Chapter 7

Document segmentation and image queries

In this chapter, two application examples of the QMF filter bank features of Section 3.2 will be presented. The motivation is to show the applicability of the texture features of Section 3.2, not to present complete systems with all pre- and post-processing for the applications. In Section 7.1, an approach to document segmentation is presented and in Section 7.2 an application to image content search is presented. A common denominator for the two approaches presented here is that the search is for known textures in images containing other textures not known a priori. Hence, several commonly used classifier design approaches [35] are not applicable here. Furthermore, for the same reasons, the filter optimization approaches presented in Chapters 4 and 5 are not applicable.

7.1 Document segmentation

An enormous amount of printed material is available on paper. The digital computer and computer networks have made it possible to search for and retrieve electronically stored documents in seconds, no matter where in the world they are stored. This is not the reality for documents stored as paper copies and there is a considerable interest in digitizing paper documents.

The straightforward approach to document digitization is to scan the whole document and store it as a bit-map representation. This approach was taken for the CD-ROM accompanying the proceedings of the IEEE 1994 International Conference on Acoustics, Speech & Signal Processing (ICASSP'94). Unfortunately, this requires a huge storage capacity (even with modern image compression techniques) and does not enable search for textual content in the documents.

A more efficient way of representing documents is to separate the text and the graphics and store the text in a character representation form. Therefore, several approaches to text-graphics segmentation have been proposed. The most popular approaches may broadly be classified as either *top-down* or *bottom-up* approaches [122]. One of the most common top-down techniques is the *run length smoothing* method [17, 135] (also known as the *constrained run length* method). With the run length smoothing algorithm, the document image is processed along vertical and horizontal runs (scan-lines). If the number of white pixels between two black pixels is less than a threshold, the white pixels are converted to black. Thereby, white gaps within and between words are converted to black, while white gaps between columns remain white. A similar discussion applies for lines. Hence, the run length smoothing method is sensitive to font-size, character spacing, line and column spacing and to some extent to the orientation of the document.

Another common top-down technique is the *recursive projection profiles* method [59, 76, 136] (also known as the *recursive X-Y cuts* method). The basic idea underlying the projection profiles method is that the numbers of black pixels in the columns and rows of the image are projected onto the horizontal and vertical axes, respectively. Hence column and line spacings generate detectable valleys in the projection profiles and it is possible to extract the block structure of the document. The recursive projection profiles method is restricted to rectangular blocks of known orientation.

The bottom-up methods are typically variants of the *connected components* method [33]. The connected components method basically extracts connected components, like a character or a connected part of a drawing. It then performs some kind of analysis on each connected component and merges connected components that are close to each other. The drawbacks with the connected components method are that it is character size dependent, sensitive to inter-line and inter-character spacings, and sensitive to the digitizing resolution.

In addition to the already mentioned restrictions with these methods, they typically operate on thresholded images. If the image capturing conditions are difficult, appropriate thresholding may be difficult.

An interesting ability of humans is the ability to tell what is and is not text in a document viewed from a distance too far away for letter discrimination. This suggests that textural clues are important for humans in separating text regions from non-text regions. In this section, an approach to segmentation of text and graphics in scanned documents is proposed. The approach is based on the assumption that the text in a document may be viewed as one texture, while the graphics is represented by different textures. Then, the documents are segmented with a texture segmentation scheme using the sub-sampled feature extraction scheme of Section 3.2. Both unsupervised and supervised segmentation schemes are proposed. While most traditional text-graphics segmentation schemes require a priori knowledge of the input, the unsupervised scheme is independent of document layout, typeface, font size, scanning resolution, etc. However, the unsupervised segmentation scheme shows a somewhat limited performance for difficult documents. By sacrificing some of the flexibility, it is shown that the segmentation is improved by using a supervised classifier. In Subsection 7.1.1, the unsupervised segmentation scheme is proposed and in Subsection 7.1.2, the supervised segmentation scheme is proposed. In both cases experimental results illustrating the performances of the suggested approaches are presented.

An approach similar to the unsupervised approach [49], uses the Gabor filter and a local energy function as the feature extractor, see Section 2.2.3. One major drawback with the Gabor approach is its computational complexity. The approach presented in [49] uses a subset of eight filters from the total set of 20 suggested in [50]. However, even when this restricted subset is used, a total data amount of eight times the input is used by the classifier. The approach presented here has much lower complexity, without any observed sacrifices in segmentation quality.

In another similar approach, supervised segmentation using eigenfilters derived by the use of neural networks are examined [84]. By this approach, good results are obtained, realizing some savings over the previous approach. However, the extension to unsupervised problems is not straight forward.

7.1.1 Unsupervised segmentation

From the results of Chapter 6, it is evident that the critically sampled IIR QMF filter banks of Section 3.2 give quite good results. Furthermore, the computational complexities are lower than for most other filter bank approaches, see Section 3.2.3. Consequently, a critically sampled IIR QMF filter bank, using the filter structure of Figure 2.6(d), will be used as feature extractor in these experiments. Furthermore, the sigmoidal nonlinearity and Gaussian smoothing filter are used in these experiments. Using the critically sampled filter bank implies calculation of only a few feature samples. The total size of the feature set is therefore identical to the size of the input image. Critical sampling introduces some edge inaccuracies, see Section 3.2.4, but the computational and memory-requirement savings are tremendous. The approach by Jain and Bhattacharjee [49], for example, implies a feature set eight times the size of the input image. An A4 page scanned at 300 dots per inch, for example, consists of more than eight million pixels, thus the size of the feature set is important.

If the features give rise to correct unsupervised segmentation, we will not have to make any assumptions about font size, scanning resolution, column layout, orientation, etc., as we will see in this section. The clustering algorithm chosen is the k-means algorithm [116] with a predetermined number of clusters.

It has been chosen to use parts of pages scanned from the March 2, 1994, issue of the newspaper “Stavanger Aftenblad” (SA), Stavanger, Norway as test images. For comparison, some images used in [49, 128] have also been included. Figures 7.1(a-c) shows a part of page 8 in SA and the quite successful segmentation of the document. Except for some problems with the heading and spurious dots in the text regions, the segmentation is as desired. Note that no a priori information on font size, scanning resolution, etc. has been assumed. This example shows the applicability of the method. To further emphasize the flexibility of the approach, Figures 7.1(d) and 7.1(e) show the segmentation results from the same page rotated by 55° and at half the resolution, respectively. These figures show one important result: the segmentation quality is practically invariant with respect to scaling and rotation.

How does the method compare with other methods? Figure 7.2 show the same images as used in [49, 128]. The segmentation results from these images are practically the same. On these images, the sub-sampling did not yield any particular degradation in the segmentation result. These results, in combination with the results of Chapter 6, indicate that there is no practical need for using the significantly more complex Gabor filter bank for this application.

Unfortunately, not all documents are as easy to segment as the ones presented thus far. The text segmentation method is based on the assumption that the text regions may

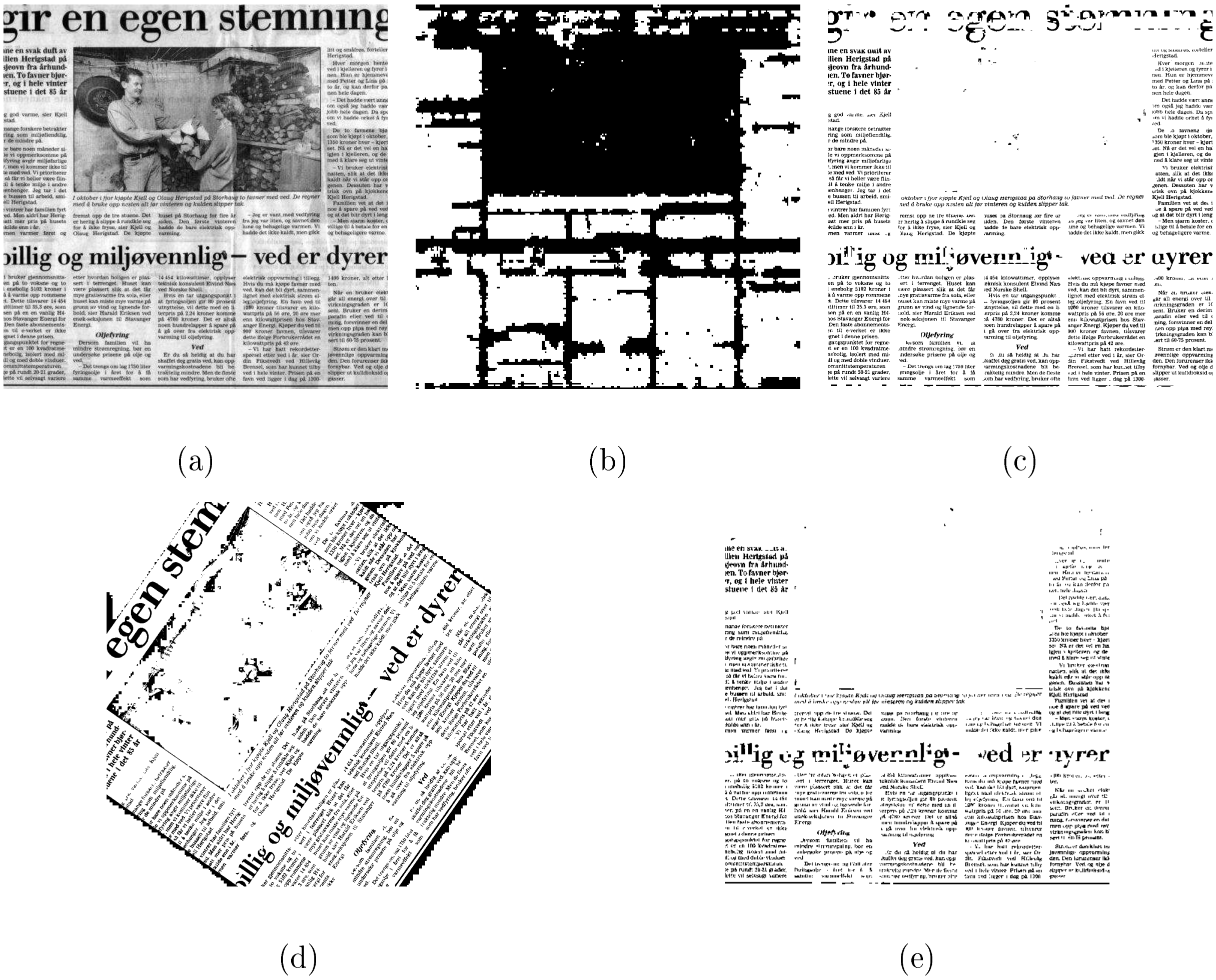


Figure 7.1: (a) The 1024×1024 pixel scanned image of parts of page 8 from SA, (b) the segments from unsupervised segmentation, and (c) with regions classified as picture excluded. (d) Unsupervised segmentation of the same page rotated by 55°, and (e) scanned at half the resolution, 512×512 pixels.

be considered as constituting a significantly different texture than the graphics regions. Therefore it is natural that the method performs worse when the graphics contain textures similar to the texture in the text region. The texture of parts of the graphics region in the document in Figure 7.3(a) is quite similar to the texture of text. As a result, we see that the segmentation, Figure 7.3(b), has included a considerable part of the graphics into the text cluster. No unsupervised experimental results on similarly difficult data have been found in other publications.

It is important to be aware that all experiments in this section were done with no assumptions on the input image and with no post-processing. Several kinds of post-processing techniques are possible and should improve the results, especially if we have information about the format of the input, e.g. that the text is organized in rectangular columns.

(a) (b) (c) (d)

Figure 7.2: (a), (c): Test images from [49] and (b), (d) the unsupervised segmentations of the images.

(a) (b)

Figure 7.3: (a) Page 2 from SA and (b) the unsupervised segmentation of the page.

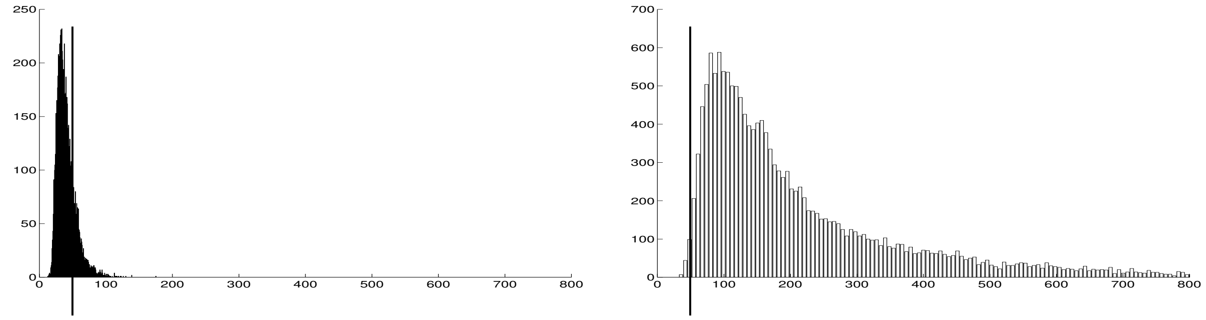
7.1.2 Supervised segmentation

Motivated by the problem with the unsupervised segmentation approach in the previous section, a somewhat different approach is proposed, utilizing supervised segmentation. A supervised approach will of course mean a limitation when compared to the flexibility of the unsupervised approach. However, in many applications it may be assumed that the font-size, scanning resolution, and page orientation are fixed. This will probably be the case if for instance a newspaper decides to digitize back issues.

A supervised approach was also presented by Jain and Bhattacharjee [49]. The specific application where the supervised approach was applied was segmentation of bar-code from text. Their supervised approach was based on training a classifier using exemplar features from bar-code and text. However, in a general text segmentation application it is not possible to assume that the non-text class is this well-defined. It will therefore not be possible to train a classifier on the non-text regions.

Training and classification

Although it is not possible to assume that the non-text class has fixed properties, it may be assumed that the text class has fixed and well-defined properties. To classify, we need



(a) (b)

Figure 7.4: Histograms of distance from codebook for feature vectors from (a) text and (b) image for page 8 of SA.

to calculate some kind of measure of similarity with the prototype text class. The training feature vectors were represented by a set of cluster center vectors obtained by clustering with the k-means clustering algorithm [116].

The means and variances of each cluster are computed from the features. The distance from the query pattern is computed by the squared weighted Euclidean distance to the closest cluster center. This measure is given by

$$d = (\mathbf{v} - \mathbf{m})^T \operatorname{diag}(\mathbf{C})^{-1} (\mathbf{v} - \mathbf{m}), \tag{7.1}$$

where $\mathbf{v}$ is the investigated feature vector, $\mathbf{m}$ is the cluster mean, and $\mathbf{C}$ is the cluster covariance matrix. The idea of clustering and distance computation is similar to the basic ideas of the Kohonen Self Organizing Map and the Learning Vector Quantization [58].

A feature vector, corresponding to one pixel or a small region of the image, is classified as belonging to the target pattern if the distance is below some threshold. An empirical distance threshold of 50 from the mean was selected. This choice was appropriate when the features were scaled to approximately the range $0 \cdots 255$. It is important to keep in mind that techniques like Bayes decision rule [29] may not be employed to determine an optimal threshold for classification as text because we do not know the non-text properties.

Experimental results

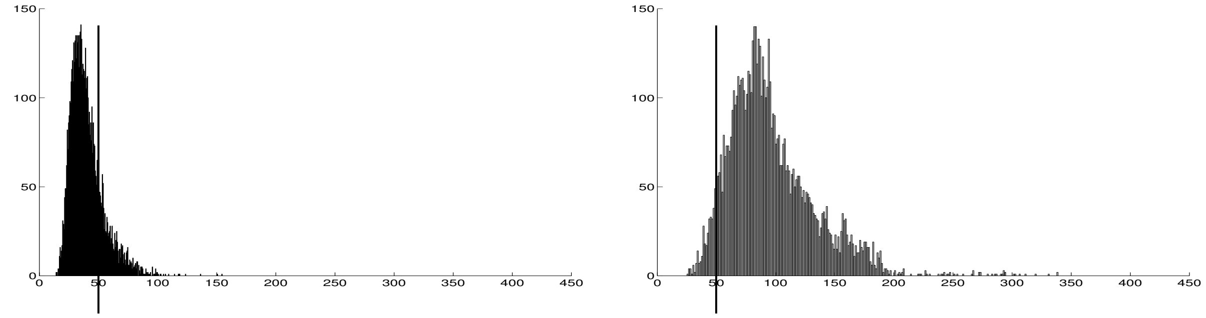
Two documents have been chosen for the reported supervised experiments. First, the results with page 8 from SA are shown, a page that was successfully segmented by the unsupervised approach. Then it is illustrated how the method performs for a more difficult page, page 2 of SA.

Page 8 of SA, Figure 7.1 (a), contains four different kinds of text; normal text, boldface text in the article introduction, and headings of two sizes. Attempts to classify the document according to each of these text types have been made, using codebooks for each of the four kinds of text. If a feature vector is close to several of the text classes, it is classified as the class with the closest prototype.

How good is the separation between text and non-text regions? Figures 7.4(a) and 7.4(b) show the distance histograms between the text codebook and the text and graphics

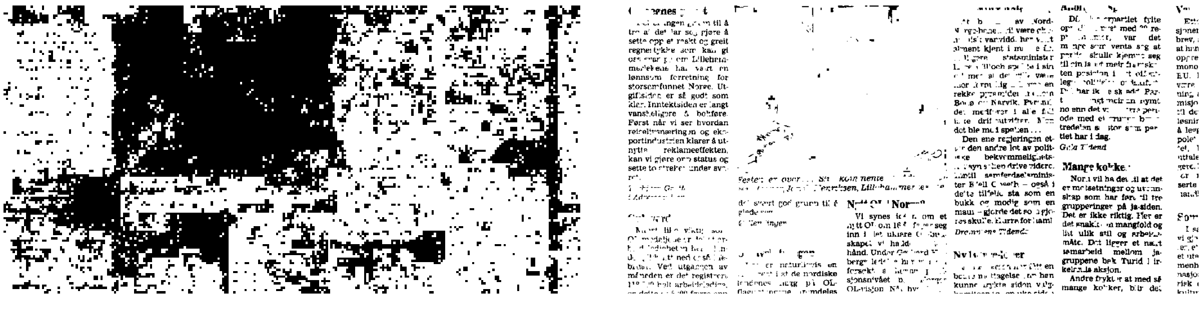
(a) (b)

Figure 7.5: Supervised segmentation of page 8 from SA, (a) shown as regions (black is non-text regions, dark gray is normal text, medium gray is boldface text and light gray and white are large and small headings, respectively) and (b) the input image masked through the segmentation result (non-text regions masked out).



(a) (b)

Figure 7.6: Histograms of distance from codebook for feature vectors from (a) text and (b) image for page 2 of SA.



(a) (b)

Figure 7.7: Supervised segmentation of page 2 from SA, (a) shown as regions and (b) the input image masked through the segmentation result (non-text regions masked out).

features, respectively. We see that the histograms are well separated. From the histograms we understand why the threshold of 50 is appropriate.

The segmentation results from page 8 are given in Figure 7.5. As we see, the segmentation is mainly correct, but with some small regions misclassified. These misclassified regions should be possible to detect and remove with various post-processing techniques, which are not addressed here. The promising results from the supervised approach were expected, since the results with the unsupervised approach were good.

The main motivation for developing a supervised approach was improvement of the results for the more difficult documents, like page 2 of SA. The feature distance histograms for this page are given in Figure 7.6. As we could expect, we see that the histograms have more overlap, but a text classification distance threshold of 50 still seems reasonable. The segmentation results are given in Figure 7.7. We see that only small parts of the graphics region are classified as text, while large parts were classified as text in the unsupervised approach. Hence, the results are much better with the supervised approach than with the unsupervised approach.

7.2 Query by image content

One of the major contributions of computers to everyday life is the ability to conduct fast and reliable text pattern searches. Text pattern searches allow us to find names in databases, books based on their keywords, etc. Searching non-textual data such as images based on contents (e.g., color and texture), on the other hand, is far more complex. However, the increased utility of computer vision and pattern recognition algorithms, along with the availability of more powerful computers at low cost, make such an application increasingly feasible. In an image database system, features may be computed in advance and the difficult issues are feature storage and comparison. In searches in images not stored in an integrated database, such as searches over a network, pre-computed features will typically not be available and the feature extraction complexity is of paramount importance.

Query by image content is a relatively new research area and an increasing number of approaches to querying by color and texture are reported. Picard et al. [88, 90, 91] focus their work on user interaction and adaptive feature selection. Claiming that no single feature extraction scheme is suitable for all image types, they apply a "society of models". Some works on retrieving a set of most similar sub-images based on matched filters [89], Markov random field parameters [37], and Gabor features [70] have been presented. Taking a more global perspective, Pentland et al. [86] and Flickner et al. [34] present complete systems, integrating many aspects of query systems. Except for the work by Liu and Picard [64] on modeling the human visual system, not much work on feature tuning for image queries has been reported.

In this section, a feature extraction scheme for color and texture content search is suggested and tested on a large number of real world images. In Subsection 7.2.1 the feature extraction and classification scheme is proposed and in Subsection 7.2.2 experiments are given.

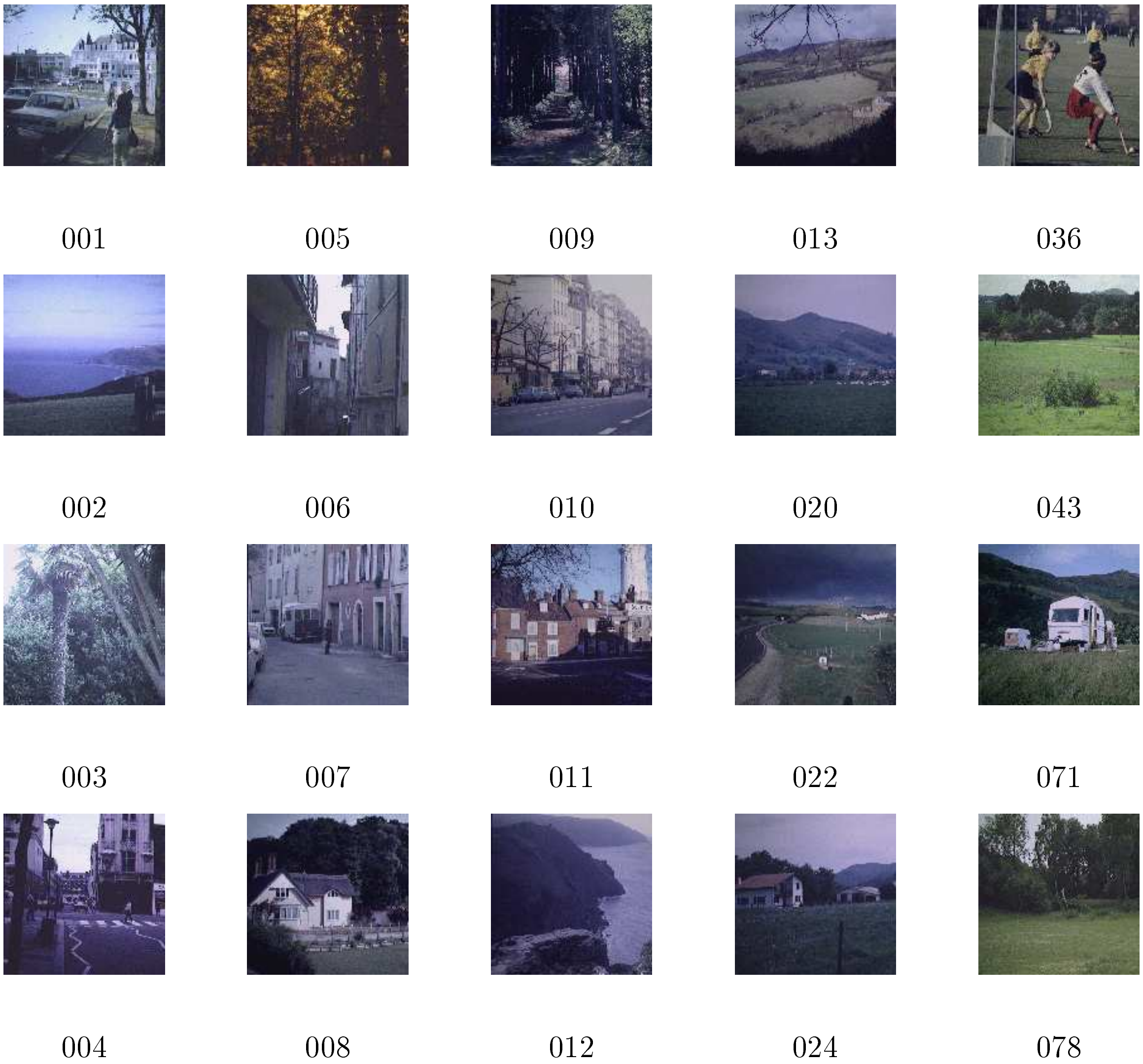


Figure 7.8: Subset of the test images used for image content search/labeling, downloaded from [73].

7.2.1 Methodology

In this work, a method for color and texture search is developed. The texture features will be extracted by the computationally attractive critically sampled IIR QMF filter bank scheme presented in Section 3.2. Color is very important for many query patterns. In this case, the color is represented in the feature space by the two chrominance components of the NTSC color representation [48]. The texture features are extracted from the luminance component.

How are the color and texture features combined in the classification? The straightforward approach is to combine them into one vector by appending the individual feature vectors. However, for filter banks with many sub-bands, this would imply high dimensionality in the *texture* feature sub-space while the *color* sub-space is limited to dimensionality two. Hence, large distance variations in color would only yield a low relative variation in the overall distance measure. Another scheme, yielding identical weight on color and texture, would be to classify using the color and texture feature vectors separately, yielding one color-based class map and one texture-based class map. An overall class map could then be formed by logical "and" of these two class maps. Both of these approaches were tested with relatively poor results.

Yet another approach is to append the color and texture features into one vector and balance the contributions from the color and the texture. The balancing may be done in the distance measure. A modification of the codebook design approach of Section 7.1.2, using a balanced squared weighted Euclidean distance measure, is applied. The distance measure is

$$\begin{aligned} d = & \frac{1}{2^2} \left(\mathbf{v_C} - \mathbf{m_C} \right)^T \operatorname{diag} \left(\mathbf{C_C} \right)^{-1} \left(\mathbf{v_C} - \mathbf{m_C} \right) \\ & + \frac{1}{N^2} \left(\mathbf{v_T} - \mathbf{m_T} \right)^T \operatorname{diag} \left(\mathbf{C_T} \right)^{-1} \left(\mathbf{v_T} - \mathbf{m_T} \right), \end{aligned} \tag{7.2}$$

where the C and T subscripts denote the sub-vectors and sub-matrices corresponding to the color and the texture features, respectively and N is the dimensionality of the texture feature vector. By this normalization, it may easily be shown that linear scaling of the distance in the color and texture domains affect the total distance measure identically.

In an image database system, query times of a few seconds, even if the database consists of thousands of images, are desirable and have been reported [90]. However, although the complexity of the proposed feature extraction scheme is very low, see Section 3.2.3, even loading all those images will violate the complexity constraints. A possible solution to this problem is to store a small amount of feature information for each image and apply a fast similarity match [9, 90, 141]. Another alternative is to classify the regions of the images off-line and store class labels. This will be illustrated by experiments in the next subsection. Hence, if a user requests images with hair, the class labels are examined and images with hair are returned. The problem of integration with higher-level information, e.g. context information, will not be treated here.

7.2.2 Experiments

An example of indexing regions with grass using the 98 vacation images in the MIT VisionTexture database [73] will be given. Some of the MIT images are shown in Figure 7.8. One of the most challenging tasks in the experiments was the selection of good training image regions. Eventually the images 008, 013, 020, 022, 024, 036, 043, 071, 078, and 095 were used as training images. When testing on any of the training images, the leave-one-out scheme was used.

How do we determine which sub-band decomposition is most appropriate for the problem? The selection of sub-band decomposition is clearly a tradeoff between the "curse of dimensionality" [29] and the need for adequate feature separation. Furthermore, classical feature analysis and selection schemes, like the scheme of Whitney [142], are not adequate for this problem due to the lack of a definition of the "everything else" class.

In the experiments, several sub-band decompositions were tested along with the dyadic Gabor features [50, 70], Section 2.2.3. The distance distributions along with the corre-

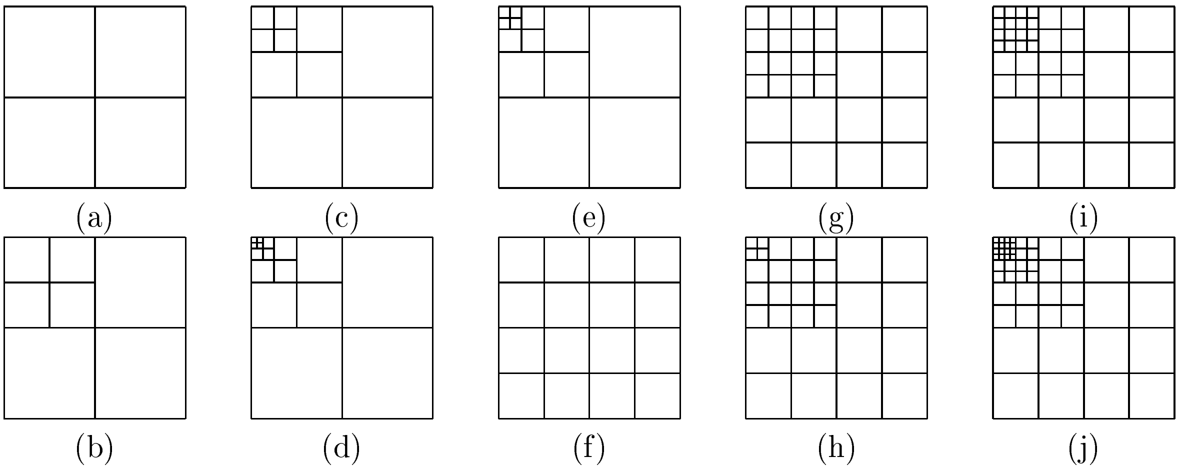


Figure 7.9: Sub-band decompositions of the filter banks used in the image content search experiments.

sponding overlaps were computed and are shown in Figure 7.10. From the overlap percentages, we can see that the filter bank structure of Figure 7.9(h) yields the most powerful features. We also see that it compares favorably with the computationally *much* more complex Gabor [50] features, Figure 7.10(l).

It is particularly interesting to see that some of the texture features hardly add any discrimination at all, and some even makes the results worse! From Figure 7.10(k) we see that the overlap without any texture measure applied at all is 20.2%. Many texture measures give results that are only slightly better. The filter structures of Figures 7.9(e) and 7.9(j) even makes the results worse, 29.5% and 33.8% overlaps.

In Figure 7.11 some classification results with this feature extractor are shown. Different intensities indicate different thresholds for the distance measure. As can be seen, if the results are studied in detail, the system detects regions with grass well. Some false detections of walls and leaves can be seen, but from a pure texture and color point of view, the falsely detected areas are very similar to certain types of grass. The experiments were restricted to grass as an illustrational example.

The results also tell us that as the dimensionality of the texture features becomes too large (e.g. Figures 7.9(j) and 7.10(j)), the performance degrades. This indicates that it becomes increasingly difficult to design a good codebook with the increased number of feature variables – the "curse of dimensionality" [29].

7.3 Summary

In this chapter, applications to content based image search and document segmentation of the critically sampled QMF filter bank scheme of Section 3.2 have been presented. The first application was an approach to document segmentation using an unsupervised classifier. This approach uses no a priori knowledge of the input, as opposed to most traditional text-graphics segmentation schemes. Still, satisfying results are obtained in many cases. The second approach was a supervised classification approach, yielding near

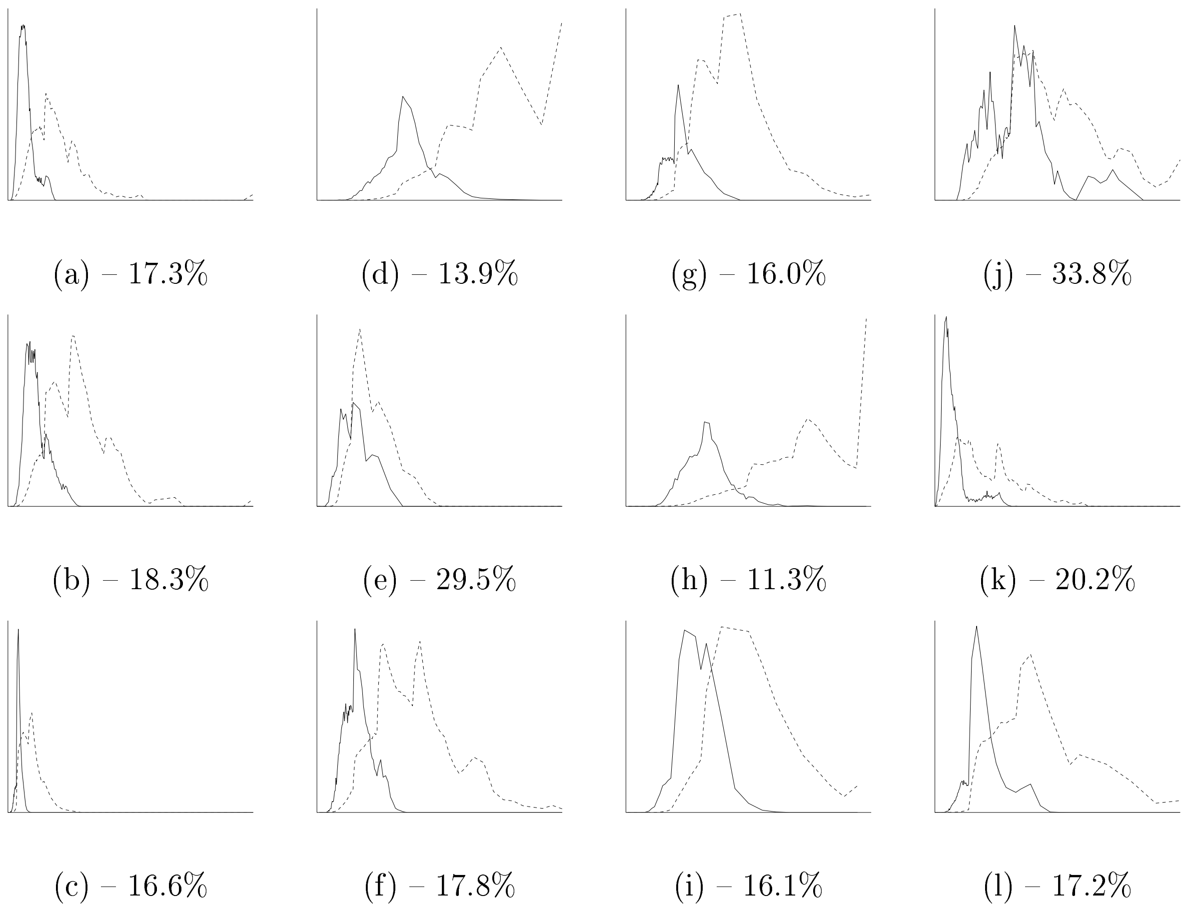


Figure 7.10: Histograms of feature distance to closest grass codebook vector for grass (solid) and non-grass (dashed) for different feature sets. The horizontal axes correspond to distance measure and the vertical to relative densities. Minimal overlap is optimal. Histograms (a) to (j) correspond to the filter structures in Figure 7.9, (k) corresponds to no texture filter bank at all (using only the luminance in addition to the color), and (l) to the dyadic Gabor filter bank [70]. The percentages of histogram overlaps are indicated.

perfect segmentation even for difficult documents where the unsupervised approach had poor segmentation.

Furthermore, a feature extraction and classification scheme for color-texture pattern search in digital image databases has been proposed. The color was represented by the two chrominance components in the NTSC scheme, while texture was represented by QMF filter bank features. The color and texture features were combined and their contributions balanced using a modified weighted Euclidean distance measure. The scheme was tested on the problem of labeling or finding regions in images with grass.

The focus of the applications has been on proving the applicability of the critically sampled QMF filter bank.[1] The issues of pre- and post-processing and integration of other types of information have been treated more superficial. This is not to indicate that these problems are less important, but rather that they are beyond the scope of this work.

[1]Both this and the previous application, document segmentation, are problems where only a subset of the possible texture classes are known a priori. Hence, the optimization approaches of Chapters 4 and 5 are not applicable.

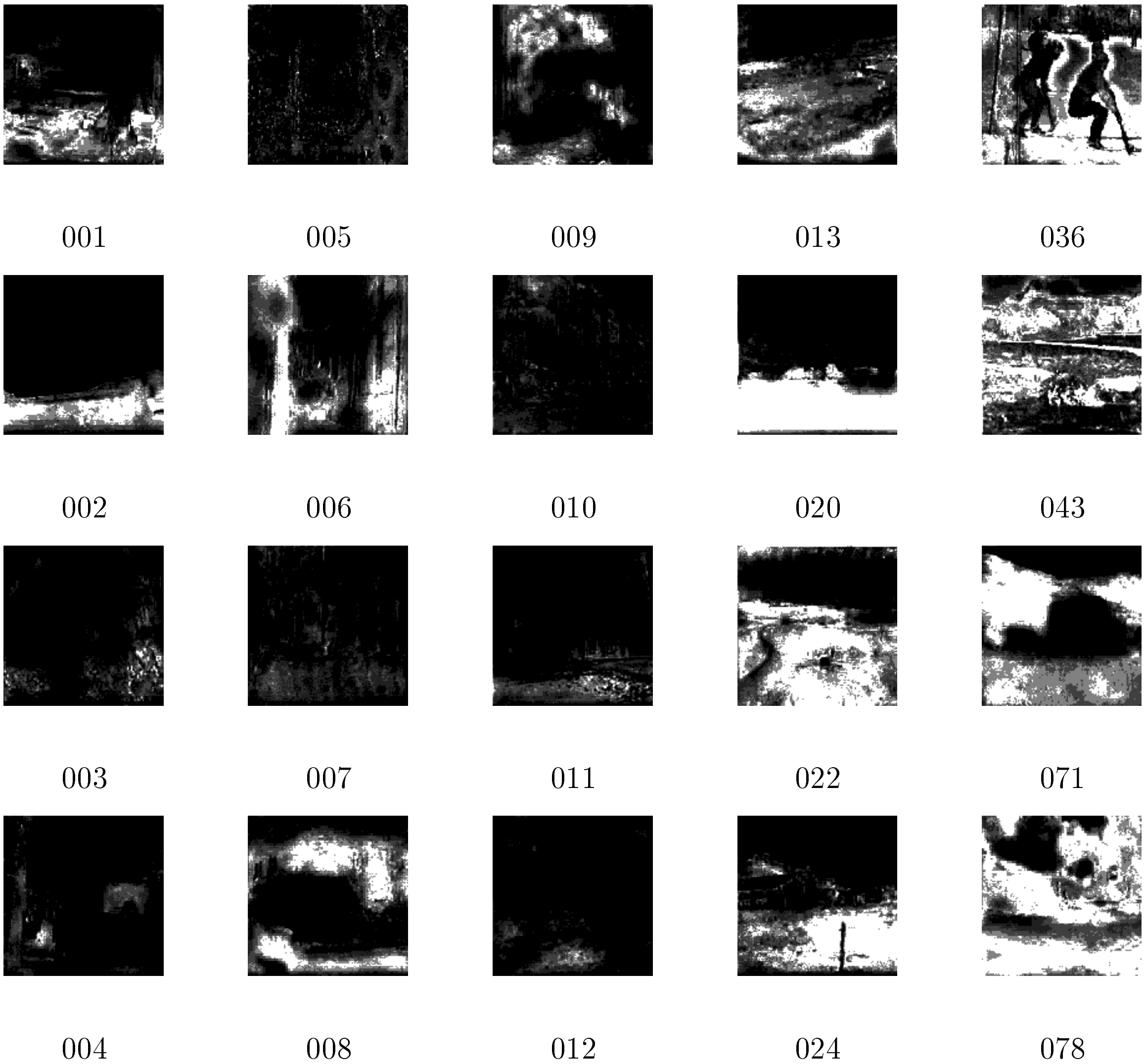


Figure 7.11: Segmentation results for the images of Figure 7.8.