CHAPTER 5

IMAGE SEGMENTATION USING SPECTRAL HISTOGRAMS

Image segmentation is a most fundamental problem in computer vision and image understanding. In this chapter, we propose an new energy functional for image segmentation which requires explicitly a feature and a distance measure for each segmented region. Using the spectral histogram proposed in the Chapter 4, we derive an iterative and deterministic approximation algorithm for segmentation. We apply the algorithm to segmenting images under different assumptions.

5.1 Introduction

Image segmentation is the central problem in computer vision, and the performance of high-level modules such as recognition critically depend on segmentation results. Roughly segmentation can be defined as a constrained partition problem. Each partition, or region, should be as homogeneous as possible and neighboring partitions should be as different as possible. Two computational issues can be identified from the definition. The first issue is to define features to be used and the associated similarity/dissimilarity measure. Given the similarity/dissimilarity measure, the second issue is to design a computational procedure to derive a solution for a given input image.

To address the first issue, one needs to define a feature, which can be a scalar or vector and a distance measure in the feature space. In the literature, this issue is not well addressed. For intensity images, mean and variance values are most widely used, which are closely related to the Gaussian assumption behind many algorithms [152] [89] [88] [13] [94] [151]. For texture images, many textural features have been proposed [44] [43] [61] [18] [65] [21] [127] [107]. Those features in general are only justified through experiments on selected texture images. Comparing with existing features, spectral histograms provide a well-justified feature vector. As demonstrated in the previous chapter, the spectral histogram can effectively capture the texture appearance and has been successfully applied to image modeling and classification.

The second issue has been studied extensively in the literature. Existing methods can be classified roughly into local algorithms and optimization of a global criterion. Local algorithms include edge detectors and region growing [152] [89] [88] [13]. While good experimental results have been obtained, the major difficulty of local algorithms is that the segmented regions are not guaranteed to be homogeneous. For example, in region growing, two different regions may be grouped together due to noise and variations in local areas. Optimization approaches can be further divided as local and global based on the segmentation criterion. If the criterion only consists of local terms, such as in pair-wise classification, those approaches may suffer the problem of local algorithms. The results may critically depend on parameter values such as the number of regions.

For the global optimization, the central problem is to derive an efficient algorithm to achieve a good solution due to the high dimensionality of the potential solution space. In this regard, Mumford-Shah energy functional for segmentation [94] given in (4.1) is representative in that most existing segmentation algorithms are special cases [92]. As pointed out earlier, the underlying assumption of the solution space is piece-wise smooth images. However, to give a solution for segmentation, the solution space f must be piece-wise constant by the definition of segmentation. In this case, the energy function becomes [94]:

$$E(\Gamma) = \sum_i \int\int_{R_i} (g(x,y) - mean_{R_i}(g))^2 dxdy + \nu|\Gamma|, \tag{5.1}$$

where

$$mean_{R_i}(g) = \frac{1}{|R_i|} \int\int_{R_i} g(x,y)dxdy.$$

Here $|R_i|$ is the area of region R_i. There are several limitations of the energy functional (5.1). The feature to be used is limited to the mean value of a region, which is not sufficient for characterizing regions as demonstrated in the previous chapter and also may give undesirable solutions when the mean values of two regions are very close. Another problem is that a result obtained by minimizing the energy functional is unpredictable in regions which cannot be described by their mean values. Some of problems are resolved by using a Gaussian model in the Region Competition algorithm by Zhu and Yuille [151].

In this chapter, we extend the model (5.1) using the spectral histogram and the associated distance measure. We develop an algorithm which couples the feature detection and segmentation steps together by extracting features based on the currently available segmentation result. We also develop an algorithm which identifies regional features in homogeneous texture regions automatically.

In Section 5.2 we give a formulation of our segmentation energy functional. Section 5.3 describes our segmentation algorithm. Section 5.4 provides experimental results when features for each region are given manually. Section 5.5 describes an automated algorithm for detecting features of homogeneous texture regions. Section 5.6 proposes a method for precise texture boundary localization. Section 5.7 discusses future research questions along this line. Section 5.8 summarizes the chapter.

5.2 Formulation of Energy Functional for Segmentation

Following the notations used in Mumford and Shah [94], let R be a grid defined on a planar domain and R_i, $i = 1, \cdots, n$ be a disjoint subset of R, Γ_i be the piecewise smooth boundary of R_i, and Γ be the union of Γ_i, $i = 1, \cdots, n$. A feature $\mathcal{F}_i$ is associated with each region R_i, $i = 1, \cdots, n$. We also define R_0, which is called *background* [135], as

$$R_0 = R - (R_1 \cup \cdots R_n).$$

Based on the energy functional by Mumford and Shah [94], given an input image I, we define an energy functional for segmentation as

$$E(R_i, n) = \lambda_R \sum_{i=1}^{n} \sum_{(x,y)\in R_i} D(\mathcal{F}_{R_i}(x,y), \mathcal{F}_i) + \lambda_{\mathcal{F}} \sum_{i=1}^{n} \sum_{j=1}^{n} D(\mathcal{F}_i, \mathcal{F}_j) + \lambda_\Gamma \sum_{i=1}^{n} |\Gamma_i| - \sum_{i=1}^{n} |R_i|. \tag{5.2}$$

Here D is a distance measure between a feature at a pixel location and the feature vector of the region, λ_R, $\lambda_{\mathcal{F}}$, and λ_Γ are weights that control the relative contributions of the corresponding terms.

The functional given in (5.2) is motivated by a special case of the functional by Mumford and Shah [94], as shown in (5.1). In (5.2), the first term encodes the homogeneity requirement in each region R_i and the second term requires that the features of the regions should be as different as possible. Here we allow R_i to consist of several connected regions and we drop the requirement of neighboring regions in the second term. Alternatively, we can require that each region must be a connected region and only include neighboring regions in the second term.

The third term requires that boundaries of regions should be as short as possible, or as smooth as possible. The last term is motivated by the fact that some regions may not be described well by the selected features. In that case, those regions should not be labeled as segmented regions. Rather, those regions should be treated as background, which can be viewed as grouping through inhomogeneity. To illustrate the motivations, Figure 5.1 shows an intensity image where two regions have similar mean values but different variances. If we use mean values as features, we argue that the most reasonable output should be one homogeneous region and another region which can not be described by the current model. We will discuss this issue later for segmentation at different scales.

5.3 Algorithms for Segmentation

Given the energy functional defined in (5.2), now the question is to compute a good solution for a given image. Obviously, due to high dimensionality of the problem, it is computationally not possible to achieve a good solution by search in the potential solution space.

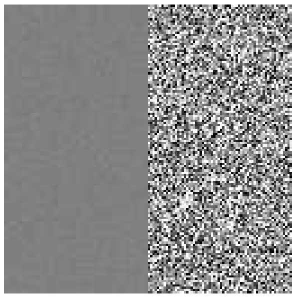


Figure 5.1: Gray-level image with two regions with similar means but different variances.

Based on different assumptions, we derive approximate solutions. A simple case is that feature vectors $\mathcal{F}_i$ are given. In this case, the problem becomes supervised classification/segmentation problem. For pixels within a homogeneous region, the label should be determined based on the minimum distance classifier, which minimizes the first term in (5.2). For pixels near boundaries between different regions, the boundary term plays an additional role besides the distance from each feature vector. Pixels that cannot be classified well by any given feature vector should be labeled as the background.

Another case is that seed points are given. In this case, the minimization of the energy functional essentially can be achieved through a procedure similar to region growing, as demonstrated in [151]. An iterative algorithm can be used by alternating feature estimation and region growing [151]. In the feature estimation phase, we fix the segmentation results R_i and estimate $\mathcal{F}_i$ by minimizing the energy functional. The region growing phase is same as the procedure for the case where feature vectors $\mathcal{F}_i$ are given. This case is a generalization of region growing algorithms.

The most difficult case is to automatically identify suitable feature vectors for a given input image and derive a solution. If we could solve this problem, we could essentially solve the segmentation problem. In this chapter, we develop an algorithm to automatically identify features from an input image based on the relationships between different scales and neighboring regions.

Another problem is how to estimate the feature at a given pixel. A straightforward way is to use a window centered at the pixel always. Another way is to utilize the currently available results and use asymmetric windows. This helps to further minimize the boundary uncertainty [151].

In this chapter, we use an iterative but deterministic algorithm. We assume that the feature vectors for regions, which may be given manually or detected automatically, are close to the true region vectors. In other words, we do not do the feature re-estimation along the iterations. The first step prior to iterative segmentation is a minimum distance classifier which generates initial segmentation results. For a given pixel (x, y) to be updated, we first estimate the spectral histogram using asymmetric windows around the pixel. Figure 5.2 gives two examples. For simplicity, we use square windows throughout this chapter, which, in general, give good results but the biases due the square shapes are visible in several cases. Circular windows provide better approximation for arbitrary shaped boundaries and are more biologically plausible. In addition, there are many possible choices for windows and the influence of different choices will be studied in the future.

Because there are several windows to choose at pixel (x, y), for each $\mathcal{F}_i$, we use the window that has the most number of labels of R_i, and thus the feature at pixel (x, y) for different labels can be different. We use spectral histograms as features

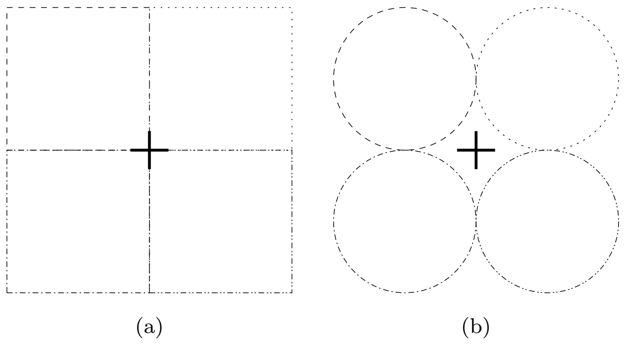


Figure 5.2: Examples of asymmetric windows. The solid cross is the central pixel. (a) Square windows. (b) Circular windows.

vectors, i.e., $\mathcal{F}_{R_i}(x,y) = H_{W^{(s)}_{(x,y)}}$, where $W^{(s)}_{(x,y)}$ is a local neighborhood, the size and shape of which are given by integration scale $W^{(s)}$ for segmentation. $W^{(s)}$ is a pre-defined neighborhood. We use χ^2-statistic as the distance measure. However, this distance measure may not provide an accurate measure close to boundaries due to the inhomogeneity. For example, in the image shown in Figure 5.1, the left region is homogeneous and the variations allowed should be small. In the right region, the variations allowed should be relatively large. To overcome this problem and provide an accurate model, we estimate a probability model of the χ^2-statistic for each given feature vector $\mathcal{F}_i$ from the initial classification result. The implementation details and an example are given in the next section. We approximate the boundary term by a local term, which is given by the percentage of pixels belonging to the region in a pre-defined neighborhood. Finally, the local updating rule at pixel (x,y) is given

$$\pi_i(x,y) = (1-\lambda_\Gamma)P(\chi^2(H_{W^{(s)}_{(x,y)}}, H_i))+\lambda_\Gamma \sum_{(x_1,y_1)\in N(x,y)} (L(x_1,y_1) == i)/|N(x,y|. \quad (5.3)$$

Here $L(x,y)$ is the current region label of pixel (x,y). $N(x,y)$ is a user-defined neighborhood, and the eight nearest neighbors are used, and λ_B is a parameter that controls the relative contributions from the region and boundary terms. The new label of (x, y) is assigned as the one that gives the maximum $\pi_i(x, y)$. To save computation, (5.3) only needs to be applied at pixels between different regions, which gives rise to a procedure similar to region growing. A special case of (5.3) is for pixels along boundaries between background region and a given region because we do not assume any model for the background region. For pixel $(x, y) \in R_0$, which is adjacent to region R_i, $i \neq 0$, if

$$\chi^2(H_{W^{(s)}_{(x,y)}}, H_i) < \lambda_B * T_i,$$

we assign label i to (x, y). Here T_i is a threshold for region R_i, which is determined automatically based on the initial classification result, and λ_B is a parameter which determines relative penalty for unsegmented pixels.

In order to extract spectral histograms, we need to select filters. For segmentation, we use the same eight filters for classification including the intensity filter, two gradient filters, LoG with two scales and three Gabor filters with different orientations unless specified otherwise.

5.4 Segmentation with Given Region Features

In this section, we study a special case of image segmentation. Here we assume that feature vector $\mathcal{F}_i$ for each region is given manually by specifying a seed pixel in the region. We assume that given features provide a good approximation of the true model underlying regions. The features are defined as spectral histograms and are estimated at integration scale $W^{(s)}$.

5.4.1 Segmentation at a Fixed Integration Scale

First feature vectors $\mathcal{F}_i$ are extracted from windows centered at given pixel locations, the size of which is specified by integration scale $W^{(s)}$. Then the image is classified using feature vectors $\mathcal{F}_i$, and the classification result is used as the initial segmentation. To save computation, the initial result is generated by sub-sampling the input image. To obtain parameters T_i and estimate a probability model, we compute the histograms of the χ^2-statistic between the computed and given spectral histograms, which are shown in Figure 5.4(b) for the image shown in Figure 5.3(a). Parameter T_i is determined by the first trough after the first peak from its histogram. Based on the assumption that feature vectors $\mathcal{F}_i$ are close to the true feature vectors, we derive a probability model by assigning zero probability for values larger than T_i. The derived probability models are shown in Figure 5.4(b).

To illustrate the effectiveness of using the derived probability model and asymmetric windows, Figure 5.5(a) shows a row from the image shown in Figure 5.3(a). Figure 5.5(b) shows the probability of the two labels at each pixel using asymmetric windows. Here we can see that the edge point is localized precisely at the true location. Figure 5.5(c) show the probability using windows centered at pixels. There is an interval where labels can not be decided because the spectral histogram computed in the interval does not belong to either of the regions. This shows that the probability model is sensitive. For comparison, Figure 5.6 shows the results using χ^2-statistic directly. In both asymmetric and central windows cases, the decision boundaries are not sharp and edge points cannot be localized accurately.

Then the initial segmentation result is refined through an iterative procedure similar to region growing but with fixed region features. Due to that spectral histograms

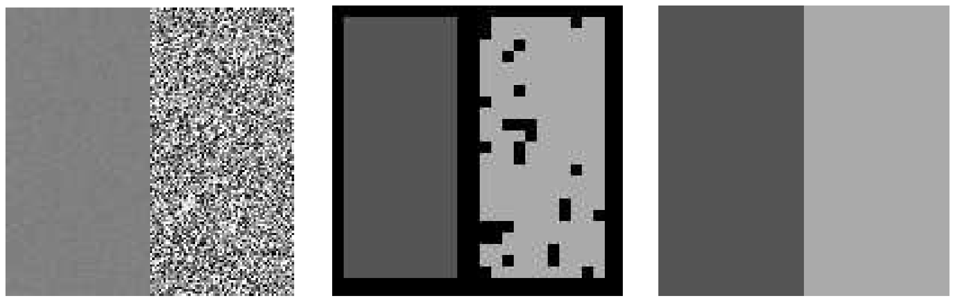
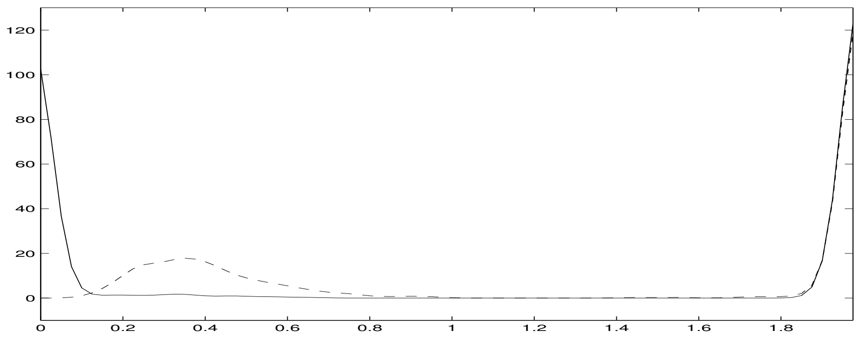

(a) (b) (c)

Figure 5.3: Gray-level image segmentation using spectral histograms. The integration scale $W^{(s)}$ for spectral histograms is a 15×15 square window, $\lambda_\Gamma = 0.2$, and $\lambda_B = 3$. Two features are given at $(32, 64)$ and $(96, 64)$. (a) A synthetic image with size 128×128. The image is generated by adding zero-mean Gaussian noise with different σ's at left and right regions. (b) Initial classification result. (c) Final segmentation result. The segmentation error is 0.00 % and all the pixels are segmented correctly.

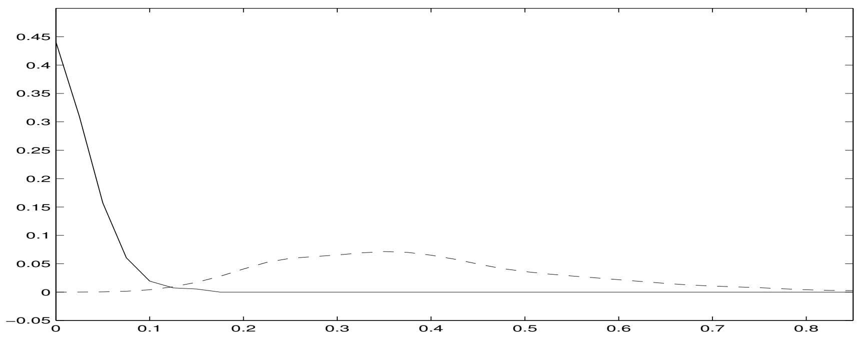
characterize texture properties well, we obtain good experimental results even with this simple algorithm. The integration scale for spectral histograms is 15×15. Figure 5.3(b) shows the initial classification result. The final result is shown in Figure 5.3(c), where all the pixels are segmented correctly. Due to that the image consists of two images with similar mean values but different variances, if we apply nonlinear smoothing algorithms, the segmentation result would be wrong because the only feature is the mean value after smoothing.

Figure 5.7 shows another example of similar means but different variances. Here the boundary is 'S' shaped to test the algorithm for irregular boundaries. Here the boundary is preserved well but artifacts due to the square windows are evident. Near top and bottom image borders, the error is the most due to the boundary effects.

Figure 5.8 shows an example with two texture regions. Because the two textures are relatively homogeneous, the segmentation result is very accurate.



(a)



(b)

Figure 5.4: The histogram and derived probability model of χ^2-statistic for the given region features. Solid lines stand for left region and dashed lines stand for right region. (a) The histogram of the χ^2-statistic between the given feature and the computed ones at a coarser grid. (b) The derived probability model for the left and right regions.

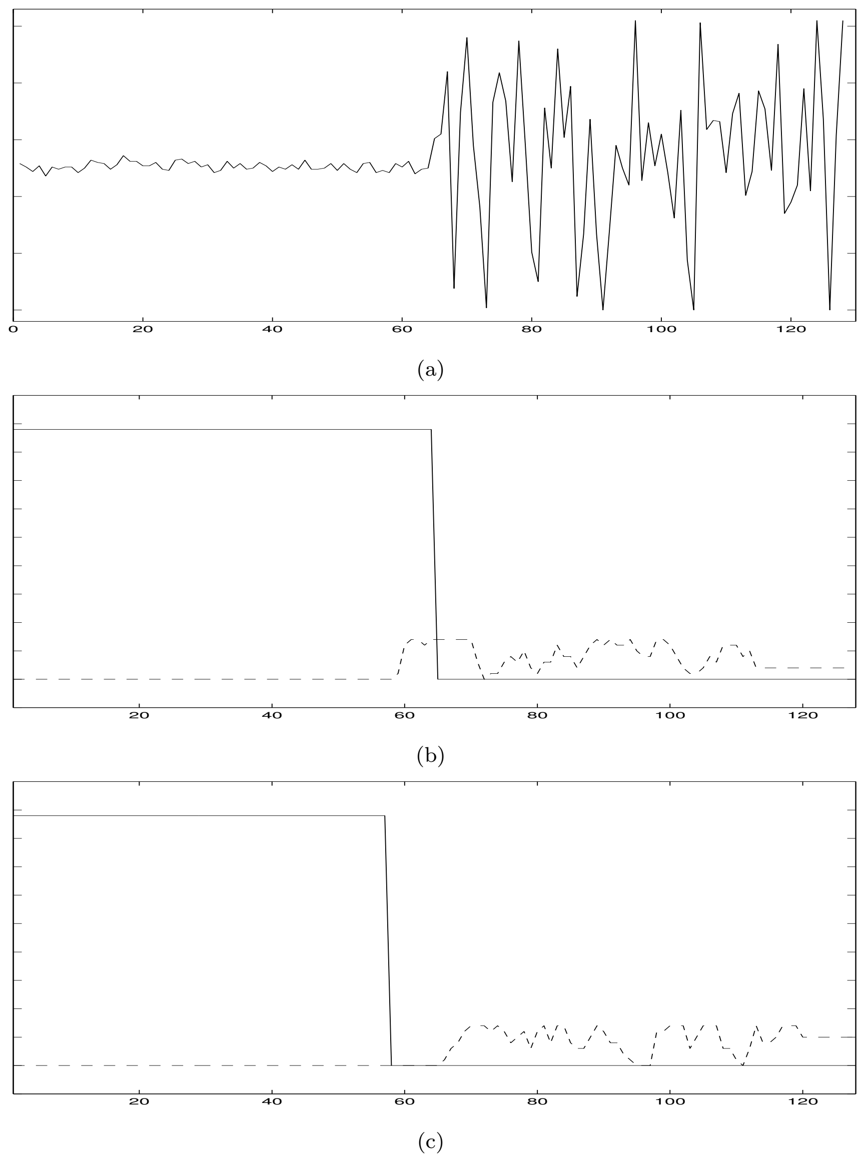
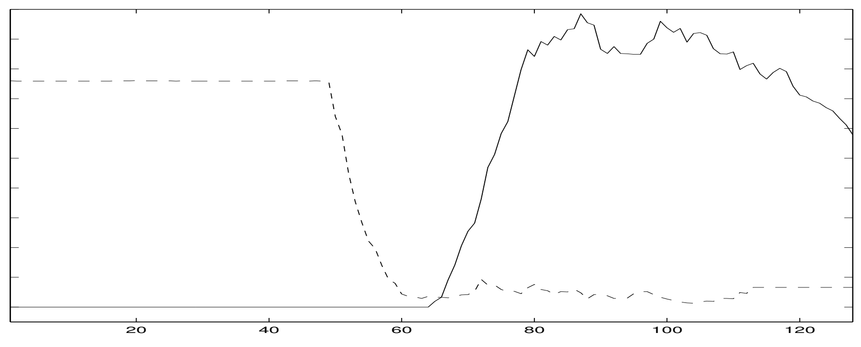
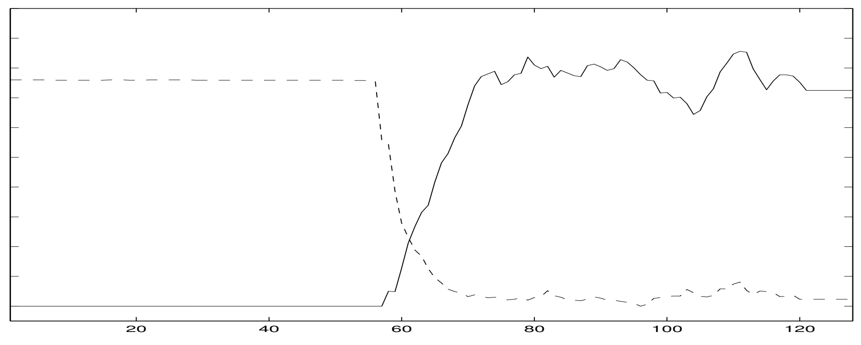


Figure 5.5: A row from the image shown in Figure 5.3 and the result using derived probability model. In (b) and (c), solid lines stand for left region and dashed lines stand for right region. (a) The 64th row from the image. (b) The probability of the two given regional features using asymmetric windows when estimating spectral histogram. The edge point is correctly located between columns 64 and 65. (c) Similar to (a) but using windows centered at the pixel to compute spectral histogram. Labels between columns 58 and 65 cannot be decided. This is because that the computed spectral histograms within that interval do not belong to either region.



(a)



(b)

Figure 5.6: Classification result based on χ^2-statistic for the row shown in Figure 5.4(a). Solid lines stand for left region and dashed lines stand for right region. (a) χ^2-statistic from the two given regional features using asymmetric windows when estimating spectral histogram. If we use the minimum distance classifier, the edge point will be located between columns 65 and 66, where the true edge point should be between columns 64 and 65. (b) Similar to (b) but using windows centered at the pixel to compute spectral histogram. The edge point is localized between 61 and 62.

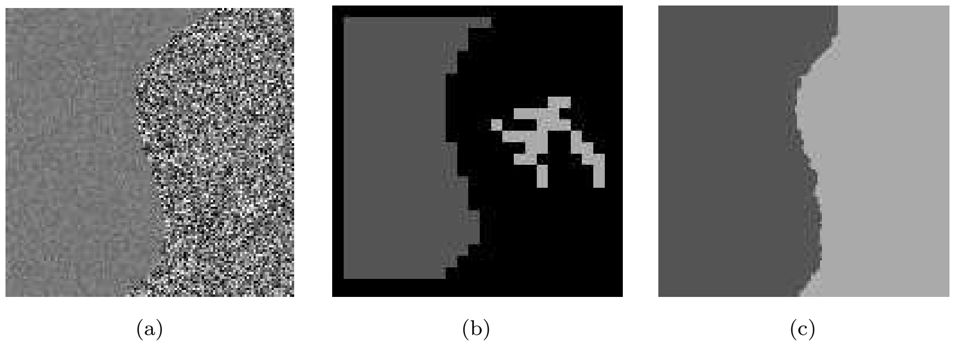


Figure 5.7: Gray-level image segmentation using spectral histograms. $W^{(s)}$ is a 15×15 square window, $\lambda_\Gamma = 0.2$, and $\lambda_B = 5$. Two features are given at $(32, 64)$ and $(96, 45)$. (a) A synthetic image with size 128×128. The image is generated by adding zero-mean Gaussian noise with different σ's at the two different regions. Here the boundary is 'S' shaped to test the segmentation algorithm in preserving boundaries. (b) Initial classification result. (c) Final segmentation result.

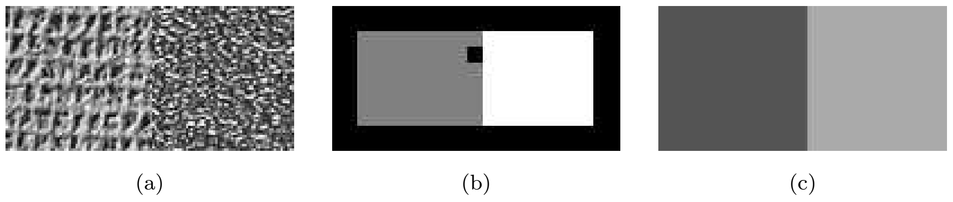


Figure 5.8: Texture image segmentation using spectral histograms. $W^{(s)}$ is a 29×29 square window, $\lambda_\Gamma = 0.2$, and $\lambda_B = 2$. Features are given at pixels $(32, 32)$ and $(96, 32)$. (a) A texture image consisting of two texture regions with size 128×64. (b) Initial classification result. (c) Final segmentation result.

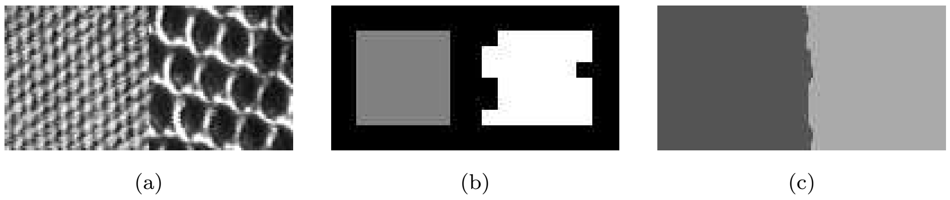


Figure 5.9: Texture image segmentation using spectral histograms. $W^{(s)}$ is a 29×29 square window, $\lambda_\Gamma = 0.2$, and $\lambda_B = 3$. (a) A texture image consisting of two texture regions with size 128×64. (b) Initial classification result. (c) Final segmentation result.

Figure 5.9 shows an example of two texture regions. Because the right region is not homogeneous with respect to the integration scale 29×29, the boundaries between the two texture regions are displaced several pixels and boundaries are not smooth due to the black and white patterns in the right region. Overall, the segmentation result is good.

Figures 5.10, 5.11, 5.12, and 5.13 show examples of textures consisting of four texture regions. The segmentation results are good given the integration scale used and the inhomogeneity in the texture regions.

Figures 5.14 and 5.15 show two challenging examples for texture segmentation algorithms. Because the boundaries are not distinctive, it is difficult even for humans to localize the boundaries precisely. We applied the same algorithm to the images. While the results are not perfect, they are quite satisfactory if compared to many segmentation methods.

Figure 5.16 shows a texton-like image. Because we only use Gabor filters at three different orientations, the spectral histogram of the top region is not representative, resulting in obvious displacement of boundary toward the upper region. But if we

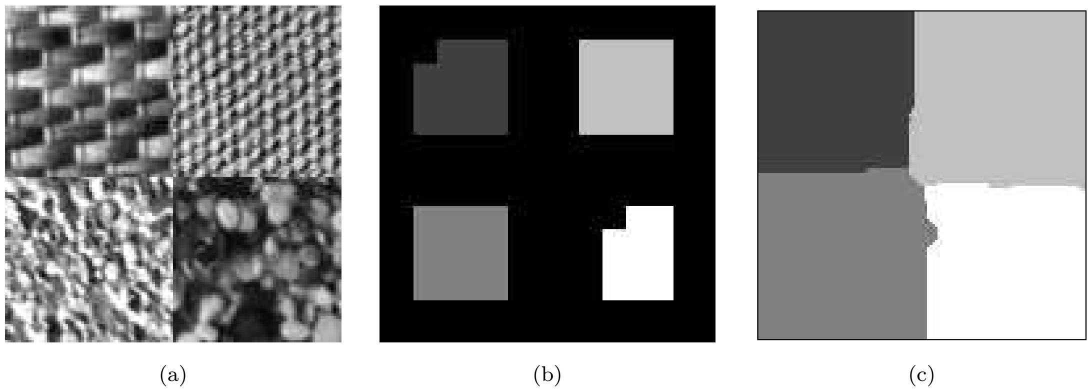


Figure 5.10: Texture image segmentation using spectral histograms. $W^{(s)}$ is a 35×35 square window, $\lambda_\Gamma = 0.4$, and $\lambda_B = 3$. Four features are given at $(32, 32)$, $(32, 96)$, $(96, 32)$, and $(96, 96)$. (a) A texture image consisting of four texture regions with size 128×128. (b) Initial classification result. (c) Final segmentation result.

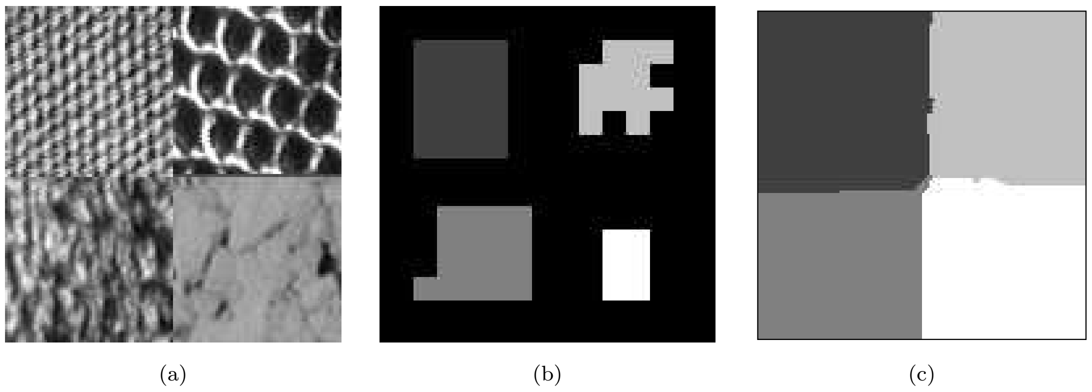


Figure 5.11: Texture image segmentation using spectral histograms. $W^{(s)}$ is a 35×35 square window, $\lambda_\Gamma = 0.4$, and $\lambda_B = 3$. Four features are given at $(32, 32)$, $(32, 96)$, $(96, 32)$, and $(96, 96)$. (a) A texture image consisting of four texture regions with size 128×128. (b) Initial classification result. (c) Final segmentation result.

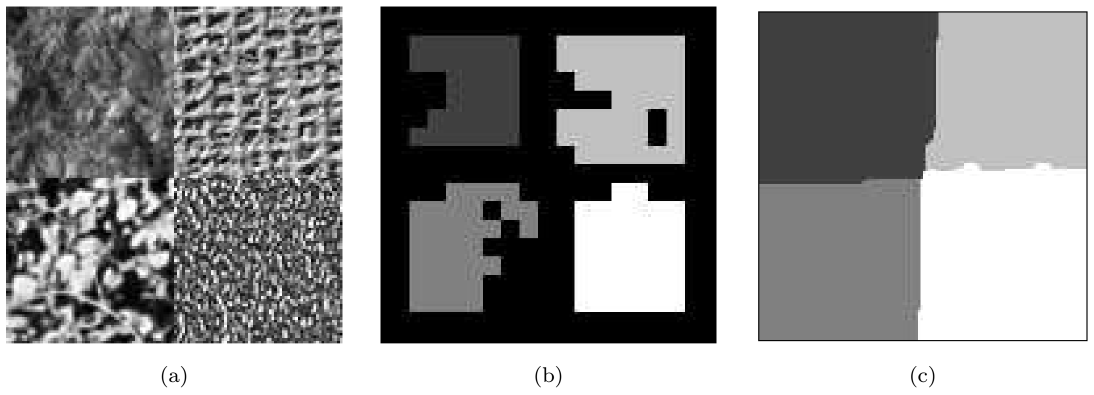


Figure 5.12: Texture image segmentation using spectral histograms. $W^{(s)}$ is a 29×29 square window, $\lambda_\Gamma = 0.2$, and $\lambda_B = 3$. Four features are given at $(32, 32)$, $(32, 96)$, $(96, 32)$, and $(96, 96)$. (a) A texture image consisting of four texture regions with size 128×128. (b) Initial classification result. (c) Final segmentation result.

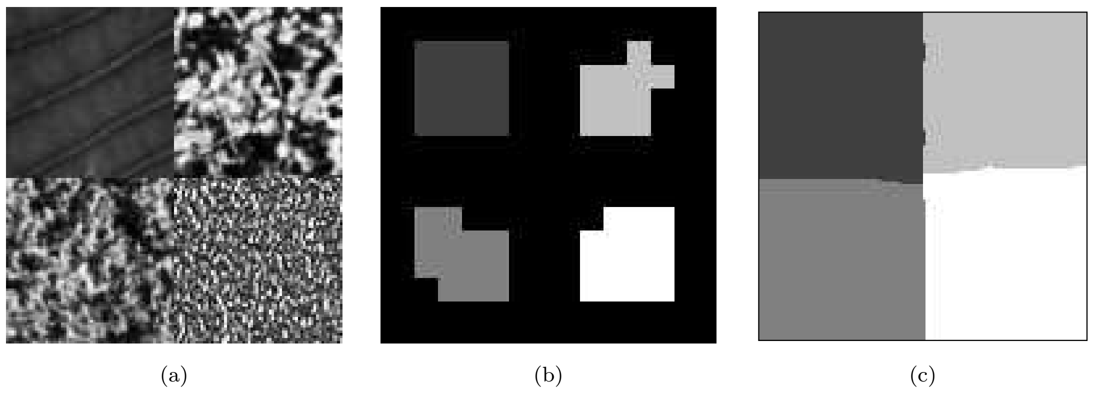


Figure 5.13: Texture image segmentation using spectral histograms. $W^{(s)}$ is a 35×35 square window, $\lambda_\Gamma = 0.4$, and $\lambda_B = 3$. Four features are given at $(32, 32)$, $(32, 96)$, $(96, 32)$, and $(96, 96)$. (a) A texture image consisting of four texture regions with size 128×128. (b) Initial classification result. (c) Final segmentation result.

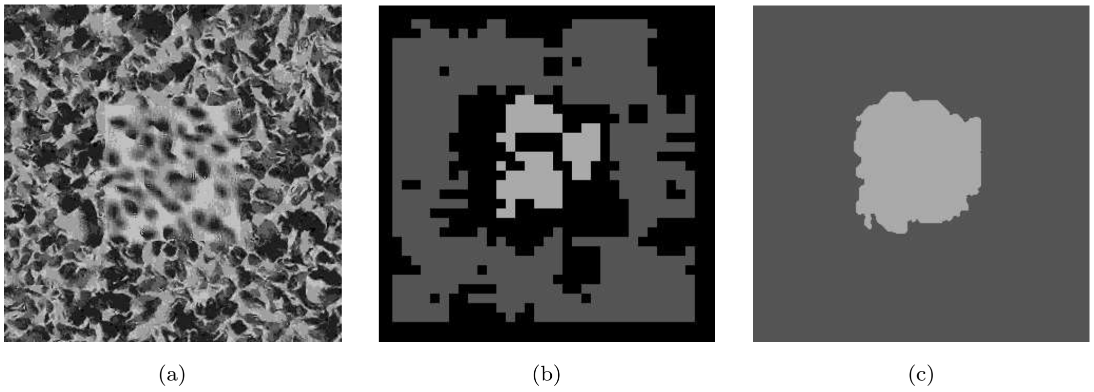


Figure 5.14: A challenging example for texture image segmentation. $W^{(s)}$ is a 35×35 square window, $\lambda_\Gamma = 0.4$, and $\lambda_B = 20$. Two features are given at $(160, 160)$ and $(252, 250)$. (a) Input image consisting of two texture images, where the boundary can not be localized clearly because of their similarity. The size of the image is 320×320 in pixels. (b) Initial classification result. (c) Final segmentation result.

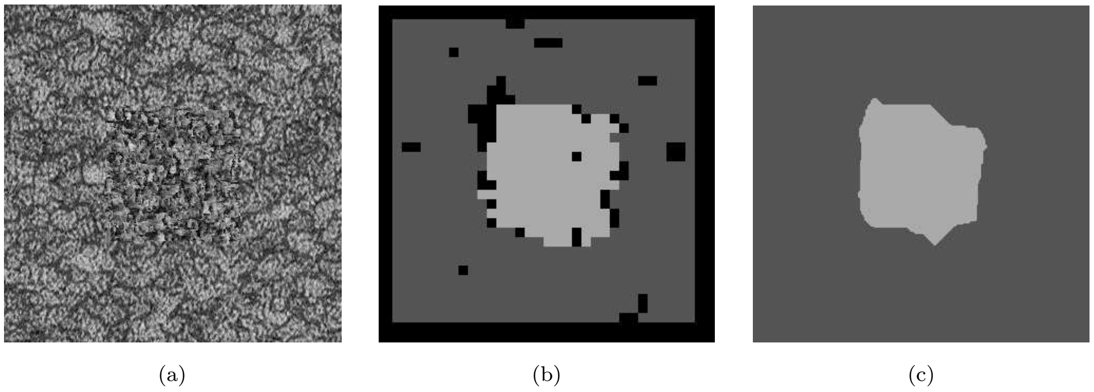


Figure 5.15: Another challenging example for texture segmentation. $W^{(s)}$ is a 35×35 square window, $\lambda_\Gamma = 0.4$, and $\lambda_B = 20$. Two features are given at $(160, 160)$ and $(252, 250)$. (a) Input image consisting of two texture images, where the boundary can not be localized clearly because of their similarity. The size of the image is 320×320 in pixels. (b) Initial classification result. (c) Final segmentation result.

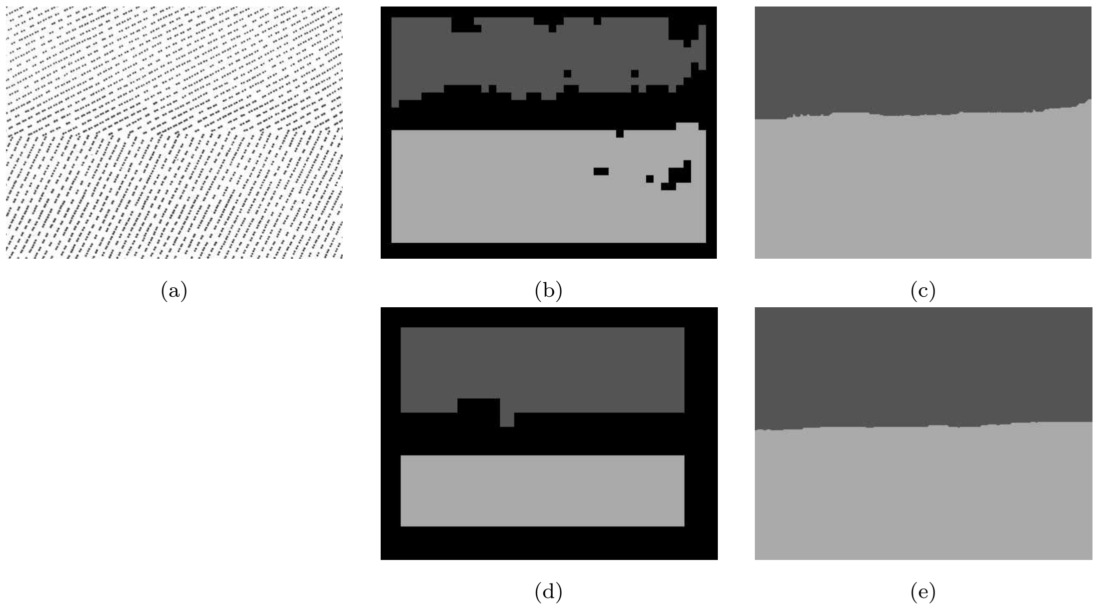
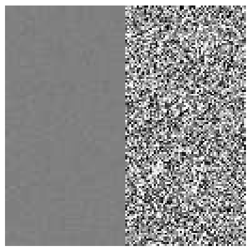


Figure 5.16: Segmentation for a texton image with oriented short lines. $W^{(s)}$ is a 35×35 square window, $\lambda_\Gamma = 0.4$, and $\lambda_B = 10$. Two features are given at $(185, 67)$ and $(180, 224)$. (a) The input image with size of 402×302 in pixels. (b) The initial classification result. (c) The segmentation result using spectral histograms. (d) The initial classification result using two Gabor filters $Gcos(10, 30°)$ and $Gcos(10, 60°)$. (e) The segmentation result using two Gabor filters. The result is improved significantly.

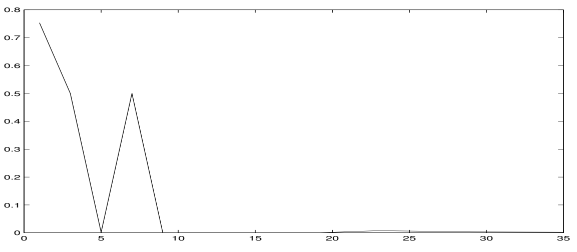
allow to change filters used, we obtain a much better result as shown in Figure 5.16(d) and (e).

5.4.2 Segmentation with Multiple Scales

In this section, we briefly study the effects of integration scales on the segmentation results. Because no textures are absolutely homogeneous with respect to the finite integration scales, in order to capture the characteristics of texture images, there is a minimum integration scale required. In the framework of spectral histograms, we



(a)



(b)

Figure 5.17: Segmentation results at different integration scales. Parameters $\lambda_\Gamma = 0.4$, and $\lambda_B = 4$ are fixed. (a) The input image. (b) The percentage of mis-classified pixels.

can estimate the minimum scale of a region using the relationships between different integration scales, which will be explained and used in Section 5.5

We use the example shown in Figure 5.17(a), which was used previously. We try the integration scales from 1×1 to 35×35. For the final segmentation result obtained at each scale, we calculate the percentage of mis-segmented pixels using the ground truth image. Figure 5.17(b) shows the result. Except at scale 7×7, the segmentation decreases until windows get larger than 23×23. While the error varies at larger windows, the error is very low. The error at scale 23×23 is less than 0.2 %.

The segmentation results for selected small scales are shown in Figure 5.18. These results reveal some of desirable properties of the segmentation algorithm. For example, at scales less than 9×9, the right region is not homogeneous. The desirable result at these scales is that the right region is classified as background. The result obtained at scale 5×5 is produced due to the randomness. Note that the given pixels are used to calculate the region feature $\mathcal{F}_i$ only and they are not used as seed pixels for region growing.

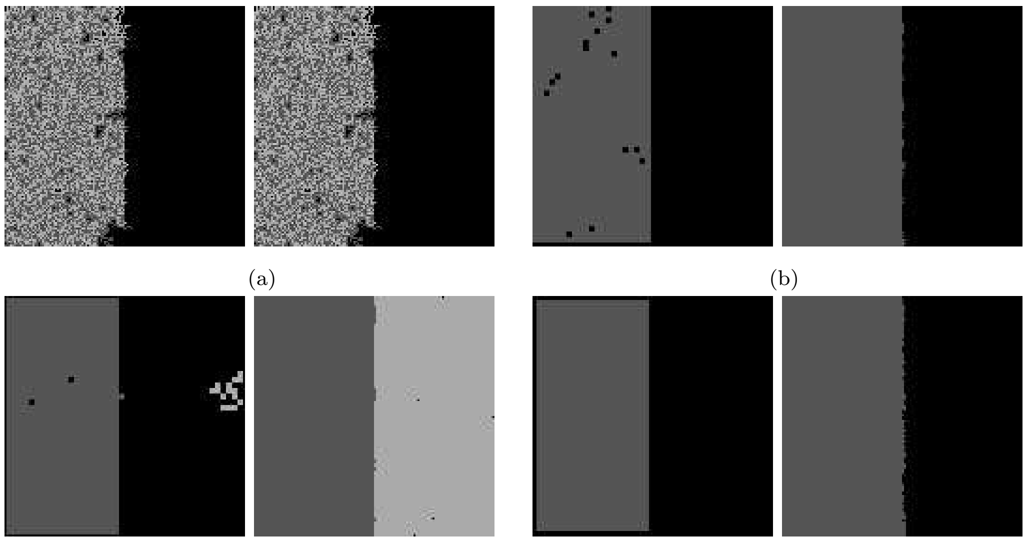


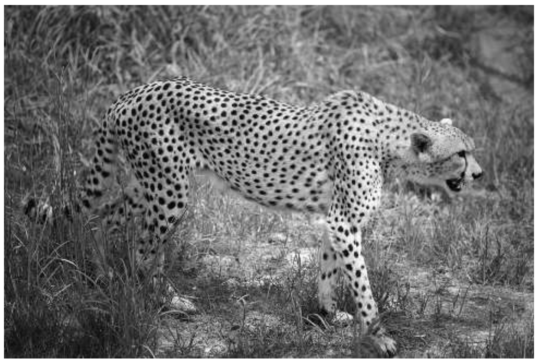
Figure 5.18: Segmentation results using different segmentation scales for the image shown in Figure 5.17(a). In each sub-figure, the left shows the initial classification result and the right shows the segmentation result. Parameters $\lambda_\Gamma = 0.4$, and $\lambda_B = 4$ are fixed. (a) $W^{(s)}$ is a 1×1 square window. (b) $W^{(s)}$ is a 3×3 square window. (c) $W^{(s)}$ is a 5×5 square window. (d) $W^{(s)}$ is a 7×7 square window.

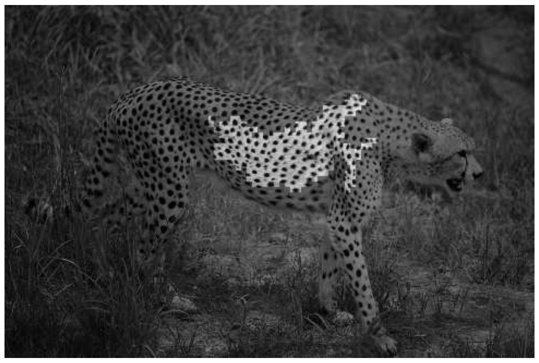
5.4.3 Region-of-interest Extraction

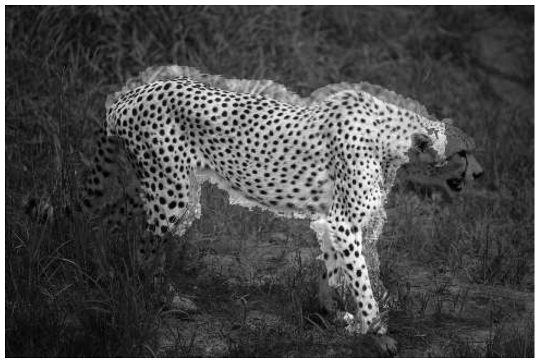
Here we try to segment natural texture images. Because natural images consist many regions that are not homogeneous texture regions, we segment regions that are specified through region features. For each image, we only give one region feature and the algorithm is essentially to segment region of interest. As mentioned before, in our algorithm, no assumption is made regarding the distributions and properties of the background regions.

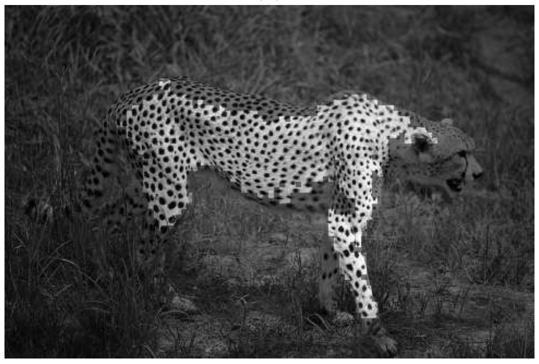
We apply the same algorithm. Figure 5.19 shows the result for a cheetah image. If we use the same filters, the initial result is shown in Figure 5.19(b) and the segmentation result is shown in Figure 5.19(c). Here the boundaries are not localized well due to the similarity between the surrounding areas and the white part of cheetah skin. The tail is not included because that area is relatively dark. If we do not use the intensity filter and use gradient and LoG filters, the tail is then correctly segmented as shown in Figure 5.19(e). If we have a database for recognition, the cheetah can be easily recognized due to its distinctive skin pattern which is characterized by its spectral histograms.

Figure 5.20 shows an indoor image which includes a sofa with texture surface. Figure 5.20(c) shows the final result. The lower boundary of the sofa is localized well due to that the floors are different from the sofa texture. However, the boundaries of top and right part are not localized well due to that the intensity values at those regions are similar to the white part of sofa texture. If we put another regional feature in the white area, the top boundary can be localized more accurately through competition as shown in Figure 5.20(d).


(a)


(b)


(c)


(d)

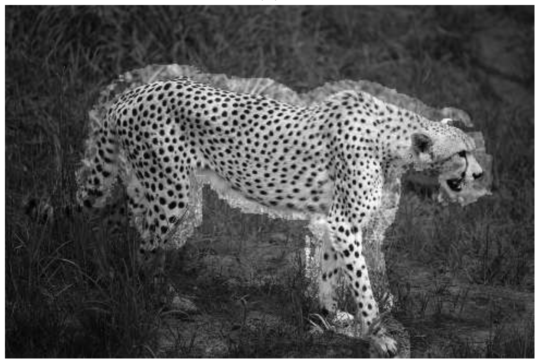

(e)

Figure 5.19: A texture image with a cheetah. The feature vector is calculated at pixel $(247, 129)$ at scale 19×19, $\lambda_\Gamma = 0.2$, and $\lambda_B = 2.5$. To demonstrate the accuracy of the results, the classification and segmentation results are embedded into the original image by lowering the intensity values of the background region by a factor of 2. (a) The input image with size 324×486. (b) The initial classification result using 8 filters. (c) The final segmentation result using 8 filters. (d) The initial classification result using 6 filters consisting of D_{xx}, D_{yy}, $LoG(\sqrt{2}/2)$, $LoG(1)$, $LoG(2)$ and $LoG(3)$. (e) The final segmentation result corresponding to (d).

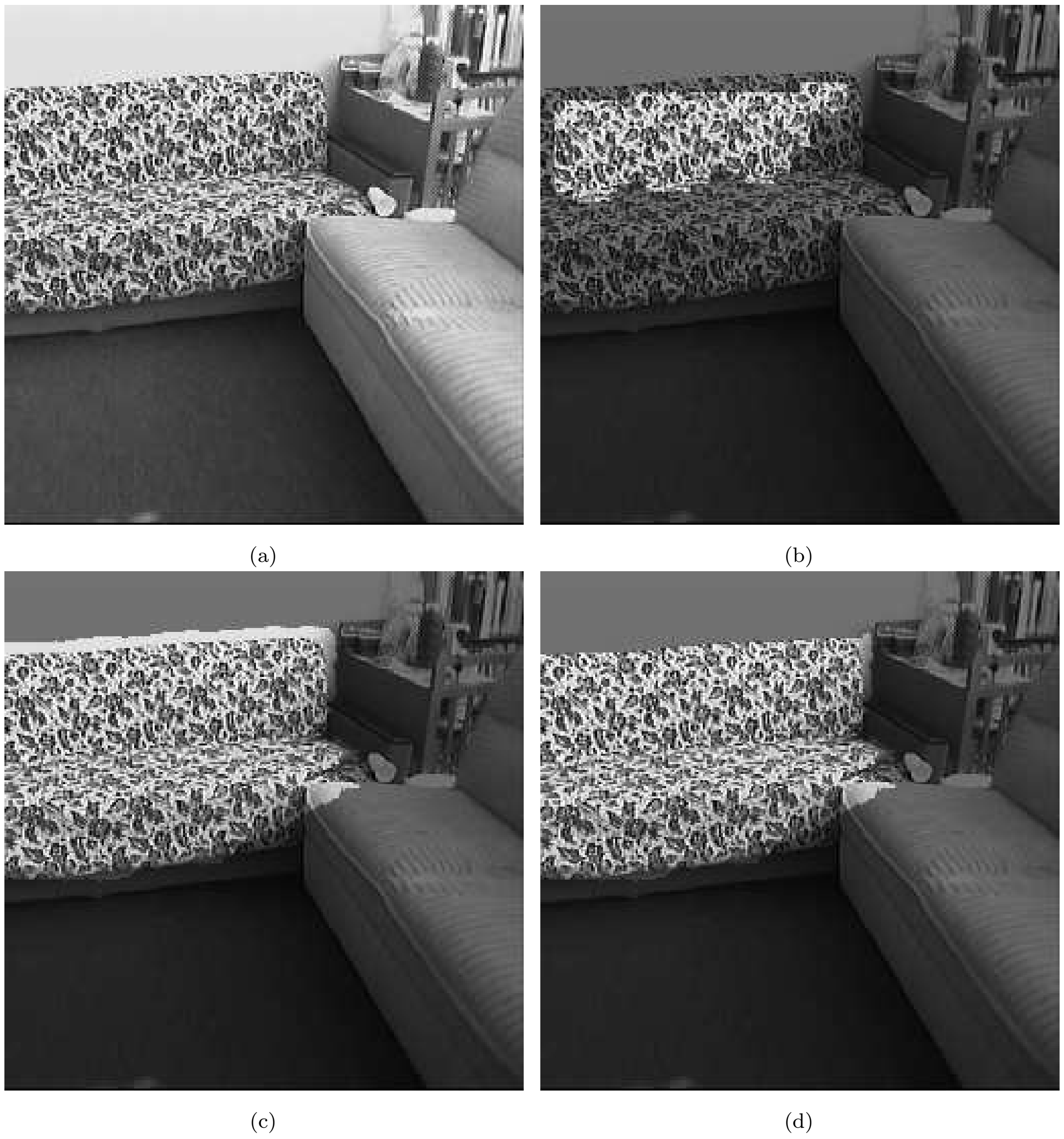


Figure 5.20: An indoor image with a sofa. The feature vector is calculated at pixel $(146, 169)$ at scale 35×35, $\lambda_\Gamma = 0.2$, and $\lambda_B = 3$. (a) Input image with size 512×512. (b) Initial classification result. (c) Final segmentation result. (d) Segmentation result if we assume there is another region feature given at $(223, 38)$.

5.5 Automated Seed Selection

In the segmentation experiments presented above, we assume that several representative pixels are given. Those pixels are used to calculate feature vectors of the corresponding regions. This assumption is obviously too restrictive for an autonomous system. Also the human visual system does not need this assumption but can deal with a wide range of texture images robustly and reliably. In this section, we attempt to develop a solution for identifying seed points automatically in an input image. The proposed method is based on the spectral histogram and is consistent with the proposed computational framework.

The basic idea under the proposed method is to identify homogeneous texture regions within a given image. As discussed in the previous chapter, the spectral histogram can be defined on image patches with different sizes and shapes and those spectral histograms defined on different patches can still be compared using the similarity/dissimilarity measure as spectral histograms are naturally normalized.

We try to identify homogeneous texture regions based on the divergence between two integration scales. Let $W^{(a)}$ be an integration scale larger than $W^{(s)}$, the integration scale for segmentation, we define the distance between the two scales centered at pixel (x, y)

$$\psi^{(s,a)}(x, y) = D(H_{W^{(s)}(x,y)}, H_{W^{(a)}(x,y)}). \tag{5.4}$$

Within a homogeneously defined texture region, $\psi^{(s,a)}$ should be small because $H^{W^{(s)}}(x, y)$ and $H^{W^{(a)}}(x, y)$ should be similar. We also define a distance measure between different windows at scale $W^{(s)}$ within the window given by $W^{(a)}$,

$$\psi^{(s,s)}(x,y) = \max_{\substack{W^{(s)}(x_1,y_1) \subset W^{(a)}(x,y) \\ W^{(s)}(x_2,y_2) \subset W^{(a)}(x,y)}} D(H_{W^{(s)}(x_1,y_1)}, H_{W^{(s)}(x_2,y_2)}). \tag{5.5}$$

Equation (5.5) is approximated at implementation using central and four corner windows within $W^{(a)}$. Finally, we want to choose features that are as different as possible from those already chosen. Suppose we choose n features already, where, $\mathcal{F}_i = H_{W^{(s)}(x,y)}$, for $i = 1, \ldots, n$, we define

$$\psi^{(c)}(x,y) = \max_{1 \le i \le n} D(H_{W^{(s)}(x,y)}, \mathcal{F}_i). \tag{5.6}$$

We have the following saliency measure

$$\psi(x,y) = (1-\lambda_C)(\lambda_A \times \psi^{(s,a)}(x,y) + (1-\lambda_A) \times \psi^{(s,s)}(x,y)) - \lambda_C \times \psi^{(c)}(x,y). \tag{5.7}$$

Here λ_A and λ_C are parameters to determine the relative contribution of each term.

To save computation, we compute $\psi(x,y)$ on a coarser grid. Feature vectors are chosen according to the value of $\psi(x,y)$ until

$$\lambda_A \times \psi^{(s,a)}(x,y) + (1-\lambda_A) \times \psi^{(s,s)}(x,y) < T_A,$$

where T_A is a threshold.

Figures 5.21-5.25 show the segmentation results for texture images. Here the algorithm identifies the feature vectors automatically instead of manually chosen features. Due to the inhomogeneity, some of texture boundaries are not as good as the results with the manually given feature points. This is due to that the windows for feature selection are large compared to the texture regions.

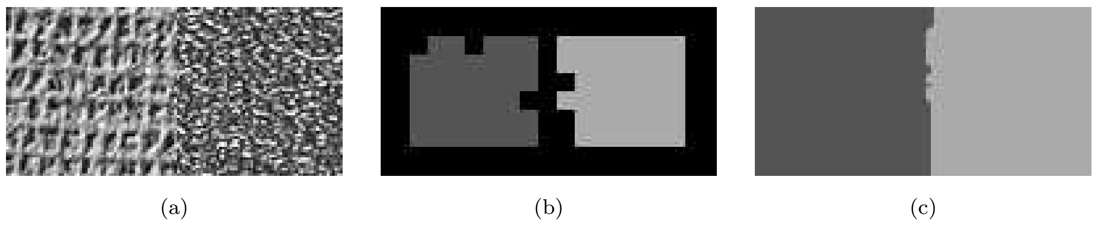


Figure 5.21: Texture image segmentation with representative pixels identified automatically. $W^{(s)}$ is a 29×29 square window, $W^{(a)}$ is a 35×35 square window, $\lambda_C = 0.1$, $\lambda_A = 0.2$, $\lambda_B = 2.0$, $\lambda_\Gamma = 0.2$, and $T_A = 0.08$. (a) Input texture image, which is shown in Figure 5.8. (b) Initial classification result. Here the representative pixels are detected automatically. (c) Final segmentation result.

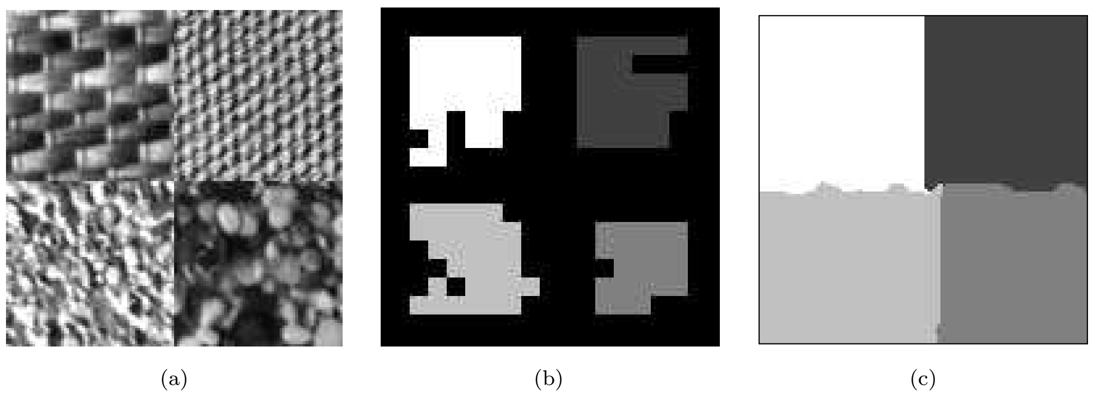


Figure 5.22: Texture image segmentation with representative pixels identified automatically. $W^{(s)}$ is a 29×29 square window, $W^{(a)}$ is a 43×43 square window, $\lambda_C = 0.4$, $\lambda_A = 0.4$, $\lambda_B = 5.0$, $\lambda_\Gamma = 0.4$, and $T_A = 0.30$. (a) Input texture image, which is shown in Figure 5.10. (b) Initial classification result. Here the representative pixels are detected automatically. (c) Final segmentation result.

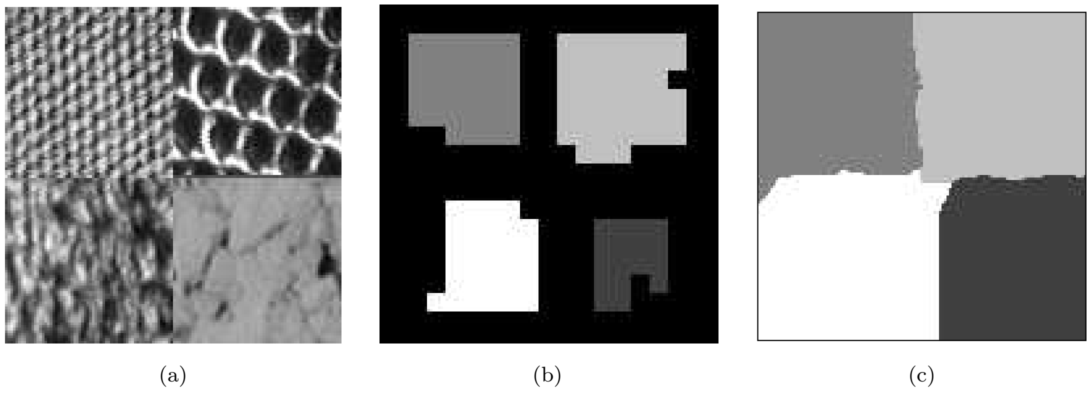
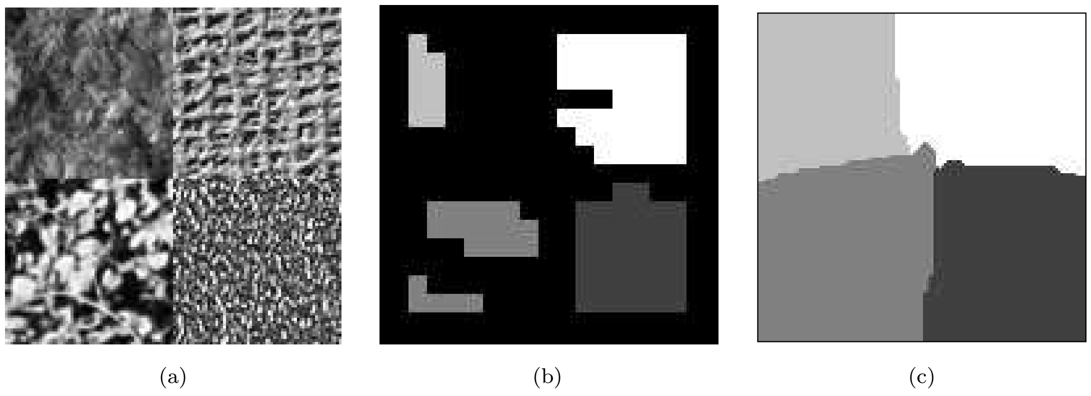


Figure 5.23: Texture image segmentation with representative pixels identified automatically. $W^{(s)}$ is a 29×29 square window, $W^{(a)}$ is a 43×43 square window, $\lambda_C = 0.1$, $\lambda_A = 0.2$, $\lambda_B = 5.0$, $\lambda_\Gamma = 0.4$, and $T_A = 0.20$. (a) Input texture image, which is shown in Figure 5.11. (b) Initial classification result. Here the representative pixels are detected automatically. (c) Final segmentation result.



$W^{(s)}$ is a 29×29 square window, $W^{(a)}$ is a 43×43 square window, $\lambda_C = 0.1$, $\lambda_A = 0.2$, $\lambda_B = 5.0$, $\lambda_\Gamma = 0.4$, and $T_A = 0.20$.

Figure 5.24: Texture image segmentation with representative pixels identified automatically. (a) Input texture image, which is shown in Figure 5.12. (b) Initial classification result. Here the representative pixels are detected automatically. (c) Final segmentation result.

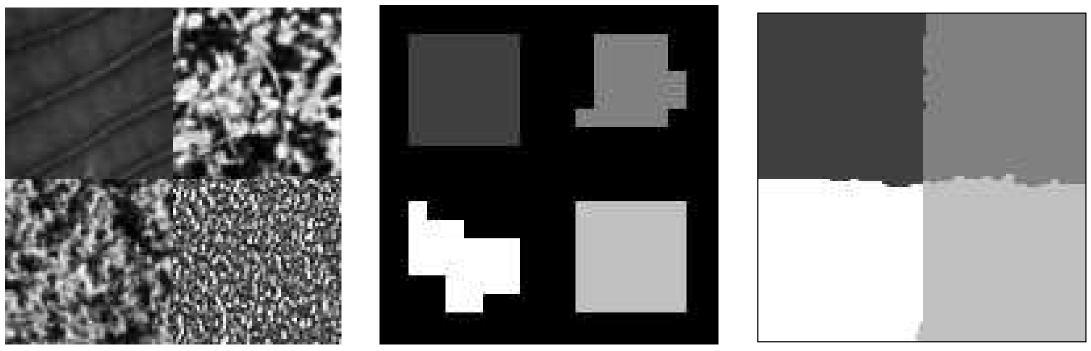

(a) (b) (c)

Figure 5.25: Texture image segmentation with representative pixels identified automatically. $W^{(s)}$ is a 29×29 square window, $W^{(a)}$ is a 43×43 square window, $\lambda_C = 0.1$, $\lambda_A = 0.2$, $\lambda_B = 5.0$, $\lambda_\Gamma = 0.4$, and $T_A = 0.20$. (a) Input texture image, which is shown in Figure 5.13. Here the representative pixels are detected automatically. (c) Final segmentation result.

5.6 Localization of Texture Boundaries

Because textures need to be characterized by spatial relationships among pixels, relatively large integration windows are needed in order to extract meaningful features, which is evident from Figure 4.23(b), which shows that the classification performance degrades dramatically when the integration scale gets too small. The large integration scale we use results in large errors along texture boundaries due to the uncertainty introduced by large windows [13]. By using asymmetric windows for feature extraction, the uncertainty effect is reduced. However, for arbitrary texture boundaries, the errors along boundaries can be large even when the overall segmentation performance is good. For example, Figure 5.26(b) shows a segmentation result using spectral histograms. While the segmentation error is only 6.55%, visually the

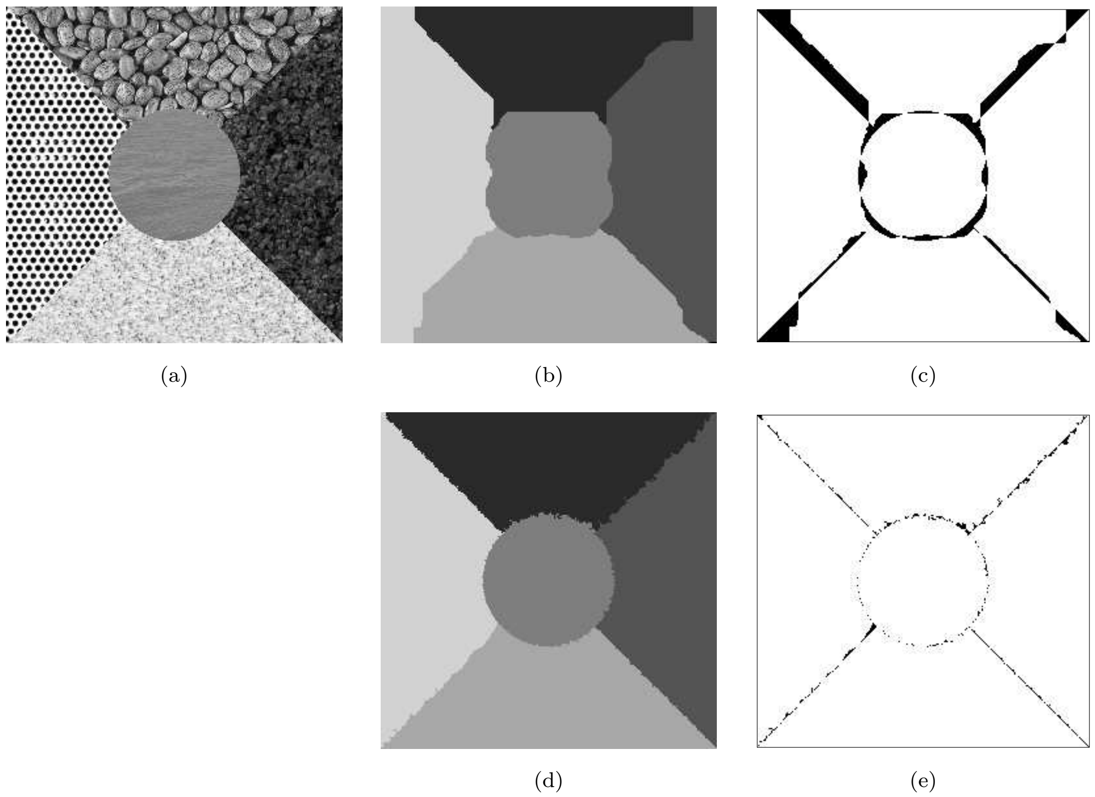


Figure 5.26: (a) A texture image with size 256 × 256. (b) The segmentation result using spectral histograms. (c) Wrongly segmented pixels of (b), represented in black with respect to the ground truth. The segmentation error is 6.55%. (d) Refined segmentation result. (e) Wrongly segmentation pixels of (d), represented in black as in (c). The segmentation error is 0.95%.

segmentation result is intolerable due to large errors along texture boundaries. as shown in Figure 5.26(c).

In order to reduce the uncertainties along boundaries, we need to use smaller windows for feature extraction. However, features extracted should capture the spatial relationships among pixels. To overcome this problem, we propose the following measure to refine the result obtained using spectral histograms. As for segmentation, we first build a probability model for given m pixels from a texture region. To capture the spatial relationship, we choose for each texture region a window as a template. In our case, the template is the same window from which the region feature $\mathcal{F}$ is extracted. For the selected m pixels, we define the distance between those pixels and a texture region as the minimum mean square distance between those pixels and the template. Based on the result from spectral histograms, we build a probability model for each texture region with respect to the proposed distance measure. Intuitively, if the m pixels belong to a texture region, it should match the spatial relationship among pixels when the m pixels are aligned with the texture structure.

After the probability model is derived, we use the local updating equation given in (5.3) by replacing $W^{(s)}_{(x,y)}$ by the m pixels in a texture region along its boundary based on the current segmentation result and $\chi^2(H_{W^{(s)}_{(x,y)}}, H_i)$ by the new distance measure. Figure 5.26(d) shows the refined segmentation result with $m = 11$ pixels. Visually the segmentation result is improved significantly and the segmentation error is reduced to 0.95%. Figure 5.26(e) shows the wrongly segmented pixels.

Figure 5.27(c) shows the result for an intensity image. It is clear that the boundary between two regions is improved significantly, especially at the top and bottom borders of the image.

Figure 5.28(c) shows the refined segmentation result for the image which was used in classification shown in Figure 4.22. Compared to the classification result, the errors along the texture boundaries are reduced significantly.

Figure 5.29(c) shows another example. Here the resulting texture boundary between the top and left regions is jagged due to the local ambiguities of a small number of pixels. This is partially because that the boundary smoothness is approximated

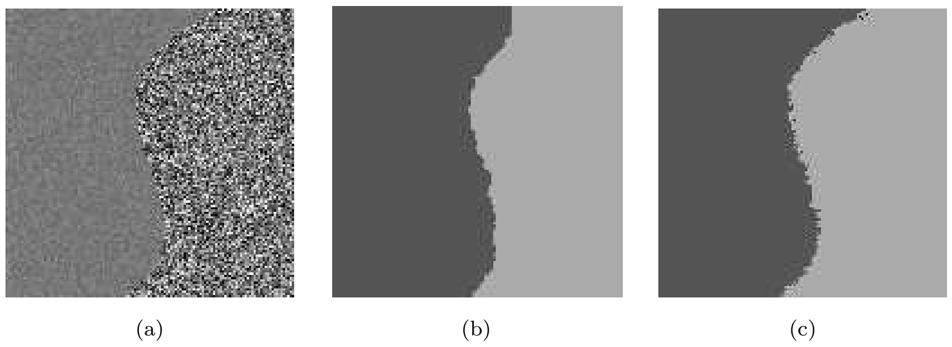


Figure 5.27: (a) A synthetic image with size 128 × 128, as shown in Figure 5.7(a). (b) The segmentation result using spectral histograms as shown in Figure 5.7(c). (c) Refined segmentation result.

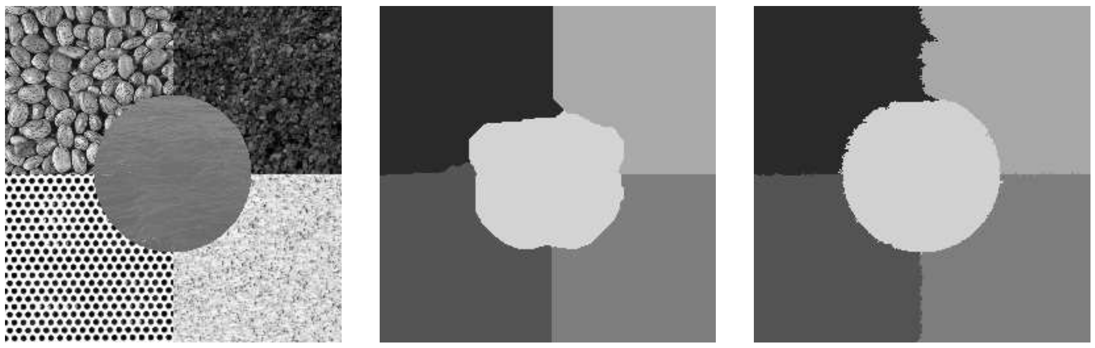


Figure 5.28: (a) A texture image with size 256 × 256. (b) The segmentation result using spectral histograms. (c) Refined segmentation result.

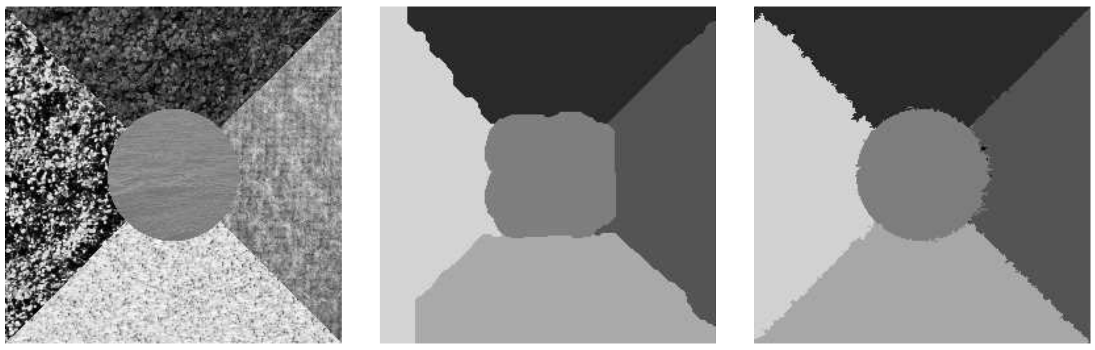

(a) (b) (c)

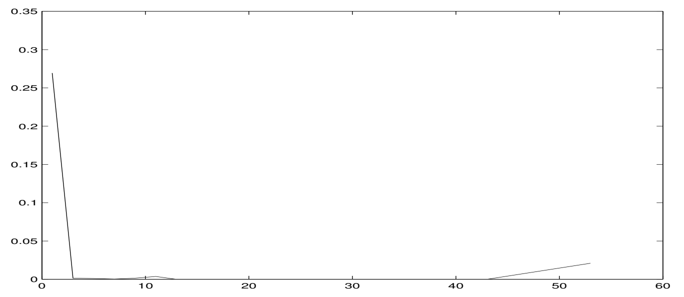
Figure 5.29: (a) A texture image with size 256 × 256. (b) The segmentation result using spectral histograms. (c) Refined segmentation result.

using a local term given in (5.3). By using a more global boundary smoothness term as in [151], the result could be improved, which will be investigated in the future.

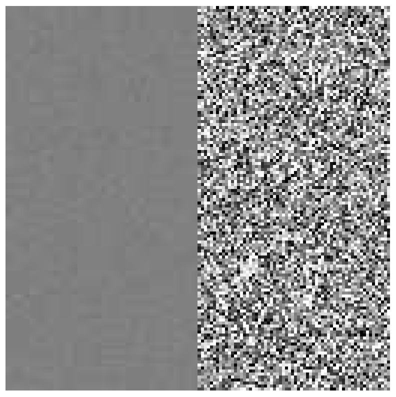
5.7 Discussions

There are several improvements that can be done using spectral histograms. As is evident from Figure 5.30, one can estimate automatically the minimum scale for a given texture region. It shows clearly in Figure 5.30 (b) and (c) that the left region needs a smaller scale than the right one. The minimum scale selection has been studied by Elder and Zucker [29] under the assumption of Gaussian noise. I did some work along this line and it is not included in this dissertation.

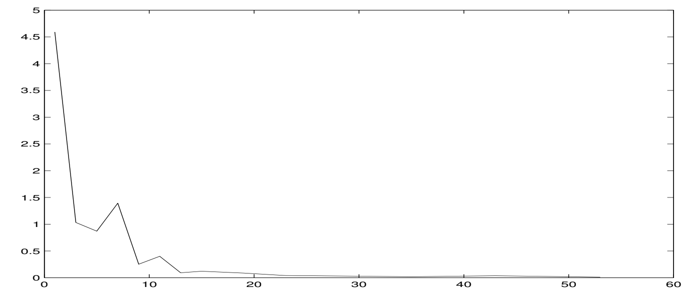
As we see from the examples, most texture images are not absolutely homogeneous, but can be better described by responses of some filters, as shown in Figures 5.16 and 5.19. A similar question is how to effectively discriminate two textures by selecting filters. A straightforward extension is to use a weighted average of filter responses,



(b)



(a)



(c)

Figure 5.30: Distance between scales for different regions. (a) Input image. (b) The distance between different integration scales for the left region at pixel $(32, 64)$. (c) The distance between different integration scales for the right region at pixel $(96, 64)$.

as in the FRAME model [149] [150]

$$\chi^2(H_{\mathbf{I}_1}, H_{\mathbf{I}_2}) = \sum_{\alpha=1}^{K} \sum_{z} \lambda_z^{(\alpha)} \frac{(H_{\mathbf{I}_1}^{(\alpha)}(z) - H_{\mathbf{I}_2}^{(\alpha)}(z))^2}{H_{\mathbf{I}_1}^{(\alpha)}(z) + H_{\mathbf{I}_2}^{(\alpha)}(z)}. \tag{5.8}$$

There are two issues here. If we have training samples available, we can use parameter estimation methods [149] [150]. We can also use a neural network to learn the weights. For image segmentation, the main problem is how we can obtain training samples along the segmentation procedure. One way is to use the initial segmentation result. The other way is to alternate between two phases. In the training phase, we can re-estimate the parameters using the current segmentation result and in the segmentation phase we can use the current parameters for segmentation.

Some of textures cannot be captured well using spectral histograms when the filter responses are not homogeneous. Rather, the spatial relationships among texture elements are more prominent. For the example, the zebra stripes shown in Figure 5.31(a) cannot be captured well using spectral histograms, especially the boundaries are not localized well. Given that the surrounding area is close to the white part of zebra stripes and zebra stripes are not homogeneous, the result is what one would expect using a homogeneous model. No matter what homogeneity measure is used, it would not work well in cases like the zebra image because the zebra stripes are simply not homogeneous. To overcome this problem, one needs to define a model which can characterize the structural relationships in a generic way. Within the spatial/frequency representational framework, short-range order coupling may provide a solution. In other words, the filter responses capture the local shape information and short-range order coupling overcomes the inhomogeneity in the region. This may lead to a new computational model and needs to be further investigated and studied.

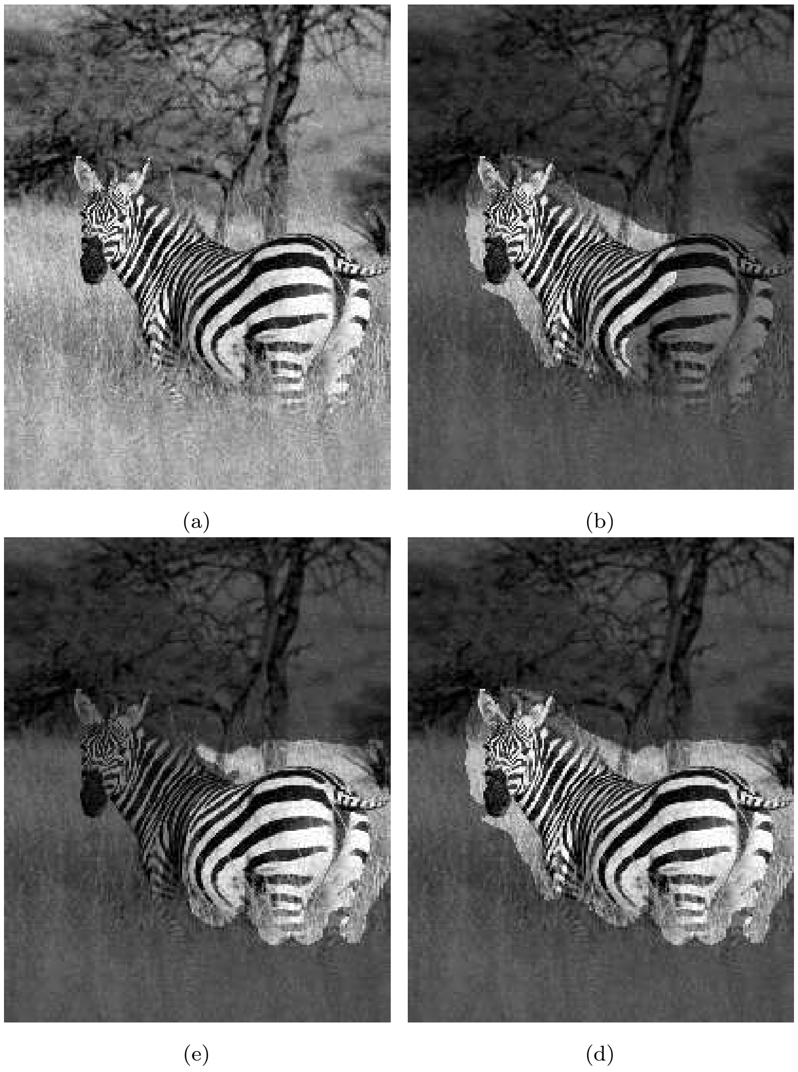


Figure 5.31: A natural image with a zebra. $\lambda_\Gamma = 0.2$, and $\lambda_B = 5.5$. (a) The input image. (b) The segmentation result with one feature computed at $(205, 279)$. (c) The segmentation result with one feature computed at $(308, 298)$. (d) The combined result from (b) and (c).

5.8 Conclusions

In this chapter, we formulate an energy functional for image segmentation by making features and homogeneity measures explicit for segmented regions. We have developed a segmentation algorithm using spectral histograms as a generic feature for natural images and χ^2-statistic as a similarity measure. We investigate the performance of the algorithms under different assumptions. Satisfactory results have been obtained using intensity, texture, and natural images. Future work along this line is discussed.