A HIERARCHICAL DOCUMENT DESCRIPTION AND COMPARISON METHOD

A Thesis

Submitted to the Faculty

of

Purdue University

by

Jacob Lee Harris

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Electrical and Computer Engineering

August 2003

ACKNOWLEDGMENTS

Burak Bitlis, Xiaojun Feng, and I worked together on this entire project. The research presented in this thesis was done either by me or by the group of us, with the following exceptions. Burak and Xiaojun researched and implemented all the latent semantic information material. Burak was very instrumental in developing and coding the merge distance function. Also, Xiaojun figured out how to convert pdf files to raster images and developed the tree matching strategy described in Section 3.2. This thesis is the summary of all our efforts, and the research bears their mark just as much as it does mine.

Professors Ilya Pollak, Charles Bouman, Mary Harper, and Jan Allebach met with us weekly to generate ideas and keep us moving forward. We are indebted to their leadership.

This work was supported in part by a National Science Foundation CAREER award CCR-0093105 and by a fellowship from the School of Electrical and Computer Engineering, Purdue University.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Harris, Jacob Lee. M.S.E.C.E., Purdue University, August, 2003. A Hierarchical Document Description and Comparison Method. Major Professor: Ilya Pollak.

We propose a method to describe and compare the content and hierarchical structure of a document given only an image of the document. A tree structure is used to capture the hierarchical structure of the document. Each node of this tree corresponds to a region of the document and contains a vector of calculated features that describe the region. Two documents are then compared using a tree matching strategy to compare their hierarchical descriptions. Because this method depends only on an image of the document, it can be used to describe any document that can be scanned or printed. Finally, we give an example to demonstrate the effectiveness of the method we have introduced.

1. INTRODUCTION

Determining the similarity of document images is an important first step for several document retrieval tasks, such as document classification, information extraction, and retrieval based on visual similarity [2]. These document retrieval tasks are important when dealing with large databases of document images. For example, document retrieval is necessary when a database user wants to find documents in the database which are similar to a particular example document. Document classification could be useful in deciding where to store a document image being added to the database, or in choosing the most appropriate document model, if the models are different for various classes. All of these tasks require some notion of document similarity. When assessing what makes document images similar, we consider both of the following to be important: the *content* of the document (words, pictures, etc.) and the *layout* of the document (how the content of the document is organized).

The document comparison problem has been studied for some time. Some methods depend only on the text content of the document, as in [3–5]. Such methods have proven to be very effective for comparing documents which are all or mostly text, but they do not take into account either the pictures and graphics which might be present in the document or the physical layout of the document.

Srihari, Zhang, and Rao [6] give an example of a method that uses both text and picture information to query a document database, but this method still does not make use of the layout of the document. The method of Hu *et al.* [2] classifies documents based on the layout of the text regions, but does not take into account the content of the text regions or picture regions. All of these methods either do not take the document layout into account or do not describe the layout in a hierarchical manner. An example of describing the layout of a document using a hierarchical structure

appears in [7], which uses a stochastic regular grammar to produce a segmentation of the document.

We introduce a method to describe both the hierarchical layout structure and the content of a single-page document image. We begin by partitioning the document into meaningful regions and calculating several features of these regions. We then define a distance between regions, and as in chapter 3 of [8], we recursively merge the two nearest regions until all regions have been merged. Thus we create a binary partition tree whose nodes correspond to regions in the document and contain features to describe these regions. The leaves of the tree correspond to the various small regions of a document such as paragraphs, images, captions, etc., while internal nodes of the tree correspond to a logical grouping of all the leaf regions descending from it. For example, an internal node could correspond to a column of text, which is the union of several paragraphs. The document tree that we form is similar to the binary space partition trees used by Qiu and Sudirman [9] in that each node contains features of the document region to which it corresponds, and that the tree is a binary partitioning of the document. However, the specific features and rules for determining the boundaries of regions are different.

We then compare two document images by comparing their document trees. We use a simple tree matching strategy to do this comparison, but ongoing work will include investigation and evaluation of standard tree and graph matching methods.

The remainder of this thesis is organized as follows. Chapter 2 describes the formation of the hierarchical document description, which we call the document tree. Chapter 3 defines the distance between document trees. Chapter 4 contains the results of an example implementation of our strategy, and Chapter 5 concludes the thesis.

2. FORMING THE DOCUMENT TREE

The document tree describes both the content and the hierarchical structure of the document. It is formed by first segmenting the document into meaningful regions, then calculating features of these regions, and finally defining a distance between regions based on these features. The distance is then used to merge the regions recursively into a binary tree describing the document. Each of these functions is described more fully below.

2.1 Summary of Approach

We describe the hierarchical structure of the document with a binary tree, called the document tree, which is created by recursively merging the regions of the document. Figure 2.1 illustrates this concept. Given a document and a partition of the document into meaningful regions, we recursively merge similar regions. The resulting tree describes the hierarchy of structure in the document. In the example of Figure 2.1, the root node corresponds to the entire document, which is split into background and content. The content is split into text and image regions, which are then split into their smaller components. So we see that we create the tree in a bottom up fashion, but it is easier to interpret the meaning of the tree in a top down fashion.

The recursive region merging procedure used to create the document tree implies the following: (i) the document has been split into regions, and (ii) there exists a way to discern which regions are "closest", that is, which regions should be merged first.

Task (i) can be accomplished with any image segmentation program that partitions the image into several regions. This could also be done by initially assuming that the individual pixels of the image are the regions to be considered. We rejected

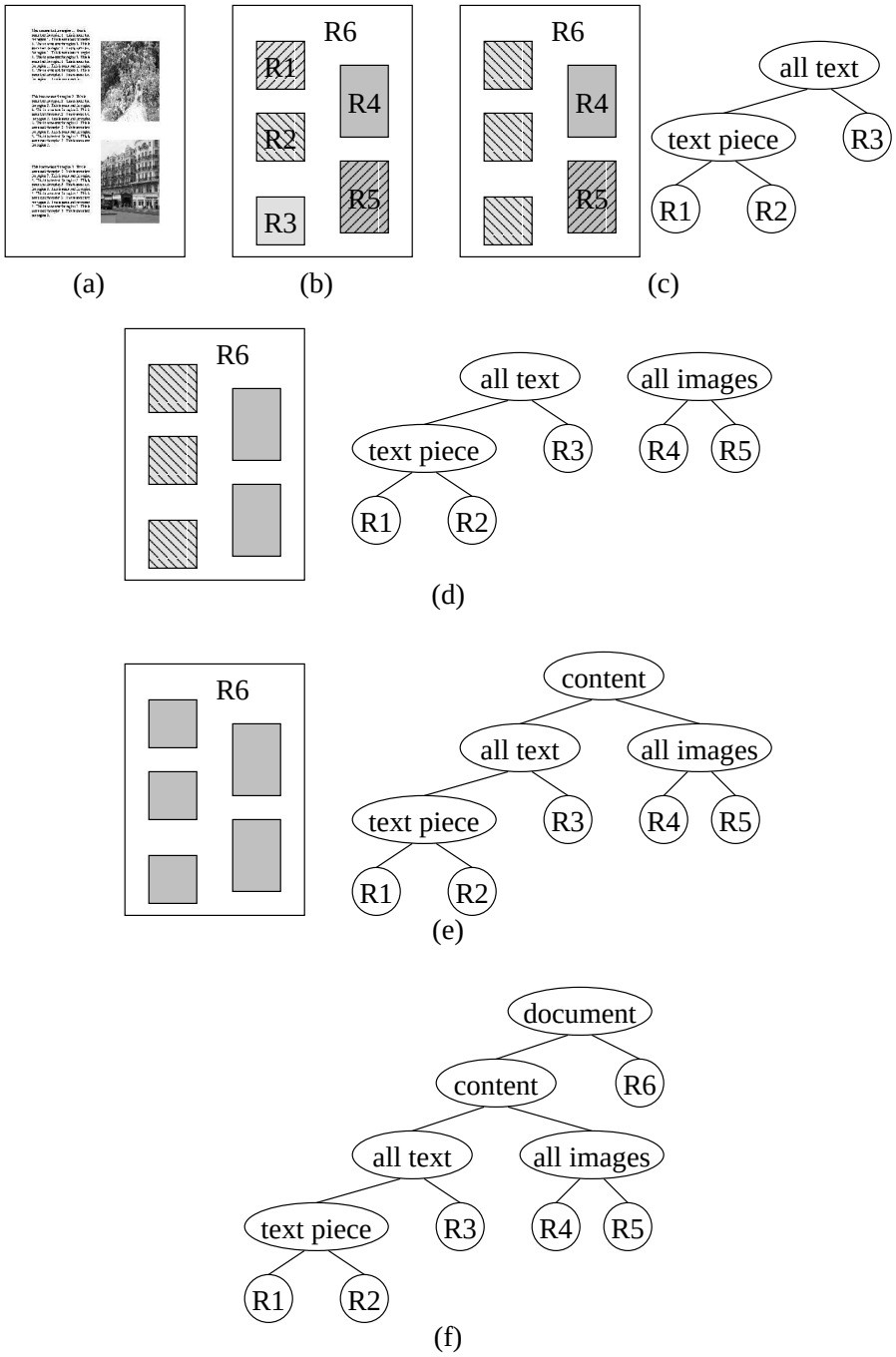


Fig. 2.1. Document tree concept demonstration. (a) A document with three text regions and two image regions. (b) The regions of the document before region merging. (c) The document tree and remaining regions of the document after two merges. (d) The document tree and remaining regions of the document after three merges. (e) The document tree and remaining regions of the document after four merges. (f) The final document tree describing the image in *(a)*.

this approach because we do not think the logical hierarchy of a document extends to the pixel level.

To accomplish task (ii), we first define for each region several features to describe the region. We choose these features so that they describe quantities that are believed to be useful for computing a distance between regions. The features should also be easy to calculate from the original image, and also easy to combine into a feature vector for the aggregate region when two regions are merged. We then define a distance function between regions that depends only on the feature vectors of the two regions being compared. This distance function is the criterion for deciding the order in which regions are merged while forming the document tree. The region pair with minimum distance is merged recursively until only one region remains.

It should be noted that we choose a segmentation algorithm, a set of features, and a distance function to describe one particular notion of what similar regions are. Other applications could use different segmentation algorithms, distance functions, and sets of features. In this thesis, we simply demonstrate the effectiveness of the ones we choose to use.

The remainder of this chapter describes the details of each step taken to form the document tree. Section 2.2 describes the file formats allowed for the original document. Section 2.3 describes the segmentation procedure by which we partition the document into regions. Section 2.4 describes our notion of which regions should be considered close to each other, motivating our choice of features and distance function. Then in Section 2.5, we describe the features that were calculated for each region, and Section 2.6 describes the distance function defined on this set of features. Finally, Section 2.7 gives a detailed procedure for forming the document tree.

2.2 Document Format Specification

The goal of this part of our document processing system is to be able to form a document tree to describe any existing document. In general, documents exist in

Fig. 2.2. Block diagram of segmentation procedure

either electronic or hard copy format, or both. Documents for which only hard copies exist can be scanned, and most documents in electronic format can be converted to a pdf file. Therefore, the system was designed to use either a scanned image or a pdf file for input. These input formats need to be handled slightly differently during the segmentation process, as described below.

2.3 Segmentation of Document

We first describe the general procedure used to break the document into regions. Then we describe the specific segmentation algorithm we used for this task, the Trainable Sequential Maximum a Posteriori (TSMAP) algorithm, and how it was trained to solve this document segmentation problem. Finally, we discuss the details of the implementation of the general segmentation procedure.

2.3.1 Segmentation procedure

The procedure for segmenting a document is illustrated in Figure 2.2. The raw input file goes through a preprocessing step to produce an image that can be passed to the particular segmentation algorithm that was chosen. The output of this segmentation algorithm is then passed through a postprocessing step that removes any unwanted noise or insignificant details in the segmentation, yielding a useful segmentation of the document. The regions defined by this useful segmentation of the original document become the leaves of the document tree.

Note that this strategy does not specify the use of one specific segmentation procedure. Any reasonable segmentation procedure with appropriate preprocessing

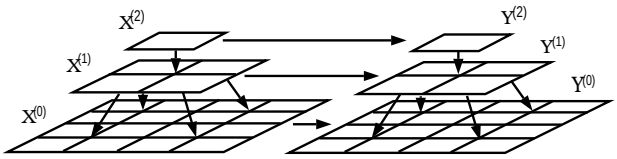


Fig. 2.3. The multiscale segmentation model proposed in the TSMAP paper [1]. $Y^{(n)}$ contains the image feature vectors extracted at scale n while $X^{(n)}$ contains the corresponding class of each pixel at scale n. Notice that both the image features, Y, and the context model, X, use multiscale pyramid structures.

and postprocessing steps could be used. The rest of this section, however, does describe the specific segmentation algorithm that was used, along with pertinent preprocessing and postprocessing steps.

2.3.2 The TSMAP algorithm

We chose to use the Trainable Sequential Maximum a Posteriori (TSMAP) segmentation algorithm proposed by Hui Cheng and Charles Bouman in [1] to perform the segmentation of the documents. The TSMAP algorithm uses Bayesian multiscale techniques to find an estimate of which class should be assigned to each pixel in a grayscale image. A multiscale quad tree model of both the class labels and image feature vectors is used. As illustrated in Figure 2.3, for each scale n, there is a random field of image feature vectors, $Y^{(n)}$, and a random field of class labels, $X^{(n)}$. $Y^{(n)}$ contains Haar basis wavelet coefficients (image texture and edge information) at scale n, and $X^{(n)}$ contains the corresponding class labels.

The class label fields $X^{(n)}$ are assumed to be a Markov chain in scale, dependent only on the next coarser scale, so that $p(x^{(n)}|x^{(>n)}) = p(x^{(n)}|x^{(n+1)})$. The image features $y^{(n)}$ are assumed conditionally independent given the class labels $x^{(n)}$ and the image features $y^{(n+1)}$ at the coarser scale. These two assumptions yield a tractable expression for $p(y, x)$. Segmenting the image then corresponds to estimating the class labels X from the image feature data Y. Although the MAP estimate is popular for performing this task, it is undesirable in this case because it assigns equal weight to

errors at all scales. The authors argue that since coarse scale misclassifications affect more pixels, they should be penalized more heavily than fine scale errors. Therefore they use the sequential MAP (SMAP) cost function proposed in [10] to perform the estimation. This cost function has the desired property that misclassifications at coarse scales are penalized more heavily than those at fine scales. The authors show that the image segmentation $\hat{x}^{(n)}$ can then be computed with one recursive coarse-to-fine pass through the tree, assuming the feature data y are known for every level of the quad tree.

However, calculation of this coarse-to-fine recursion requires knowledge of several parameters of the model. These parameters are estimated in a training step, hence the title "Trainable SMAP algorithm". The idea is to provide several images for which the true segmentations are known. Normally, these true segmentations are produced by a human. For these images with corresponding hand segmentations, the algorithm calculates both the class label quadtree X and the feature vector quadtree Y, and then uses this information to estimate the parameters of the model.

Once the training step is complete, any number of images can be segmented, because the model parameters are then known. The SMAP estimate of the class label field, $\hat{x}^{(n)}$, can be calculated for any grayscale image with one fine-to-coarse recursion to calculate the image features y followed by one coarse-to-fine recursion that calculates $\hat{x}$, as mentioned above. It should be noted that because of the nature of the wavelet decomposition used to calculate the image features $y^{(n)}$, the finest segmentation level that can be estimated is only half the resolution of the image being segmented. For instance, if the image being segmented has a resolution of 100 dots per inch (dpi), which is 39.37 dots per centimeter (dpcm) , the finest resolution image output from the TSMAP program will have a resolution of 50 dpi (19.69 dpcm).

2.3.3 Training the TSMAP algorithm for use with documents

We trained the TSMAP algorithm to assign one of the following four classes to the pixels of the image being segmented: header text, body text, image, and background. The output of the TSMAP segmentation step is then an image whose pixels contain the class numbers assigned to the corresponding pixels in the original document. Because images generated directly from pdf files contain much less noise than scanned images, using the same set of training parameters to segment images from both input formats produced incorrect segmentations of the images. Therefore, two sets of training parameters have been developed, one for documents in scanned image format, and one for documents in pdf format.

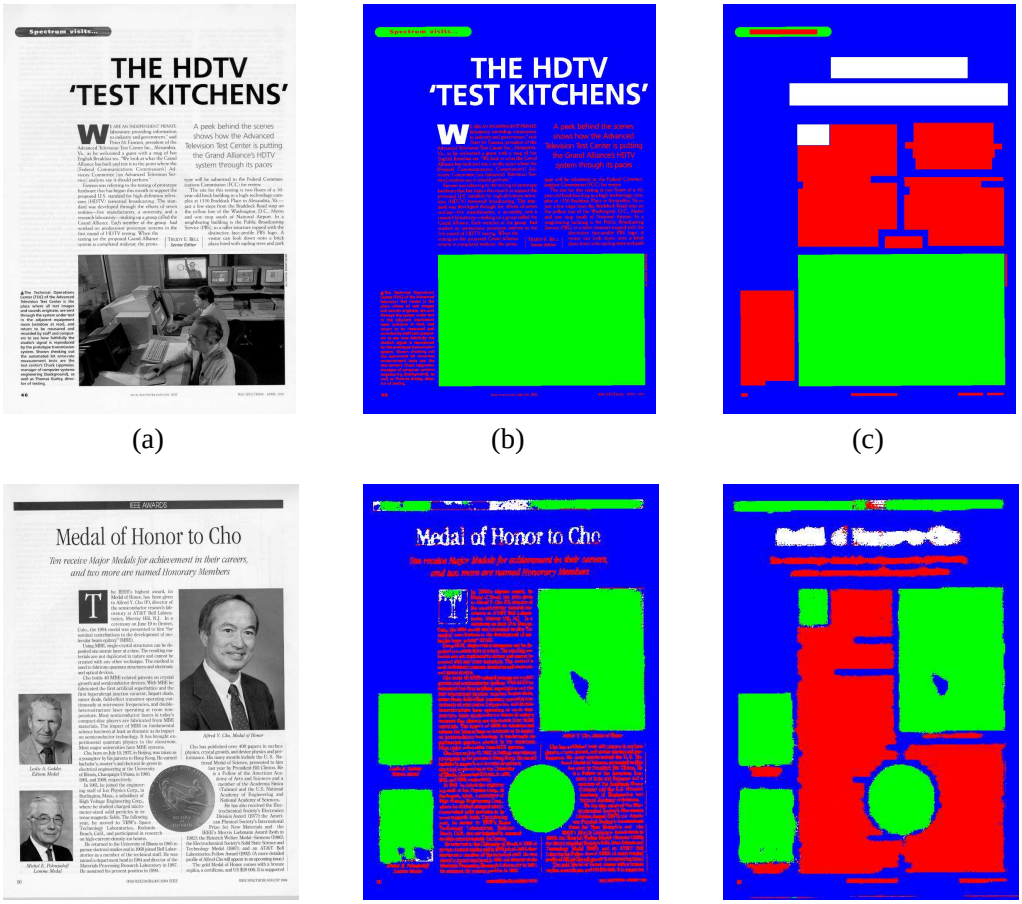
The case of scanned documents was illustrated in [1], and we proceeded much the same as the authors of the paper. Twenty training images with corresponding hand segmentations were provided to the TSMAP training algorithm to calculate the model parameters for use with scanned documents. The same twenty 100 dpi (39.37 dpcm) training images as those used for [1] were used, but the corresponding hand segmentations were modified so that the resulting segmentations would produce text regions instead of individual letters. Figure 2.4 illustrates the difference.

For the case of documents in pdf format, eight images were generated from pdf files and then hand segmented. These images formed the training set for the TSMAP algorithm to calculate the model parameters for use with pdf documents.

2.3.4 Implementation of the segmentation procedure

These are the details of our implementation of the segmentation procedure outlined in Section 2.3.1.

The preprocessing step prepares the document to be segmented. Since the TSMAP segmentation program has been trained on 100dpi (39.37 dpcm) images, the input to the segmentation step needs to be a 100dpi (39.37 dpcm) grayscale image. For a raw image in scanned format, the preprocessor performs a simple color-to-grayscale



(a) (b) (c)

(e) (e) (g)

Fig. 2.4. (a) Typical training image from the Cheng and Bouman paper [1] introducing the TSMAP algorithm. (b) Manual segmentation of *(a)* used in [1] to train the TSMAP segmentation program. (c) Manual segmentation of *(a)* used in this thesis to train the TSMAP segmentation program. (d) A grayscale scanned image. (e) Segmentation of *(d)* produced by the TSMAP algorithm when trained using manual segmentations as in *(b)*. (f) Segmentation of *(d)* produced by the TSMAP algorithm when trained using manual segmentations as in *(c)*.

conversion followed by subsampling. For a raw image in pdf format, a 100dpi (39.37 dpcm) grayscale image is generated directly from the pdf file.

The segmentation algorithm used for this paper is the TSMAP algorithm. The 100dpi (39.37 dpcm) grayscale image from the preprocessing step is the input to the TSMAP segmentation program, and the output of the program is a new 50dpi (19.69 dpcm) image whose pixels contain the class numbers assigned to the corresponding pixels in the original image. The format of the original image (scanned or pdf) determines which set of model parameters are used to calculate the segmentation.

The postprocessing step cleans up the output from the segmentation step. While the TSMAP segmentation procedure does well from a large-scale perspective, there are normally many small spots of noise in the segmentations. Many of these are singleton pixels, and are removed in this postprocessing step by the use of a rather strict despeckling filter. When a pixel is classified differently from all its neighbors, we consider it to be misclassified. Therefore, each singleton pixel in the segmentation image is replaced with a majority vote among the members of its 8-point neighborhood. This concludes the postprocessing step.

The despeckled segmentation image is then used to partition the original document into regions. We define the regions of the document to be the groups of pixels in the original document that correspond to the connected components of the despeckled segmentation image.

2.4 Assumptions Leading To Region Feature Vector and Distance Function Definitions

Once the document has been split into regions, we need to define the region feature vector and a corresponding distance function which will be used to create the document tree. To do this, we need to have some notion of what makes two regions close to each other. Of course, this notion of closeness is dependent on the application, giving rise to many possible feature vectors and distance functions. For this thesis,

we restrict our notion of closeness to depend on the following five characteristics of the regions: class, size, position, color, and text content.

- *Class:* Section 2.3.3 explains that the TSMAP segmentation program assigns to each region one of the following four classes: header text, body text, image, and background. Our notion of closeness depends on class in the following way. The classes of the regions being compared determine how heavily to weight the other four closeness factors. For instance, the text content of the regions is much more important in a comparison of two body text regions than in a comparison of two image regions.

- *Size:* Small regions should be merged before large regions. Small-small pairs should be merged first, then small-large, and finally large-large pairs. For instance, we want the various text regions of a document such as paragraphs to merge with each other into one large text region before any of them merge with a huge region such as the background. This keeps the hierarchy of the document evident in the tree that describes the document.

- *Position:* Region pairs which are adjacent or spatially close should be merged before region pairs which are spatially far from each other. For instance, in a column of three paragraphs, the first merge should not be the top-bottom pair.

- *Color:* Regions with similar color characteristics should be considered close. This applies especially to regions classified as image.

- *Text Content:* Regions with similar text content should be considered close. This applies more to regions containing much text than to regions containing little or no text.

We emphasize that these five notions of nearness are assumptions specific to our particular example, and that for other applications, other assumptions guiding the notion of nearness could be more appropriate.

2.5 Regions and Region Features

The regions of the image are defined as the connected components of the despeckled segmentation of the image. Several features are defined that describe the region in terms of the five similarity characteristics from Section 2.4. Once the features are calculated, they are the only information about the regions that is saved and used. The features alone determine the distance between regions when the document tree is formed, and each node of the document tree contains the aggregate features that describe the section of the document descending from that node.

Whenever possible, the features are chosen to be independent of the resolution of the raw input image. This is not so important for document tree generation because this procedure only compares regions that originate in the same image. However, we eventually want to compare documents by comparing their document trees (and therefore their feature vectors). When this happens, for example a region from a 400dpi (157.5 dpcm) image can be compared to a region from a 300dpi (118.1 dpcm) image, and it becomes useful to have features whose values do not depend on the resolution of the image they come from. Otherwise, when comparing a document scanned at two different resolutions, the images might be considered quite far apart when logically they are not.

To this end, those quantities which contain a number of pixels are represented as a fraction of the number of pixels in the document. Also, quantities which contain pixel locations are divided by the maximum dimension of the image. Under this scheme, a 640 by 480 image and a 160 by 120 image are both represented as a 1.0 by 0.75 image.

The features are also chosen so that they are easy to calculate initially, and so that the features of the aggregate region formed by merging two regions are also easy to calculate given only the features of the two component regions. The following sections describe both the features themselves and the method to be used to find the new feature value when two regions are merged. In all, we define one class feature,

two size features, three position features, three color features, and one text content feature.

2.5.1 Class feature

Recall that the TSMAP algorithm assigns a class to each region of the original image. The four classes are header text, body text, image, and background. This feature stores the class information for a region.

Class ratio

Each region in the document initially has only one class, that which is assigned to it by the TSMAP program. As regions are merged during the formation of the document tree, however, an aggregate region can contain regions of differing classes. Therefore we save the class information in a four-element vector called the class-ratio vector. The i-th element of the class-ratio vector contains the fraction of pixels in the entire document that are both contained in the region and of class i.

$$classratio[i] = \frac{\text{number of pixels of class } i \text{ in the region}}{\text{number of pixels in the entire document}}$$

Note that using the fraction instead of just the number of pixels of class i makes this feature independent of the raw image resolution, as desired.

When merging regions R_1 and R_2, the class-ratio vector of the aggregate region R_{new} is the element by element sum of the class-ratio vectors of the regions being merged.

$$classratio_{R_{new}}[i] = classratio_{R_1}[i] + classratio_{R_2}[i]$$

2.5.2 Size features

Two features are defined that describe the size of the region.

Number of pixels

The number of pixels (numpixels) feature is the number of pixels in the original image that are contained in the region. This feature is not independent of the resolution of the raw image input, but it is useful for calculating the values of other features which are input resolution independent.

$$numpixels = \text{number of pixels in region}$$

When merging regions R_1 and R_2, the numpixels value of the aggregate region R_{new} is the sum of the numpixels values of the regions being merged.

$$numpixels_{R_{new}} = numpixels_{R_1} + numpixels_{R_2}$$

Size ratio

The size-ratio feature is the fraction of pixels in the entire document that are contained in the region.

$$sizeratio = \frac{\text{number of pixels in region}}{\text{number of pixels in document}}$$

Note that this feature is independent of input image resolution.

When merging regions R_1 and R_2, the size-ratio of the aggregate region R_{new} is the sum of the size-ratio values of the regions being merged.

$$sizeratio_{R_{new}} = sizeratio_{R_1} + sizeratio_{R_2}$$

2.5.3 Position features

Three features are defined that either describe the position of the region or are calculated using the position information of the region.

Bounding box

The bounding box of a region is defined as the smallest rectangle which encloses the entire region. We represent this rectangle by storing its upper left and lower right pixels. Thus the upper left pixel contains the minimum row value and the minimum column value over all pixels in the region. Similarly, the lower right pixel contains the maximum row and column values over all pixels in the region. The pixel locations are represented relative to the maximum dimension of the document, so they are independent of the resolution of the original image.

$$
\begin{aligned}
boundbox.ul.m &= \frac{\text{min row location of any pixel in the region}}{\text{max dimension of image}} \\
boundbox.ul.n &= \frac{\text{min column location of any pixel in the region}}{\text{max dimension of image}} \\
boundbox.lr.m &= \frac{\text{max row location of any pixel in the region}}{\text{max dimension of image}} \\
boundbox.lr.n &= \frac{\text{max column location of any pixel in the region}}{\text{max dimension of image}}
\end{aligned}
$$

When merging regions R_1 and R_2, the bounding box of the aggregate region R_{new} is

$$
\begin{aligned}
boundbox_{R_{new}}.ul.m &= \min(boundbox_{R_1}.ul.m, boundbox_{R_2}.ul.m) \\
boundbox_{R_{new}}.ul.n &= \min(boundbox_{R_1}.ul.n, boundbox_{R_2}.ul.n) \\
boundbox_{R_{new}}.lr.m &= \max(boundbox_{R_1}.lr.m, boundbox_{R_2}.lr.m) \\
boundbox_{R_{new}}.lr.n &= \max(boundbox_{R_1}.lr.n, boundbox_{R_2}.lr.n)
\end{aligned}
$$

The bounding box feature is most useful at the leaf level of the document tree, before regions are merged. For instance, the bounding box of a text region can define a section of the original image to send to an optical character recognition program to extract the text content of the region. However, the bounding box can become unuseful once two regions are merged.

If for example a very small region is merged with a region far from it, which is likely to happen because we encourage small regions to merge first, the bounding box expands a lot while the overall region does not go through much change. This behavior is illustrated in Figure 2.5. We conclude that some other way that is not so

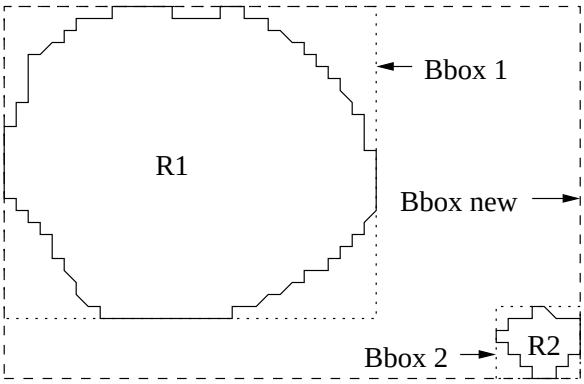


Fig. 2.5. Illustration of the bounding box feature, and how it can become non-descriptive when a large region is merged with a small region far from it.

susceptible to tiny bits of segmentation noise should be used to describe the position of the pixels in a region, especially at the aggregate region level of the document tree. To this end, we define the spatial variance bounding box later in this section. However, we must first define the centroid and the spatial variance of a region.

Centroid

The centroid feature of a region R is defined as the average location of the pixels in the region. The pixel locations are stored relative to the maximum dimension of the document, so the numerical value is independent of the resolution of the input image.

$$\begin{aligned} centroid.m &= \frac{1}{numpixels} \sum_{\text{pixels in R}} \frac{\text{row location of pixel}}{\text{max dimension of image}} \\ centroid.n &= \frac{1}{numpixels} \sum_{\text{pixels in R}} \frac{\text{column location of pixel}}{\text{max dimension of image}} \end{aligned}$$

When merging regions R_1 and R_2, the centroid of the aggregate region R_{new} is a weighted average of the centroids of the regions being merged, where the weighting constants are the number of pixels in each region.

$$centroid_{R_{new}} = \frac{numpixels_{R_1} \times centroid_{R_1} + numpixels_{R_2} \times centroid_{R_2}}{numpixels_{R_1} + numpixels_{R_2}}$$

Spatial variance

The spatial variance (svar) feature of a region R is defined as the variance of the location of the pixels in the region. It is an ordered pair whose members are the sample variances of the row and column location of the pixels in the region, respectively. These variances are calculated from pixel locations that are represented relative to the maximum dimension of the image, so the numerical values of the variances do not depend on the resolution of the original image.

$$\begin{aligned} svar.m &= \frac{1}{numpixels} \sum_{\text{pixels in R}} \left(\frac{\text{row location of pixel}}{\text{max dimension of image}} - centroid.m \right)^2 \\ svar.n &= \frac{1}{numpixels} \sum_{\text{pixels in R}} \left(\frac{\text{column location of pixel}}{\text{max dimension of image}} - centroid.n \right)^2 \end{aligned}$$

When merging regions R_1 and R_2, the spatial variance of the aggregate region R_{new} is calculated in the following way.

$$\begin{aligned} sumx^2_{R_1}.m &= numpixels_{R_1} \times [svar_{R_1}.m + (centroid_{R_1}.m)^2] \\ sumx^2_{R_2}.m &= numpixels_{R_2} \times [svar_{R_2}.m + (centroid_{R_2}.m)^2] \\ sumx^2_{R_{new}}.m &= sumx^2_{R_1}.m + sumx^2_{R_2}.m \\ svar_{R_{new}}.m &= \frac{sumx^2_{R_{new}}.m}{numpixels_{R_{new}}} - (centroid_{R_{new}}.m)^2 \end{aligned}$$

$$\begin{aligned} sumx^2_{R_1}.n &= numpixels_{R_1} \times [svar_{R_1}.n + (centroid_{R_1}.n)^2] \\ sumx^2_{R_2}.n &= numpixels_{R_2} \times [svar_{R_2}.n + (centroid_{R_2}.n)^2] \\ sumx^2_{R_{new}}.n &= sumx^2_{R_1}.n + sumx^2_{R_2}.n \\ svar_{R_{new}}.n &= \frac{sumx^2_{R_{new}}.n}{numpixels_{R_{new}}} - (centroid_{R_{new}}.n)^2 \end{aligned}$$

Spatial variance bounding box

We point out that the centroid and svar features may be used to generate a concept that is similar to the bounding box of the region yet much less sensitive than the bounding box to small bits of noise. We define the spatial variance bounding box (*svar.bbox*), to be a rectangle whose center is the centroid of the region,

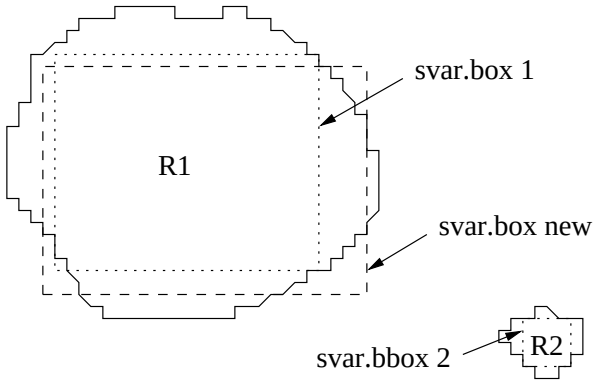


Fig. 2.6. Illustration of the Spatial Variance Bounding Box, and how it is less susceptible than the Bounding Box to becoming non-descriptive when a large region is merged with a small region far from it.

and which extends one spatial standard deviation to either side of the centroid in each of the row and column directions. That is, the corners of the *svar.bbox* are $(centroid.m \pm \sqrt{svar.m}, centroid.n \pm \sqrt{svar.n})$. The *svar.bbox* concept, along with its lower sensitivity to the noise that disrupts the bounding box feature, are illustrated in Figure 2.6.

2.5.4 Color features

Three features are defined that describe the color content of a region.

RGB mean

The rgbmean feature is a vector of length three containing the sample mean of the red, green, and blue intensities of pixels in the region, respectively.

$$
\begin{aligned}
rgbmean[0] &= \frac{1}{numpixels} \sum_{\text{pixels in R}} \text{red intensity of pixel} \\
rgbmean[1] &= \frac{1}{numpixels} \sum_{\text{pixels in R}} \text{green intensity of pixel} \\
rgbmean[2] &= \frac{1}{numpixels} \sum_{\text{pixels in R}} \text{blue intensity of pixel}
\end{aligned}
$$

When merging regions R_1 and R_2, the rgbmean of the aggregate region R_{new} is an element by element weighted sum of the rgbmean vectors of the regions being merged, where the weighting constants are the number of pixels in each region.

$$rgbmean_{R_{new}}[i] = \frac{numpixels_{R_1} \times rgbmean_{R_1}[i] + numpixels_{R_2} \times rgbmean_{R_2}[i]}{numpixels_{R_1} + numpixels_{R_2}}$$

RGB variance

The rgbvar feature is a vector of length three containing the sample variances of the red, green, and blue intensities of pixels in the region, respectively.

$$\begin{aligned}
rgbvar[0] &= \frac{1}{numpixels} \sum_{\text{pixels in R}} \{(\text{red intensity of pixel}) - rgbmean[0]\}^2 \\
rgbvar[1] &= \frac{1}{numpixels} \sum_{\text{pixels in R}} \{(\text{green intensity of pixel}) - rgbmean[1]\}^2 \\
rgbvar[2] &= \frac{1}{numpixels} \sum_{\text{pixels in R}} \{(\text{blue intensity of pixel}) - rgbmean[2])\}^2
\end{aligned}$$

When merging regions R_1 and R_2, the rgbvar of the aggregate region R_{new} is an element by element variance update of the rgbvar vectors of the regions being merged.

$$\begin{aligned}
sumx^2_{R_1}[i] &= numpixels_{R_1} \times [rgbvar_{R_1}[i] + (rgbmean_{R_1}[i])^2] \\
sumx^2_{R_2}[i] &= numpixels_{R_2} \times [rgbvar_{R_2}[i] + (rgbmean_{R_2}[i])^2] \\
sumx^2_{R_{new}}[i] &= sumx^2_{R_1}[i] + sumx^2_{R_2}[i] \\
rgbvar_{R_{new}}[i] &= \frac{sumx^2_{R_{new}}[i]}{numpixels_{R_{new}}} - (rgbmean_{R_{new}}[i])^2
\end{aligned}$$

RGB histograms

The rhist, ghist, and bhist features are histograms of the red, green, and blue intensities of pixels in the region, respectively. The histograms have 32 bins which are uniformly spaced over the 255 possible values of the RGB intensities. This means, for example, that rhist is a vector of length 32, and that rhist[0] is the number of pixels whose red intensity value falls into the range [0,7], rhist[1] is the number of pixels whose red intensity value falls into the range [8,15], and so on.

When merging regions R_1 and R_2, the histogram of the aggregate region R_{new} is an element by element sum of the histograms of the regions being merged.

$$\begin{aligned} rhist_{R_{new}}[i] &= rhist_{R_1}[i] + rhist_{R_2}[i] \\ ghist_{R_{new}}[i] &= ghist_{R_1}[i] + ghist_{R_2}[i] \\ bhist_{R_{new}}[i] &= bhist_{R_1}[i] + bhist_{R_2}[i] \end{aligned}$$

2.5.5 Text content feature

One feature is defined that describes the text content of a region.

Latent semantic information

The latent semantic information (latsem) feature is a vector containing information related to the words contained in the region. If the original input is a scanned image, the words are extracted from the document via an optical character recognition (OCR) package, and if the original input is a PDF file, the words are extracted directly from the pdf file using a program called pstotext.

The technique of latent semantic analysis is laid out in [3]. The general idea is to create a space describing word frequency in documents, and then to describe the region in terms of this space. We create this space using text from several documents. We begin with a dictionary of English words (terms). For each document, we count the number of times each dictionary term occurs in the document, and we put this information in a matrix. Each row of this matrix corresponds to a term from the dictionary, and each column of this matrix corresponds to a document. We denote this matrix X. We perform a singular value decomposition of X, yielding an exact model given by

$$X = T_0 S_0 D_0'$$

where T_0 and D_0 have orthogonal, unit length columns, and S_0 is a diagonal matrix of singular values. See Figure 2.7.

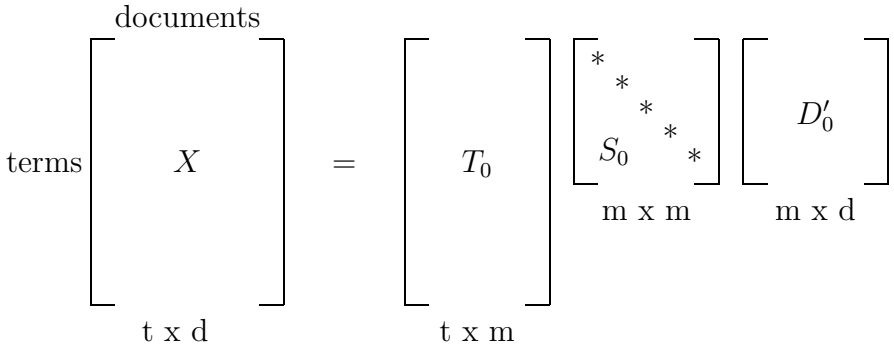


Fig. 2.7. Illustration of the singular value decomposition of the matrix X.

We then approximate X by deleting some of the small singular values and keeping only k of them, along with the corresponding columns of T_0 and D_0. This yields a reduced model for X given by

$$\widehat{X} = TSD'$$

as shown in Figure 2.8.

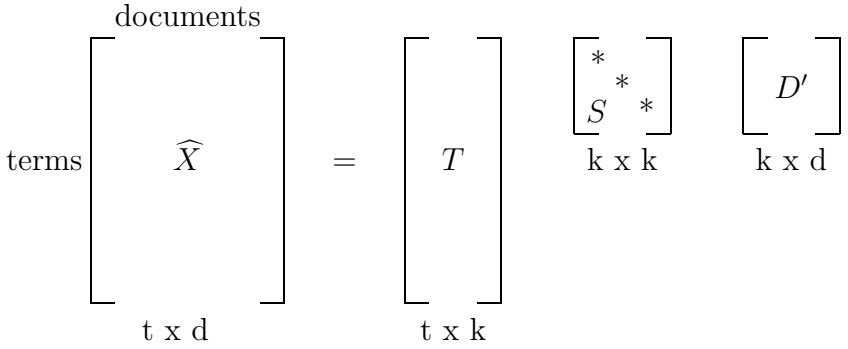


Fig. 2.8. Illustration of the reduced model $\widehat{X}$ for the matrix X.

We use the matrix T to generate the latsem vector for a region R. We count the number of times each dictionary word appears in the region and store this in a t by 1 vector W. Then the latent semantic vector describing a region is defined to be the matrix product of its word frequency vector with the matrix T. See Figure 2.9.

$$latsem_R = W'T$$

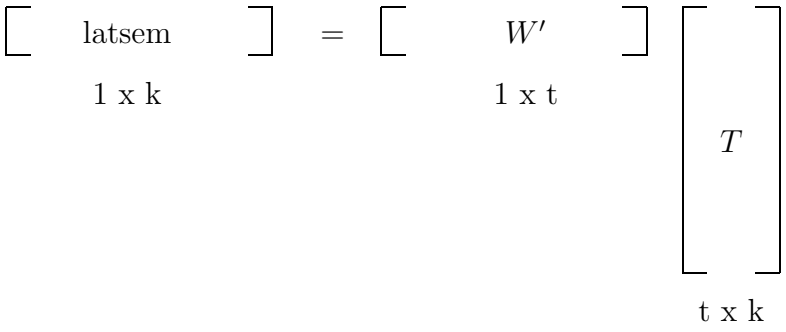


Fig. 2.9. Illustration of the definition of a region's latent semantic feature vector

When merging regions R_1 and R_2, the latsem vector of the aggregate region R_{new} is the element by element sum of the latsem vectors of the regions being merged.

$$latsem_{R_{new}}[i] = latsem_{R_1}[i] + latsem_{R_2}[i]$$

2.6 Merge Distance Between Regions

The final part of our method of forming the document tree is to define a distance between regions that will be used to determine the order in which regions are merged. The distance between regions is a function of the region features defined in Section 2.5. We define this distance function to implement the assumptions from Section 2.4 about what makes regions similar. We emphasize again that different assumptions could lead to the use of distance functions quite different from the one we describe here. This section simply records one implementation of a specific set of similarity assumptions.

We define the distance between two regions in the following way. We first define seven similarity functions to describe the similarity of the regions with regard to size, position, color, and text content. We denote these merge similarity functions

$$md_i(R_1, R_2),\ i \in \{0, 1, 2, 3, 4, 5, 6\}$$

We combine them by taking a weighted sum, where the weights are a function of the classes of the regions being merged. Recall from Section 2.4 that this is exactly

the role that class is intended to play in the distance calculation. We denote these class-dependent weights as

$$mw_i(c_1, c_2),\ i \in \{0, 1, 2, 3, 4, 5, 6\}$$

We then multiply this sum by a function of the size of the regions, also weighted according to the classes of the regions. This size function encourages small-small and small-large region pair merges to happen before large-large merges, as specified in Section 2.4. We denote the size function and its weight as

$$f_{size}(R_1, R_2) \text{ and } w_{size}(c_1, c_2),\ \text{respectively}$$

This yields the following distance function:

$$mclassdist_{c_1,c_2}(R_1, R_2) = [f_{size}(R_1, R_2) \times w_{size}(c_1, c_2)] \times \sum_{i=0}^{6} [md_i(R_1, R_2) \times mw_i(c_1, c_2)]$$

We define this to be the distance between regions R_1 and R_2 if every pixel of region R_1 is of class c_1 and every pixel of region R_2 is of class c_2. In general, however, both R_1 and R_2 may contain pixels of each of the four classes.

To find the total distance between the regions R_1 and R_2 in the general case, we first calculate $mclassdist_{c_1,c_2}(R_1, R_2)$ for all possible class pairings. Then we use the class-ratio vectors of the regions to assign a weight to each of the possible class pairings, and sum over all possible class pairings. This yields the total distance between regions. Let cr_R denote the class-ratio vector for a region R. Then

$$mergedist(R_1, R_2) = \sum_{c_1=0}^{3} \sum_{c_2=0}^{3} \frac{cr_{R_1}[c_1]}{\sum_{j=0}^{3} cr_{R_1}[j]} \times \frac{cr_{R_2}[c_2]}{\sum_{k=0}^{3} cr_{R_2}[k]} \times mclassdist_{c_1,c_2}(R_1, R_2)$$

The following sections contain the specific definitions of the quantities used to calculate this distance.

2.6.1 Definitions of similarity functions

The similarity functions $md_i(R_1, R_2)$ implement the various notions of region closeness assumed in Section 2.4. A few comments are in order before the definitions are

given. Recall that the functions eventually get combined by a weighted sum. We want the values of these weights to have approximately the same meaning for all similarity functions. That is, a weight of 1.0 should indicate the same level of strength being assigned to the function, independent of the similarity function to which it is applied. Therefore, each similarity function is normalized so its values fall approximately in the range [0,1].

Similarity as a function of RGB mean

The rgbmean similarity function is motivated by the color assumption from Section 2.4. It is defined as the ℓ^2 norm of the difference between the rgbmean vectors of two regions. The function is normalized as follows. Since the individual mean RGB values are in the range [0, 255], the maximum difference between any pair of means is 255. Therefore, the maximum value of the distance is $255 \times \sqrt{3}$, which is the value of the normalizing constant.

$$md_0(R_1, R_2) = \frac{\sqrt{\sum_{i=0}^{2}(rgbmean_{R_1}[i] - rgbmean_{R_2}[i])^2}}{255 \times \sqrt{3}}$$

Similarity as a function of RGB variance

The rgbvar similarity function is also motivated by the color assumption from Section 2.4. It is defined as the ℓ^1 norm of the difference between the rgbvar vectors of two regions. The function is normalized to approximately the range [0,1] as follows. As a rough calculation, the maximum variance case occurs when half the pixels are black, and half are white, so that all the pixels are half the range from the mean. In this case the variance is approximately $128^2 = 16\,384$. Since we are adding three quantities which could each be at most approximately 16 384, the normalizing constant is $1/(3 \times 16\,384)$.

$$md_1(R_1, R_2) = \frac{\sum_{i=0}^{2} |rgbvar_{R_1}[i] - rgbvar_{R_2}[i]|}{3 \times 16\,384}$$

Similarity as a function of RGB histograms

The histogram similarity function is motivated by the color assumption from Section 2.4. It is defined to be the ℓ^1 norm of the difference between the normalized histograms of two regions. We give an example to illustrate the need for normalizing the histograms. The following two histograms should have distance zero:

H1: [1, 2, 1]

H2: [100, 200, 100]

because corresponding bins contain the same percentage of the total number of pixels in the region. So before taking the ℓ^1 norm, we normalize the histograms by dividing each region's histogram entries by the number of pixels in the region. Each bin then contains the percentage of the region's pixels assigned to the bin.

After taking the ℓ^1 norm, we normalize the similarity function so its value will be in the range [0,1]. Because the sum over all bins in each normalized histogram is equal to 1, the maximum value of the ℓ^1 norm of the difference between two normalized histograms is 2. This is the value of the normalizing constant.

$$md_2(R_1, R_2) = \frac{1}{2}\sum_{i=0}^{31}\left|\frac{rhist_{R_1}[i]}{numpixels_{R_1}} - \frac{rhist_{R_2}[i]}{numpixels_{R_2}}\right|$$

$$md_3(R_1, R_2) = \frac{1}{2}\sum_{i=0}^{31}\left|\frac{ghist_{R_1}[i]}{numpixels_{R_1}} - \frac{ghist_{R_2}[i]}{numpixels_{R_2}}\right|$$

$$md_4(R_1, R_2) = \frac{1}{2}\sum_{i=0}^{31}\left|\frac{bhist_{R_1}[i]}{numpixels_{R_1}} - \frac{bhist_{R_2}[i]}{numpixels_{R_2}}\right|$$

Similarity as a function of latent semantic information

The latent semantic similarity function is motivated by the text content assumption from Section 2.4. It is defined to be the cosine of the angle included by the latent semantic vectors of the regions being compared. We normalize this quantity to the range [0,1] by taking one minus the cosine of the angle and then dividing by two. That is,

$$md_5(R_1, R_2) = \frac{1}{2} * \left(1 - \frac{< latsem_{R_1}, latsem_{R_2} >}{\| latsem_{R_1} \| \| latsem_{R_2} \|}\right)$$

where $< \cdot, \cdot >$ is the inner product operator, and $\|\|$ is the ℓ^2 norm.

Similarity as a function of spatial variance

The spatial variance similarity function is motivated by both the size and position assumptions from Section 2.4. It is the most complicated similarity function, and is the product of two terms. We begin by introducing these terms, and then we define them exactly later in the section.

The first term encourages the regions to be close spatially—the spatial variance bounding box term (svar.bbox). As illustrated in Figure 2.10 and explained later in this section, this term has the desired effect when the regions are approximately the same size. However, if the regions are of vastly different size, and the small one is actually close to the large region but is far from the corners of the large one, the small region can have quite a large distance from the large one.

To counteract this effect, we multiply the svar.bbox term by a term which depends on the minimum spatial variance difference (svar.diff). This term encourages the merging of two regions when the difference between the spatial variance of the aggregate region and the spatial variance of at least one of the component regions is small. As described more fully below, this term is smallest when one region contains the other, small when one region is much larger than another, and large when the regions are of similar size and not spatially close.

We originally used this product as the definition for $md_6(R_1, R_2)$, but we noticed that small regions very far from each other were merged instead of merging a small region with a large region near it. To encourage large-small region merges above small-small region merges, we divide the product of $md_{svar.bbox}(R_1, R_2)$ and $md_{svar.diff}(R_1, R_2)$ by the sum of the size ratios of the two regions. Because it differs from the other similarity functions in this way, we point out that the range of $md_6(R_1, R_2)$ is not equal to [0,1].

$$md_6(R_1, R_2) = \frac{md_{svar.bbox}(R_1, R_2) \times md_{svar.diff}(R_1, R_2)}{sizeratio_{R_1} + sizeratio_{R_2}}$$

Spatial Variance Bounding Box Term In Section 2.5.3, we defined the Spatial Variance Bounding Box (*svar.bbox*) of a region to be a rectangle whose center is the centroid of the region, and which extends one spatial standard deviation on each side of the centroid. That is, the corners of the *svar.bbox* are $(centroid.m \pm \sqrt{svar.m}, centroid.n \pm \sqrt{svar.n})$.

We define the distance between two *svar.bbox* as follows. Find the Euclidean distance from each corner of $svar.bbox_1$ to each corner of $svar.bbox_2$. There are sixteen such pairs. Keep the smallest two, and add them together.

We normalize the function as follows. When pixel locations are stored relative to the maximum dimension of the document, they are in the range [0,1]. Therefore the maximum Euclidean distance between any two points is $\sqrt{2}$. We are adding two distances, so we normalize $md_{svar.bbox}(R_1, R_2)$ by the constant $2 \times \sqrt{2}$.

Let D_1 be the minimum distance from any corner of $svar.bbox_{R_1}$ to any corner of $svar.bbox_{R_2}$, and let D_2 be the second smallest such distance. Then

$$md_{svar.bbox}(R_1, R_2) = \frac{D_1 + D_2}{2 \times \sqrt{2}}$$

The reason we chose to add the smallest two distances is summed up in the following observation. "Close" rectangles will generally have a side in common or close to each other, which puts at least two sets of corner pairs fairly close to each other.

When comparing regions of similar size, $md_{svar.bbox}(R_1, R_2)$ has the advantage that regions situated directly above or beside each other are closer than regions situated diagonally from each other. This advantage is illustrated in Figure 2.10(a). When comparing regions of vastly different size, the distance we chose has the advantage that if the small region is near the corner of a large region (as is often the case with an image and a caption), the distance between the regions is small, because two corners of the small region are close to the same corner of the large region. This behavior is illustrated in Figure 2.10(b). The primary disadvantage of $md_{svar.bbox}(R_1, R_2)$ is that if two regions have vastly different sizes, and the small region is near the centroid of

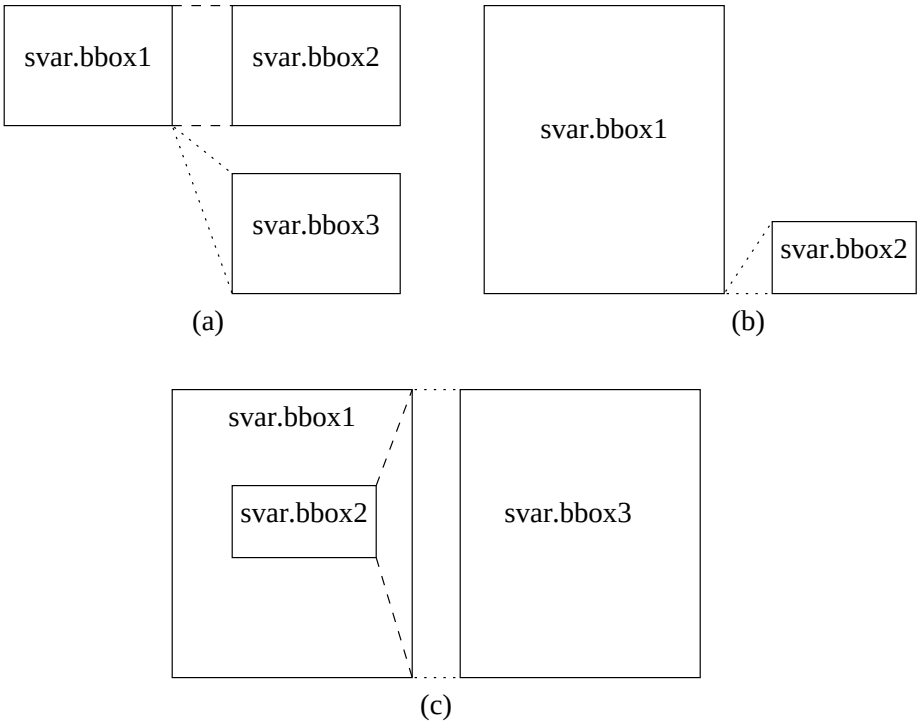


Fig. 2.10. (a) Spatial variance bounding box distance from Region 1 to two similarly sized regions. (b) Spatial variance bounding box distance from a large region to a small region close to a corner of the large region. (c) Small region R_2 inside large region R_1. $md_{svar.bbox}()$ mistakenly considers R_3 and R_1 closer than R_2 and R_1.

the large region, we would definitely want to merge the regions, but our distance puts the regions further apart than, say, a pair of similarly sized regions right next to each other. Figure 2.10(c) illustrates this concept. We would probably want to merge the small region with the large region containing it before merging the two large regions.

Spatial Variance Difference Term We make two observations which motivate the definition of the Spatial Variance Difference distance function $md_{svar.diff}(R_1, R_2)$. Then we define the function.

Let R_1 and R_2 be regions. Our first observation is that if R_1 is much larger than R_2, then the spatial variance of $R_1 \cup R_2$ is approximately equal to that of R_1, because there aren't enough pixels in R_2 to significantly affect the centroid and spatial variance sums. This is essentially the idea behind using the spatial variance bounding box, which is introduced in Section 2.5.3 and illustrated in Figure 2.6.

Our second observation is that the spatial variance of $R_1 \cup R_2$ can be smaller than the spatial variance of either R_1 or R_2, but not both. This happens when, say, the pixels of R_2 lie closer to the centroid of $R_1 \cup R_2$ than the pixels of R_1. An example of this is when R_1 is a ring, and R_2 is the interior of the ring. Since the pixels of R_2 are closer to the centroid than R_1, we consider it sensible to merge the regions, because R_2 is probably contained inside R_1.

We conclude from these two observations that encouraging the minimum spatial variance difference to be small encourages first merging regions where one region is inside the other, and second regions with large size differences. Both of these effects are consistent with the assumptions of Section 2.4. In particular, the minimum spatial variance difference is small for the $md_{svar.bbox}(R_1, R_2)$ problem case illustrated in Figure 2.10(c). Therefore we multiply $md_{svar.bbox}(R_1, R_2)$ by the $md_{svar.diff}(R_1, R_2)$ defined below in an attempt to correct this problematic behavior of $md_{svar.bbox}(R_1, R_2)$.

We define the $md_{svar.diff}(R_1, R_2)$ as follows. Let R_1 and R_2 be the two regions being compared. We calculate the spatial variance of $R_1 \cup R_2$, find the minimum change in variance in the row and column directions, and sum them up. The range

of this sum is $[-\frac{1}{2}, \frac{1}{2}]$, because the row and column changes are each in the range $[-\frac{1}{4}, \frac{1}{4}]$. Therefore we take the exponential of *min.diff* to yield a positive quantity, and normalize so the multiplier is in the range [0,1].

$$\begin{aligned} min.diff.m(R_1, R_2) &= \min\{svar_{R_1 \cup R_2}.m - svar_{R_1}.m, svar_{R_1 \cup R_2}.m - svar_{R_2}.m\} \\ min.diff.n(R_1, R_2) &= \min\{svar_{R_1 \cup R_2}.n - svar_{R_1}.n, svar_{R_1 \cup R_2}.n - svar_{R_2}.n\} \\ mult(R_1, R_2) &= \exp\{2 \times [min.diff.m(R_1, R_2) + min.diff.n(R_1, R_2)]\} \end{aligned}$$

$$md_{svar.diff}(R_1, R_2) = \frac{mult(R_1, R_2) - \exp(-1)}{\exp(1) - \exp(-1)}$$

2.6.2 The size function

The size function was included to encourage small-small and small-large region merges. Its form is analogous to a parallel combination of resistors. Since $sizeratio_{R_1}$ and $sizeratio_{R_2}$ are both in the range (0,1), and since $sizeration_{R_1} + sizeratio_{R_2}$ is also in (0,1), this function would be in the range (0, 1/4] without the normalizing constant.

$$f_{size}(R_1, R_2) = 4 \times \frac{1}{\frac{1}{sizeratio_{R_1}} + \frac{1}{sizeratio_{R_2}}}$$

2.6.3 The class-dependent weights

So far, we have only made the general statement that the weights w_{size} and $mw_i,\ i \in \{0, 1, 2, 3, 4, 5, 6\}$ discussed at the beginning of Section 2.6 depend on the classes of the regions. We have not yet stated specifically how they depend on these classes. In this section we describe these dependencies. We begin by explaining for each possible class pairing which features we believe are pertinent, and what the total weight of the class pairing should be. Then we report the specific values that were used to generate the examples in this thesis. It should be noted that we picked these values essentially as a well thought out guess based on our notions of how each class pairing should be treated. We make no claim regarding the optimality of the weights

that were used. We do, however, show that using the chosen weights produces results that seem to make sense.

We now describe the notions behind the weights assigned for each class pairing. We abbreviate the four classes as follows: TH is header text, TB is body text, I is image, and BG is background. For the remainder of this section, let c_1 and c_2 be the classes of regions R_1 and R_2, respectively.

We first explain the $w_{size}(c_1, c_2)$ assignments. With only one exception, our assignment of weight to the size function is independent of the classes of the regions being merged. Because the TSMAP algorithm often classifies small portions of image and text regions as background, and because we want these small background regions to merge with their respective image or text regions instead of with the large document background region, we assign more weight to $w_{size}(BG, BG)$. The values we use are arbitrary numbers whose only significance is that $w_{size}(BG, BG)$ is greater than $w_{size}(c_1, c_2)$ for all (c_1, c_2) not equal to (BG, BG).

$$w_{size}(c_1, c_2) = \begin{cases} 1.0 & , \quad (c_1, c_2) \neq (BG, BG) \\ 5.0 & , \quad (c_1, c_2) = (BG, BG) \end{cases}$$

We now explain the $mw_i,\ i \in \{0, 1, 2, 3, 4, 5, 6\}$ assignments. Because all the similarity functions $md_i(R_1, R_2)$, $i \in \{0, 1, 2, 3, 4, 5\}$ are normalized to be approximately in the range [0,1], we consider the sum over i of $mw_i(c_1, c_2)$ to be a reasonable measurement of the total strength of the weighting given to a particular class pairing. Therefore, we first choose the relative weighting of each class pairing as compared to the other pairings by setting $\sum_{i=0}^{6} mw_i(c_1, c_2)$ for each (c_1, c_2) pair. Then we choose for each class pair (c_1, c_2) how much of the total class pairing weight should be assigned to each similarity function.

Class pairing weights

We set the class pairing weights, $\sum_{i=0}^{6} mw_i(c_1, c_2)$, equal to a number in the range [1.0,10.0]. A Small class pairing weight corresponds to the class pair being more likely to merge, because it leads to a smaller distance between the regions.

We assign the class pairing weights as follows. We begin with class pairings which contain a body text (TB) region. We assume that the most likely class to merge with TB is TB, so we set

$$\sum_{i=0}^{6} mw_i(TB, TB) = 1.0$$

We assume that the next most likely class to merge with TB is TH, and that since header text is also text, the weight should be close to that for the (TB,TB) class pair. Therefore we set

$$\sum_{i=0}^{6} mw_i(TB, TH) = 2.0$$

We assume that the third most likely class to merge with TB is I, and we set

$$\sum_{i=0}^{6} mw_i(TB, I) = 5.0$$

Finally, because we want all the "content" regions to be merged with each other before they are merged with the background of the document, we assume that any merge with BG should be hard to do. Therefore we set

$$\sum_{i=0}^{6} mw_i(TB, BG) = 10.0$$

Next we consider class pairings which contain an image region. We assume that the most likely class to merge with I is I, and we set

$$\sum_{i=0}^{6} mw_i(I, I) = 1.0$$

The weight of the (I,TB) pair is set in the previous paragraph to 5.0. We assume that TB is more likely to merge with I than TH, because body text can be a caption. Therefore we set

$$\sum_{i=0}^{6} mw_i(I, TH) = 7.0$$

For the same reason as for the (TB,BG) pair, we assume that the (I,BG) pair should be difficult to merge, and we set

$$\sum_{i=0}^{6} mw_i(I, BG) = 10.0$$

We next consider class pairings which contain a header text region. We assume that the most likely class to merge with TH is TH, so we set

$$\sum_{i=0}^{6} mw_i(TH, TH) = 1.0$$

The (TH,TB) pair weight is set above to 2.0, and the (TH,I) pair weight is set above to 7.0. For the same reason as for the (TB,BG) and (I,BG) pairs, we set

$$\sum_{i=0}^{6} mw_i(TH, BG) = 10.0$$

The only remaining undefined class pair weight is that for a (BG,BG) pair. Since some regions which are classified as BG by the TSMAP program are actually portions of other regions such as images, we assume that the (BG,BG) pair should be just as hard to merge as any other pair involving BG. Then the order of merging for pairs containing BG regions is determined by the similarity functions, some of which depend on the regions' locations. We set

$$\sum_{i=0}^{6} mw_i(BG, BG) = 10.0$$

Similarity function weights

The individual $mw_i(c_1, c_2)$ assignments are given in Table 2.6.3. The following paragraphs contain the assumptions that are used to generate these weights. The latent semantic information similarity function, $md_5(R_1, R_2)$, is not yet implemented, so currently $mw_5(c_1, c_2) = 0$ for all (c_1, c_2) pairs. Therefore only two types of similarity functions are available: (1) a position and size function—the spatial variance distance, and (2) color content functions—rgbmean, rgbvar, rhist, ghist, and bhist distances.

We first give our assumptions for region pairs containing a body text region. For (TB,TB) and (TB,TH) pairs, we assume that the position is much more important

Table 2.1
Table of weights used for the merge distance function.

		TB	I	BG	TH
TB	mw_0:	0	0	1	0
	mw_1:	0	0	0	0
	mw_2:	0	0	0	0
	mw_3:	0	0	0	0
	mw_4:	0	0	0	0
	mw_5:	0	0	0	0
	mw_6:	1	5	9	2
	w_{size}:	1	1	1	1
I	mw_0:		1/17	1	0
	mw_1:		3/17	0	0
	mw_2:		3/17	0	0
	mw_3:		3/17	0	0
	mw_4:		3/17	0	0
	mw_5:		0	0	0
	mw_6:		4/17	9	7
	w_{size}:		1	1	1
BG	mw_0:			1	1
	mw_1:			0	0
	mw_2:			0	0
	mw_3:			0	0
	mw_4:			0	0
	mw_5:			0	0
	mw_6:			9	9
	w_{size}:			5	1
TH	mw_0:				0
	mw_1:				0
	mw_2:				0
	mw_3:				0
	mw_4:				0
	mw_5:				0
	mw_6:				1
	w_{size}:				1

Classes

TB — body text
TH — header text
I — image
BG — background

Weights correspondence

mw_0 = rgbmean
mw_1 = rgbvar
mw_2 = rhist
mw_3 = ghist
mw_4 = bhist
mw_5 = latsem
mw_6 = svar
w_{size} = size-ratio

than the color content of the text regions, so we assign all the weight to the spatial variance function. For (TB,I) pairs, we assume that the color information of the text region is irrelevant, so we assign all the weight to the spatial variance function. For (TB,BG) pairs, we note that sometimes text is colored to match the background somewhat, so we let this pair's weights take into account both rgbmean and spatial variance. We assume that position is much more important than color, however, and we assign 90% of the class pair weight to the spatial variance function, while the remaining 10% of the weight is assigned to the rgbmean function.

Next we give our assumptions for the remaining region pairs containing an image region. For (I,I) pairs, we assume that both the position and color features are important. We weight the position function slightly more heavily than the color content functions because there are more color content functions and so the color content would dominate the sum if all functions were given equal weight. We assign equal weight to the rhist, ghist, and bhist functions. For (I,TH) pairs, we use the same assumption as for (TB,I) pairs that text region color is irrelevant, and we assign all the weight to the spatial variance function. For (I,BG) pairs, we assume that the color information is nearly irrelevant, so we use the same weight breakup as that for (TB,BG) pairs.

For the remaining region pairs, we assume the following. For (TH,TH) pairs, we assume that the weighting should be the same as that for (TB,TB) pairs. For (TH,BG) pairs, we assume that the weighting should be the same as for (TB,BG) pairs. The last set of pair weights is that for the (BG,BG) pair. Since a document can have more than one background region, and they could be different colors (for example, a section of gray and a section of white), we assume that both color and position are important. We assign the same weight split between the rgbmean and spatial variance functions as for the other pairs involving BG regions.

We point out that the weights given in Table 2.6.3 are generated by our assumptions. Perhaps a better way to assign the weights is to define a training step wherein a user provides several images with known distances between regions, and the program

attempts to learn the best weights to produce the desired distances. However, this takes time and has not been implemented yet.

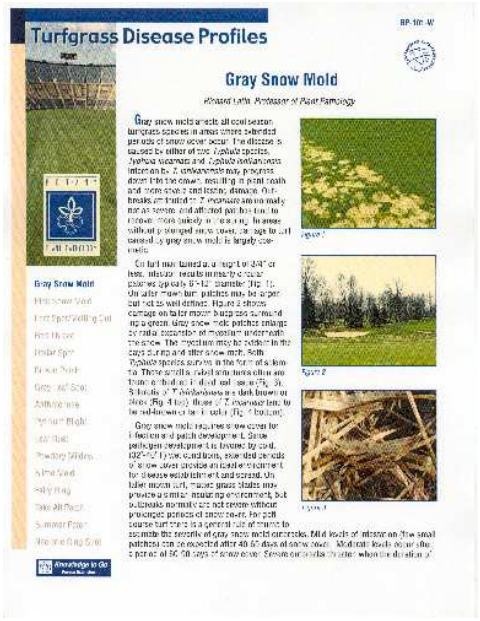
2.7 Specifics of Document Tree Formation

We have said that the document tree is formed by recursively merging the two closest regions under the distance function defined in Section 2.6. This is not entirely true. We do recursively merge the two closest regions until only one region remains; however, we do not record all of these merges in the document tree.

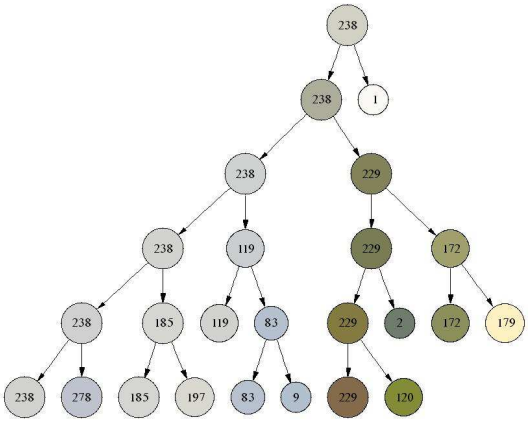
Initially, we did include all merges in the document tree, and we found that despite our effort to make small regions merge early in the process, some tiny regions slipped through and occupied positions close to the root of the document tree. Therefore we use a size threshold, only including the merge in the document tree if both regions are above the threshold. The small region is still merged, and the features are still updated; we just do not record the merge in the document tree. This means that in a document tree, the region corresponding to a parent node may not be exactly the union of the regions corresponding to its children; it may be the union of these two regions and one or more tiny regions in the document. We consider this to be less of a problem than having these tiny regions occupy nodes close to the root of the document tree.

Figure 2.11 illustrates the correspondence between a document tree and the merging procedure which creates it. In Figure 2.11(b), the nodes of the document tree have been given the color of the rgbmean feature contained in the node, and each node also contains the identification number which is assigned to the region by our connected components routine. In Figure 2.11(c) and (d), the regions corresponding to nodes of the document tree have been assigned random colors. The root node of the document tree corresponds to the entire document. The final merge in document tree formation is between the background (right child of root) and everything else (left child of root). This state of region merging is captured in Figure 2.11(c). The second

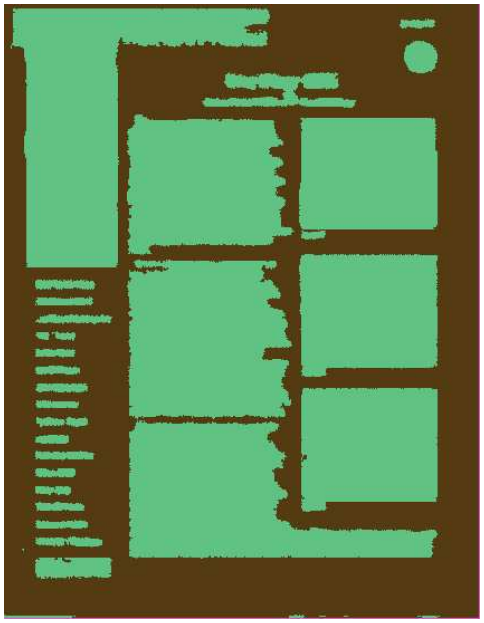
to last merge is between the text regions of the document and the image regions of the document. These regions are shown in Figure 2.11(d). Traveling further from the root of the document tree corresponds to the regions of the document becoming split further.



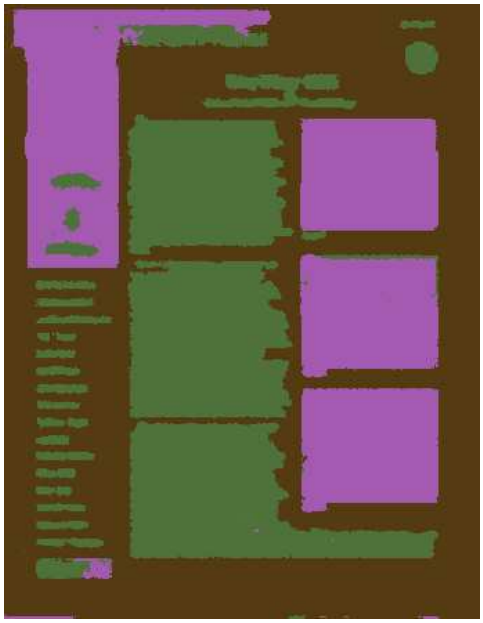
(a) Original document



(b) Document tree for the document



(c) Regions at last merge



(d) Regions at next to last merge

Fig. 2.11. An example of the correspondence between a document tree and the merging process that creates it.

3. DISTANCE BETWEEN DOCUMENT TREES

The goal of our document processing system is to describe the similarity of two documents based on their content and hierarchical structure. This section defines the similarity measurement that we developed. As in Section 2, we begin by describing the general strategy, and then proceed to describe the assumptions and detailed description of our specific implementation of the general strategy.

3.1 General Strategy

We assume that a good way to find the distance between two documents is to find a distance between the document trees that describe them. Therefore we begin by defining a method to calculate the distance between two document trees which have been created by the methods of Section 2. Since the only data stored at the nodes of the document tree are region feature vectors, we note that this document tree distance in general depends only on the region feature vectors and the hierarchical structure of the document trees themselves, because no other information is available.

Our particular method of calculating the distance between document trees depends heavily on calculating a distance between a pair of document tree nodes, where one node comes from each tree. Since nodes of the document tree correspond to regions in the document, this node comparison is actually a comparison of a region from one document with a region from another document. Therefore, we state assumptions about what makes two regions similar when comparing documents. We point out that in most cases, at least some of these assumptions will differ from the assumptions of Section 2.4 which are used to form the merging distance function that is used to create the document tree in the first place. Finally we define a distance between regions based on these similarity assumptions.

We emphasize that other assumptions about what makes documents and document trees similar could yield other methods for calculating the distance between documents. In addition, other distance functions could be found which would also satisfy the assumptions given in Section 3.3. The remainder of this section describes our particular implementation of the general strategy.

3.2 Method For Calculating Distance Between Document Trees

Our primary assumption about how to calculate the distance between two documents is that similar documents should have similar document trees. We calculate the distance between two document trees using a tree matching strategy. This strategy assumes that there exists a meaningful distance function to compare a node from one document tree (the feature vector of a region from one document) with a node from another document tree (the feature vector of a region from another document). We call this distance $node.dist(n_1, n_2)$, and we define the function used for this thesis in Section 3.4.

We begin the definition of our tree matching strategy by pointing out several characteristics of the document tree which result from the way it is formed.

1. Each node of the document tree, whether leaf or internal, represents a region in the document, and contains only the feature vector calculated for that region.

2. A document tree is not necessarily a complete tree.

3. There are no nodes with only one child. Any node is either a leaf node with zero children, or an internal node with two children.

4. There is no meaning associated with whether a node is the left or right child of its parent. This can cause several document tree structures to be logically equivalent. For example, the four document trees of Figure 3.1 are exactly equivalent. They all mean that D and E are merged first to form B, and then B and C are merged to form A. The interchangeability of the left and right children

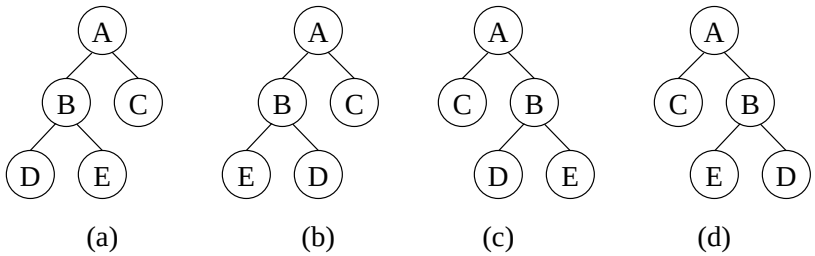


Fig. 3.1. Four equivalent document trees. A, B, C, D, and E are region feature vectors.

is true at every level of the document tree, so any tree matching method should search all equivalent rearrangements of the nodes to find the one that produces the best tree match.

5. Since the document tree is not necessarily a complete binary tree, it is possible for the case to arise where there is no exact tree structure match for two document trees, even considering the equivalence of the left and right children throughout the trees. Therefore the document tree distance calculation must find the best possible tree match, and then assign a penalty for the parts of the trees that do not match.

6. The nodes near the top of the document tree correspond to large regions of the document. We therefore assume that the document tree distance calculation should assign more weight to distances between nodes near the root of the document trees than to distances between nodes which are further down in the trees.

Tree Matching Algorithm Let n_1 be a node from document tree 1, and n_2 be a node from document tree 2. We define a recursive algorithm $tree.dist(n_1, n_2)$ to find the distance between nodes n_1 and n_2 given the nodes and the (possibly empty) subtrees descending from them. Thus if p_1 and p_2 are the root nodes of document trees 1 and 2, respectively, then $tree.dist(p_1, p_2)$ is the distance between the trees.

We assign larger weight to the nodes near the root of the tree as follows. We assign a fraction w of the weight to the distance between two parent nodes, while the remaining $(1 - w)$ fraction of the weight is assigned to the distance between their children. The following is pseudocode for the tree distance function. In the pseudocode, $left(n)$ and $right(n)$ are the left and right children, respectively, of the node n.

1. $tree.dist(n_1, n_2)$
2. if (n_1 has no children) and (n_2 has no children)
3. return $node.dist(n_1, n_2)$
4. else if ((n_1 has children) and (n_2 has no children)) OR
5. ((n_1 has no children) and (n_2 has children))
6. return $w \times node.dist(n_1, n_2) \times$ number of unmatched nodes
7. else
8. $$sizefrac.ll = \frac{sizeratio[left(n_1)] + sizeratio[left(n_2)]}{sizeratio(n_1) + sizeratio(n_2)}$$
9. $$sizefrac.rr = \frac{sizeratio[right(n_1)] + sizeratio[right(n_2)]}{sizeratio(n_1) + sizeratio(n_2)}$$
10. $$sizefrac.lr = \frac{sizeratio[left(n_1)] + sizeratio[right(n_2)]}{sizeratio(n_1) + sizeratio(n_2)}$$
11. $$sizefrac.rl = \frac{sizeratio[right(n_1)] + sizeratio[left(n_2)]}{sizeratio(n_1) + sizeratio(n_2)}$$
12. $$\begin{aligned} cost_1 = \; & w \times node.dist(n_1, n_2) + \\ & (1 - w) \times [sizefrac.ll \cdot tree.dist(left(n_1), left(n_2)) + \\ & \quad sizefrac.rr \cdot tree.dist(right(n_1), right(n_2))] \end{aligned}$$
13. $$\begin{aligned} cost_2 = \; & w \times node.dist(n_1, n_2) + \\ & (1 - w) \times [sizefrac.lr \cdot tree.dist(left(n_1), right(n_2)) + \\ & \quad sizefrac.rl \cdot tree.dist(right(n_1), left(n_2))] \end{aligned}$$
14. return $\min(cost_1, cost_2)$

Some comments about this tree matching algorithm are in order. The "number of unmatched nodes" term in pseudocode line 6 is our implementation of the penalty

for tree structures that do not match. The motivation for this penalty is described earlier in this section in the remarks about the structure of the document tree.

The intuition behind lines 12 – 14 of the pseudocode is as follows. If both n_1 and n_2 have children, we give a fraction w of the weight to the node distance between n_1 and n_2, and we give the remaining $(1 - w)$ fraction of the weight to the sum of the distance between subtrees rooted at the children of n_1 and n_2. Keeping in mind the interchangeability of the left and right children, we check the tree distance for both ways of pairing the children of n_1 and n_2, and we use the pairing which produces a smaller distance because this corresponds to the best tree match.

Within the sum of subtree distances for the children of n_1 and n_2 in pseudocode lines 12 and 13, we assign weight to each subtree distance function based on the fraction of the total size of the parents which is occupied by the children being compared. We do this because we assume that if n_1 and n_2 have one large and one small child region, and if the large regions match well while the small ones do not, we want the similarity of the large regions to dominate the distance term. Because this fraction depends both on the nodes being compared and on their parents, and because the $tree.dist(\cdot, \cdot)$ function compares the subtrees rooted at its arguments, the fraction is not included in the $tree.dist(\cdot, \cdot)$ definition itself. Otherwise the $tree.dist(\cdot, \cdot)$ function would no longer depend just on the subtrees rooted at its arguments.

3.3 Assumptions Leading To The Definition of Distance Between Regions

As in Section 2.4, we state our assumptions about what makes two regions similar. The key difference between this section and Section 2.4, however, is that this section deals with a comparison of regions from different documents, whereas that section deals with a merging criterion for regions from the same document. Since the distance function must be a function of the region feature vector which describes the class, size, position, color, and text content of the region, we state our assumptions for how each of these characteristics should affect the distance.

- *Class:* As in Section 2.4, the class characteristics of the regions being compared are used to determine how heavily to weight the contributions of the other four similarity factors.

- *Size:* Regions of similar size should be considered similar. Note that this differs from the assumptions of Section 2.4, wherein a large-small region pair is considered close, and a large-large region pair is considered far apart.

- *Position:* Regions which occupy similar positions in their respective documents should be considered similar.

- *Color:* Regions with similar color characteristics should be considered similar. This applies especially to regions classified as image.

- *Text Content:* Regions with similar text content should be considered similar. This applies more to regions containing much text than to regions containing little or no text.

Again, we emphasize that these five notions of similarity are assumptions specific to our particular example, and that for other applications, other assumptions guiding the notion of similarity could be more appropriate.

3.4 Distance Between Regions For Comparing Document Trees

The only undefined piece of the method of Section 3.2 for finding the distance between two document trees is the distance function $node.dist(n_1, n_2)$ which compares a node from one document tree with a node from another document tree. This distance takes into account both the feature vectors of the regions corresponding to the nodes and the hierarchical structure of the document tree itself. The feature vectors are defined in Section 2.5, and we define $node.dist(n_1, n_2)$ to implement the assumptions from Section 3.3 about what makes regions similar. We emphasize again that different assumptions could lead to the use of distance functions quite different

from the one we describe here. This section simply records one implementation of a specific set of similarity assumptions.

We define the distance between two document tree nodes n_1 and n_2 in the following way. Let R_1 and R_2 be the regions corresponding to n_1 and n_2, respectively. We first define nine similarity functions which describe the similarity of the regions R_1 and R_2 with regard to size, position, color, and text content. We denote these similarity functions

$$d_i(R_1, R_2),\ i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$$

We combine them by taking a weighted sum, where the weights are a function of the classes of the regions being merged. Recall from Section 3.3 that this is exactly the role that class is intended to play in the distance calculation. We denote these class-dependent weights as

$$w_i(c_1, c_2),\ i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$$

This yields the following distance function:

$$classdist_{c_1,c_2}(R_1, R_2) = \sum_{i=0}^{8} [d_i(R_1, R_2) \times w_i(c_1, c_2)]$$

We define this to be the distance between regions R_1 and R_2 if every pixel of region R_1 is of class c_1 and every pixel of region R_2 is of class c_2. In general, however, both R_1 and R_2 may contain pixels of each of the four classes.

To find the total distance between the regions R_1 and R_2 in the general case, we first calculate $classdist_{c_1,c_2}(R_1, R_2)$ for all possible class pairings. Then we use the class-ratio vectors of the regions to assign a weight to each of the possible class pairings, and sum over all possible class pairings. This yields the total distance between regions. Let cr_R denote the class-ratio vector for a region R. Then

$$node.dist(n_1, n_2) = \sum_{c_1=0}^{3} \sum_{c_2=0}^{3} \frac{cr_{R_1}[c_1]}{\sum_{j=0}^{3} cr_{R_1}[j]} \times \frac{cr_{R_2}[c_2]}{\sum_{k=0}^{3} cr_{R_2}[k]} \times classdist_{c_1,c_2}(R_1, R_2)$$

The following sections contain the specific definitions of the quantities used to calculate this distance.

3.4.1 Definitions of similarity functions

The similarity functions $d_i(R_1, R_2)$ implement the various notions of region closeness assumed in Section 3.3. A few comments are in order before the definitions are given. First, recall that the functions eventually get combined by a weighted sum. We want the values of these weights to have approximately the same meaning for all similarity functions. That is, a weight of 1.0 should indicate the same level of strength being assigned to the function, independent of the similarity function to which it is applied. Therefore, each similarity function is normalized so its values fall approximately in the range [0,1].

The second comment is that we have two basic types of similarity functions: those which depend strictly upon the feature vectors of the nodes being compared, and those which depend both on the structure of the document tree and on the feature vectors at the nodes in the tree. We currently have only one similarity function, $d_7(\cdot, \cdot)$, which takes into account the relation between a node and its children in the document tree, but we believe that other useful such distances could be defined.

Similarity as a function of RGB mean

The rgbmean similarity function is motivated by the color assumption from Section 3.3, and is exactly the same as the rgbmean similarity function for merging regions given in Section 2.6.1. It is defined as the ℓ^2 norm of the difference between the rgbmean vectors of two regions. The function is normalized as follows. Since the individual mean RGB values are in the range [0, 255], the maximum difference between any pair of means is 255. Therefore, the maximum value of the distance is $255 \times \sqrt{3}$, which is the value of the normalizing constant.

$$d_0(R_1, R_2) = \frac{\sqrt{\sum_{i=0}^{2}(rgbmean_{R_1}[i] - rgbmean_{R_2}[i])^2}}{255 \times \sqrt{3}}$$

Similarity as a function of RGB variance

The rgbvar similarity function is also motivated by the color assumption from Section 3.3, and also happens to be exactly the same as its counterpart in Section 2.6.1. It is defined as the ℓ^1 norm of the difference between the rgbvar vectors of two regions. The function is normalized to approximately the range [0,1] as follows. As a rough calculation, the maximum variance case occurs when half the pixels are black, and half are white, so that all the pixels are half the range from the mean. In this case the variance is approximately $128^2 = 16\,384$. Since we are adding three quantities which could each be at most approximately 16 384, the normalizing constant is $1/(3 \times 16\,384)$.

$$d_1(R_1, R_2) = \frac{\sum_{i=0}^{2} |rgbvar_{R_1}[i] - rgbvar_{R_2}[i]|}{3 \times 16\,384}$$

Similarity as a function of RGB histograms

The histogram similarity functions are motivated by the color assumption from Section 3.3, and are defined exactly the same as the histogram similarity functions for merging regions given in Section 2.6.1. They are defined to be the ℓ^1 norm of the difference between the normalized histograms of two regions. We give an example to illustrate the need for normalizing the histograms being compared. The following two histograms should have distance zero:

H1: [1, 2, 1]

H2: [100, 200, 100]

because corresponding bins contain the same percentage of the total number of pixels in the region. So before taking the ℓ^1 norm, we normalize the histograms by dividing each region's histogram entries by the number of pixels in the region. Each bin then contains the percentage of the region's pixels assigned to the bin.

After taking the ℓ^1 norm, we normalize the similarity function so its value will be in the range [0,1]. Because the sum over all bins in each normalized histogram is

equal to 1, the maximum value of the ℓ^1 norm between two normalized histograms is 2. This is the value of the normalizing constant.

$$d_2(R_1, R_2) = \frac{1}{2} \sum_{i=0}^{31} \left| \frac{rhist_{R_1}[i]}{numpixels_{R_1}} - \frac{rhist_{R_2}[i]}{numpixels_{R_2}} \right|$$

$$d_3(R_1, R_2) = \frac{1}{2} \sum_{i=0}^{31} \left| \frac{ghist_{R_1}[i]}{numpixels_{R_1}} - \frac{ghist_{R_2}[i]}{numpixels_{R_2}} \right|$$

$$d_4(R_1, R_2) = \frac{1}{2} \sum_{i=0}^{31} \left| \frac{bhist_{R_1}[i]}{numpixels_{R_1}} - \frac{bhist_{R_2}[i]}{numpixels_{R_2}} \right|$$

Similarity as a function of latent semantic information

The latsem similarity function is motivated by the text content assumption from Section 3.3. As for the merging distance, this similarity function is defined to be the cosine of the angle included by the latent semantic vectors of the regions being compared. We normalize this quantity to the range [0,1] by taking one minus the cosine of the angle and then dividing by two. That is,

$$d_5(R_1, R_2) = \frac{1}{2} * \left(1 - \frac{< latsem_{R_1}, latsem_{R_2} >}{\| latsem_{R_1} \| \| latsem_{R_2} \|} \right)$$

where $< \cdot, \cdot >$ is the inner product operator, and $\|\|$ is the ℓ^2 norm.

Similarity as a function of spatial variance

The spatial variance similarity functions are motivated by both the size and position assumptions from Section 3.3. They depend on the notion of the spatial variance bounding box introduced in Section 2.5.3.

We use the spatial variance bounding box to calculate two similarity functions. The first is mostly a measure of the position difference between the regions, with some dependence on size as well. The second is an attempt to capture some of the hierarchical information contained in the document tree. It measures the difference in how spread out the regions being compared are.

Spatial Variance Position Distance This is a modified version of the spatial variance similarity function from Section 2.6.1 that is used when creating the document tree. To compare two regions, we omit the $md_{svar.diff}(R_1, R_2)$ term, and we change the $md_{svar.bbox}(R_1, R_2)$ distance a little.

We defined the Spatial Variance Bounding Box (*svar.bbox*) of a region in Section 2.5.3 to be a rectangle whose center is the centroid of the region, and which extends one spatial standard deviation on each side of the centroid. That is, the corners of the *svar.bbox* are $(centroid.m \pm \sqrt{svar.m}, centroid.n \pm \sqrt{svar.n})$. The goal in using the spatial variance bounding box is to form something that captures the notion of a bounding box of a region yet remains useful if a large region is merged with a small region far from it.

For this similarity function, we define the distance between two *svar.bbox* as follows. We find the Euclidean distance between corresponding corners of $svar.bbox_{R_1}$ and $svar.bbox_{R_2}$, and add them all together. There are four such pairs of corners: upper left to upper left, upper right to upper right, lower left to lower left, and lower right to lower right. Figure 3.2 illustrates the idea behind this distance function.

Since the maximum distance between any two points when the pixel locations are in the range [0,1] is $\sqrt{2}$, and we are adding four distances, we normalize this quantity by the constant $4 \times \sqrt{2}$.

$$\begin{aligned}
\text{Let } D_{ul} &= \text{distance from upper left corner of } svar.bbox_{R_1} \\
&\quad \text{to upper left corner of } svar.bbox_{R_2} \\
\text{Let } D_{ur} &= \text{distance from upper right corner of } svar.bbox_{R_1} \\
&\quad \text{to upper right corner of } svar.bbox_{R_2} \\
\text{Let } D_{ll} &= \text{distance from lower left corner of } svar.bbox_{R_1} \\
&\quad \text{to lower left corner of } svar.bbox_{R_2} \\
\text{Let } D_{lr} &= \text{distance from lower right corner of } svar.bbox_{R_1} \\
&\quad \text{to lower right corner of } svar.bbox_{R_2}
\end{aligned}$$

$$d_6(R_1, R_2) = \frac{D_{ul} + D_{ur} + D_{ll} + D_{lr}}{(4 \times \sqrt{2}) \times (sizeratio_{R_1} + sizeratio_{R_2})}$$

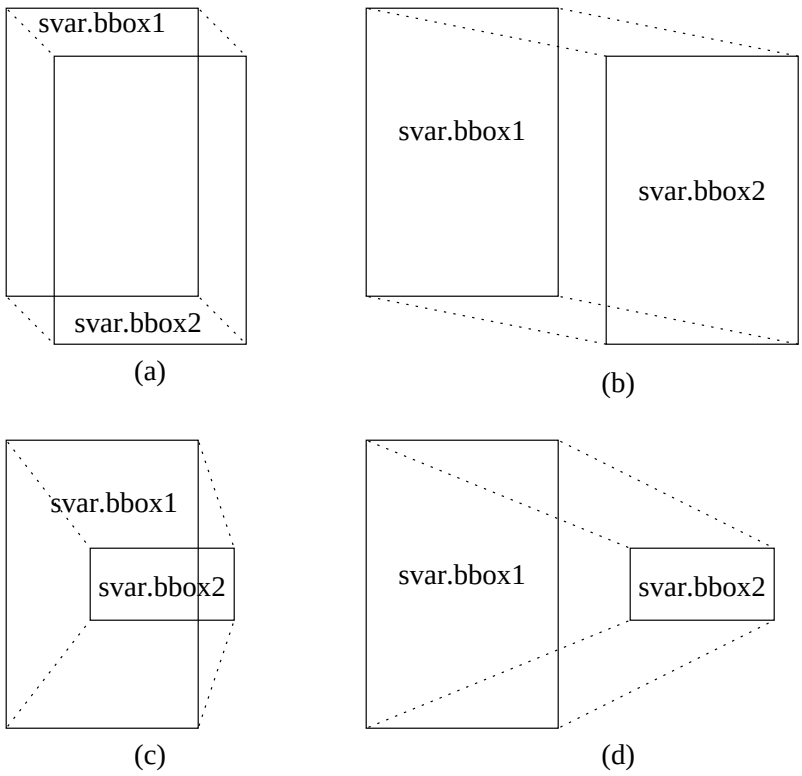


Fig. 3.2. Four descriptive cases important for understanding the spatial variance position distance. (a) Regions of similar size and position are considered close. (b) Regions of similar size and different position are considered far apart. (c) Regions of different size and same position are considered fairly close. (d) Regions of different position and different size are considered far apart.

Note that, like the corresponding similarity function in 2.6.1, we divide by the sum of the size ratios of the regions.

This similarity function encourages the regions to be both spatially close and of the same size. It is distinct from a distance between centroids because the distance between regions can only be zero under this metric if they have the same centroid and spatial variance. That is, size matters too.

Spatial variance hierarchy distance This is our only similarity function that makes use of the structure of the document tree to compare the characteristics of a node and its children to the characteristics of another node and its children. The function first calculates a "spatial variance factor" for each node being compared by assessing the similarity of the spatial variance bounding boxes of a node's children to the *svar.bbox* of the node itself. Then the distance between two regions is defined to be the absolute value of the difference between the spatial variance factors for the regions.

We denote the spatial variance factor for a region R as $sfact(R)$. We point out that two regions can have approximately the same *svar.bbox* yet have quite dissimilar appearance, as illustrated in Figure 3.3. Therefore we incorporate the hierarchical structure of the document tree, and we consider the spatial variance bounding boxes of the region R, and the left and right children of R, denoted $left(R)$ and $right(R)$, respectively.

We define $sfact(R)$ to be the total area of the *svar.bbox* of the children lying outside the *svar.bbox* of the parent region minus the total area of the *svar.bbox* of the children lying inside the *svar.bbox* of the parent region. That is,

$$\begin{aligned} sfact(R) = \quad & \text{area of } svar.bbox_{left(R)} \text{ lying outside } svar.bbox_R + \\ & \text{area of } svar.bbox_{right(R)} \text{ lying outside } svar.bbox_R - \\ & \text{area of } svar.bbox_{left(R)} \text{ lying inside } svar.bbox_R - \\ & \text{area of } svar.bbox_{right(R)} \text{ lying inside } svar.bbox_R \end{aligned}$$

Recall that pixel values are relative to the maximum dimension of the document they came from, and are in the range [0,1] independent of the resolution of their source

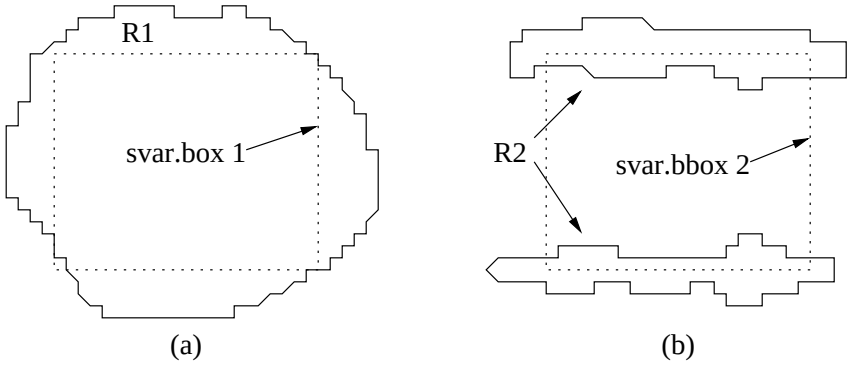


Fig. 3.3. Two regions with similar spatial variance bounding box but dissimilar appearance.

document. This means that the *sfact*(R), which depends on the areas of *svar.bbox*'s, can be a meaningful measure of distance between regions that come from two different documents. Figure 3.4 shows how this method based on *sfact*(R) is able to detect the difference between the regions from Figure 3.3. Therefore, we define the spatial variance hierarchy distance between two regions R_1 and R_2 as

$$d_7(R_1, R_2) = |sfact(R_1) - sfact(R_2)|$$

Similarity as a function of size

When comparing regions, according to the assumptions of Section 3.3, we want distance to be small when the regions are approximately the same size. We define the distance to be the absolute value of the difference in size ratio between the two regions.

$$d_8(R_1, R_2) = |sizeratio_{R_1} - sizeratio_{R_2}|$$

3.4.2 The class-dependent weights

So far, we have only made the general statement that the weights $w_i,\ i \in \{0, \ldots, 8\}$ discussed at the beginning of Section 3.4 depend on the classes of the regions. We

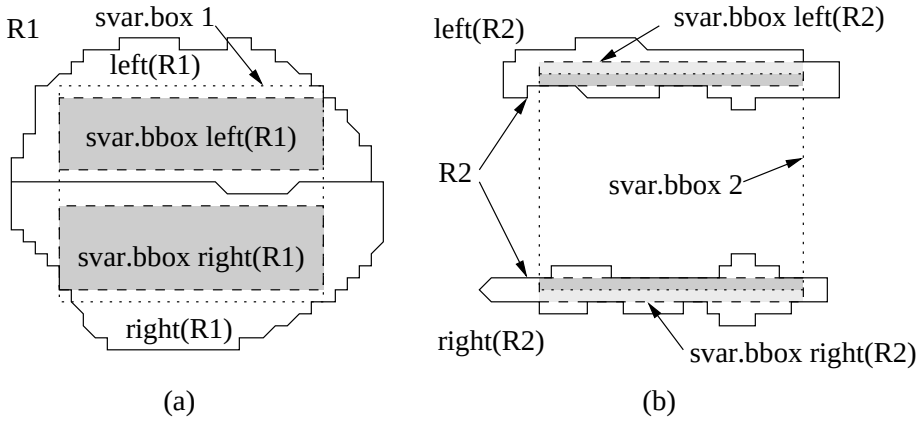


Fig. 3.4. Illustration of how the spatial variance hierarchy distance can distinguish regions with similar spatial variance bounding boxes but dissimilar spatial variance factor. The light gray areas are counted as positive area for the spatial variance factor, while the dark gray areas are counted as negative area.

have not yet stated specifically how they depend on these classes. In this section we describe these dependencies. As in Section 2.6.3, we begin by explaining for each possible class pairing which features we believe are pertinent, and what the total weight of the class pairing should be. Then we report the specific values that were used to generate the examples in this thesis. It should be noted that we picked these values essentially as a well thought out guess based on our notions of how each class pairing should be treated. We make no claim regarding the optimality of the weights that were used. We do, however, show that using the chosen weights produces results that seem to make sense.

We now describe the notions behind the weights assigned for each class pairing. We abbreviate the four classes as follows: TH is header text, TB is body text, I is image, and BG is background. For the remainder of this section, let c_1 and c_2 be the classes of regions R_1 and R_2, respectively.

We now explain the $w_i,\ i \in \{0, \ldots, 8\}$ assignments. Because the similarity functions $d_i(R_1, R_2)$, $i \in \{0, \ldots, 8\}$ are normalized to be approximately in the range [0,1], we consider the sum over i of $w_i(c_1, c_2)$ to be a reasonable measurement of the total strength of the weighting given to a particular class pairing. Therefore, we first

choose the relative weighting of each class pairing as compared to the other pairings by setting $\sum_{i=0}^{8} w_i(c_1, c_2)$ for each (c_1, c_2) pair. Then we choose for each class pair (c_1, c_2) how much of the total class pairing weight should be assigned to each similarity function.

Class pairing weights

We set the class pairing weights, $\sum_{i=0}^{8} w_i(c_1, c_2)$, equal to a number in the range [1.0,10.0]. A small class pairing weight corresponds to the class pair being more similar, because it leads to a smaller distance between the regions.

We assign the class pairing weights as follows. We begin with class pairings which contain a body text (TB) region. We assume that the most similar class to TB is TB, so we set

$$\sum_{i=0}^{8} w_i(TB, TB) = 1.0$$

We assume that the next most similar class to TB is TH, and that since header text is also text, the weight should be close to that for the (TB,TB) class pair. Therefore we set

$$\sum_{i=0}^{8} w_i(TB, TH) = 2.0$$

We assume that the third most similar class to TB is I, and we set

$$\sum_{i=0}^{8} w_i(TB, I) = 5.0$$

Finally, because we want all the "content" regions to be considered more similar to each other than to the background of the document, we assume that TB and BG are very dissimilar. Therefore we set

$$\sum_{i=0}^{8} w_i(TB, BG) = 10.0$$

Next we consider class pairings which contain an image region. We assume that the most likely class to merge with I is I, and we set

$$\sum_{i=0}^{8} w_i(I, I) = 1.0$$

The weight of the (I,TB) pair is set in the previous paragraph to 5.0. We assume that TB is more similar to I than TH, because body text can be a caption. Therefore we set

$$\sum_{i=0}^{8} w_i(I, TH) = 7.0$$

For the same reason as for the (TB,BG) pair, we assume that the (I,BG) pair should be considered dissimilar, and we set

$$\sum_{i=0}^{8} w_i(I, BG) = 10.0$$

We next consider class pairings which contain a header text region. We assume that the most likely class to merge with TH is TH, so we set

$$\sum_{i=0}^{8} w_i(TH, TH) = 1.0$$

The (TH,TB) pair weight is set above to 2.0, and the (TH,I) pair weight is set above to 7.0. For the same reason as for the (TB,BG) and (I,BG) pairs, we set

$$\sum_{i=0}^{8} w_i(TH, BG) = 10.0$$

The only remaining undefined class pair weight is that for a (BG,BG) pair. We assume that two background regions are very similar, and we set

$$\sum_{i=0}^{8} w_i(BG, BG) = 1.0$$

Similarity function weights

The individual $w_i(c_1, c_2)$ assignments are given in Table 3.4.2. The following paragraphs contain the assumptions that are used to generate these weights. The latent semantic information similarity function, $d_5(R_1, R_2)$, is not yet implemented, so currently $w_5(c_1, c_2) = 0$ for all (c_1, c_2) pairs. Therefore only two types of similarity functions are available: (1) position and size functions—the sizeratio, svar position, and svar hierarchy distances, and (2) color content functions—the rgbmean, rgbvar, rhist, ghist, and bhist distances.

Table 3.1
Table of weights for the tree comparison region distance function.

		TB	I	BG	TH
TB	w_0:	0	0	0	0
	w_1:	0	0	0	0
	w_2:	0	0	0	0
	w_3:	0	0	0	0
	w_4:	0	0	0	0
	w_5:	0	0	0	0
	w_6:	0.25	4.0	6	0.5
	w_7:	0.25	0.5	2	0.5
	w_8:	0.5	0.5	2	1.0
I	w_0:		1/24	2	0
	w_1:		3/24	0	0
	w_2:		4/24	0	0
	w_3:		4/24	0	0
	w_4:		4/24	0	0
	w_5:		0	0	0
	w_6:		2/24	6	5
	w_7:		4/24	1	1
	w_8:		2/24	1	1
BG	w_0:			0.25	0
	w_1:			0	0
	w_2:			0	0
	w_3:			0	0
	w_4:			0	0
	w_5:			0	0
	w_6:			0.25	6
	w_7:			0.25	2
	w_8:			0.25	2
TH	w_0:				0
	w_1:				0
	w_2:				0
	w_3:				0
	w_4:				0
	w_5:				0
	w_6:				1/3
	w_7:				1/3
	w_8:				1/3

Classes

TB	—	body text
TH	—	header text
I	—	image
BG	—	background

Weights correspondence

w_0	=	rgbmean
w_1	=	rgbvar
w_2	=	rhist
w_3	=	ghist
w_4	=	bhist
w_5	=	latsem
w_6	=	svar position
w_7	=	svar hierarchy
w_8	=	size-ratio

We first give our assumptions for region pairs containing a body text region. For (TB,TB) and (TB,TH) pairs, we assume that the position is much more important than the color content of the text regions, so we assign all the weight to the position and size functions. For (TB,I) pairs, we assume that the color information of the text region is irrelevant, so we assign all the weight to the position and size functions. We also assume that the most important of these functions is the svar position function, so we assign it the most weight. For (TB,BG) pairs, we assume that the position and size of a text region is much more important than the color content of the text region, so we only take into account the position and size similarity functions.

Next we give our assumptions for the remaining region pairs containing an image region. For (I,I) pairs, we assume that both the position and color features are important. We assign equal weight to the histograms, and we assign the largest weight to the histogram and svar hierarchy similarity functions. For (I,TH) pairs, we use the same assumption as for (TB,I) pairs that text region color is irrelevant, and we assign all the weight to the position and size functions. We also use roughly the same proportions of the weight for each of these functions as those used for (TB,I) pairs. For (I,BG) pairs, we assume that the color information is only slightly relevant, so we assign most of the weight to the svar position function and only a little weight to the rgbmean, svar hierarchy, and size-ratio functions.

For the remaining region pairs, we assume the following. For (TH,TH) pairs, we assume that the weighting should be quite similar to that for (TB,TB) pairs. We simply weight all the position and size functions equally instead of preferring the size-ratio function slightly, as is done for (TB,TB) pairs. For (TH,BG) pairs, we assume that the weighting should be the same as for (TB,BG) pairs. The last set of pair weights is that for the (BG,BG) pair. Since a document can have more than one background region, and they could be different colors (for example, a section of gray and a section of white), we assume that both color and position are important. We split the weight evenly among the rgbmean function, and the position and size functions.

4. EXAMPLES

In Sections 2 and 3, we have described a general strategy for forming and comparing hierarchical document descriptions called document trees, and we have defined one example of how this strategy can be implemented. This section describes some results of applying this specific strategy to some real document images.

We began with two multipage documents in pdf format. Since the system only describes and compares single page documents, we converted each page of these pdf files to a 400 dpi (157.5 dpcm) color image. Because we wanted to use the TSMAP parameters for scanned images, we then printed the pages on a color printer and scanned the pages at 400 dpi (157.5 dpcm). These scanned images of the pages were the input to our system.

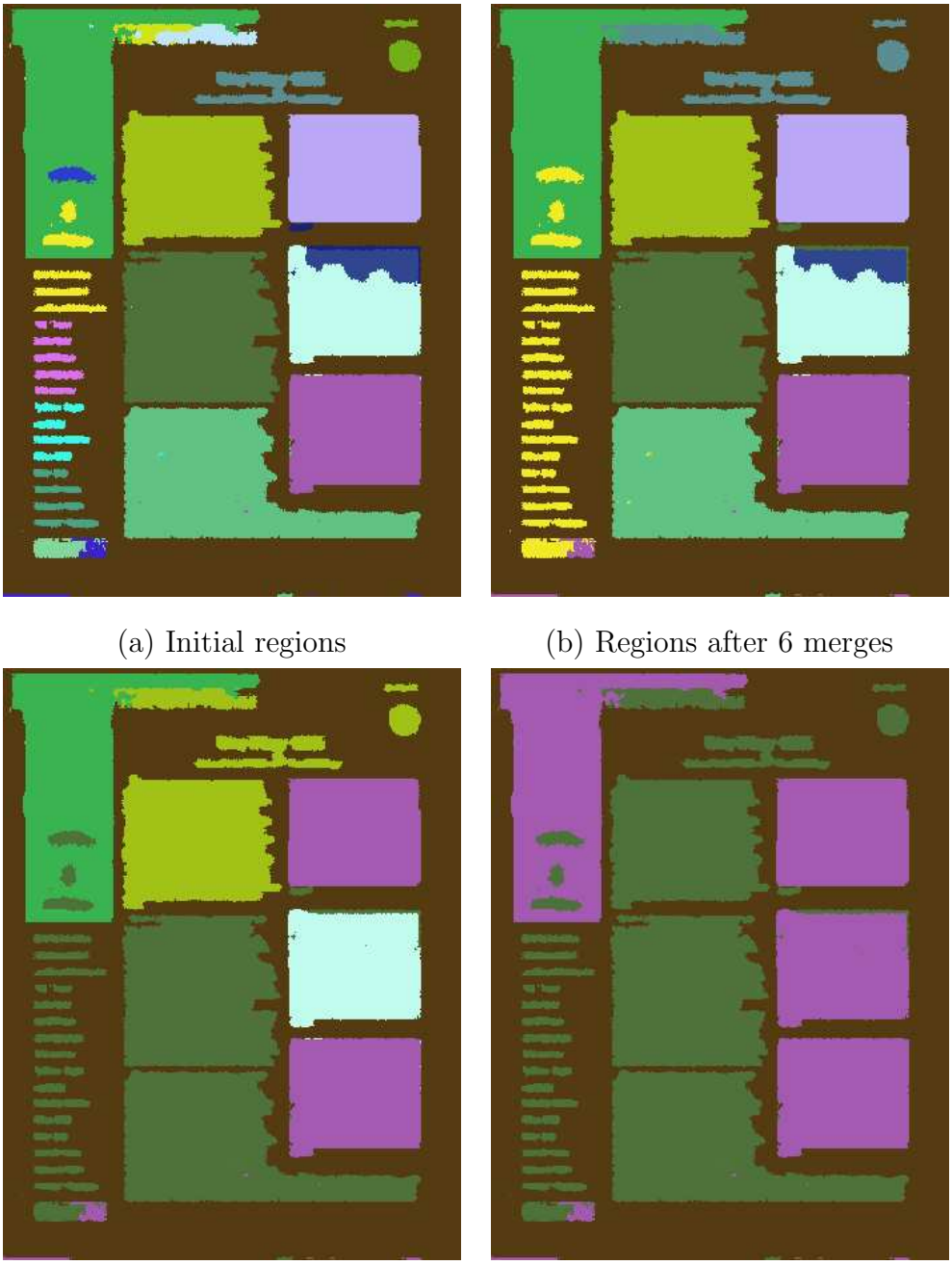
We then formed a document tree to describe each of these pages. The first step of this process is to segment the document into regions. The scanned images were first put through the segmentation procedure whose block diagram appears in Figure 2.2. The output at three stages of this procedure is shown in Figure 4.2. Figure 4.2(a) shows the 400 dpi (157.5 dpcm) color image input to the segmentation procedure. The preprocessing step converted the document to the 100dpi (39.37 dpcm) grayscale image shown in Figure 4.2(b). The TSMAP program, using the parameters obtained by training on scanned images, produced a 50 dpi (19.69 dpcm) color segmentation image, and the postprocessing step replaced all singleton pixels with a majority vote among their eight point neighborhood, yielding the segmentation of Figure 4.2(c). In this segmentation image, white corresponds to pixels classified as header text, red corresponds to body text regions, green corresponds to image regions, and blue corresponds to background regions.

Once we had a segmentation of the original image, we found the connected components of the segmentation to define the regions of the document and recursively

merged these regions to form the document tree. Figure 4.1 illustrates some snapshots of this process. For this figure, each region was initially assigned a random color. When two regions were merged, the smaller region was assigned the color of the larger region, forming an aggregate region of uniform color.

Figure 4.3 shows the top six levels of the resulting document tree. For this figure, the color of each node in the tree is the mean rgb value of the region of the original document to which the node corresponds. The numbers inside the nodes of the tree are arbitrary region numbers assigned by our connected components program. For a description of how a document tree corresponds to the region merging process, see Section 2.7.

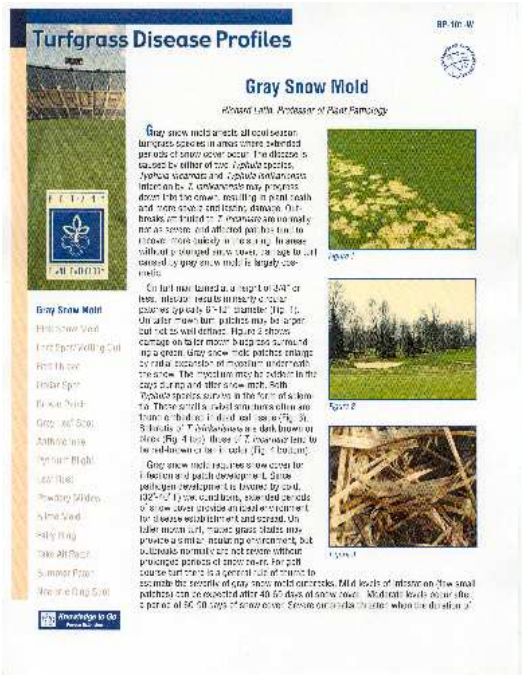
Once we had formed the document tree for each document, we chose a document and calculated the distance between its document tree and all the other document trees to see if the distance function induced a sensible ordering of the documents. The resulting distances are shown in Table 4.1. Figure 4.4 shows the comparison document and the closest document to it, along with the document tree that describes each document. Figure 4.5 shows the next four closest documents to the comparison document, and Figure 4.6 shows the three furthest documents.



(a) Initial regions (b) Regions after 6 merges

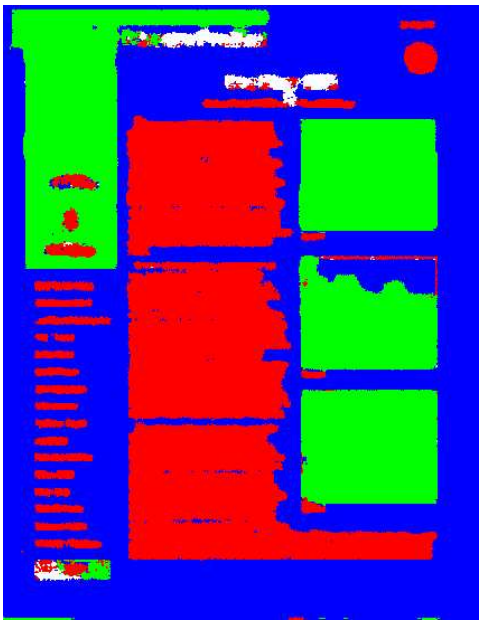
(c) Regions after 11 merges (d) Regions after 16 merges

Fig. 4.1. Several snapshots from the region-merging process which creates the document tree for the document BP-101-W-1.



(a) Original 400 dpi (157.5 dpcm) color image

(b) 100 dpi (39.37 dpcm) grayscale image



(b) 50 dpi (19.69 dpcm) segmentation image

Fig. 4.2. Several snapshots from the document segmentation process for the document BP-101-W-1.

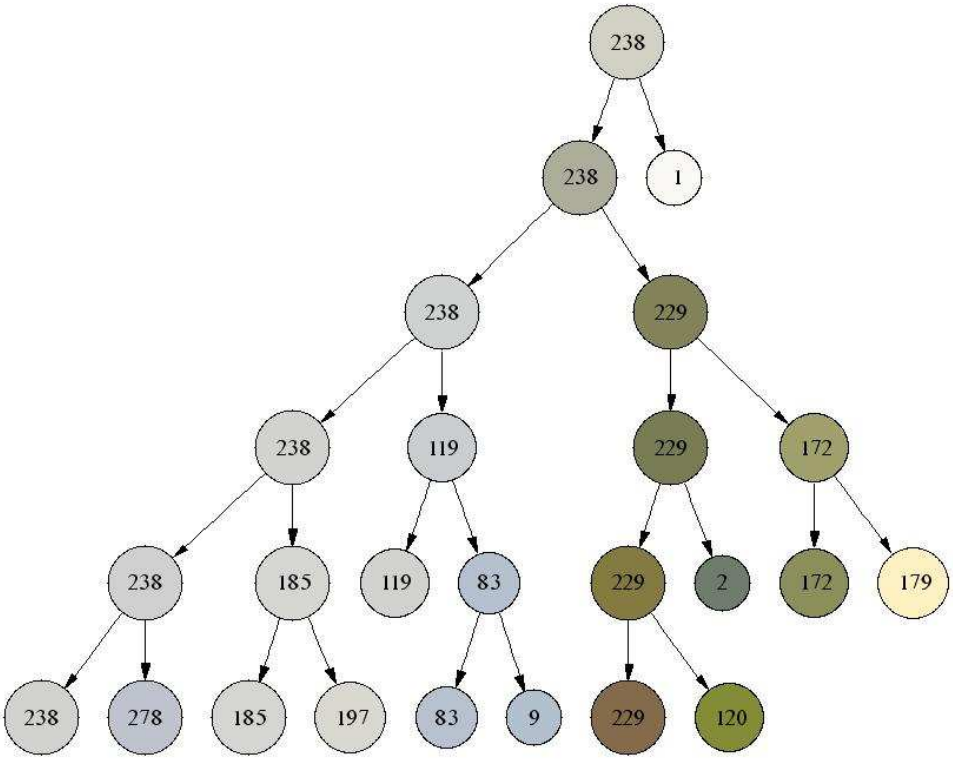
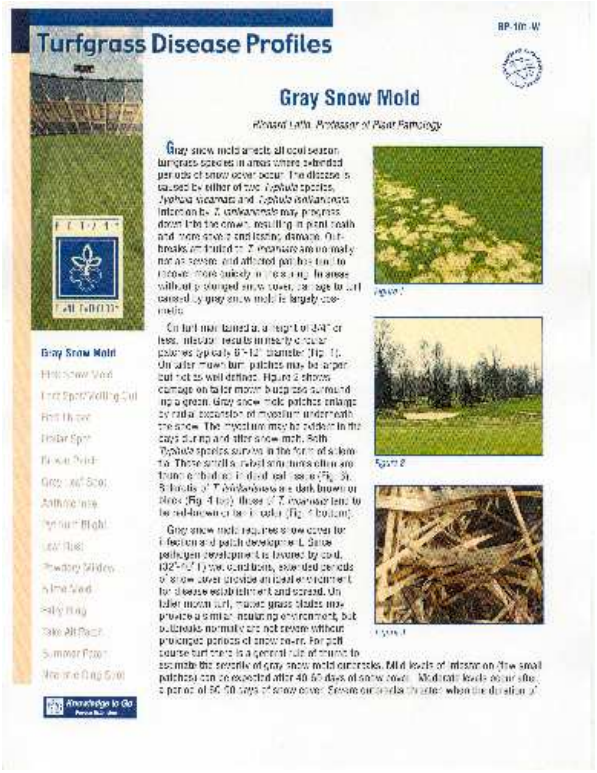
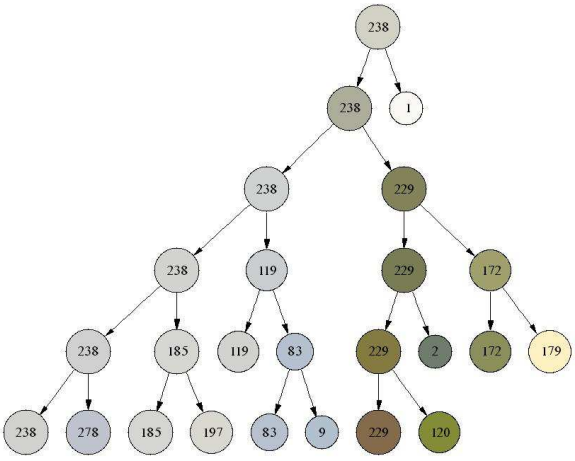


Fig. 4.3. The document tree for the document BP-101-W-1.

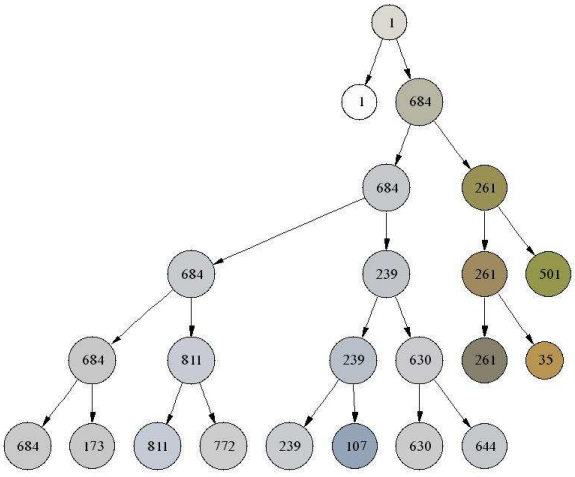


BP-101-W-1

BP-101-W-2



tree for BP-101-W-1



tree for BP-101-W-2

Fig. 4.4. Our system classified document BP-101-W-2 as the most similar to the document BP-101-W-1, among eight document images. The bottom row shows the document trees for the two images.

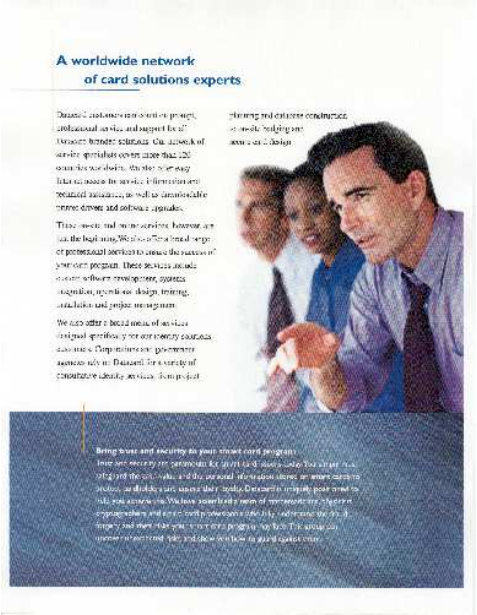
(3) CorpBroch-5

(4) CorpBroch-3

(5) CorpBroch-4

(6) CorpBroch-6

Fig. 4.5. The next four closest documents to BP-101-W-1



(7) CorpBroch-7

(8) CorpBroch-8

(9) CorpBroch-1

Fig. 4.6. The three documents furthest from BP-101-W-1.

Table 4.1
Distances from comparison document to all other pages

Distances from BP-101-W-1

Rank	Distance	Document
(1)	0.00000	BP-101-W-1
(2)	0.01729	BP-101-W-2
(3)	0.02485	CorpBroch-5
(4)	0.02572	CorpBroch-3
(5)	0.03459	CorpBroch-4
(6)	0.04994	CorpBroch-6
(7)	0.06026	CorpBroch-7
(8)	0.06045	CorpBroch-8
(9)	0.13990	CorpBroch-1

5. CONCLUSION

We have proposed a general framework for capturing and comparing the hierarchical structure of document images. At this early stage of implementation, the method produces results that make sense intuitively, at least for the small example presented in this thesis. Because of this, we believe that the method is worth further refinement. Our current work consists of adding the text content feature (latent semantic information) to the implementation and testing the implementation on a larger database of document images.

Several opportunities for further work exist. We begin with improvements at the detail level and move toward improvements at the general level. Our first suggestion deals with the method of assigning the weights used in calculating the value of the two distance functions. The weights are currently assigned by way of educated guessing, so some method could be found to generate an optimal weight assignment.

At a slightly more general level, if we keep the assumptions from Section 2.4 and Section 3.3 about what makes regions similar, several improvements are available. First, we could evaluate the discriminative power of each of the similarity functions we compute. Some of them may be unnecessary. Second, we could create new similarity functions defined on the existing features to implement the assumptions more effectively. Third, we could define additional features with corresponding similarity functions to implement the assumptions more effectively. And fourth, we could investigate other, more standard, methods to calculate the distance between two document trees.

We close with two suggestions at the most general level. First, we could investigate how humans perceive information in documents so we could make better assumptions, resulting in different tree formation and tree comparison strategies. Second, we could investigate other methods of forming a tree to describe the segmented document.

For instance, we could use a probabilistic grammar to describe the document image. Kanungo and Mao [7] have demonstrated that regular grammars can describe a document image quite well for the task of segmentation. We propose that the powerful 2-D probabilistic context-free grammar model along with the center-surround algorithm for performing computations with the model from [11] might be a very good method to investigate further for this task. Of course, using a different document tree generating method implies the need to re-evaluate the tree comparison method as well.

LIST OF REFERENCES

LIST OF REFERENCES

[1] H. Cheng and C. A. Bouman, "Multiscale Bayesian segmentation using a trainable context model," *IEEE Trans. on Image Processing*, vol. 10, no. 4, pp. 511 – 525, April 2001.

[2] Jianying Hu, Ramanujan Kashi, and Gordon Wilfong, "Comparison and classification of documents based on layout similarity," *Information Retrieval*, vol. 2, pp. 227 – 243, May 2000.

[3] S. Deerwester, S. T. Dumais, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society For Information Science*, vol. 41, no. 6, pp. 391 – 407, 1990.

[4] Liping Ma, John Shepherd, and Anh Nguyen, "Document classification via structure synopses," in *Proceedings of the Fourteenth Australasian Database Conference on Database Technologies*, Xiaofang Zhou and Klaus-Dieter Schewe, Eds., Adelaide, Australia, 2003, vol. 17.

[5] James W. Cooper, Anni R. Coden, and Eric W. Brown, "Information retrieval models: Detecting similar documents using salient terms," in *Proceedings of the Eleventh International Conference on Information and Knowledge Management*. ACM, November 2002, pp. 245 – 251.

[6] Rohini K. Srihari, Zhongfei Zhang, and Aibing Rao, "Intelligent indexing and semantic retrieval of multimodal documents," *Information Retrieval*, vol. 2, pp. 245 – 275, May 2000.

[7] Tapas Kanungo and Song Mao, "Stochastic language models for style-directed layout analysis of document images," *IEEE Trans. on Image Processing*, vol. 12, no. 5, pp. 583 – 596, May 2003.

[8] Jean-Michel Morel and Sergio Solimini, *Variational Methods in Image Segmentation*, vol. 14 of *Progress in Nonlinear Differential Equations and Their Applications*, Birkhäuser, Boston, 1995.

[9] G. Qiu and S. Sudirman, "Color image coding, indexing and retrieval using binary space partitioning tree," in *Proc. of IEEE Int'l Conf. on Image Proc.*, October 7-10 2001, vol. 1, pp. 682 – 685.

[10] C. A. Bouman and M. Shapiro, "A multiscale random field model for Bayesian image segmentation," *IEEE Trans. on Image Processing*, vol. 3, no. 2, pp. 162 – 177, March 1994.

[11] I. Pollak, J. M. Siskind, M. Harper, and C. A. Bouman, "Spatial random trees and the center-surround algorithm," Tech. Rep. TR-ECE-03-03, Purdue University, School of Electrical and Computer Engineering, 2003, Submitted to *IEEE Trans. on Information Theory*.