QUANTITATIVE EVALUATION OF FEATURE SETS, SEGMENTATION ALGORITHMS AND COLOR CONSTANCY ALGORITHMS USING WORD PREDICTION

by

Prasad Gabbur

A Thesis Submitted to the Faculty of the

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

In Partial Fulfillment of the Requirements
For the Degree of

MASTER OF SCIENCE

In the Graduate College

THE UNIVERSITY OF ARIZONA

2 0 0 3

ACKNOWLEDGEMENTS

I take this opportunity to gratefully acknowledge all those people who have been my support during this work and also during my entire stay. Firstly I would like to thank Dr. Kobus Barnard for his supervision and without whom this work would not have been possible. The enthusiasm shown by him towards my work has always been inspiring and has been the driving factor for being able to complete this thesis. I would really like to thank him for his patience and time to guide me at every step of this work and provide me with moral and financial support.

My special thanks to Dr. Malur K. Sundareshan without whose help I would not have been able to come to this institution to pursue my higher education. He has been the person whom I have looked for during good and bad times. His advice has kept me on the right path throughout and made me complete things on time. It were his words that taught me the essence of research. As told by him once, "Research may be sometimes disappointing but the results out of it are rewarding", is something that I will always remember.

I am grateful to Dr. Robin N. Strickland for his excellent course in Advanced digital signal processing and some of the fundamentals learnt in that course have helped in the pursuit of this work. I thank him for having agreed to be on my thesis committee on a short notice and taking time to review this work. I am indebted to him for his suggestions and helping me out with financial support when it was most needed.

Many thanks to the faculty of the ECE department for providing me with a great learning experience through their excellent courses. I would like to thank the staff of both the Electrical and Computer Engineering and Computer Science departments for their help on a number of occasions.

My gratitude to my parents and sisters for being a constant source of encouragement and putting up with me during this period. I would also like to thank my relatives for their moral support. I have no words to describe the support offered to me by my friends especially Nikhil V. Shirahatti, and Ananth Kini. My deepest word of thanks to my grandmother, Ms. Shantabai Patil, who has always been my mentor and to whom I would like to dedicate this thesis.

TABLE OF CONTENTS

TABLE OF CONTENTS - *continued*

TABLE OF CONTENTS - *continued*

TABLE OF CONTENTS - *continued*

LIST OF FIGURES

LIST OF FIGURES - *continued*

LIST OF TABLES

ABSTRACT

Recent research in the field of multimedia indexing and retrieval has tried to exploit the semantic information carried by keywords attached to images. Given a large annotated database of images, a joint distribution between the visual and semantic descriptions of scenes can be obtained. This can be used to annotate a new image with the most probable words conditioned on its visual description. The process of predicting words for new images is called "auto-annotation" and it has links to general object recognition. Also it is possible to evaluate the accuracy of word prediction on a large scale due to the availability of large annotated databases. In this thesis, an approach to model the joint distribution between visual and semantic descriptions of scenes is discussed. This model is used to evaluate a few low-level computer vision algorithms. Specifically different feature sets, segmentation algorithms and color constancy algorithms are evaluated quantitatively using the word prediction tool. The annotation accuracy is the quantitative measure. Further, modifications are proposed to a segmentation algorithm called Normalized Cuts to achieve better grouping of regions in images to aid the process of auto-annotation. The effects of illumination color change on object recognition are studied using the joint image-word model. Different strategies to deal with illumination change in an object recognition framework are evaluated using the annotation measure. Results suggest that color and texture are the most important features in this model. The performance of segmentation algorithms is a function of number of regions used for annotation. All the strategies used to compensate for illumination change are helpful.

Chapter 1

INTRODUCTION

Recent research in the field of multimedia indexing and retrieval has tried to exploit the semantic information carried by keywords attached to images. There exist huge databases of images that come with words describing the context of each image. The semantic information carried by the words associated with images can be very helpful in organizing and indexing the data. Since these words describe the content of the images−individual objects or their characteristics−there exists a correlation between them and the visual features computed from the images. Some of these links can be extracted with the help of image analysis, natural language processing and machine learning techniques applied to such annotated image datasets. Visual and semantic descriptions that tend to co-occur frequently imply a strong connection between each other. Given a huge annotated image database that contains sufficient repetitions of these co-occurences, it is possible to learn which visual and semantic descriptions are strongly connected. An annotated database can be viewed as a collection of a number of such connected entities where each entity possibly describes a concept. For example, repetition of the entity—blue untextured region (visual) and the word "sky" (semantic)—in a number of images in the database could imply the concept of a sky. Assuming that a finite number of such concepts exist and that their visual and semantic descriptors may be affected by noise, clustering techniques can be used to recognize the concepts. Either hard or soft clustering techniques can be used. The clustering process is nothing but an organization or indexing of the concepts in the dataset.

Taking this approach to organize annotated datasets, soft clustering techniques introduced by Barnard et. al. [1, 5, 49] have also led to statistical models that can link images to words. The approach is to model the joint statistics of image regions and words probabilistically. The image regions are obtained using a segmentation technique and a set of features is extracted from these segments. These features form a visual description of the image regions and are used in learning the relationship between them and the words. Once a joint probability model is available, a number of applications are possible. One of these applications is a more meaningful organization of annotated databases where clusters are found based on both visual and linguistic descriptions [2] of scenes. Another important application is in generating words for images automatically, called "auto-annotation". The process of generating words for images that are not used during training has clear ties to object recognition. This is because the predicted words carry semantic information about the scene described by the image and hence word-prediction can be viewed as a process of translating from visual to linguistic description. It is for this reason, and also due to the similarity of modeling approach to methods in statistical machine translation [3, 4], that the visual-semantic model is called a *translation model for object recognition* [5]. A few other systems that use visual and text information for image database organization and querying include the Blobworld [55] and Webseer [56]. Blobworld uses keywords in conjunction with image features for querying by narrowing down only on those images that contain these keywords during search. Webseer also uses a similar approach for image query on the web in that it obtains visual information from images by classifying them as photographs or artificial images and also by analyzing them using a face finder. Cascia et. al. [57] combine text with color and orientation histograms of images to exploit the two modalities in a web image database. The work of Srihari et. al. [58, 59] uses text information associated with photographs for scene understanding. None of these systems explicitly learn direct relationships between text and visual components of a scene. The approach of the model used in this work is to recognize links between visual components (segments) and words using feature-word co-occurrence data. This makes it possible to predict words for images of new scenes (not used in training) and thereby recognizing objects in them. From a browsing viewpoint, this model allows querying and retrieval with queries formed by just images, words or a combination of both.

1.1 Annotated database of images

A number of annotated image databases are available; examples include online museum data, stock photo collections such as the Corel image dataset, and web images with captions. For this work, the Corel dataset is used. The Corel database we use has 392 directories of images with each directory containing 100 images on one specific topic such as “aircraft”. Each image is annotated with a set of keywords that pertain to the content of the scene depicted by the image. A few examples of annotated images from the dataset are shown in Fig. 1.1.

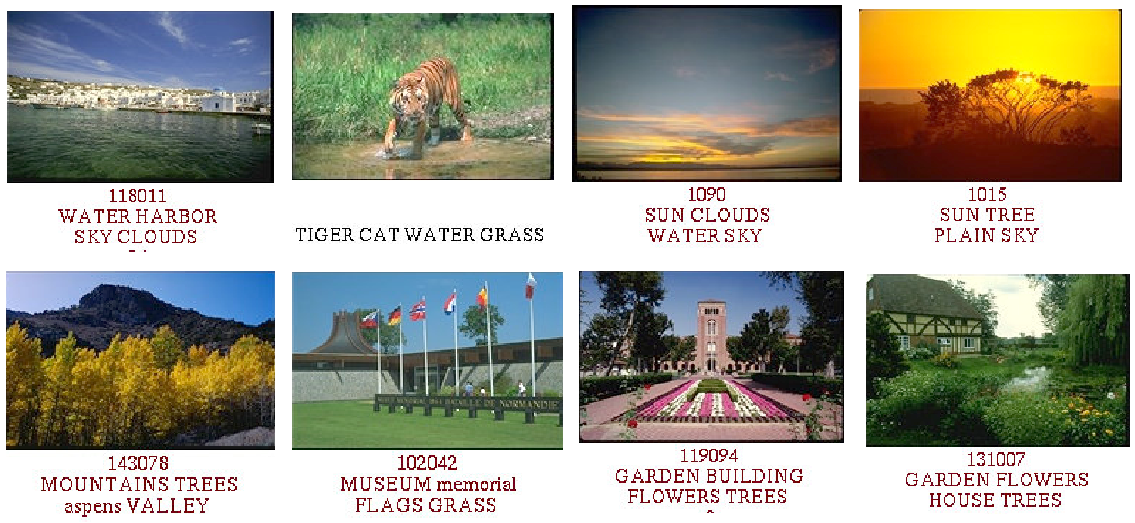


Fig. 1.1. *Annotated database*: *Example images from the Corel dataset along with their annotations.*

Notice that each image is accompanied by a set of words that together describe the content of the image. However, there is no information as to which keyword goes with which component of the scene. This has to be learnt as part of the modeling procedure from the co-occurrence of visual and linguistic data.

1.2 Image preprocessing

To obtain a visual representation, each image is processed by a segmentation algorithm to partition it into distinct regions. A number of segmentation algorithms are available that aim at splitting up an image into coherent regions, such as Blobworld [6], Normalized Cuts [7], and Mean Shift [8]. Part of this work (Chapter 3) is to evaluate some of these methods using word prediction performance. Once a segmentation is available, a set of visual features is extracted from each of the regions. These features can be broadly classified into size, position, color, texture, and shape features. A detailed description of the features used is given in Chapter 2. It remains an open question as to which set of features is more suitable for this task and an attempt to answer this question by evaluating different feature sets on word prediction performance also forms a part of Chapter 2. The purpose of feature extraction is to obtain a visual description of image segments using a set of numbers so that the joint probability distribution between these numbers and words can be estimated. Note that the set of numbers representing a region is also referred to as a *blob* in the text.

1.3 Joint modeling of image regions and words

A number of models have been considered for the purpose of modeling joint distribution between image regions and words [1, 5, 49]. Different models are aimed at different applications. For all experiments conducted in this thesis a particular model suitable for the object recognition task is used. For completeness, a brief description of the model is provided here and for further details the reader is referred to [1]. Image items (regions and words) are assumed to be generated by a statistical process, with words and regions considered analogously. Let D (blob features and words) be the set of observations associated with a document or image. The probabilistic model assumes that these observations ($D = \{(w, b)\}$ where w denotes a word and b denotes a blob feature vector) are generated from a set of nodes. M such nodes can be visualized as in Fig. 1.2.

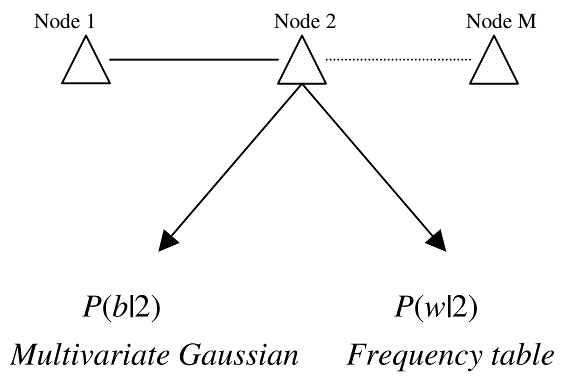


Fig. 1.2. *Visual-semantic model*: *Generative model for the joint distribution of image regions and words.*

The basic assumption in the generative model is that each node generates certain blobs and words together with high probability. In other words, each node is responsible for generation of entities together (blobs and words) that pertain to some concepts. For example, if a node is responsible for generating “zebra” concept, then this node will have a high joint probability over black-and-white stripy blobs and the word “zebra”. For generation of a blob and word in any image, the joint distributions of this blob-word pair are summed over all nodes. Therefore the joint probability of a blob-word pair is given by:

$$P(w,b) = \sum_{l} P(w,b \mid l)P(l) \tag{1.1}$$

where l is an index over nodes. Furthermore, it is assumed that a word and a blob are conditionally independent given a node. Hence,

$$P(w,b) = \sum_{l} P(w \mid l) P(b \mid l) P(l) \tag{1.2}$$

$P(b|l)$ is assumed to be Gaussian over the feature space with a diagonal covariance matrix and $P(w|l)$ is a table of probabilities. Estimating the parameters of the model given the dataset is a missing-data problem where the missing data is which node generated a particular blob and/or word. Parameters are estimated using the Expectation Maximization algorithm [9] by maximizing an objective function proportional to the likelihood of the dataset. The likelihood of the dataset is the probability of generating all the documents (the set of blobs and words of an image is referred to as a document) in the dataset assuming each document is independently generated. In the likelihood function however, $P(W,B)$, where W is the set of all words and B is the set of all blobs in a document, is considered for each document and the objective function maximizes this taking all training documents into consideration. This is because there is no information in the database as to which blob b and which word w are tied together in a document. More details regarding the method of training and testing the system can be found in [1].

Once the parameters of the model are determined, it can be used to predict words for images. The generalization ability of the model is measured by how well it can predict words for blobs in those images that are not used in training. This is also indicative of its object recognition performance viewed as machine translation. With the modeling scheme as described above, it is possible to obtain a posterior distribution over words given a blob as:

$$P(w \mid b) = \frac{P(w,b)}{P(b)}$$

$$= \frac{\sum_{l} P(w \mid l) P(b \mid l) P(l)}{P(b)} \tag{1.3}$$

The model is used to calculate the probability of generating each word in the vocabulary given a blob, using the above equation. To label each region in the image with the most probable word it can predict, only *P(w/b)* is used for that blob. Roughly speaking, each segment of an input image is annotated with the most probable word that it can co-occur with according to the trained model. Thus we have an image that is broken down into regions and each region is annotated with the most probable word it can predict according to the co-occurrence model (see Fig. 1.3). For generating words for an image, the probability distributions are added up for N largest blobs in the image. Then the image is annotated with the most probable words in the distribution so obtained.

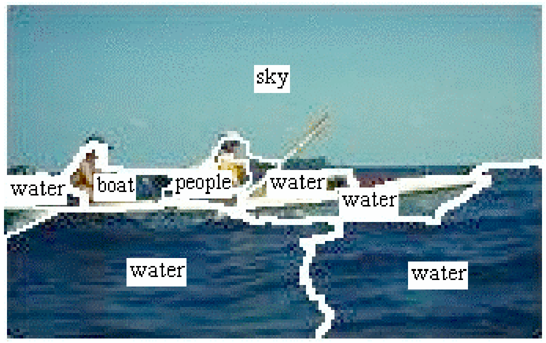


Fig. 1.3. *Region-labeling*: *A segmented image with each region labeled with the most probable word given the model.*

1.4 Evaluating recognition performance

A natural way to evaluate recognition performance would be to predict a word for each region in an image and score the prediction positively if the word is relevant for the region. This is not feasible on a large scale since such segmented and labeled databases are not available. Generation of such databases would require hand labeling of image regions and doing this on a large scale is impractical. However, it is easy to measure the annotation performance on a large scale. This can be done by predicting M words for a test image where M is the number of keywords supplied for that image. The predicted words can be compared with the keywords provided. A number of methods to evaluate annotation performance are given in [1]. For the experiments here, annotation measure is used as a proxy for recognition performance. This is not a perfect measure since the annotation process produces words for an image as a whole. It does not directly suggest correspondence between words and image segments. However, good annotation performance implies that the system is capable of recognizing contents of the scene.

The process of generating words for an image utilizes information from individual blobs of the image. This is because the final distribution from which words are chosen to annotate the image is obtained by summing up the word distributions given by the blobs. So, doing well on annotation would require the system to learn useful relationships between blobs and words. Therefore it makes sense to use annotation performance as a proxy for recognition performance. The approach here is to predict M words for a test image if this test image is provided with M actual keywords. The ratio of the number of correct predictions to M is the absolute score. This is further converted into a relative score as follows. A baseline word prediction accuracy is obtained by annotating a test image with M words as before but using an empirical word distribution. The empirical word distribution is obtained by calculating the frequencies of the words as they occur in the training database. The relative score is the absolute score minus the score obtained using the empirical distribution. This reduces variance due to varied test sample difficulty. A positive value for the relative score implies that the system has learnt non-trivial information, enabling it to achieve higher annotation accuracy than what is possible with a more obvious strategy of annotating the images using the empirical distribution. The higher the relative score, the better the performance. Further, the nature of the Corel dataset makes the relative score a sensible performance measure. The actual image annotations typically contain several common words like “sky”, “water”, “people” and fewer less common words like “tiger”. Annotating all the images with words like “sky”, “water and “people” (as implied by the empirical distribution) will usually lead to reasonable accuracy. Therefore relative performance is indicative of the ability to recognize more specific concepts in the images.

1.5 Experimental protocol

Each directory (CD) of the Corel dataset is composed of images pertaining to a specific topic. Hence training with a certain set of directories may bias the model towards good word prediction for images describing similar concepts to those in the training images. Also the model performance is dependent on the initialization point of the EM algorithm as it is inherently a local maxima-based optimization technique. To compensate for these, a systematic sampling scheme is adopted.

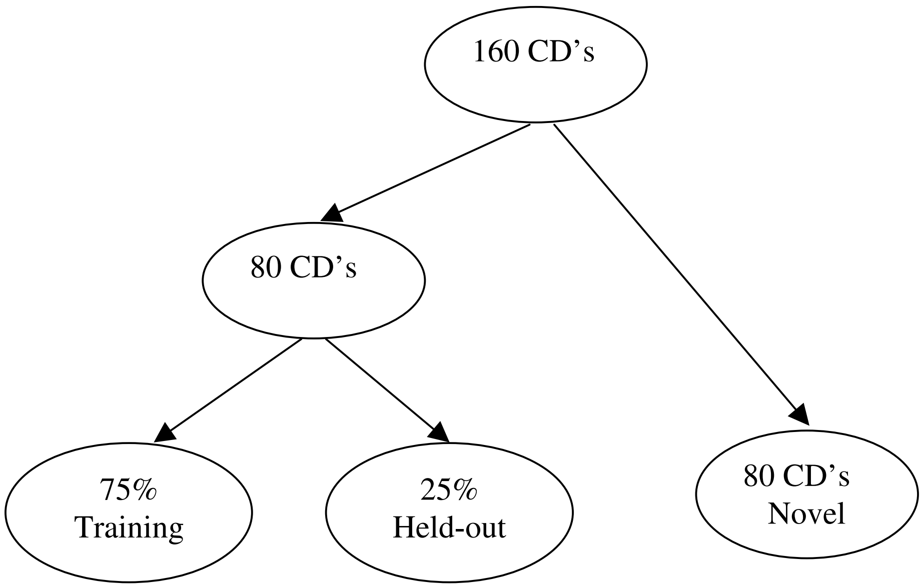


Fig. 1.4. *Sampling scheme*: *Illustration of the sampling scheme to obtain training and test datasets.*

The entire Corel dataset is divided into two parts. 75% of the images in each directory in the first half are used for training and the remaining 25% are used as a held-out test set. The images in the other half form a novel held-out test set. Predicting words for the images in the novel set is difficult since these are most likely composed of concepts not depicted by the training images. Predicting words for the novel set even with a reasonable accuracy is a good indicator of generalization ability. Also, the performance on the novel images is more relevant to object recognition because the system needs to learn about concepts, rather than memorize instances present in the training images, to do well on novel images. 10 such random samplings are done and the results are averaged over the samplings. Each sampling renders a different set of training and test images. If in a sampling the test images are similar in concept to training images this will bias the system towards higher word prediction accuracy. On the other hand if the training and test images are comprised of totally different concepts this will have a tendency to lower the prediction accuracy. Averaging the results tends to reduce this bias due to varied test sample difficulty in different samples. Also, variance of the measurements can be calculated in such a scheme. Thus the performance measures obtained using different feature sets, segmentation methods or even different statistical models can be compared since the variances are available.

1.6 Use of translation model for evaluating computer vision algorithms

Computer vision algorithms involve a number of low-level processes to achieve the task intended. Segmentation, edge detection, filtering, and feature extraction are a few low-level tasks that form an initial step in a number of vision applications. There exists a great volume of literature describing several techniques to perform these tasks. But not much work has been done to evaluate these algorithms on a common ground. Also a good general task has not been forthcoming to perform this evaluation. It can be argued that word prediction is an excellent task because it is associated with higher-level image semantics and recognition. It is general since it is not necessary to specify in advance which objects or scene semantics are to be considered. The availability of vast datasets with labeled image-word data provides an added advantage to use this task for evaluation. Large-scale experiments provide reliable values for the performance indices and their variance. Since auto-annotation is general and testable, it can be used to develop and evaluate computer vision tools that support discovering scene semantics.

1.7 Organization of the thesis and contributions

1.7.1 Organization of the thesis

In Chapter 2 of the thesis, feature set evaluation is described. This chapter begins by giving a detailed description of features presently used in the system. These features are classified as size, location, shape, color, context and texture features. Under each category different features encode information relating to that category in different ways. The performance of word-prediction depends on the feature set used. The effects of using different feature sets on the performance of word prediction are studied. Features belonging to different categories are added to a base set of features to determine which of the above categories or combinations of categories help the process of auto-annotation. Quantitative results of evaluating different feature sets using annotation performance as a measure are tabulated. The results of this evaluation are discussed at the end of the chapter.

Chapter 3 is concerned with segmentation algorithms. The performance on the word prediction task is demonstrated as a quantitative measure to evaluate different classes of segmentation algorithms. The classes of segmentation algorithms considered are the Normalized Cuts [7] and Mean Shift [8]. Illustrative segmentations using these classes of algorithms are shown followed by a comparison of these algorithms. Specifically, for each segmentation algorithm, the annotation performance is plotted as a function of the number of regions used for annotation. Then the chapter describes the details of the Normalized Cuts segmentation algorithm. This is followed by an account of problems identified in the original algorithm and a description of possible modifications to overcome these problems. Results of segmentation on a few images before and after applying these modifications are also illustrated. Then the word prediction tool is used to perform a quantitative evaluation of the original and the modified versions of the algorithm.

Chapter 4 presents the effects of illumination on the translation model for object recognition. The focus in this chapter is on changes in image color due to change in the color of light illuminating a scene. Possible degradation of object recognition performance due to this illumination change is studied. Different ways to compensate for illumination change are described. Specifically, two paradigms are considered. One is to train the recognition system for illumination change by including training images taken under different expected illumination changes. The other is to use color constancy processing to compensate for the effects of illumination color change. Two color constancy algorithms—"gray world" and "scale-by-max"—are studied. The improvement in word prediction obtained by using each of these strategies is evaluated quantitatively using annotation performance. It is shown that word prediction can be used as a tool to evaluate different color constancy algorithms within a single framework.

Conclusions of the thesis are given in Chapter 5. This chapter also proposes a few directions for further research based on the results of the work in this thesis.

1.7.2 Contributions of the thesis

The main emphasis in this thesis is that the translation model for object recognition can be used as a tool to evaluate different low-level computer vision processes quantitatively. Quantitative evaluation is possible by computing the annotation accuracies obtained by using these different low-level algorithms for the task of word prediction. Segmentation and feature extraction form preprocessing steps in the translation model used here. Hence, different segmentation methods and different feature sets lead to models with different performances in terms of annotation. The annotation performance using different segmentation methods and different feature sets are compared.

Two new features are added to the existing set of features in the system. These features encode the outer shape and context information of regions in images. It is shown that context information is useful in that it helps to disambiguate objects that appear similar in terms of a few visual features but exist in different surroundings. It is also shown that shape information is not of much help because the present day segmentation techniques cannot group objects as a whole. In this framework, the objects to be recognized are not specified beforehand. The system learns about different objects from their instances in the training images and hence the task can be regarded as general object recognition. The present work emphasizes that the features or feature combinations that help this task, as implied by annotation performance, carry useful information to help recognize objects in general.

Though there exist a number of algorithms to segment natural images, comparison of these algorithms has been through visual inspection on some set of images. It is possible that a segmentation algorithm may do a good job on images of some specific type(s) but not on others. Performing visual inspection on images of all possible categories is impractical. In this evaluation methodology, results are typically based on the performance of segmentation algorithms on the huge Corel dataset that contains images conveying many different themes. Hence the results of such an evaluation should be indicative of the ability of a segmentation algorithm in grouping objects meaningfully in a variety of images. Moreover, this evaluation is a quantitative evaluation of different segmentation algorithms that has not been possible before. Among the existing segmentation algorithms, a specific algorithm called Normalized Cuts is particularly useful with the joint image-word model discussed here. This algorithm is considered in detail and modifications are proposed to the original version of this algorithm to achieve better grouping of regions in natural images. An evaluation of the modified version against the original version is carried out using annotation measure.

The translation model is also used to compare different strategies to deal with illumination change in an object recognition framework. Specifically the focus in on strategies that aim to compensate for changes in image color due to change in color of illumination of a scene when it is imaged. These strategies are to train the recognition system with images taken under expected illumination changes or to use some kind of color-constancy processing to compensate for changes in illumination color. To our knowledge, such an evaluation has not been done before. The effects of varying illumination on the object recognition model are also studied. The results of such a comparison could be indicative of which strategies and which color constancy algorithms are more suitable in the context of object recognition than others.

Chapter 2

EVALUATION OF FEATURE SETS

The process of auto-annotation exploits the correlation between visual and semantic descriptions of natural scenes. The semantic description is provided by the keywords attached to the images. To obtain visual description, images are segmented and features are extracted from each of the resulting segments. These features are intended to provide a meaningful characterization of objects present in the scene. The focus is on objects because the keywords used in the joint modeling are mostly nouns and hence pertain to objects present in the scene. The features extracted could describe the color, texture, geometry, shape or any other characteristic of an object. The question arises as to what is the best feature set to be used. The choice of a feature set influences the joint image-word model and hence the accuracy of auto-annotation. Considering the task to be achieved, i.e. auto-annotation, the feature set chosen should correlate well with the type of keywords being used. The better the correlation, the better should be the model to predict words for images more accurately. It is not clear as to which type of visual features go well with words in describing images. It is also possible that combining feature sets describing different aspects (color, texture, shape etc.) of objects can perform better than using them individually. On the other hand, this may lead to redundancy in terms of information carried by the features. For example, it is possible that some of the color descriptors can carry texture information and using color and texture features together may be redundant. This may also lead to a high degree of correlation between feature sets that in turn may cause problems while training. This is particularly the case when the model being trained assumes independence or no correlation among visual features. To address these issues, a thorough evaluation of the performance of different feature set combinations is needed. This is precisely the goal of this part of the thesis. To start with, a description of the features presently used in the system is provided. This is followed by details of the feature evaluation experiments and their results.

2.1 Features in the present system

Segmentation of an image renders a partition of the image into distinct regions. Features are extracted from each of the individual regions to characterize the objects implied by those regions. The various features extracted can be described under the following categories:

2.1.1 Region size

Region size is the area of the region normalized by the size of the image. The idea is to encode the amount of space occupied by the region in the image using a single number. In a scene containing a bird flying in the sky, the sky region will have a high value for this feature whereas the bird region will have a small value. Of course, the underlying assumption is that the segmentation algorithm is able to separate out the two different entities in the scene in a meaningful way. If A represents the area of a region, W represents the width and H the height of the image, then the region size RS is given by:

$$RS = \frac{A}{W \times H} \tag{2.1}$$

2.1.2 Region location

This feature carries information about the relative position of an object in a scene. In natural scenes containing sky and an ocean, sky always appears at the top and ocean at the bottom. This can be encoded with the row and column coordinates of the center of mass of the region. To achieve scale invariance, these coordinates are normalized by the height and width of the image respectively. So the two numbers describing the location of a region are given by:

$$x_{loc} = \frac{x_{CM}}{W}$$

$$y_{loc} = \frac{y_{CM}}{H} \tag{2.2}$$

where, x_{CM} and y_{CM} are the column and row coordinates of the center of mass of the region, and x_{loc} and y_{loc} are their normalized counterparts with respect to the width (W) and height (H) of the image respectively. The center of mass coordinates are obtained as the means of the histograms (probability mass functions) representing the distribution of the region pixels along the column and row axes.

2.1.3 Shape features

Shape of an object can be encoded in several ways. The review papers [10, 11] are a good source of techniques used by vision researchers to encode and match/differentiate shapes of objects. More recent work also suggests using shape context [12] and wavelet descriptors [13] as possible shape features. Shape descriptors can be classified depending on whether they carry information about the internal structure of an object or the outer boundary structure. In this work, both the types of descriptors are used to represent shapes of objects. The internal shape descriptors include the second moment, compactness and convexity features. A 30-component vector encodes outer boundary information whose Fourier transform serves as a useful shape feature and which has been referred to as a type of Fourier descriptor in the shape literature [14]. These features are described below.

2.1.3.1 Second moment

Second moment is the standard deviation of region pixels from the region center of mass. The standard deviation is computed along both the row and coordinate axes. In order to account for different sizes of the same object at different instances, the standard deviations are normalized with respect to half the region width and height along the row and column axes respectively. This normalization also forces the standard deviation values to lie in the range [0, 1] independent of the scale of the objects. Formally, let x_{CM}

and y_{CM} be the column and row coordinates of the center of mass of a region, and then the second moments of the region along those axes are given by:

$$(Second\_Moment)_x = \frac{1}{(W/2)}\sqrt{\frac{\sum_{n=1}^{N}(x_n - x_{CM})^2}{N}}$$

$$(Second\_Moment)_y = \frac{1}{(H/2)}\sqrt{\frac{\sum_{n=1}^{N}(y_n - y_{CM})^2}{N}} \tag{2.3}$$

where n indexes pixels in the region that is assumed to have a total of N pixels in the above equations.

2.1.3.2 Compactness

The compactness of a region is given by the ratio of its area to the square of its outer boundary length. Hence,

$$Compactness = \frac{A}{P^2} \tag{2.4}$$

where A is the area of the region and P is its perimeter. The compactness operator assumes a high value for regions that are circular in shape [15]. For regions filled with holes and those that are concave it assumes a low value. Hence this feature can be classified as an internal shape descriptor.

2.1.3.3 Convexity

As implied by the name, this feature measures how convex a region is. To measure convexity of a region, the area A_{hull} of the convex hull of the region is calculated. The ratio of region area A to its convex hull area A_{hull} is suggestive of how well the region boundaries follow the convex hull and hence gives a measure of convexity. Therefore,

$$Convexity = \frac{A}{A_{hull}} \tag{2.5}$$

2.1.3.4 Outer boundary descriptor and its Fourier transform

A contribution of this thesis is the addition of a shape feature that serves to describe the outer boundary shape of objects. We chose to use the simplest descriptor to represent the shapes of objects in terms of their outer boundaries. The contour of a 2D object is considered as a closed sequence of successive boundary pixel coordinates (x_s, y_s), where $0 \le s \le N-1$ and N is the total number of pixels on the boundary. An example of this coordinate chain is shown in the Fig. 2.1:

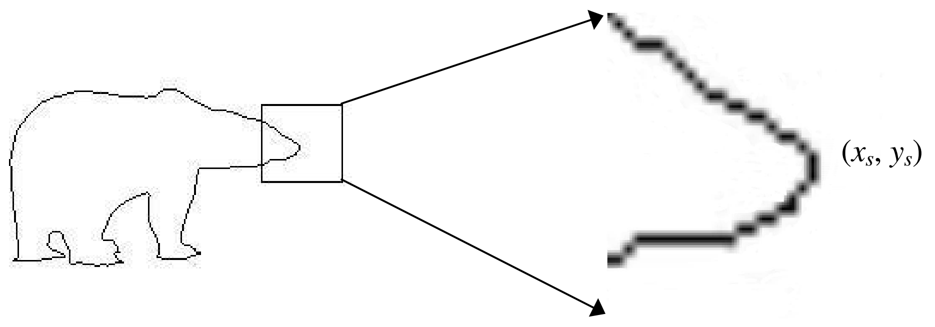
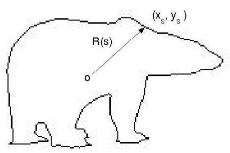


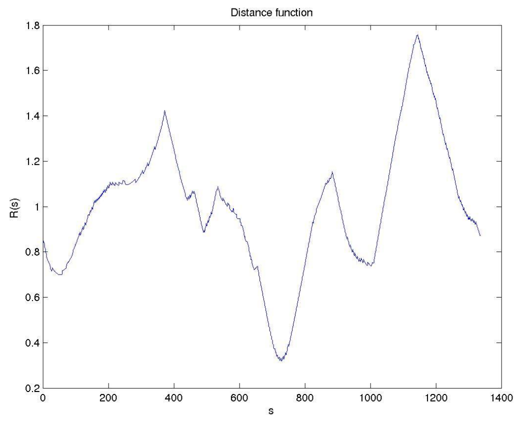
Fig. 2.1. *Shape contour*: *An example shape contour and a magnified portion of the same showing a coordinate chain.*

The outer boundary feature vector is the vector of distances of these boundary pixels from the region centroid or center of mass. If (x_c, y_c) denotes the centroid coordinates of a region and (x_s, y_s) denotes the boundary pixel coordinates at the pixel indexed by *s*, then the distance function for the region, *R*(*s*), is given by:

$$R(s) = \sqrt{(x_s - x_c)^2 + (y_s - y_c)^2} \tag{2.6}$$

Fig. 2.2 shows a typical distance function *R*(*s*) for a shape and its plot against the arc-length (pixel-index) parameter, *s*. Note that in the figure, *o* is the center of mass of the region and the distance function *R*(*s*) is normalized with respect to the average value of *R*(*s*) over all the boundary pixels. The reason for this normalization will be explained further.





(a) (b)

Fig. 2.2. *Distance function*: *(a) Distance function R(s) measured from the center of mass o of the region. (b) Plot of R(s) against the arc length parameter s.*

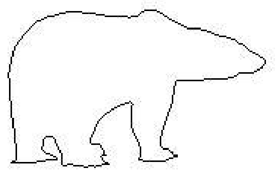
It can be seen that if segmentation does not produce smooth boundaries between regions then the function $R(s)$ varies significantly within a small range of s. This can be thought of as a noisy version of the actual distance function where the noise is due to the jagged boundaries produced during segmentation. The noisy version of the distance function is not a good representative of outer boundary shape since it varies a lot for different instances of the same object. A smoothed version of the distance function $R_{smooth}(s)$, is obtained by low-pass filtering $R(s)$ using a Gaussian kernel $G(s)$ of appropriate width σ. Therefore,

$$R_{smooth}(s) = \sum_{k=0}^{N-1} R(k)G(s-k) \tag{2.7}$$

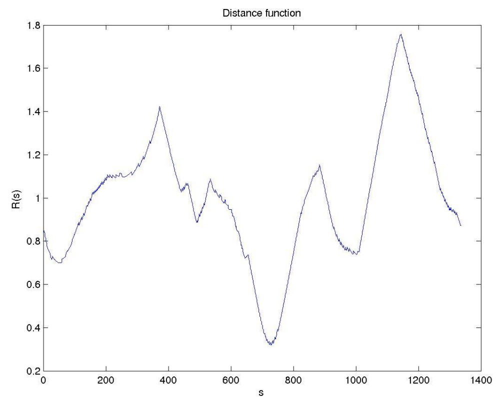
where the Gaussian kernel $G(s)$ is given by:

$$G(s) = \exp(-s^2 / \sigma^2) \tag{2.8}$$

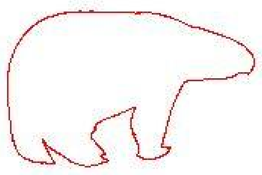
where σ is the width parameter of the Gaussian kernel. The value of σ is chosen proportional to the total number *N* of boundary pixels. This makes the smoothing process invariant to different sizes of the same object in different images. Before using it for smoothing, the kernel *G*(*s*) is L_1 normalized so that the filter has unit response at zero frequency and hence there is no DC gain in $R_{smooth}(s)$ with respect to *R*(*s*). Examples of original shapes and their smoothed versions along with their corresponding *R*(*s*) and $R_{smooth}(s)$ are shown in Fig. 2.3:



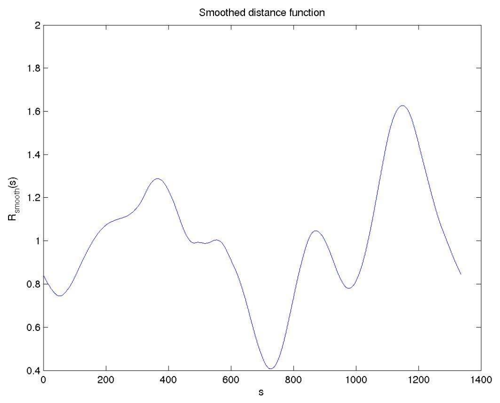
(a)



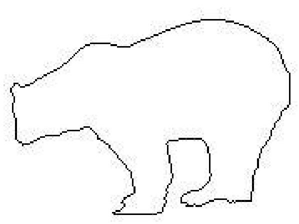
(b)



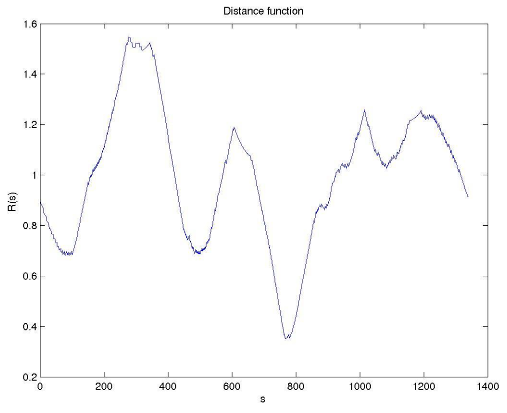
(c)



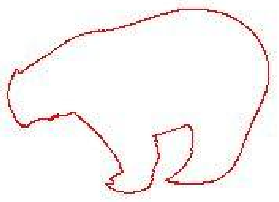
(d)



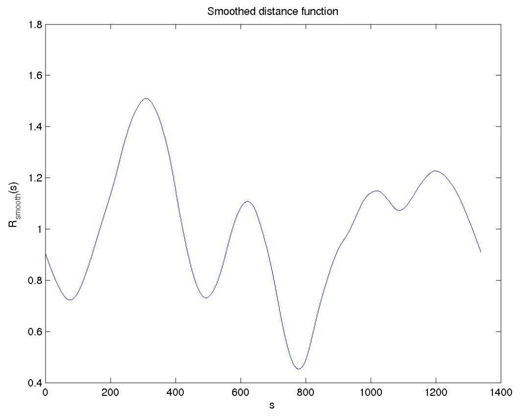
(e)



(f)



(g)



(h)

Fig. 2.3. *Smoothing of distance function*: *An example shape contour as obtained from a segmentation output is shown in (a) with a plot of its distance function R(s) in (b). (c) shows the smoothed version of the shape in (a) where smoothing is done using a Gaussian kernel as described in the text. (d) is the plot of* $R_{smooth}(s)$. *Similarly (e), (f), (g) and (h) describe another shape contour of a different instance of the same entity (bear) as in (a)-(d) but from a different view (equivalent to mirror reflection).*

The idea is that though *R*(*s*) may differ a lot for different instances of the same object (due to noise), the smoothed versions of these noisy instances $R_{smooth}(s)$ should almost resemble one another. This is because noise is contained mostly in the high frequency range, which is eliminated by passing through the low pass filter. The function $R_{smooth}(s)$ is made scale invariant by first normalizing it with respect to the average distance of the outer boundary from the region centroid to obtain $R_{smooth,n}(s)$. Specifically:

$$\overline{R} = \frac{1}{N}\sum_{k=0}^{N-1} R(k) \tag{2.9}$$

$$R_{smooth,n}(s) = \frac{R_{smooth}(s)}{\overline{R}} \tag{2.10}$$

Furthermore, the number of samples *N* is dependent on the size of the object in the particular image. In different images, the same object can occur in different sizes resulting in different values for *N* across those images. To overcome this problem and also to let the descriptor have the same length for different objects, the function $R_{smooth,n}(s)$ is uniformly sampled at *M* points to obtain the final descriptor $R_f(s)$, $0 \le s \le M-1$. We chose *M* to be 30 for this work to capture as much shape information as possible while keeping the shape feature dimensionality manageable by the learning algorithm. In our sampling scheme we choose 1 sample from the center of every window of size *N*/*M* samples to obtain the *M* samples from the entire signal of length *N*. To ensure that signal values from the left and right neighboring windows contribute in the smoothing process of our sample value in the present window, a value of 2*(*N*/*M*) was chosen for the width parameter σ of the Gaussian kernel $G(s)$. [10] describes an analysis of shapes at different scales by using a range of values for σ during smoothing. Following this approach would lead to a very high dimensional shape feature with the choice of M here. Hence we chose to encode the shape feature at one scale, as our aim is to test if shape is useful at all. We do this without making the feature dimensionality excessively large so as to be manageable by the learning algorithm. We also allowed the symmetric Gaussian to have non-zero support over 2.2*(N/M) samples to allow for at least 60% of the samples from left and right neighboring windows to contribute towards the smoothed values in the present window.

The descriptor $R_f(s)$ is translation invariant inherently by the definition of $R(s)$. Normalization w.r.t. $\overline{R}$ makes it scale-invariant. The first point ($s = 0$) is always chosen to be the top left corner point of the outer boundary to avoid ambiguity regarding the starting point. This makes the descriptor $R_f(s)$ rotationally variant. There is another problem with this descriptor. The contours in (a) and (e) of Fig. 2.3 represent different instances of the same entity (bear). But their descriptor signals as seen in Fig. 2.3 (d) and (h) are reflections of each other along with a phase shift. With this representation the two descriptors are very different from each other and there is no way to incorporate information that they represent the same entity. To overcome this and also to achieve rotational invariance, Fourier descriptors can be used. In the area of shape analysis and classification, several shape feature representation schemes based on autoregressive (AR) models [16, 17] and Fourier descriptors [18, 14, 19] of contours have been proposed. An experimental comparison of shape classification methods based on these two principles has been carried out in [20], which indicates that Fourier-based methods provide better performance than AR-based approaches, especially for noisy images. This also provides motivation to use Fourier descriptors to encode the shape feature.

A Fourier descriptor for shape is obtained by projecting any shape descriptor into the frequency domain by taking its Fourier transform [14]. For our work, we project the *M*-component vector $R_f(s)$ into the frequency domain to obtain a Fourier-based shape descriptor. Fourier transform of a contour representation generates a set of complex coefficients. These coefficients represent the shape of an object in the frequency domain, with lower frequency describing the general shape property and higher frequency denoting the shape details. Taking the T-point Discrete Fourier transform of $R_f(s)$, we have,

$$\Re(k) = \sum_{s=0}^{M-1} R_f(s) e^{-j\frac{2\pi ks}{T}} \qquad k = 0,1,....,T-1 \tag{2.11}$$

The coefficients $\Re(k)$ are the complex Fourier coefficients of $R_f(s)$. Note that always $T \geq M$ to avoid aliasing. For the particular case of $M = 30$ here, we took a 61 point DFT to obtain the Fourier transform:

$$\Re(k) = \sum_{s=0}^{29} R_f(s) e^{-j\frac{2\pi ks}{61}} \qquad k = 0,1,....,60 \tag{2.12}$$

From the conjugate symmetry property of Fourier transform of a real signal $R_f(s)$, we have,

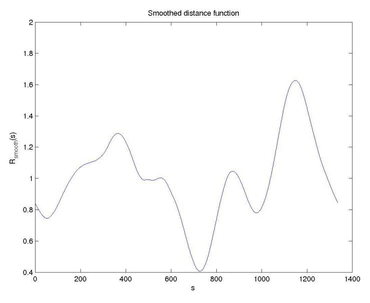
$$|\Re(k)| = |\Re(61-k)|$$

$$\arg(\Re(k)) = -\arg(\Re(61-k)) \qquad k = 0,1,....,60 \tag{2.13}$$

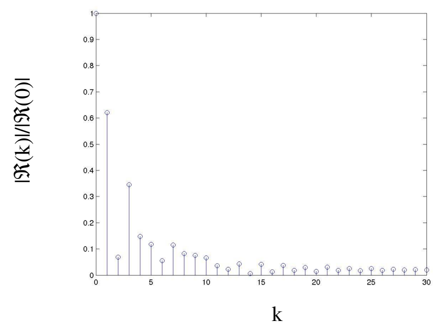
The phase, $\arg(\Re(k))$, of the Fourier transform coefficients is dependent on the starting point (s = 0) of the signal $R_f(s)$. If we ignore the phase and consider only the magnitude of the coefficients, the starting point of the signal $R_f(s)$ on the boundary does not make a difference and the resulting descriptor becomes rotationally invariant. Recall another property that the Fourier magnitude coefficients of a real signal and its reflected and phase-shifted versions are the same. This makes the Fourier descriptor invariant to mirror reflections and rotations thereafter of object instances in different images (Fig. 2.3 (a) and (e)) if we consider only the magnitude of the Fourier coefficients $|\Re(k)|$. The Fourier descriptor in our case becomes:

$$\mathbf{F} = \left[\frac{|\Re(1)|}{|\Re(0)|}, \frac{|\Re(2)|}{|\Re(0)|}, ..., \frac{|\Re(30)|}{|\Re(0)|} \right] \tag{2.14}$$

where each frequency coefficient is normalized by the magnitude of the zero-frequency component to make the descriptor scale-invariant. Figure 2.4 illustrates Fourier descriptors by plotting $R_f(s)$ for a shape and its corresponding descriptor vector **F**.



(a)



(b)

Fig. 2.4. *Fourier descriptor*: *(a) Smoothed distance function* $R_{smooth,n}(s)$ *for a shape. (b) The first 30 samples of the 61-point DFT magnitude response of the sampled function* $R_f(s)$ *obtained from* $R_{smooth,n}(s)$.

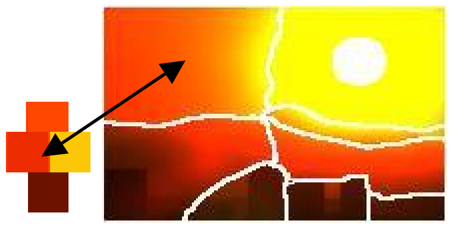
The remaining coefficients $\frac{|\Re(k)|}{|\Re(0)|}$ for k = 31, 32,…., 60 do not appear in the descriptor **F** because they would be redundant as implied by the conjugate symmetry property of Eq. (2.13). The 30-component vector **F** forms the Fourier shape descriptor, which is invariant to translation, rotation and scale.

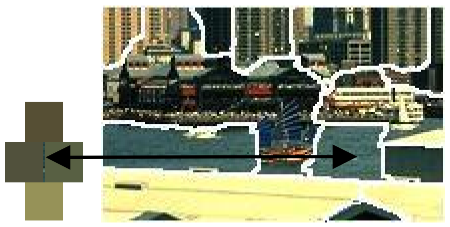
2.1.4 Color features

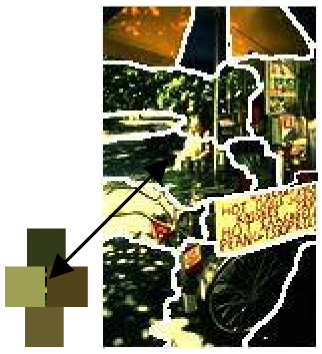
Color is a very useful and distinguishing characteristic of an object. In the present framework, color of a region is encoded by computing the mean and the standard deviation of color of the pixels present in the region. To compute the mean and standard deviation of color a suitable color space needs to be chosen. There are several color spaces that are suited for different applications. For this work, three different color spaces are considered. Those are the RGB, CIE L*a*b and the rgS (chromaticity with brightness) spaces. The rgS color values are directly obtained from RGB values as S=R+G+B, r=R/S, and g=G/S. Thus with any one of the above color spaces chosen, color of a region is represented by a set of 6 numbers.

2.1.5 Context feature

In addition to using average color and its standard deviation over a region, color is also encoded as context information around the region. Incorporating context features into the system and testing its usefulness is another contribution of this thesis. Our description of color context of a region is the average color adjacent to the region in various directions. The motivation to use this feature is that the context of an object helps to ascertain its presence and disambiguate it from objects whose other features are almost similar to that of the object. For example, a brown region is more likely to be a bird, and less likely to be rock, if it surrounded by a light blue (sky) region. To compute the color context of a region, 4 quadrants are considered with origin of the coordinate system located at the center of mass of the region. Axes that are aligned at 45 degrees with the image row and column axes separate the quadrants. Let $\overline{R}$ be the average distance of the outer boundary of the region from its center of mass. In each quadrant, the color of all pixels that belong to other regions but lie within a radius of $2\overline{R}$ is averaged if the number of such pixels is greater than or equal to 100. Otherwise, the average of the region itself is used as color context. Therefore, color context is encoded into 12 numbers, 3 for each quadrant assuming that a 3 dimensional color space is used. In the following images, the average colors around a region of interest (in the four quadrants) are represented by four patches beside each image. In each case, the left patch represents the average color in the left quadrant with respect to the center of mass of the region and the right patch represents the average color in the right quadrant. Similarly patches are shown for the top and bottom quadrants.


(a)


(b)


(c)

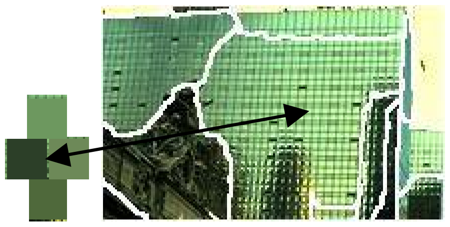

(d)

Fig. 2.5. *Color context feature*: *Images (a), (b), (c) and (d) illustrate the color context feature captured as average color around a region of interest in each of those images. Beside each image, the 4 rectangular patches filled with uniform color represent the context information for a particular region of interest. The double-sided arrow between the rectangular patches and the image points to the region of interest. The uniform colors of the rectangular patches are the average colors of the surrounding regions in the 4 quadrants as described in the text.*

2.1.6 Texture features

The variation of intensity patterns within a region constitutes the texture within that region. To extract texture information of a region, the image is first convolved with a linear filter bank. The filter bank consists of both even and odd symmetric filters at different scales and orientations. Then the responses to these filters at all the pixels within the region are averaged separately. The average values along with the standard deviations of the responses encode the texture within the region. The idea is that if a region has most of the edges oriented along a certain direction, then the response to filters with that orientation will be high for pixels within the region. So the average of the responses to filters at that orientation will also be high and it is representative of texture information within the region. The notion of texture is always associated with a scale. If we look at a scene from two different distances, the texture of the same region in the two cases can appear different. Therefore in order to capture texture at different scales, responses to filters at different scales are considered.

For this work, the even part of the filter bank is constituted by second derivative of Gaussian kernels and the odd part by the kernels formed by taking their Hilbert transform. In addition, 4 different radially symmetric filters are also used. These are formed by the Difference of Gaussian (DOG) filters having different width parameters as in [21]. Even and odd symmetric filters at 12 different orientations each at 4 different scales are used. These along with the 4 DOG filters constitute a total of 52 filters.

2.2 Feature evaluation

The performance on the task of auto-annotation is dependent on the set of features used for training. Here we test out the performance of different feature combinations using the translation model. For these experiments, the Normalized Cuts segmenter is used and the number of regions is fixed at eight. To study the usefulness of each type of feature, we start out with a base set of features consisting of size, location, second moment and compactness. Then the features describing color, texture, shape and context or a combination of them are added to the base set. The improvement in word prediction performance by adding any feature set over that with just the base set is suggestive of the helpfulness of the feature set for the task of auto-annotation. Table 2.1 shows the results of feature evaluation experiments using annotation performance. The values in the Table are the averaged relative scores of word prediction accuracy as described in Section 1.4 of Chapter 1. Also the Table lists results for 3 types of test images ("Training", "Held-out" and "Novel") as described in Section 1.5 of Chapter 1.

The results suggest that color is the most useful feature for this task using the Corel dataset. Among the three different spaces used for encoding color, rgS space gives the greatest improvement in performance. This may be due to the fact that there is a weaker correlation between the color components in the rgS space as compared to the RGB or L*a*b spaces. Recall that independence among different features at the node level is assumed as part of the modeling procedure. To fully support this notion experiments are needed by considering color spaces obtained by further decorrelating color coordinates using techniques such as independent component analysis. Texture is encoded by considering responses to 4 DOG filters and 12 oriented filters at one scale. The improvement in word prediction performance upon adding texture features to the base set is indicative of helpfulness of texture for object recognition. When texture is used in conjunction with color, the increment is not that large. A possible reason could be that color variance features carry some amount of texture information and hence are correlated with the texture features.

Using shape proved to be problematic. Results using both the outer boundary descriptor and its Fourier transform (Fourier descriptor) are shown. It is clear from the results on the training data that the shape descriptor carries useful information but the results on the held out and novel data suggest that what is captured does not generalize well. This is because shape can be useful only when the segmentation process hypothesizes objects as a whole. State-of-the-art in segmentation techniques utilize only low level information for grouping regions which is not sufficient to isolate objects in images. Hence shape can be useful only when a segmentation technique capable of grouping objects as a whole is available. Adding the color context feature helped improve the accuracy of word prediction supporting the notion that context information is useful for the process of object recognition. The way in which context information is incorporated into the system is relatively simple. In order to describe context information around a region only the average color information around the region is exploited. It is possible to test the usefulness of surrounding texture as also context information. This may further help improve the performance. A recent work using an approach similar to the one described here for object recognition is worth mentioning [50]. In their approach context information is inherent to the translation model for object recognition by assuming that the probability of linking an image patch to a particular word is dependent on patch-word alignments of adjacent regions. The contextual model is shown to perform better than a corresponding model assuming independence between different patch-word alignments of an image. This offers encouragement to explore more sophisticated strategies of incorporating context information either in terms of features or using a contextual model itself.

Also note that training using all features tends to degrade the overall performance. This may be a result of correlation between different feature dimensions or possible over-training because of longer feature vectors providing more information than necessary. Over-training could also be another reason for the degradation in performance using shape feature because the dimensionality of the feature space increases by 30 due to addition of this feature. More experiments are needed to investigate these effects.

Feature set	Word prediction performance on the various data sets (error is roughly 0.003)		
	Training	Held out	Novel
Base set	0.019	0.020	0.018
Base set, RGB	0.076	0.057	0.044
Base set, L*a*b	0.097	0.085	0.061
Base set, rgS	0.109	0.092	0.065
Base, rgS, color context	0.134	0.094	0.055
Base set, texture	0.079	0.048	0.041
Base, rgS, texture	0.109	0.072	0.059
Base, RGB, color context, texture	0.116	0.073	0.055
Base set, shape	0.046	0.013	0.011
Base set, shape (Fourier)	0.043	0.018	0.018
Base set, rgS, shape	0.065	0.029	0.027
Base set, rgS, shape (Fourier)	0.064	0.034	0.030
Base, rgS, texture, shape	0.083	0.043	0.038
Base, rgS, texture, shape (Fourier)	0.079	0.041	0.038
Everything	0.097	0.055	0.039

Table 2.1. *Feature evaluation*: *Word prediction performance for a variety of feature sets. Color is clearly the best single cue, followed by texture.*

Chapter 3

EVALUATION OF SEGMENTATION ALGORITHMS AND MODIFICATIONS TO NORMALIZED CUTS ALGORITHM

The task of auto-annotation, which is the main focus of this thesis, utilizes a model based on joint distribution of image region features and words. Typically the words that accompany images in annotated databases and that have been used for experiments in this work are nouns. Examples include “sky”, “water”, “tiger”, and “people”. With the approach being followed here where segmentation is an initial processing step, it makes more sense to split up images into regions that correspond to real-world objects. The task here also demands a generic segmentation algorithm that proceeds without the goal of looking for specific objects or regions. This makes the segmentation process all the more difficult. If the objects to be segmented are known beforehand, attributes specific to those could be used to the advantage of the segmentation algorithm. Since this is not the case here, one has to rely only on low-level visual features like brightness, color, texture etc. to group perceptually similar entities in an image. The grouping process should also give a global impression of a scene in that it should treat objects as a whole even though their constituent parts are slightly different perceptually and are distributed across different parts of the scene. It is our feeling that a segmentation method that achieves these goals should lead to a better model correlating image regions and words and hence better auto-annotation.

There have been a number of segmentation algorithms proposed in the literature that aim at splitting up images into coherent regions using low-level visual cues. Usually performance of these algorithms is demonstrated by visual inspection on some set of images. Recently efforts have been made to compare segmentations to those provided by human subjects [27]. But doing a comprehensive evaluation on images of all kinds needs a lot of human effort in this approach. It is also possible to compare different segmentation techniques based on how they impact the overall performance of a specific computer vision task where segmentation forms an initial preprocessing step. We are not aware of a single task with an associated quantitative measure to compare different segmentation methods in a common framework. We argue that the task of object recognition as being approached here gives a systematic methodology for evaluation of various segmentation algorithms on a common ground. In this chapter, we evaluate segmentation algorithms based on their performance on the task of word prediction keeping all the other aspects (features, training/testing method) constant. The algorithms considered here are variants of the Normalized Cuts [7] and Mean Shift [8] class of segmentation algorithms.

The Normalized Cuts framework is well suited for the approach of joint image-word modeling as being approached here. This is because it gives the flexibility to choose the granularity at which segmentation is performed. By choosing a threshold for the normalized cut value we can control whether we need an over-segmentation (lots of regions) or under-segmentation (few regions) of an image. We consider the details of this segmentation algorithm as originally proposed by Shi and Malik [7] and the code for which was made available to us by Tal and Malik of the Berkeley segmentation group. And we propose several modifications to the original version to achieve improved segmentation in terms of localizing objects and obtaining a global impression of an image. In Section 3.1, word prediction performance is used to evaluate Normalized Cuts and Mean Shift classes of segmentation algorithms. Section 3.2 gives a description of some of the main aspects of the Normalized Cuts algorithm as in [7, 21]. This is followed by an account of the problems identified with the algorithm along with proposed modifications in Section 3.3. Finally Section 3.4 evaluates the original and modified versions of the Normalized Cuts algorithm using word prediction performance.

3.1 Evaluation of segmentation algorithms

Two classes of segmentation algorithms are considered here for comparison based on word prediction performance. These are the Mean Shift and the Normalized Cuts algorithms. Mean Shift technique initially performs kernel density estimation in a feature space and then delineates arbitrary shaped clusters to form segments in an image. More details can be found in [8]. The code for the Mean Shift algorithm is made available online by Georgescu et. al. of Rutgers University at [51]. Example Corel images segmented using the Mean Shift algorithm are shown in Fig. 3.1.

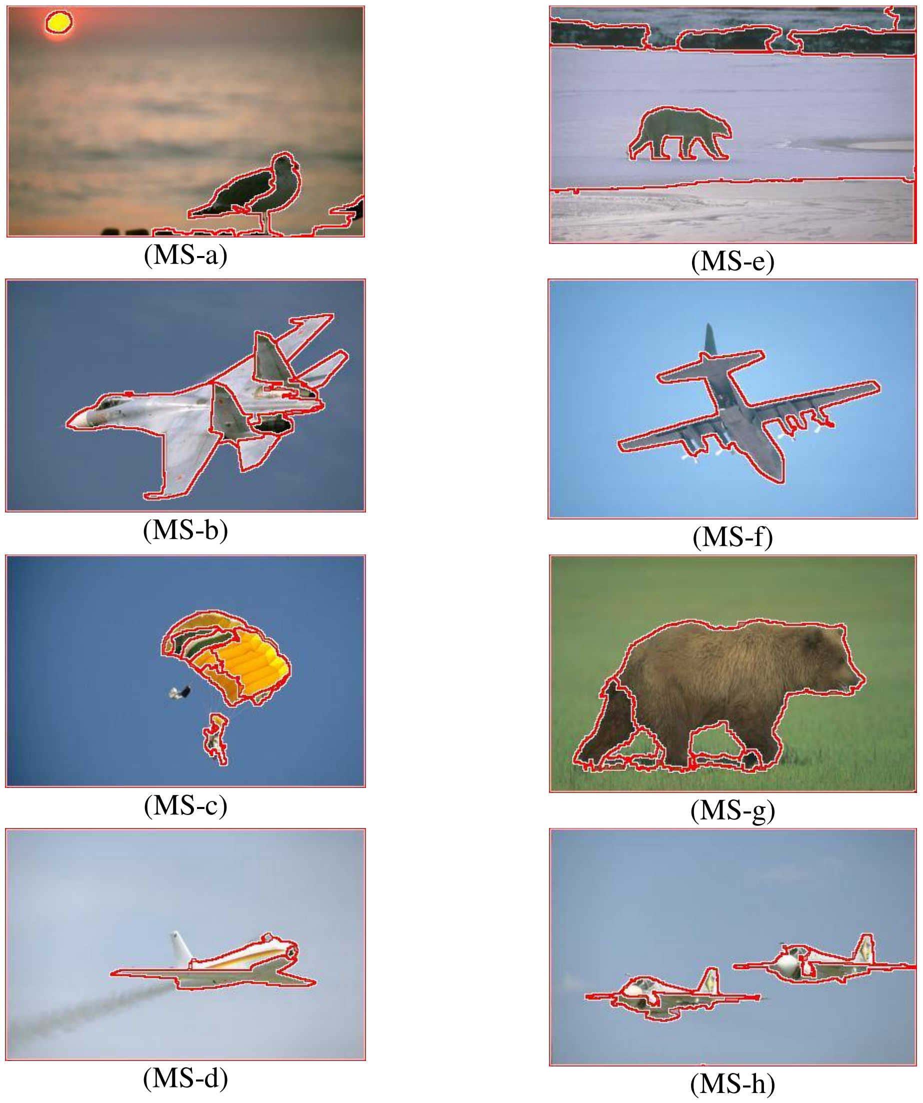


Fig. 3.1. *Mean Shift segmentation*: *Example images showing segmentation results using Mean Shift algorithm.*

Normalized Cuts (Ncuts) is a graph theoretic approach that derives a weighted undirected graph representation out of an image and then recursively partitions the graph so as to minimize a normalized objective function [7]. The same images as above segmented using the Ncuts algorithm are shown in Fig. 3.2.

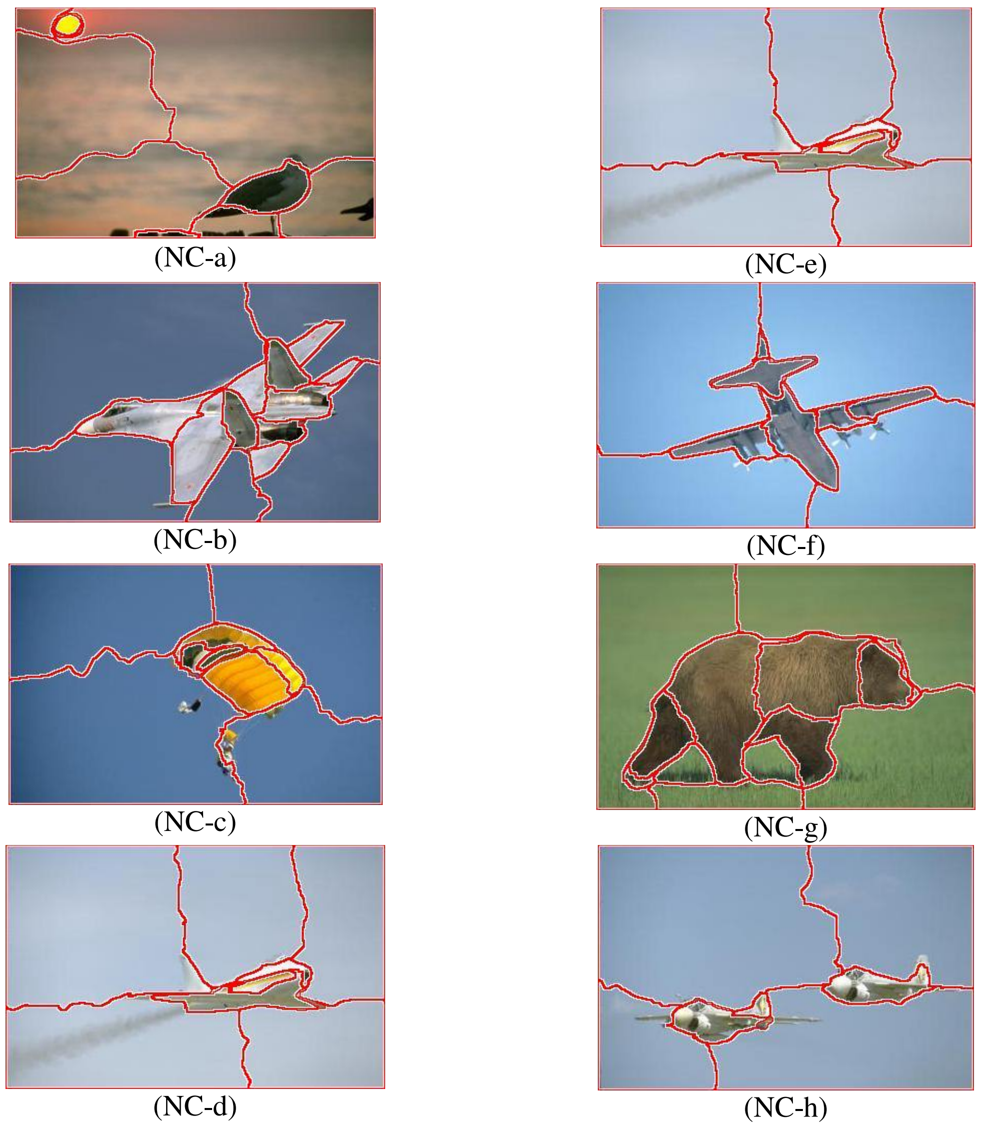


Fig. 3.2. *Normalized Cuts segmentation*: *Example images showing segmentation results using Normalized Cuts algorithm.*

Here a quantitative evaluation of the two classes of algorithms is carried out to illustrate that auto-annotation can be used as a tool to evaluate different segmentation techniques. To carry out this evaluation, annotation performance is plotted as a function of number of regions used for annotating the test images. Regions in the order of decreasing area are considered. A fixed set of features is used for all the experiments reported here. The following plot shows the annotation performance for the above described segmentation methods.

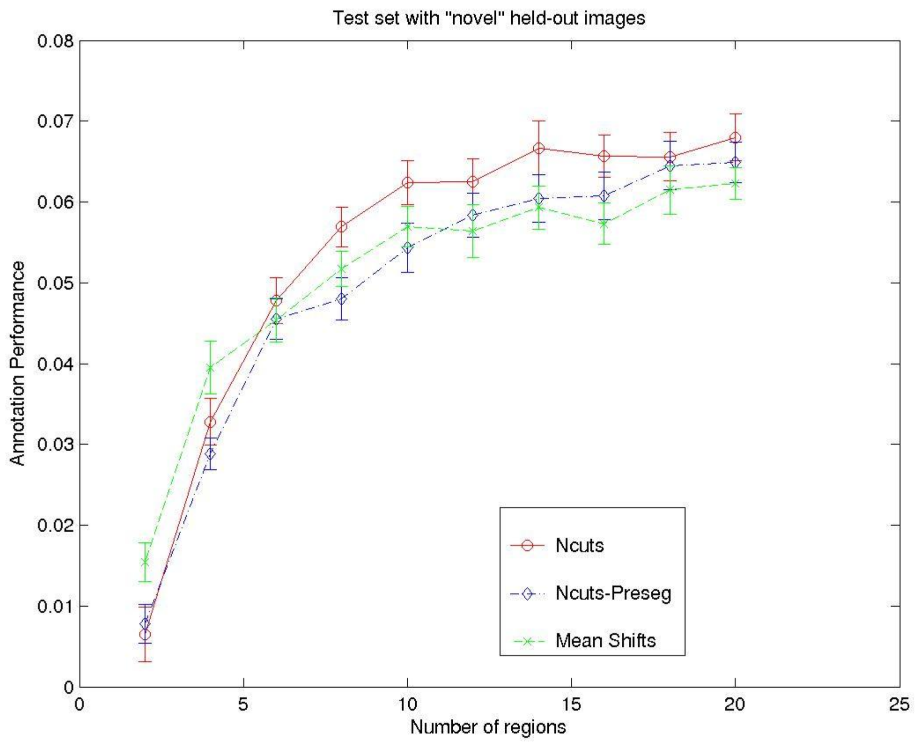


Fig. 3.3. *Normalized Cuts vs. Mean Shift*: *Comparison of Normalized Cuts and Mean Shift algorithms based on word prediction measure. The test images are different in concept from training images and hence the values on the curve imply generalization performance indicative of object recognition. The annotation performance is computed using the relative measure as described in Chapter 1.*

The images used for testing the algorithms are chosen from a held-out set that comes from Corel CD's that are not used in training. Therefore the performance is indicative of the generalization ability achieved by the model using a particular segmentation algorithm. Two versions of Normalized Cuts algorithm are used. The Ncuts-Preseg (pre-segmentation) version is an intermediate stage in the Normalized Cuts algorithm that produces an over-segmentation of an image (lots of regions). The details about pre-segmentation will be described in a later section. The results suggest that the performance of any segmentation algorithm is a function of the number of regions used for producing annotations for the images. In the plot above, Mean Shift performs better than both the versions of Normalized Cuts for number of regions less than 6. However its performance degrades in comparison to the Normalized Cuts algorithm as the number of regions increases. The results do not clearly suggest as to which segmentation method is superior as it is still a function of the number of regions. A possible reason for such a behavior is explained in the context of comparison of the original and a modified version of Normalized Cuts algorithm in a later section. More experiments with many more different segmentation algorithms may be needed to clearly bring out the usefulness of word prediction measure as a tool for segmentation evaluation. In the sections that follow, the Normalized Cuts segmentation algorithm is considered in detail to incorporate several modifications to the original version leading to better grouping of regions in an image.

3.2 Normalized Cuts algorithm

An overview of the normalized cut criterion and its use in image segmentation as in the code obtained from Tal and Malik of the Berkeley segmentation group is provided here to facilitate the understanding of the modifications proposed in Section 3.3. Normalized Cuts (Ncuts) is a graph partitioning technique that splits up a weighted graph into two parts in an optimal sense. A weighted graph is formed out of a set of points in an arbitrary space where each pair of points is connected with a weighted edge between them. The weight on the edge connecting any two points is indicative of the degree of similarity between the points. A graph partitioning technique seeks to partition the graph into disjoint sets such that some measure of similarity among points within a set is high and across different sets is low. To achieve this, the Normalized Cuts method advocates a criterion for measuring the goodness of a graph bi-partition—the *normalized cut* [7]. This criterion is explained below.

3.2.1 Normalized cut criterion

Let $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ be a weighted undirected graph where $\mathbf{V}$ is the set of all vertices and $\mathbf{E}$ is the set of all edges in the graph. The edge weight $w(i, j)$ is a measure of similarity between nodes i and j. Then the graph can be partitioned into two disjoint sets A and B, i.e., $A \cup B = \mathbf{V}$ and $A \cap B = \varnothing$, by removing edges that exist between point pairs such that one of the points belongs to A and the other belongs to B. The total weight of the edges removed is a measure of dissimilarity between the two sets and in graph theoretic terminology it is called the *cut*.

$$cut(A,B) = \sum_{u\in A, v\in B} w(u,v) \tag{3.1}$$

Shi and Malik [7] propose a normalized measure of dissociation between the two sets. They compute the cut cost as a fraction of the total edge connections to all the nodes in the graph. This dissociation measure is called the *normalized cut* (*Ncut*):

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)} \tag{3.2}$$

where *assoc*(*A*,*V*) is the total connection from nodes in *A* to all nodes in the graph, and *assoc*(*B*,*V*) is similarly defined. An optimal bi-partitioning of a graph can be achieved by splitting it up into two sets *A* and *B* such that *Ncut*(*A*,*B*) is minimized. Also an approximate optimal partition minimizing this criterion can be found as a solution to a generalized eigensystem as follows.

3.2.2 Computing the optimal partition

Given a partition of nodes of a graph, **V**, into two sets *A* and *B*, let $\boldsymbol{y}$ be an $N = |\mathbf{V}|$ dimensional binary indicator vector. That is $\boldsymbol{y}_i$ assumes only one of two possible values depending on whether *i* is in *A* or *i* is in *B* [7]. Let $\boldsymbol{d}(i) = \sum_j w(i,j)$ be the total connection weight from node i to all other nodes. Also let $\mathbf{D}$ be a N x N diagonal matrix with $\boldsymbol{d}$ on its diagonal and $\mathbf{W}$ be a N x N symmetrical matrix with $\mathbf{W}(i, j) = w_{ij}$. If $\boldsymbol{y}$ is allowed to take on real values, then the minimization of normalized cut measure between the sets A and B reduces to a generalized eigenvalue problem [7]:

$$(\mathbf{D} - \mathbf{W})\boldsymbol{y} = \lambda \mathbf{D}\boldsymbol{y} \tag{3.3}$$

The second smallest eigenvector of the generalized eigensystem is the real valued solution to the normalized cut problem. It is not the exact solution since $\boldsymbol{y}$ is not discrete. However an interesting property of the indicator vector $\boldsymbol{y}$ is that for nodes i and j that are tightly coupled (large w_{ij}), it is forced to take on similar real values. Appropriately thresholding the second smallest eigenvector $\boldsymbol{y}$ can delineate two groups from $\boldsymbol{y}$ such that the normalized cut value between the two groups is minimized. Thus a good approximation to the optimal partition according to *Ncut* criterion can be found as a solution to the generalized eigensystem of Eq. (3.3).

3.2.3 Normalized cut criterion applied to image segmentation

Normalized cut criterion is applied to image segmentation by treating each pixel in the image as a point in some arbitrary feature space. The edge weights between pixel-pairs are specified based on some similarity between the pairs in terms of the features considered. Segmentation is achieved by partitioning the graph into coherent groups using the *Ncut* criterion. So the grouping algorithm consists of the following steps:

1. Given an image, set up a weighted graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, and set the weight on the edge connecting two nodes being a measure of the similarity between the two nodes. Form the matrices $\mathbf{W}$ and $\mathbf{D}$.
2. Solve $(\mathbf{D} - \mathbf{W})\boldsymbol{x} = \lambda \mathbf{D}\boldsymbol{x}$ for eigenvectors with the smallest eigenvalues.
3. Use the eigenvector with second smallest eigenvalue to bi-partition the graph.
4. Decide if the current partition should be sub-divided, and recursively repartition the segmented parts if necessary. A threshold for *Ncut* could be set so that the recursion stops when the normalized cut value between two partitions at any stage is greater than this threshold.

The quality of segmentation is dependent on the choice of feature space and the resulting weight matrix **W** in the Ncuts procedure. Any of the low-level attributes like brightness, color, texture, etc. at a pixel can be used to form a **W** that can be input to the normalized cut grouping algorithm to come up with hierarchical partitions sequentially. Using each attribute individually has its own advantages/disadvantages depending on the class of images being segmented. [21] gives a good analysis of low-level cues, mainly contour and texture, used for grouping in traditional segmentation methods.

The Ncuts algorithm makes use of three low-level cues–contour, texture and color–to come up with a weight matrix **W** that can be used to partition the image into coherent segments based on minimizing the normalized cut criterion. The details of the steps involved in the partitioning algorithm will be described later. The element $W_{ij} = w(i, j)$ of the weight matrix **W** is a measure of similarity between nodes *i* and *j*. The total similarity measure is calculated by combining all the three cues.

3.2.4 Combining the cues

Each of the texture, contour and color cues gives a similarity measure between pixels *i* and *j* denoted by W_{ij}^{TX}, W_{ij}^{IC}, and W_{ij}^{COLOR} respectively. Assume that each of these weights is in the range [0, 1] and each carries information about similarity between pixels. The three cues are integrated by multiplying them to obtain the final combined similarity weight between pixels *i* and *j*, W_{ij}, as:

$$W_{ij} = W_{ij}^{TX} \times W_{ij}^{IC} \times W_{ij}^{COLOR} \tag{3.4}$$

The idea is that if any of the cues suggests that *i* and *j* should be separated, then the composite weight W_{ij} is small. Note that W_{ij} is also in the range [0, 1]. The composite weights W_{ij} are used in forming the weight matrix **W**.

3.2.5 Texture

Texture cue is encoded by introducing the notion of *textons* [21]. The image is first convolved with odd and even linear filters. These filters detect edges at a scale. The vector of responses to these filters at a pixel can be considered as a point in a high dimensional space. Since texture is assumed to be characterized by some spatially repeating structures, the filter responses within a uniformly textured region will not be very different from each other. This suggests that in an image, a prototype vector of responses can represent each uniformly textured region and the small variations in responses within that region are noisy versions of this prototype. *K*-means clustering is done on the filter responses obtained from all the pixels in the image. The converged means represent the texture prototypes referred to as *textons*. By mapping each pixel to the texton nearest to its vector of filter responses, the image can be analyzed into texton channels. A texton channel k consists of all pixels in the image that are mapped to the texton k.

Texture at a point is the pattern of intensity variations in a neighborhood around that point. So texture descriptor at a point is dependent on the size of the neighborhood. In other words texture is always associated with scale (size of neighborhood). Analyzing an image into texton channels makes it possible to determine a texture scale $\alpha(i)$ at each pixel i in the image and also to derive a texture descriptor based on the scale. The texture descriptor at a pixel i is the texton histogram obtained by considering only those pixels that fall within a square window of size $\alpha(i)$ centered at i [21]. Pairwise texture similarities between pixels are computed by comparing windowed texton histograms using the χ^2 histogram distance measure. This is further converted into a number in the range [0, 1] that forms the texture similarity weight W_{ij}^{TX} between pixels i and j (0 implies least similarity and 1 implies maximum similarity).

3.2.6 Contour

Contour cue is encoded using the oriented energy approach [21]. Orientation energy at a pixel is the edge strength at that pixel at a given orientation and scale. At all pixels, the orientation energies are computed at different scales and orientations. Only those pixel orientation energies are retained that are a local maximum over different scales and orientations and others are set to 0. Further, an intervening contour framework is adopted [25]. If the orientation energy along the line joining two pixels i and j is strong, then the similarity weight between these pixels based on the contour cue, W_{ij}^{IC}, is made low (towards 0). On the contrary, if there is little energy, for example in a constant brightness region, the link between the two pixels is made strong (towards 1). Again W_{ij}^{IC} is forced to be in the range [0, 1] as in the case of texture cue [21].

3.2.7 Color

To obtain a similarity measure between pixels *i* and *j* based on color cue, W_{ij}^{COLOR}, color histograms are computed at each pixel. The χ^2 metric between the two histograms gives a measure of similarity. The local windows considered for computing these histograms are the same as those used for texton histogram computation. The histogram is computed in the two-dimensional a*b* coordinates of the CIE L*a*b* color space. The a*b* space is first discretized into 64 bins with 8 equally spaced bins along the a* and b* axes respectively. When computing the histogram, each color value is quantized to the bin nearest to it in terms of Euclidean distance in the a*b* space.

3.2.7.1 Soft binning

Note that the above quantization scheme does not take into account the perceptual similarity between colors that belong to adjacent bins. The quantization scheme may assign two nearby colors to two different bins even though they are perceptually very similar. This causes problems because the χ^2 metric measures distance between discrete histograms and has no clue about the perceptual similarity between adjacent bins. To compensate for this, a soft binning scheme is adopted. In this scheme, when the count of a bin *k* is being incremented, the counts of the adjacent bins are also incremented. The magnitude of increment at an adjacent bin depends on the value of a Gaussian at that bin. This Gaussian is centered at the present color value. Note that the present color value can fall anywhere within the quantization region of the bin *k*. Thus existence of a color belonging to a bin causes increments in the adjacent bins and the increments are proportional to the Gaussian window magnitude at those bins. Color histograms are computed at all pixels in an image with the soft-binning scheme. The similarity between two pixels in terms of color cue is then the χ^2 distance between the 2D histograms at those pixels.

3.2.8 Local connectivity

The weight matrix **W** carries information about similarity between pixel-pairs. If every pixel-pair in the image is assumed to be connected by a non-zero weight, then even for a moderately sized image, **W** becomes a huge dense matrix. Pixels very far away from each other in the image have a very small likelihood of belonging to the same region. So, a sparse sampling scheme is adopted where each pixel is connected to all pixels around it falling only within some radius (dense radius). Further, it is randomly connected to pixels that fall outside this circle but within another circle of a larger radius (maximum radius) so that the total number of connections for a pixel is a constant. This results in a sparse matrix **W**.

3.2.9 Two-step segmentation procedure

The weight matrix **W** is only an approximation to the ideal weight matrix for two reasons. The first reason is it is forced to be sparse to allow for computational feasibility. The second is that the scales used for texture and color descriptors are only estimates computed using texton channels. Region boundaries need to be considered to determine exact scales. A two-step procedure is adopted to deal with these issues by starting with initial estimates of the scales computed using texton channels.

3.2.9.1 Step 1: Initial segmentation

An initial segmentation is intended to give an over-segmentation of the image so that no boundaries are missed. In other words, an image needs to be split up into a lot of regions. The image is partitioned in a single step utilizing the information contained in the higher order eigenvectors. Each pixel is mapped into a corresponding vector in a high dimensional space where each dimension is represented by the component coming from the corresponding pixel from a higher order eigenvector. A property of the higher order eigenvectors is that they put all pixels within coherent regions into tight clusters in the high dimensional space. These clusters are delineated using K-means to produce the initial segmentation [21].

3.2.9.2 Step 2: Final segmentation

The clustering process above renders an initial segmentation of the image that is usually an over-segmentation. Let this be called $\boldsymbol{S}_0$ and the number of segments in it be N_0. The initial segmentation provides a good approximation of region boundaries to modify the weight matrix. To compute the updated texton histograms for a pixel i in region R_k, textons are collected only from the intersection of R_k and the isotropic window of size determined by the texture scale, $\alpha(i)$. A similar approach is used to compute the updated color histograms. Contour probability is set to zero for pixels that are not in the region boundaries of $\boldsymbol{S}_0$. The modified weight matrix is an improvement over the original local estimation of weights. It is the assumption that initial over-segmentation of the image does not miss any region boundaries. Hence the set of boundaries in the desired final segmentation of the image is a subset of the boundaries in $\boldsymbol{S}_0$. For the next stage, each region in $\boldsymbol{S}_0$ is treated as a point in the graph and a condensed graph is obtained. The weight between two nodes (regions) in the condensed graph is the sum of the weights between all pixel-pairs such that first pixel belongs to one region and the second pixel belongs to the other region. Let $\hat{\mathbf{W}}$ represent the condensed weight matrix that is of size $N_0 \times N_0$. With the coarsened weight matrix $\hat{\mathbf{W}}$, a recursive procedure is followed to compute the final segmentation using *Ncut* threshold as the stopping criterion:

1. Compute the second smallest eigenvector for the generalized eigensystem using $\hat{\mathbf{W}}$ and the corresponding $\hat{\mathbf{D}}$ (see Section 3.2.2).
2. Threshold the eigenvector to delineate two groups of regions in the image. This produces a bi-partitioning of the image. A suitable threshold that minimizes the normalized cut value between the two partitions is chosen. The corresponding partition is the best way to segment the image into two regions.
3. Recursively repeat steps 1 and 2 for each of the partitions until the normalized cut value is larger than the *Ncut* threshold.

Fig. 3.7: O-a to O-h shows a few examples of natural images from the Corel image dataset segmented using the Normalized Cuts algorithm. These images show that although the Normalized Cuts algorithm tries to achieve most of the goals of good segmentation, it has a tendency to split up homogeneous regions into multiple segments. For example, in image O-b, the sky region is split into more than one segment. Similarly, the body of the bear in image O-g is over-segmented even though the different parts are visually coherent. The results suggest that investigating a few parts of the algorithm should lead to segments of better quality in terms of grouping together perceptually coherent regions. Towards this end, we recognize a few aspects of the algorithm that need attention. We propose modifications to these aspects and demonstrate improvement in performance after incorporating them. Some of the obvious problems that we recognized and our approach to deal with them are described below.

3.3 Proposed modifications to the Normalized Cuts algorithm

3.3.1 Averaging the weights

After obtaining the initial segmentation $\boldsymbol{S}_0$, each region in $\boldsymbol{S}_0$ is treated as a point in the graph for the next stage and a contracted weight matrix $\hat{\mathbf{W}}$ is computed using the following equation:

$$\hat{W}_{kl} = \sum_{i \in R_k} \sum_{j \in R_l} W_{ij} \tag{3.5}$$

where R_k and R_l indicate regions in $\boldsymbol{S}_0$. $\hat{\mathbf{W}}$ is the weight matrix of the coarsened graph and $\mathbf{W}$ is the weight matrix of the original graph where each pixel is a node in the graph. The idea is to treat each region in the initial segmentation as a point in the graph for the next stage. The final segmentation process merges some of these regions based on the normalized cut measure computed from the contracted graph. The weight between points (regions) k and l in the contracted weight matrix $\hat{\mathbf{W}}$ is the sum of the weights between every pixel pair i, j such that i is in k and j is in l. With this scheme there are two problems.

1. The self-weights $\hat{W}_{kk}$ in the contracted matrix are proportional to the size of the regions k because of the number of pixel pairs involved in the sum. Hence a region is more self-similar if it is larger in size and vice-versa. This leads to an asymmetry in the diagonal elements of $\hat{\mathbf{W}}$ whereas no such asymmetry existed in $\mathbf{W}$.
2. The weight $\hat{W}_{kl}$ between two regions k and l depends on the number of pixel-pairs that have non-zero weight W_{ij} in the summation in Eq. (3.5). In addition, because of the local connectivity assumption in $\mathbf{W}$, each weight $\hat{W}_{kl}$ depends on the length of the common boundary between regions k and l.

The first problem is eliminated by forcing the self-weights to be 1. The second problem can be overcome by averaging the pixel weights instead of summing them to obtain the region weights. With such a scheme, the region weights $\hat{W}_{kl}$ are given by:

$$\hat{W}_{kl} = \frac{1}{T} \sum_{i \in R_k} \sum_{j \in R_l} W_{ij} \tag{3.6}$$

where T is the number of pixel-pairs (i, j) that have non-zero weights between them. This forces all the weights $\hat{W}_{kl}$ to be in the range [0, 1] and hence removes the dependency on common boundary length.

3.3.2 Region-based texton and color histograms

Scale is important for texture and color descriptors at a pixel. But scale is not available unless perceptually coherent regions are grouped together which is essentially the goal of segmentation. This chicken-and-egg problem is solved iteratively by starting with initial estimates of different texture scales using texton channels. These estimates are then refined using boundaries obtained in the initial segmentation stage which is forced to be an oversegmentation. Based on the refined scale, texture and color descriptors are recomputed for the next stage of segmentation. Although it is a good way to re-estimate the scale, it does not exploit the fact that the regions obtained at any stage of segmentation are perceptually coherent and the texture and color within them is roughly uniform. This is evident from Fig. 3.4, which shows an initial segmentation of an image obtained using initial rough estimates of texture scales.

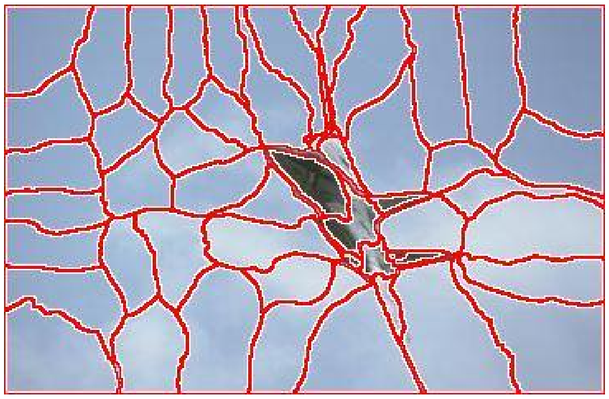
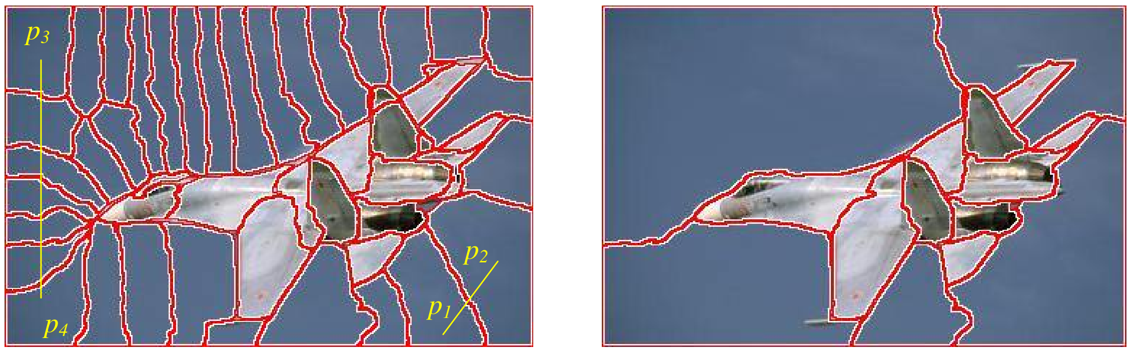


Fig. 3.4. *Initial segmentation*: *Regions obtained in the initial segmentation stage of an image are perceptually coherent. Region-based texture and color descriptors are computed for each region by considering all the pixels in the region.*

Recall that for the next stage of segmentation, the points in the graph are the individual regions of the initial segmentation. Pixel-based texture and color descriptors are computed and combined by Eq. (3.5) to obtain region similarities. It may be computationally wasteful to follow this approach given the fact that we have estimates of coherent regions at this stage. On the other hand, it would be wise to have a region-based texture and color descriptor instead of a pixel-based descriptor. That is, the texton and color histograms for a region are computed by taking into account all the pixels that lie within that region. The weights between region-pairs based on the texture and color cues are computed from the χ^2 distances between these histograms respectively. This scheme implicitly uses region sizes as scales for different textured regions. It is also computationally more efficient.

3.3.3 Meta-segmentation

To make the problem of solving for eigenvectors of a $N_{pix} \times N_{pix}$ weight matrix computationally tractable, a sparse sampling scheme is adopted to build the initial weight matrix **W**. In this scheme, every pixel has non-zero connections to only a few pixels in its neighborhood. Pixels that lie far apart in the image have zero weights on the edges connecting them. Even though they may belong to the same region, they may get separated during the initial segmentation stage. However in the next stage, each region is treated as a point and a contracted weight matrix is built. Now the weights indicate similarity between regions. This scheme tries to bring about a relationship between pixels that lie far way in the image. This can be illustrated with Fig. 3.5 below. The pixel pair p_1-p_2 is not connected in the initial estimate **W** because neither of them lies within the dense radius area of the other. However, they lie in adjacent regions (R_1 and R_2 respectively) in the initial segmentation. The weight $\hat{W}_{12}$ is non-zero in the contracted matrix obtained from initial segmentation and indirectly conveys similarity between pixels p_1 and p_2. This suggests that there is a possibility of declaring p_1 and p_2 as belonging to one region if the two regions R_1 and R_2 get merged in the final segmentation stage.



(a) (b)

Fig. 3.5. *Local connectivity*: *(a) Initial segmentation result on an image. Similarity between points p_1 and p_2 is implied by the non-zero weight between their corresponding regions in the contracted weight matrix. The same is not true for points p_3 and p_4. (b) Final segmentation output of the same image.*

The same thing is not true for the pixel pair p_3-p_4. Although both the pixels can be seen to belong to the same region (sky), they are not connected even after initial segmentation because the regions R_3 and R_4 to which they belong do not have a non-zero weight between them in the contracted matrix $\hat{\mathbf{W}}$. This is again due to the assumptions of local connectivity and the contracting scheme adopted in Eq. (3.5). Therefore, there is a tendency for pixels belonging to coherent regions to get separated if they lie sufficiently far away from each other in the image. The goal of grouping pixels globally is not completely achieved in this scheme. The main problem is the local connectivity assumption. The final segmentation step tries to circumvent it but stops after one iteration. This suggests that more iterations of the final segmentation step are needed. We call these iterations *Meta-segmentations* because they operate on regions. The so-called final segmentation step in the original Ncuts algorithm forms the first iteration in our *Meta-segmentation* framework.

However, the weight matrix contraction scheme according to Eq. (3.5) does not change the results after the first iteration of meta-segmentation unless the *Ncut* threshold is varied. This is because after the first stage of meta-segmentation (or the final segmentation step in the original algorithm) each point in the graph for the next stage is composed of a group of regions that have been combined together in the previous stage. Let these points for the next stage be called *super-regions* because each point is composed of a group of regions that have been joined together in the final segmentation stage. If any set of regions was combined in the previous stage to form a *super-region*, then it means that the *Ncuts* value between this set of regions as one partition and the remaining set as another partition was less than the threshold used to stop the recursive cut procedure. From Eq. (3.5) the weight between a *super-region* pair is obtained by simply summing the weights between region-pairs of the previous stage such that one region is a part of one *super-region* and the other one is a part of the other *super-region*. Hence, in the next stage a cut that isolates one of the *super-regions* from all others will have a normalized cut value less than the threshold used in the previous stage. So, if a single threshold is used for all the iterations, then the *super-regions* get singled out as separate segments in each iteration and the result will be the same as that after the first iteration of meta-segmentation.

The scheme of averaging the weights as in Eq. (3.6) circumvents the above problem. The self-similarity for the *super-regions* at any stage is forced to be 1 and the weights between the *super-region* pairs are obtained by averaging the weights between all region-pairs (of the previous stage) such that one region belongs to one *super-region* and the other region belongs to the other *super-region*. Then the normalized cut value obtained by separating out a single *super-region* from the rest is not necessarily less than the threshold. However it was less than the threshold by separating out the regions that formed this *super-region* from the rest of the regions in the previous stage. Thus, there is no need to change the *Ncut* threshold at each iteration of meta-segmentation. This is also meaningful in the sense that a single threshold implies that one is looking for a final segmentation such that the normalized dis-association between different regions is less than a fixed threshold for every image. This also gives way to a systematic scheme where the meta-segmentation can be iterated until there is no change in the number of regions for two successive iterations. That is, during each iteration of meta-segmentation, regions are merged if they are sufficiently similar and this merging stops when there is no possibility of any two regions being merged. The region merging procedure helps achieve the goal of grouping pixels globally even after starting with the local connectivity assumption between pixels.

The framework of *Meta-segmentation* gives a robust procedure to merge regions from a global perspective. This means that even though initial segmentations split up perceptually coherent regions into a lot of segments, the meta-segmentation iterations at a later stage will be able to merge them. But note that if any region boundaries are missed during initial segmentation, then they cannot be recovered at any further stage since all further stages always tend to merge regions. Hence, it is necessary to make the initial segmentation produce a lot of boundaries even though it means producing boundaries within coherent regions. We propose the following additional steps to the standard Normalized Cuts algorithm in an attempt to capture region boundaries as much as possible in the initial segmentation step.

3.3.4 Making the contour cue stronger

The intervening contour framework is used to obtain similarity between two pixels based on the contour cue, i.e., W_{ij}^{IC}. Recall that the final weights between pixel-pairs are obtained by multiplying the weights due to texture, contour and color cues as in Eq. (3.4). To make a cue stronger, the weight due to that cue can be raised to a higher power before multiplying it to obtain the composite weight. In order to better capture region boundaries, the weight due to contour cue W_{ij}^{IC} is squared before multiplying with the weights due to texture and color cues. So the composite weight between pixels *i* and *j* is now given by:

$$W_{ij} = W_{ij}^{TX} \times (W_{ij}^{IC})^2 \times W_{ij}^{COLOR} \tag{3.7}$$

This scheme will have a tendency to produce extra boundaries than necessary but it is possible to merge them in the meta-segmentation procedure.

3.3.5 Using average region color cue

In addition to the color histogram difference, color cue is also used by encoding it in the form of average region color for all iterations of meta-segmentation. The idea is to make the color cue stronger and further reduce similarity between regions that differ in their average color. To implement this scheme, we chose to use the rgS color space, where S = R+G+B, r = R/S and g = G/S. One reason to use this color space is that both the r and g color coordinates are normalized in the range [0, 1]. Hence, it is easy to compare distances in the rgS space if S is also normalized. Also it is shown to exhibit minimal amount of correlation between the color axes [22]. To compute distances in the rgS space, the S plane of the input image is also normalized in the range [0, 1]. The weight between two regions *k* and *l* based on the average color cue, $\hat{W}_{kl}^{AVG\_COLOR}$, is a function of the Euclidean distance between the average colors.

Precisely, the similarity between two regions is an exponential function of the Euclidean distance. Let $\bar{r}_k$, $\bar{g}_k$, $\bar{S}_k$ be the average r, g, S values for region *k* and similarly $\bar{r}_l$, $\bar{g}_l$, $\bar{S}_l$ be the average values for region *l*. Then $\hat{W}_{kl}^{AVG\_COLOR}$ is given by:

$$\hat{W}_{kl}^{AVG\_COLOR} = \exp(-d_{kl} / \tau_{AVG\_COLOR}) \tag{3.8}$$

where d_{kl} is the Euclidean distance between ($\bar{r}_k$, $\bar{g}_k$, $\bar{S}_k$) and ($\bar{r}_l$, $\bar{g}_l$, $\bar{S}_l$),

$$d_{kl} = \sqrt{(\bar{r}_k - \bar{r}_l)^2 + (\bar{g}_k - \bar{g}_l)^2 + (\bar{S}_k - \bar{S}_l)^2} \tag{3.9}$$

The value of τ_{AVG_COLOR} is chosen to be 0.3 so that the weight between regions reduces to 0.5 if the Euclidian distance is about 0.2. The exponential weighting function for the average color similarity is as shown in Fig. 3.6.

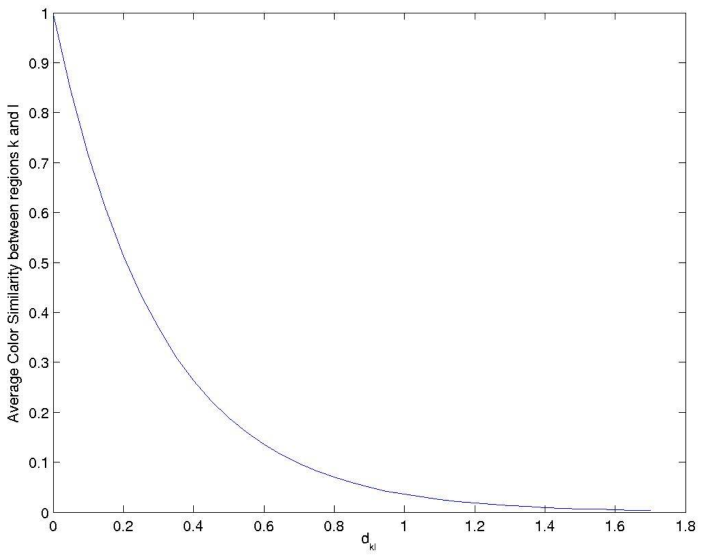


Fig. 3.6. *Weighting function*: *Exponential weighting function for average color similarity between regions.*

During any stage of meta-segmentation, the total similarity weight between regions is computed by combining the similarity weights calculated using region-based texton and color histograms, average region color and contour cues. The similarity weight between regions based on contour cue is obtained as in the original version but raised to the power of 2 to make it stronger. Thus the combined weight $\hat{W}_{kl}$ between regions *k* and *l* during any iteration of meta-segmentation is given by:

$$\hat{W}_{kl} = \hat{W}_{kl}^{TX} \times \left(\frac{1}{T} \sum_{i \in R_k, j \in R_l} \left(W_{ij}^{IC} \right)^2 \right) \times \hat{W}_{kl}^{COLOR} \times \hat{W}_{kl}^{AVG\_COLOR} \tag{3.10}$$

where,

$\hat{W}_{kl}^{TX}$	- Similarity computed using region-based texton histograms
$\hat{W}_{kl}^{COLOR}$	- Similarity computed using region-based color histograms
$\hat{W}_{kl}^{AVG\_COLOR}$	- Similarity based on average region colors
$\left(\frac{1}{T} \sum_{i \in R_k, j \in R_l} \left(W_{ij}^{IC} \right)^2 \right)$	- Similarity based on contour cue between regions

Also $\hat{W}_{kk} = 1$ for all regions k during any iteration of meta-segmentation. Note that we also chose to compute the color histograms in the rg color space to be consistent. Recall that in the original version a*b* space was used for color histogram computation.

After incorporating these modifications to the original version of the Normalized Cuts algorithm, the images M-a to M-h of Fig. 3.7 show segmentation results for comparison with the ones in O-a to O-h.

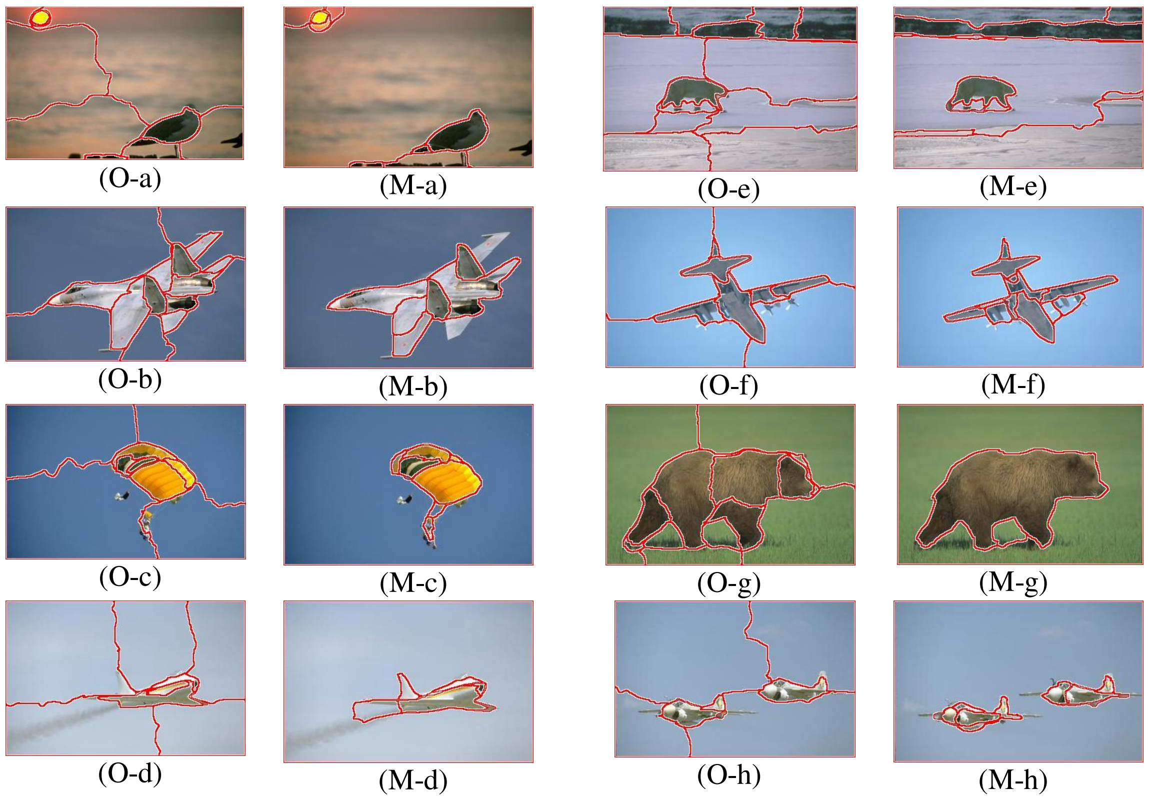


Fig. 3.7. *Modified Normalized Cuts*: *Comparison of segmentations produced by original and modified versions of the Normalized Cuts algorithm. (O-a)-(O-h) show sample segmented Corel images produced by the original Ncuts algorithm. (M-a)-(M-h) show the segmentations on the corresponding image after incorporating the proposed modifications.*

The quality of segmentation in these images can be seen to be superior in comparison with those obtained with the original Normalized Cuts algorithm, at least from a visual perspective.

3.3.6 Faster soft update scheme for color histogram computation

In addition to the modifications proposed above, we observed that a significant amount of time is spent in the soft updating scheme of color histogram computation. Recall that in the soft updating scheme, when a color bin *k* is being incremented, the histogram values in the adjacent bins are also incremented. The increment at any neighboring bin is proportional to the value of a Gaussian at that bin. The center of this Gaussian coincides with the color value being considered presently (see Section 3.2.7.1). This requires that the computation of Gaussian samples be performed for every pixel because the present color value at a pixel can fall anywhere within the quantization region of bin *k*. Considerable speed up in time can be achieved by avoiding the Gaussian computation step at every pixel with a scheme as shown in the following figure.

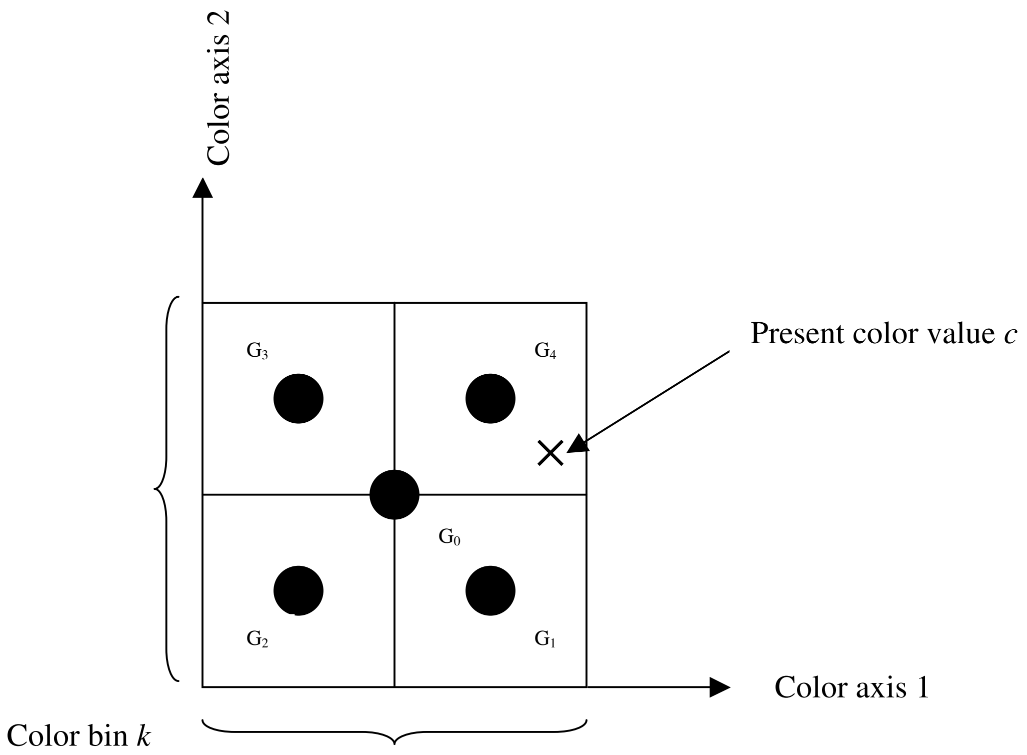


Fig. 3.8. *Faster soft update scheme*: *5 Gaussian scheme for faster soft update of color histogram. G_0, G_1, G_2, G_3, and G_4 are the centers of the 5 precomputed Gaussians. c is any color value within the quantization region of bin k. In the original scheme, the Gaussian for soft update is centered at c. In the faster scheme, the precomputed Gaussian with center at G_4 is chosen since this is the nearest to c.*

We precompute 5 Gaussians. The centers of the 5 Gaussians are systematically chosen so as to minimize the approximation error. One Gaussian is centered at the bin center. The other Gaussian centers are situated at equal distances from the central Gaussian as shown in Fig. 3.8. Whenever a color value within bin *k* occurs, the Gaussian with its center nearest to the color value is chosen. This Gaussian is used for the soft updating scheme. The advantage is that any one of the 5 Gaussians is used every time for soft updating and hence they need to be calculated only once and stored in memory. This leads to a significant improvement in time performance. The original version of the algorithm takes approximately 720 seconds on an average for a single image on a Pentium 4 machine whereas the modified version takes approximately 310 seconds. The comparison is based on images of size 364x236. Although this is an approximation, visual inspection of results on a number of images shows very little difference between the 5 Gaussian soft update scheme and the original one.

3.4 Evaluation of the modified version vs. original version

Comparison of images in Fig. 3.7: (O-a)-(O-h) with those in (M-a)-(M-h) illustrates the usefulness of the proposed modifications to achieve better grouping. For purposes of completeness and to further investigate if the Normalized Cuts algorithm with the modifications incorporated can give better performance in terms of word prediction, a plot of annotation performance as a function of number of regions is shown below.

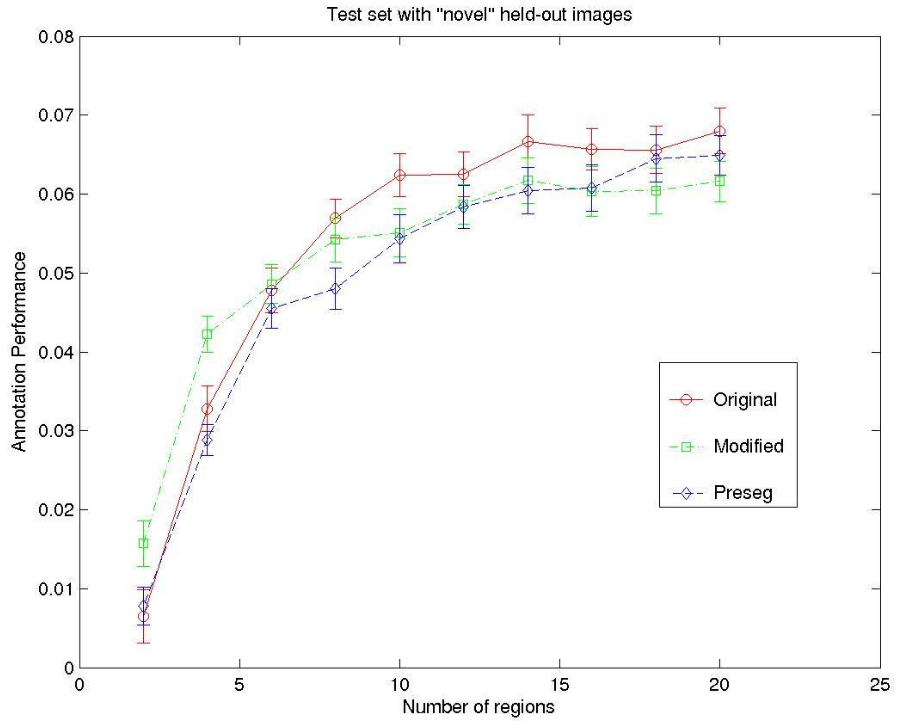


Fig. 3.9. *Normalized Cuts – original vs. modified*: *Comparison of the original and modified versions of the Normalized Cuts segmentation algorithm applied to the task of word prediction.*

Again the performance is a function of the number of regions used for annotation. For smaller number of regions (<6), the modified version performs better than the original version. However it degrades slightly as the number of regions increases. We suspect that the nature of the Corel dataset is partly responsible for the mixed behavior of different segmentation algorithms being compared here. This is because presently the system is over-reliant on color features as is clear from the feature evaluation experiments in Chapter 2. If a segmentation method is able to group homogeneous regions like “sky” or “water” into single segments, then in most of the Corel images these segments form the first few largest regions. However this is not necessarily true with those segmentation algorithms that have a tendency to split up homogeneous regions. The modified version of Normalized Cuts and Mean Shift algorithms are examples of the former class and the original version of Normalized Cuts and its Preseg version are examples of the latter class of segmentation algorithms. This can be seen from a few segmentation results illustrated in Figs. 3.1, 3.2 and 3.7. With the annotation approach, where the word distributions from the first few largest regions are used to annotate a test image, the former class of algorithms is bound to perform equivalently well or better than the latter class using fewer regions. This is because the more common words (“sky”, “water”, “people”) that are frequent in the true annotations of Corel images go well with the first few largest regions using the first class of segmentation algorithms, but not necessarily using the second class of segmentation algorithms. However, as the number of regions for annotation increases, contributions from smaller unreliable regions is more probable with the first class of segmentation algorithms than with the second class. This is more or less the trend suggested by the curves. A more robust evaluation of different segmentation algorithms can be carried out using the shape feature. Better segmentation algorithms lead to better shape characterization, thus enabling shape to contribute towards the process of annotation. But this is not possible unless grouping techniques become available that can isolate individual objects in images. (See also Chapter 2.) Inspection of results of the modified Normalized Cuts algorithm on a number of images clearly demonstrates better grouping ability of the algorithm in comparison to the original version. Possibly it could be further corroborated by carrying out a human evaluation of segmentations as in [27].

Chapter 4

EVALUATION OF COLOR CONSTANCY ALGORITHMS

4.1 Introduction

In this chapter of the thesis, we use the translation model of object recognition as a tool to evaluate computational color constancy algorithms. Color constancy is an area of research in itself that focuses on the effects of changes in scene color due to changes in the color of light illuminating the scene. If a scene is imaged under two different lights with different chromaticities, then there is invariably a color shift between the two images captured. A good explanation of this phenomenon is given in [39]. In simple terms, it is because a surface cannot reflect more than what is incident on it. Hence, a white surface, which ideally reflects all the wavelengths incident upon it, appears reddish when imaged under a red illuminant. This is because the incident light spectrum is more peaked towards the red region and hence the reflected light spectrum from the white surface also has this property. Color constancy algorithms attempt to compensate for this shift in color and derive an illumination independent description of the underlying scene. This can form an important preprocessing step in any object recognition system using color as a cue for recognition because it removes dependence of object color on illumination color. Here we test out the performances of a few approaches to computational color constancy using the translation model of object recognition. Word prediction performance is the quantitative measure for evaluation. Color is a prominent feature for the performance of this model. Hence changes in color due to illumination changes affect the recognition performance significantly. This chapter discusses the issues of color space choice, degradation due to illumination change and procedures for dealing with this degradation.

The testing approach adopted here and the results obtained also have implications in other areas of machine learning and computer vision. Illumination variation has always been a problem for many computer vision tasks. For example, consider a face recognition system that has been trained with a database of faces taken under only frontal illumination conditions. Unless the issues of varying illumination are specifically addressed while building the system, it is highly likely that its performance degrades considerably when presented with the same faces taken under a different lighting. To achieve robustness against lighting changes, one very popular approach has been to make the system learn about different lighting conditions it can encounter by presenting it with exemplars under those conditions [28, 29]. Another approach has been to compensate for changes in illumination by some kind of preprocessing like histogram equalization in order to remove the effects of illumination changes [48]. Although both the approaches claim to have achieved invariance to some extent, quantitative evaluation of the two approaches using the same system has been lacking. Since we adopt similar strategies to deal with illumination color change on the object recognition model here, the results will be suggestive of which would be a better strategy to follow.

The following section gives a brief overview of the effects of illumination color on the recorded image color by using a model of image formation in an image capture device. The method of simulating illumination changes to obtain experimental data is described in Section 4.3. Section 4.4 discusses a few computational approaches to achieving color constancy as a means of compensating for scene color changes due to illumination effects. The results of evaluating different color spaces for the task of word-prediction is given in Section 4.5. In Section 4.6, the effect of illumination color change on this task is considered. Sections 4.7 and 4.8 discuss results of using the two different strategies, discussed before, to compensate for illumination changes. Finally, Section 4.9 discusses how color normalization applied to Corel data set for training helps with the color constancy methods used.

4.2 Effects of illumination color on image color

When a camera captures an image of a scene, it records the light reflected from different surfaces present in the scene. Color cameras record both the relative intensities and colors of light reflected from the different surfaces. The image recorded by the camera is a function of illumination incident on the scene, the reflectance properties of the different surfaces and the characteristics of the camera. If the illumination incident on the scene changes, the appearance of different surfaces in the image changes. This is because a surface cannot reflect more than what is incident on it. For example, if a perfectly white surface is illuminated with a light whose wavelength spectrum is more peaked towards the red region, then the light reflected from that surface will also have this property. Hence the white surface appears reddish under such an illuminant.

An image capture device usually has a number of sensors tuned to different wavelength ranges of visible light spectrum. In conventional cameras, there are usually three sensors designated by R, G and B that are tuned to the red, green and blue regions of the visible light spectrum respectively. Each sensor integrates the incident light energy in its range of wavelengths to produce its response. Assuming a digital system, the output image is an array of pixels and each pixel consists of the different sensor responses centered over the same location. The appearance color and intensity of different objects are represented by values of these sensor responses at different pixels. The sensor responses are functions of the reflected light from different surfaces. The reflected light from a surface is dependent on the light illuminating the surface. Hence, the appearance color and intensity of objects are functions of the incident illumination. Computer vision systems that rely on appearance of objects in the input images are seriously affected by this phenomenon because appearance is influenced by the properties of the incident illumination. Specifically we are interested in changes in colors of objects in images due to changes in illumination color. The process of compensating for changes in scene appearance color due to changes in illumination color is called color constancy [30] and it is exhibited by the human vision system to some degree [31, 32, 33]. This process derives an illumination independent description of the scene content.

To illustrate the effects of change in image color due to change in illumination color, consider Fig. 4.1. It shows the image of a set of colored papers taken under two different illuminants [45]. The image on the left is taken under an illuminant for which the camera responses are well balanced and the image on the right is taken under an illuminant that is more bluish in color. Notice that there is a systematic change in color in the right image in that under the bluer light, all pixels seem to tend towards blue. It is this systematic change in response that forms the basis for formulation of computational color constancy algorithms.

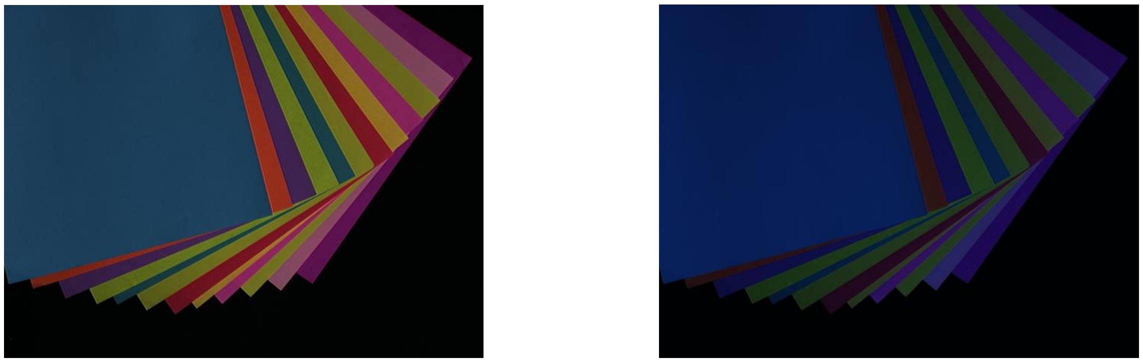


Fig. 4.1. *Color shift due to illumination change*: *(a) Image of a set of colored papers taken under an illuminant for which the camera responses are well balanced. This illuminant is the Sylvania 50MR16Q in [45]. (b) The same set of papers when imaged under a bluish illuminant (Solux-3500+3202) exhibit a systematic color shift towards blue in the image.*

4.3 Simulating illumination variation

To test the two different approaches for compensating for illumination variation in the general object recognition framework, a database of images under different illumination conditions is needed. Unfortunately, appropriate large-scale datasets with controlled illumination variation are not available. As a compromise, a semi-synthetic dataset was constructed as follows. A comprehensive controlled illumination data set was available [44, 45]. This data set was constructed to be representative of the changes in illumination chromaticity generally encountered. Each pixel in those images was scaled by the sum of R+G+B for that pixel so that the effect of overall brightness is removed. Let $r=R/(R+G+B)$, $g=G/(R+G+B)$, and $b=B/(R+G+B)$, be the normalized color values for a pixel. The dataset consisted of images of 17 different objects taken under 11 different illuminants. One of the illuminants was chosen to be the canonical illuminant (Sylvania 50MR16Q) for which the camera responses were well balanced. It is possible to approximate the mapping of an image of a scene taken under one illuminant to its corresponding image under a different illuminant using a matrix transformation [30]. That is the r, g, b values of pixels in the first image are multiplied with a 3-by-3 matrix to obtain the r, g, b values of pixels in the second image. The best 3-by-3 matrices (in the least squares sense) mapping the images under each of the 11 illuminants to the canonical illuminant were computed. To obtain the transformation matrices for an illuminant-canonical pair, all the pixels from all the images under the two illumination conditions were considered. Since the number of such pixel-pairs far exceeded the number of unknowns (9 for the 3-by-3 matrix), the least squares solution was computed.

Specifically, let $[r_1^c g_1^c b_1^c]^T$, $[r_2^c g_2^c b_2^c]^T$ … $[r_N^c g_N^c b_N^c]^T$ be the pixel values under the canonical illuminant and $[r_1^i g_1^i b_1^i]^T$, $[r_2^i g_2^i b_2^i]^T$ … $[r_N^i g_N^i b_N^i]^T$ be the corresponding values under the illuminant indexed by *i*. Representing the transformation matrix by **T**, the color transformations due to illumination change can be written using matrix notation as:

$$\begin{bmatrix} r_1^i & r_2^i & \dots & r_N^i \\ g_1^i & g_2^i & \dots & g_N^i \\ b_1^i & b_2^i & \dots & b_N^i \end{bmatrix} = \mathbf{T} \cdot \begin{bmatrix} r_1^c & r_2^c & \dots & r_N^c \\ g_1^c & g_2^c & \dots & g_N^c \\ b_1^c & b_2^c & \dots & b_N^c \end{bmatrix} \tag{4.1}$$

$$\mathbf{B} = \mathbf{T} \cdot \mathbf{A} \tag{4.2}$$

where **B** is the matrix of color values from images under illuminant *i* and **A** is that under the canonical illuminant. The least squares solution for **T** is given by multiplying the pseudo-inverse of **A** with **B** as follows:

$$\mathbf{T} = \mathbf{B} \cdot \mathbf{A}^T \cdot (\mathbf{A} \cdot \mathbf{A}^T)^{-1} \tag{4.3}$$

Hence 11 transformation matrices were obtained for the 11 different illuminant-canonical pairs. One of these matrices is the identity, i.e. for the canonical-canonical pair. These 11 matrices were used to simulate illumination changes in the Corel data set. For each image, the gamma correction was removed. Gamma correction is a non-linear transformation applied to true image pixel values when they are input to a display device.

This is to compensate for the non-linear characteristics of display devices in transforming from input pixel intensity value to output voltage value for display [60]. The pixel values in the images of the Corel dataset are gamma corrected values and therefore to obtain true pixel values it is necessary to remove gamma correction. Then the pixels were normalized by their (R+G+B) to remove the effects of overall brightness and also to be consistent with the way the transformation matrices were obtained. Then one of these 11 matrices was applied to simulate an illumination change and the normalized color values were multiplied back by the corresponding (R+G+B) value for each pixel. For the experiments, the image features were computed based on the new (R, G, B) values. This process produced some (R, G, B) values that were above the usual maximum value of 255. When color constancy processing was applied to such images, values over 255 were used, but the image pixel values were truncated to 255 before they were used for recognition experiments. This simulation of illumination change is only a gross approximation of what would occur if the illumination striking the scene underwent analogous changes. For example, the process makes no sense for sources, such as the sky. However, the procedure is more justified if we think of the database as being prints of the images, not the scenes themselves.

4.4 Computational color constancy

Computational approaches to color constancy aim to compensate for changes in recorded image color due to changes in scene illumination color. A common approach is to derive an illumination independent description of contents of the scene. This is equivalent to deriving features that are illumination-invariant. These features can then be used in a computer vision system to achieve illumination-invariance in its overall functioning. Once an illumination independent description of a scene is available, it can be used to render an image of the scene as if it were imaged under an illuminant of our choice. A number of vision applications follow this approach of mapping to a standard illuminant referred to as the canonical illuminant. This is usually the illuminant for which the system is properly trained. Using an image mapped to the canonical illuminant space itself as the illumination invariant descriptor of the underlying scene has proved to be useful [39, 40, 36]. Many computational color constancy algorithms aim at converting an image taken under an unknown illuminant to the one under the canonical illuminant. This approach can do away with the intermediate step of first deriving an illumination invariant descriptor of the scene and then using it to render an image using the canonical illuminant. The mapping from the unknown illuminant space to canonical space can be found directly by exploiting statistical properties of scenes imaged under the two illuminants. A description of a number of color constancy algorithms is given in [30]. A brief description of two of those algorithms used in the context of this work follows.

4.4.1 Gray-world algorithm

Gray-world algorithm [22, 61] assumes that the mean color of any scene imaged under a given illuminant is some fixed known color referred to as “gray”. Therefore any scene imaged under an unknown illuminant will have its mean color equal to the gray color for the unknown illuminant. Similarly any scene imaged under the canonical illuminant will have its mean color equal to the gray color for the canonical illuminant. The gray-world mapping takes an image under unknown illuminant to that under canonical illuminant so as to make the gray assumption hold. In other words, the mean color of the mapped image should equal the canonical "gray". A number of such mappings could exist. But the gray-world algorithm works with a diagonal assumption that each channel in the image is scaled independently. For concreteness, let $[r \ g \ b\ ]^T$ denote the 3 channel responses at a pixel under an unknown illuminant. The mapping that takes these responses to the canonical space is of the form:

$$\begin{bmatrix} r_t \\ g_t \\ b_t \end{bmatrix} = \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix} \begin{bmatrix} r \\ g \\ b \end{bmatrix} \tag{4.4}$$

where, $[r_t \ g_t b_t\ ]^T$ is the corresponding estimated 3 channel response under the canonical illuminant and *a*, *b* and *c* are the diagonal scaling factors for the 3 channels.

A suitable value of "gray" is assumed (here it is estimated as the average of the entire Corel data set used in these experiments). Formally, let $\overline{r_c}$, $\overline{g_c}$, $\overline{b_c}$ denote the "gray" color for the canonical illuminant and $\overline{r}$, $\overline{g}$, $\overline{b}$ be the average image color for an input image taken under an unknown illuminant. Then the normalization is achieved if we perform the transformation of *r*, *g*, *b* values of image pixels to get r_t, g_t, b_t such that:

$$r_t = \frac{\overline{r_c}}{\overline{r}} r \tag{4.5}$$

$$g_t = \frac{\overline{g_c}}{\overline{g}} g \tag{4.6}$$

$$b_t = \frac{\overline{b_c}}{\overline{b}} b \tag{4.7}$$

Note that r, g, b are normalized color values, i.e., *r=R/*(*R+G+B*), *g=G/*(*R+G+B*), *b=B/*(*R+G+B*), to remove the effects of brightness during color constancy processing. To see how the gray world algorithm causes the average of the image to be the "gray" value, the average of the color values in the transformed image are given by:

$$\overline{r_t} = \frac{1}{S}\sum_{1}^{S}\left(\frac{\overline{r_c}}{\overline{r}} r\right) = \overline{r_c} \tag{4.8}$$

$$\overline{g_t} = \frac{1}{S}\sum_{1}^{S}\left(\frac{\overline{g_c}}{\overline{g}} g\right) = \overline{g_c} \tag{4.9}$$

$$\overline{b_t} = \frac{1}{S}\sum_{1}^{S}\left(\frac{\overline{b_c}}{\overline{b}} r\right) = \overline{b_c} \tag{4.10}$$

where *S* is the size of the image. Fig. 4.2 illustrates the process of achieving color constancy with gray-world normalization. In the figure, (a) shows an original image from the Corel data set and (b) shows a simulated illumination change (bluish illuminant) on it. The result of applying gray-world normalization to the original image is shown in (c) and that to the image under bluish illuminant is shown in (d).

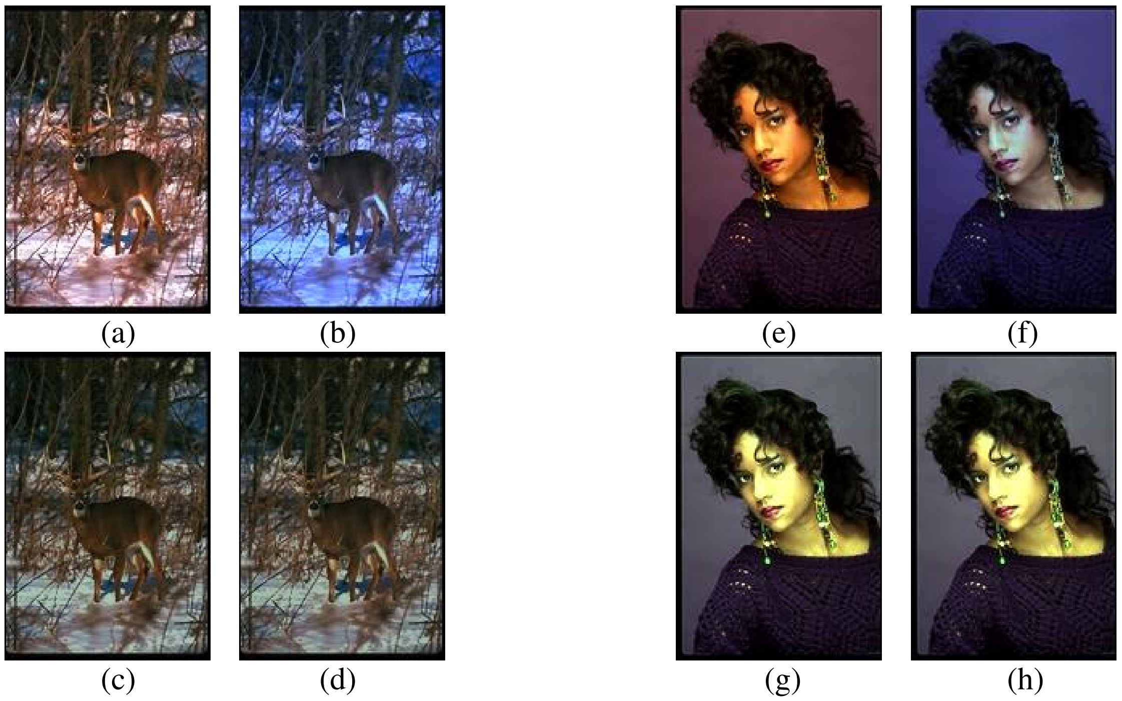


Fig. 4.2. *Gray-world color constancy*: *Illustration of gray-world color constancy processing. (a) Original Corel image. (b) Simulated illumination change image under a bluish illuminant. (c) Gray-world normalization applied to original image in (a). (d) Gray-world processing applied to the illumination-changed image in (b). Similarly for images (e), (f), (g) and (h).*

Note that the color shift between images in (c) and (d) is very little compared to the color shift between images in (a) and (b). Also observe that the original image and its normalized version are slightly different from each other. This is due to the fact that the gray-world assumption does not perfectly hold for the Corel data set. The “gray” value computed from the entire Corel data set does not seem to be exhibited by individual images in the dataset. If the gray-world assumption were to be true then the original image and its gray-world normalized image would exactly be the same. This fact about the Corel data set will be further proved by the word-prediction results that will be discussed towards the end of this chapter. The images in Fig. 4.2: (e), (f), (g) and (h) provide another illustration of the gray-world normalization process on a different image from the Corel data set.

4.4.2 Scale-by-max algorithm

Scale-by-max algorithm [54, 61] is similar to the gray world algorithm but it assumes that the maximum of scene color under any given illuminant is a fixed value. Then the image is normalized with each color channel being scaled independently so that the maximum assumption holds. Note that since each channel is being scaled independently of the other, again the diagonal model of illumination change is assumed. Let r_c^m, g_c^m, b_c^m be the maximum color values expected in each image (assumed maximum values for each channel). For the experiments here, r_c^m, g_c^m, b_c^m are estimated as the maximum values of color channels over all the images of the Corel dataset. If r^m, g^m, b^m are the maximum values in an image, then the transformations to obtain r_t, g_t, b_t are given by:

$$r_t = \frac{r_c^m}{r^m} r \tag{4.11}$$

$$g_t = \frac{g_c^m}{g^m} g \tag{4.12}$$

$$b_t = \frac{b_c^m}{b^m} b \tag{4.13}$$

As before, r, g, b are normalized color values. Using similar arguments as in Eq. (4.8), (4.9), (4.10), it is easy to see that the maximum of the image after normalization is r_c^m, g_c^m, b_c^m in the corresponding channels. The color constancy achieved with this algorithm is illustrated in Fig. 4.3. (a) shows an original image and (b) shows the same image under a bluish illuminant. (c) and (d) are scale-by-max normalized versions of the images in (a) and (b) respectively.

Again notice the relatively small color shift between images in (c) and (d) due to color constancy normalization with the scale-by-max algorithm. Also observe that there is negligible difference, if any, between the original image and its normalized version. This suggests that the maximum of the color channels computed from the entire Corel database correspond to the maximum in this particular image. Hence there is no effect of normalizing the original image. In fact, this is true of the entire Corel data set as will be proved by the word prediction results. This is also not surprising because the maximum in each channel came out to be 255 (absolute value), which is the maximum allowed in any color channel for the 8-bit Corel images used for the experiments. Fig. 4.3: (e)-(h) gives another example of color constancy normalization on a different image.

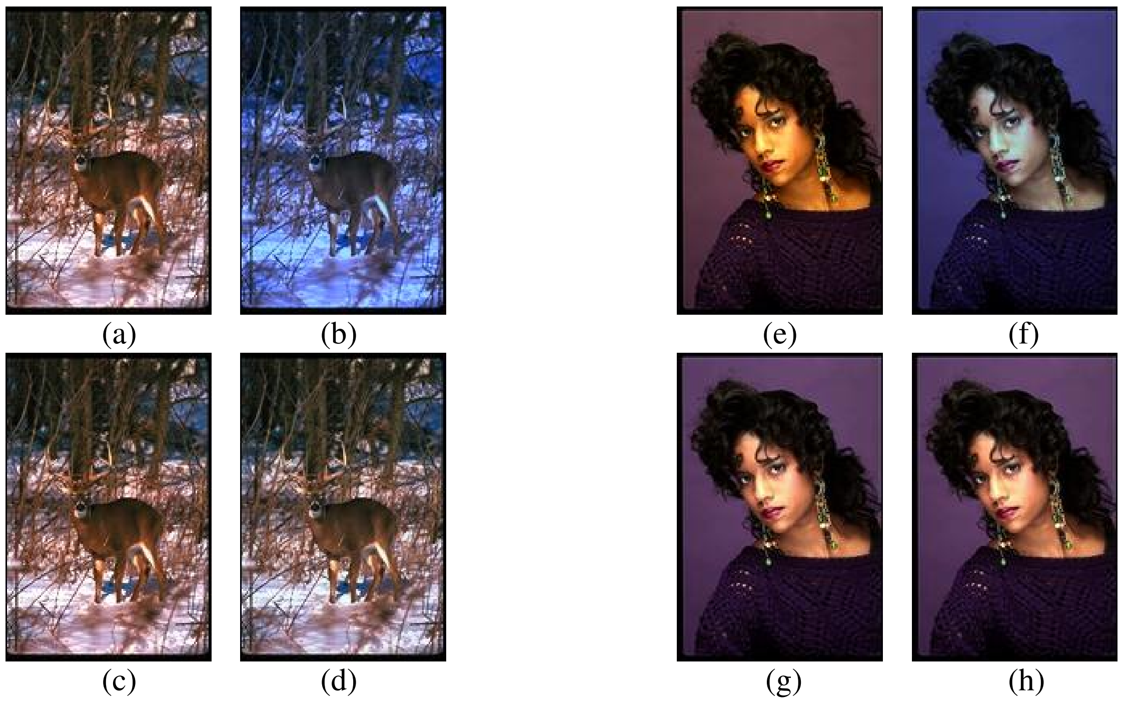


Fig. 4.3. *Scale-by-max color constancy*: *Illustration of scale-by-max color constancy processing. (a) Original Corel image. (b) Simulated illumination change image under a bluish illuminant. (c) Scale-by-max normalization applied to original image in (a). (d) Scale-by-max processing applied to the illumination-changed image in (b). Similarly for images (e), (f), (g) and (h).*

4.5 Color space evaluation

Various researchers have explored using different color spaces for different applications. Color space choice is often difficult and the choice should reflect the application. One issue is the degree to which the three values are correlated. For example, in natural images R, G, and B, tend to be correlated because variations in illumination intensity and direction (shading) cause similar effects on the 3 channels. In a normalized color space (r,g,S), this correlation is reduced and it can be further decorrelated using Principal Components Analysis (PCA) or Independent Components Analysis (ICA). Another issue is the degree to which the color space aligns with human perception. The CIE L*a*b color space was introduced to make the distances in color space roughly correspond to color differences as perceived by humans. In this section, the performance of these three color spaces on the task of word-prediction is evaluated. Since the system focuses on the canonical computer vision task – linking image features with semantics – it is likely that the results apply to other systems as well.

In all the color spaces considered both the average color and its variance over the region are used. Care is taken that in all tests the color features occupy the same proportion of total feature dimensionality. Duplicating the chosen color features appropriately does this. A weighting scheme is used for the average color and standard deviation features in each color space. The color features are duplicated as many times as the weight applied to the features in that color space. In the case of using all the 3 color spaces, a weight of 2 is used for average color features of R,G,B and L*a*b spaces and a weight of 1 is used for the standard deviation features in all spaces and the average color feature in rgS space. This results in a total dimensionality of 24 for color representation. Any weighting scheme could be used for the purpose here since the relative performance is of interest. But care has to be taken to keep the color feature dimensionality same across different experiments. This scheme was chosen just to be consistent with the prior work in [1, 5] that uses the same weights for the color features. The same dimensionality is obtained with a weighting factor of 4 for both the average color and standard deviation features in the case when only one of the color spaces is used. Word prediction performance using each color space is reported in Table 4.1. The table shows that both L*a*b and rgS spaces perform better than RGB space for this task. The improvement in performance is evident in all the 3 cases of training, held-out and novel test data.

Feature set	Word prediction performance on the various data sets (error is roughly 0.003)		
	Training	Held out	Novel
RGB, L*a*b, and rgS	0.140	0.090	0.055
RGB	0.112	0.064	0.044
L*a*b	0.148	0.096	0.059
rgS	0.149	0.094	0.060

Table 4.1. *Color space evaluation*: *Word prediction performance for the most common color spaces in computer vision. The numbers are the amount by which word prediction exceeds that of using the empirical distribution (bigger is better).*

4.6 Effect of illumination variation

To test the performance of the recognition system under varying illumination conditions, the model is trained using images from the original Corel data set. But the test images are chosen by sampling from the simulated illumination change data set. The sampling of this data set is done such that images under all 11 simulated illumination changes occur in roughly equal proportions. The results of this experiment are shown in row 2 of Table 4.2. The results clearly show that for this application, the range of illumination expected in natural images causes substantial degradation in performance. This is not surprising as color is an important feature for the translation process and varying illumination causes severe shifts in the color in test images.

4.7 Training with illumination variation

As discussed earlier, making the system learn about different illuminations by including exemplars taken under varying lighting conditions is a popular approach to deal with illumination variation. With the focus being on color here, it amounts to making the system learn about the variance of colors of objects under expected illumination changes [52, 53]. This hypothesis is tested here by including images from the simulated illumination change data set into the training set. One can include images under all 11 different illuminants for each scene during training. But this may demand a model with a larger number of nodes in order to better capture the conceptual entities in the 11 times larger dataset. Using a different sized model it would not be clear if the improvement/degradation in performance were due to the strategy of training with illumination change or due to the change in model. To use the model with the same number of nodes as for other experiments, image of each scene taken under one of the 11 illuminants is included in training. But each illumination receives equal share. The idea is that the model will be able to learn variance of colors of an object by observing different instances of it in different images taken under different illuminants in the training database. The results of this experiment are in row 3 of Table 4.2. Results show that exposing the training process to the expected illumination variation is helpful. The performance increases by about 31.5% in the training, 20% in the held-out and 50% in the novel test set. The significant improvement in performance on the novel test set might imply that training with illumination change improves generalization ability in addition to compensating for illumination change. Including images with different illuminations allows the system to see a particular object in different instances with greater color variation than that is possible without including them. This might be a plausible reason for improvement in generalization ability.

Experiment	Word prediction performance on the various data set (error estimates are shown in parentheses)		
	Training	Held out	Novel
No illumination variation	0.140 (0.003)	0.090 (0.002)	0.055 (0.005)
Train with no illumination variation and test with illumination variation	0.092 (0.0025)	0.060 (0.002)	0.030 (0.004)
Train and test with illumination variation	0.121 (0.003)	0.072 (0.002)	0.045 (0.005)
Train with no illumination variation and test with illumination variation and GW color constancy pre-processing	0.062 (0.003)	0.038 (0.003)	0.039 (0.003)
Train with no illumination variation and test with illumination variation and SBM color constancy pre-processing	0.122 (0.003)	0.082 (0.003)	0.053 (0.004)
Train with no illumination variation and GW normalization and test with illumination variation and GW color constancy pre-processing	0.121 (0.003)	0.073 (0.002)	0.053 (0.004)
Train with no illumination variation and SBM normalization and test with illumination variation and SBM color constancy pre-processing	0.135 (0.002)	0.086 (0.002)	0.059 (0.003)

Table 4.2. *Effects of illumination change and subsequent processing to deal with it*: *The numbers are amounts by which word prediction exceeds that of using the empirical distribution (bigger is better). The held out test set was composed of images hidden from the training process but from the same Corel CD's as the training data. The novel test set was composed of images from CD's different from those used in training. Errors were estimated based on the variance of the 10 samples taken. The results confirm that the range of color variation from typical illumination variation significantly degrade recognition system where color is an important cue, and that the right color constancy processing can help. In this data set, the conditions for scale-by-max are good, and it is clearly better than the gray world method. Further, if it makes sense for the application, applying color constancy to the training data (bottom two rows) can improve performance even further. This is the "normalization" strategy.*

4.8 Color constancy preprocessing

The other obvious solution to the illumination variation problem in object recognition is color constancy pre-processing. For the experiments here, two simple color constancy methods: gray-world (GW) and scale-by-max (SBM) are used. Descriptions of these color constancy methods were given in Section 4.4. For the gray-world method using mean statistic of the image or “gray”, expected value of the (R, G, B) over all 34,000 Corel images was estimated by averaging the (R, G, B) of all the pixels of all the images in the database. This “gray” came out to be (52.9, 51.0, 43.0). The color cast from the images is removed by assuming that the average (R,G,B) for each image is the “gray” value, and that the diagonal model of illumination change holds. With the scale-by-max method, each channel in an image is scaled so that the maximum in the image is that observed in the entire data set. For the Corel data set this maximum came out to be 255 for each channel.

Note that for this experiment, color constancy preprocessing is applied to only test images and training is carried out using images from the standard Corel database. The results with gray-world processing are in row 4 of Table 4.2 and with scale-by-max processing are in row 5. Note that scale-by-max method performs much better than the gray-world method. This can be reasoned out by observing the images in the database. The color balance of many or most of them is consistent with the maximum in each channel being close to 255. There are exceptions, such as the entire CD of sunsets, but each CD makes up less than 1% of the data. On the other hand, the gray-world assumption that the average of each image is a constant equal to “gray” does not hold for this data set. And attempting to deal with illumination change by exploiting it did not yield good results.

4.9 Color normalization

As a final experiment, the same normalization, either GW or SBM is applied to the training data as well as the test data. This approach does not make sense if the reference data is simple objects. Consider a scenario where there is an image consisting of a red ball only and an image consisting of a green ball. With the gray-world or scale-by-max method, both the images are exactly the same after normalization. Hence for an application where the intention is to recognize a ball of a particular color, there would be no difference between these 2 images as seen by the system. However, for the task here, the training images are neither treated as objects to be recognized, nor images to be found. Rather they are used to learn about image regions from images that typically have a wide range of colors. Thus training in a normalized space might make sense if illumination variation is expected and this is what is suggested by the results. Using this strategy improves upon that possible using color constancy processing for the test images only.

For the scale-by-max algorithm, the increment in performance with normalization over just color constancy preprocessing is small compared to that for the gray-world algorithm. This is because, with normalization we are making the assumptions of the algorithms hold well on images of the Corel dataset. The SBM assumption did hold well on the images even before normalization and hence a small difference is made by normalization. On the other hand, the GW assumption did not go well with the original Corel dataset. However we forced it by performing normalization on the training images and this is the reason for the significant improvement in performance in this case.

Chapter 5

CONCLUSIONS AND SCOPE FOR FUTURE WORK

In this thesis, a recently proposed model for object recognition viewed as translation from image regions to words is used to evaluate various computer vision tools within a single framework. There has not been a single general task to compare different low-level computer vision algorithms quantitatively. It is proposed here that the process of auto-annotation gives a good general evaluation tool. This is because auto-annotation has links to general object recognition and performance on this task can be measured on a large scale due to the availability of huge annotated image datasets. Specifically, evaluation of different feature sets, segmentation algorithms, and color constancy algorithms is performed in this thesis. The conclusions from these experiments are described in the Sections 5.1, 5.2 and 5.3 and a few possible directions for further research are identified in Section 5.4.

5.1 Evaluation of feature sets

Feature evaluation results suggest that color and texture are the two most important cues in that order for recognizing objects using this methodology. Color context information and an outer shape descriptor are incorporated in addition to the existing set of features. Experimental results show that context information helps in disambiguating objects that are similar with respect to other features. Outer shape descriptor and its Fourier transform carry usable information but fail to generalize well. Segmentation algorithms that split up images into meaningful semantic entities using both low level and high level cues are needed for shape to be useful.

5.2 Evaluation of segmentation algorithms

The use of the word prediction tool to evaluate different segmentation algorithms is demonstrated. Comparison of two classes of segmentation algorithms, namely Mean Shift and Normalized cuts is carried out using this tool. Results indicate that the performance is a function of number of regions used for annotation. This could be a side effect of the importance of color feature for this model. Segmentation algorithms that produce visually superior groupings perform better than others given that the number of regions used for annotations is small. But the performance degrades as the number of regions for annotation increases. Furthermore, Normalized cuts algorithm is considered in detail and modifications are proposed to improve the quality of grouping achieved with the algorithm.

5.3 Evaluation of color constancy algorithms

The effects of changes in image color due to change in illumination color on the translation model of object recognition is studied. Two paradigms for compensating for this are considered. One is to train the system for different possible illuminations by presenting it with exemplars taken under those conditions. The other is to use color constancy preprocessing. Results indicate that both the paradigms are useful. Two color constancy algorithms namely “gray-world” and “scale-by-max” are compared based on word prediction performance measure. The nature of the Corel dataset is well suited for using the scale-by-max algorithm. However normalization is required with the gray-world algorithm to make the underlying gray assumption to hold on images of the Corel dataset.

5.4 Scope for future work

It is possible that using some set of features together may lead to redundancy in representation of visual information. For example, color variance features may carry information about texture of a region and hence using the two together may be redundant. Experiments are needed to quantify the effects of feature redundancy during model training and subsequent word prediction. Transformations like Principal components analysis (PCA) and Independent components analysis (ICA) are aimed at reducing the correlation between different dimensions of a high dimensional space. The usefulness of these techniques to reduce the correlation between different feature dimensions is yet to be studied.

Present day segmentation algorithms use low level cues to form partitions in images and this is not sufficient to group together semantically meaningful entities. For example, using only low-level visual cues there is no way to group together black and white halves of a penguin into a single region. Some form of high-level information is needed to achieve this. It is possible to use word prediction as a tool to incorporate this information in segmentation algorithms. For example, in the Normalized Cuts framework, a weight between regions could be computed based on their similarities in terms of word prediction to propose region merges. With a properly trained word prediction model it is possible that the black and white halves of a penguin may both imply a high probability for the word “penguin” suggesting a merge between the two halves. A systematic methodology for incorporating high-level information into segmentation algorithms is well within reach.

To study the effects of illumination change, an artificial illumination change is simulated in images of the Corel dataset. It is possible to build an annotated dataset with real illumination change in the images and use this dataset for the experiments to see the effects in a more natural setting. Further experiments using more sophisticated color constancy algorithms are required to carry out a full evaluation of these algorithms on the task of object recognition.

REFERENCES

[1] K. Barnard, P. Duygulu, N. de Freitas, D. Forsyth, D. Blei, and M. I. Jordan, "Matching words and pictures," *J. Machine Learning Research*, vol.3, pp. 1107-1135, 2003.

[2] K. Barnard, P. Duygulu, and D. Forsyth, "Clustering art," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Hawaii, vol. 2, pp. 434-441, 2001.

[3] P.F. Brown, S.A.D. Pietra, V.J.D. Pietra and R.L. Mercer, "The mathematics of machine translation: Parameter estimation," *Computational Linguistics*, vol. 19, pp. 263-311, 1993.

[4] P.F. Brown, J.Cocke, S.A.D. Pietra, V.J.D. Pietra, F. Jelinek, J.D. Lafferty, R.L. Mercer and P.S. Roossin, "A statistical approach to machine translation," *Computational Linguistics*, vol. 16, pp. 79-85, 1990.

[5] P. Duygulu, K. Barnard, J.F.G. de Freitas, and D. Forsyth, "Object recognition as machine translation-I: Learning a lexicon for a fixed image vocabulary," *Seventh European Conf. Computer Vision*, vol. 4, pp. 97-112, 2002.

[6] C. Carson, S. Belongie, H. Greenspan, and J. Malik, "Blobworld: Image segmentation using expectation maximization and its application to image querying," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 8, pp. 1026-1038, August 2002.

[7] J. Shi and J. Malik, "Normalized cuts and image segmentation," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, San Juan, Puerto Rico, pp. 731-737, 1997.

[8] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603-619, May 2002.

[9] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Royal Statistical Society B*, vol. 39, pp. 1-38, 1977.

[10] R. C. Veltkamp, M. Hagedoorn, "State of the art in shape matching," Tech. Rep. UU-CS-1999-27, Utrecht University, Netherlands, 1999.

[11] H. Alt, L.J. Guibas, "Discrete geometric shapes: Matching, interpolation and approximation - a survey," J.R. Sack, J. Urrutia, editors, *Handbook of Computational Geometry*, pp. 121-153, Elsevier Science Publishers B.V. North-Holland, Amsterdam, 1999.

[12] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 24, pp. 509-521, April 2002.

[13] J. Neumann, H. Samet, and A. Soffer, "Integration of local and global shape analysis for logo classification," *Proc. 4th Int'l Wkshp. Visual Form* (*IWVF4)*, Lecture Notes in Computer Science 2059, Springer, pp. 769-778, Berlin 2001.

[14] E. Persoon, K.S. Fu, "Shape discrimination using Fourier descriptors," *IEEE Trans. Systems, Man and Cybernetics*, vol. SMC-7, no.3, pp. 170-179, March 1977.

[15] B. Menser, and M. Wien, "Automatic face detection and tracking for H.263 compatible region-of-interest coding," *Proc. SPIE Image and Video Communications and Processing,* vol. 3974, pp. 882-891, January 2000.

[16] S. R. Dubois, and F.H. Glanz, "An autoregressive model approach to two-dimensional shape classification," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 8, pp. 55-66, 1986.

[17] I. Sekita, T. Kurita, and N. Otsu, "Complex autoregressive model for shape recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 14, pp. 489-496, 1992.

[18] K. Arbter, W.E. Snyder, H. Burkhardt, and G. Hirzinger, "Application of affine-invariant Fourier descriptors to recognition of 3D objects," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 12, pp. 640-647, 1990.

[19] C.T. Zahn, and R.Z. Roskies, "Fourier descriptors for plane closed curves," *IEEE Trans. Computers*, vol. 21, no. 3, pp. 269-281, 1972.

[20] H. Kauppinen, T. Seppnäen, and M. Pietikäinen, "An experimental comparison of autoregressive and Fourier-based descriptors in 2D shape classification," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 17, no. 2, pp. 201-207, 1995.

[21] J. Malik, S. Belongie, T. Leung and J. Shi, "Contour and texture analysis for image segmentation," *Int'l J. Computer Vision*, vol. 43, no. 1, pp. 7-27, June 2001.

[22] G. Buchsbaum, "A spatial processor model for object color perception," *J. Franklin Institute*, vol. 310, pp. 1-26, 1980.

[23] G.H. Golub and C.F. Van Loan, *Matrix Computations*. Johns Hopkins Press, 1989.

[24] A. Pothen, H.D. Simon, and K.P. Liou, "Partitioning sparse matrices with eigenvectors of graphs," *SIAM J. Matrix Analysis and Applications*, vol. 11, pp. 430-452, 1990.

[25] T. Leung and J. Malik, "Contour continuity in region-based image segmentation," *Proc. European Conf. Computer Vision*, Freiburg, Germany, vol. 1, pp. 544-559, 1998.

[26] C. Fowlkes, D. Martin, and J. Malik, "Learning affinity functions for image segmentation: Combining patch-based and gradient-based approaches," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 2, pp. 675-682, 2003.

[27] D. Martin, C. Fowlkes, D. Tai, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," *Proc. Int'l Conf. Computer Vision*, vol. 2, pp. 416-421, 2001.

[28] L. Wiskott, J.M. Fellous, N. Kruger, and C. von der Malsburg, "Face recognition by elastic bunch graph matching," *Proc. IEEE Int'l Conf. Image Processing*, Santa Barbara, vol. 1, pp. 129-132, Oct. 26-29, 1997.

[29] A.S. Georghiades, P.N. Belhumeur, and D.J. Kriegman, "From few to many: Illumination cone models for face recognition under variable lighting and pose," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, pp. 643-660, 2001.

[30] K. Barnard, "Modeling scene illumination colour for computer vision and image reproduction: A survey of computational approaches," Simon Fraser University, Ph.D. depth paper, Dec. 1998.
http://vision.cs.arizona.edu/kobus/research/publications/index.html

[31] J.J. McCann, S.P. McKee, and T.H. Taylor, "Quantitative studies in Retinex theory," *Vision Research*, vol. 16, pp. 445-458, 1976.

[32] D. A. Brainard and B. A. Wandell, "Asymmetric color matching: How color appearance depends on the illuminant," *J. Optical Society of America A*, vol. 9, no. 9, pp. 1433-1448, 1992.

[33] M. Lucassen, "Quantitative studies of color constancy," Utrecht University, PhD. thesis, 1993.

[34] G. Sharma and H. J. Trussell, "Characterization of scanner sensitivity," *Proc. IS&T/SID Color Imaging Conf.: Transforms & Transportability of Color*, pp. 103-107, 1993.

[35] G.E. Healey and R. Kondepudy, "Radiometric CCD camera calibration and noise estimation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 16, no. 3, pp. 267-276, 1994.

[36] K. Barnard, "Computational colour constancy: Taking theory into practice," MSc. thesis, Simon Fraser University, School of Computing, 1995.

[37] P. L. Vora, J. E. Farrell, J. D. Tietz, and D. H. Brainard, "Digital color cameras–1–response models," available from: http://color.psych.ucsb.edu/hyperspectral/

[38] P. L. Vora, J. E. Farrell, J. D. Tietz, and D. H. Brainard, "Digital color cameras–2–spectral response," available from: http://color.psych.ucsb.edu/hyperspectral/

[39] D. Forsyth, “A novel algorithm for color constancy,” *Int’l J. Computer Vision*, vol. 5, pp. 5-36, 1990.

[40] G. D. Finlayson, “Coefficient color constancy,” Simon Fraser University, School of Computing, Ph.D. thesis, 1995.

[41] J. A. Worthey and M. H. Brill, “Heuristic analysis of von Kries color constancy,” *J. Optical Society of America A*, vol. 3, pp. 1708-1712, 1986.

[42] G. West and M. H. Brill, “Necessary and sufficient conditions for von Kries chromatic adaptation to give colour constancy,” *J. Mathematical Biology*, vol. 15, pp. 249-258, 1982.

[43] G. D. Finlayson, “Coefficient color constancy,” Simon Fraser University, School of Computing, Ph.D. thesis, 1995.

[44] K. Barnard, L. Martin, B. Funt, and A. Coath, “A data set for color research,” *Color Research and Application*, vol. 27, pp. 147-151, 2002.

[45] K. Barnard, L. Martin, B. Funt, and A. Coath, “Data for colour research,” available from: http://www.cs.sfu.ca/~colour/data

[46] P. Duygulu, K. Barnard, J.F.G. de Freitas, and D. Forsyth, “Object recognition as machine translation-I: Learning a lexicon for a fixed image vocabulary,” *Seventh European Conf. Computer Vision*, vol. 4, pp. 97-112, 2002.

[47] J. Shi and J. Malik, “Normalized cuts and image segmentation,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, pp. 888-905, 2000.

[48] J. Zhu, B. Liu, and S.C. Schwartz, “General illumination correction and its application to face normalization,” available from:
http://www.ee.princeton.edu/~juhuazhu/Acad/illum.htm

[49] K. Barnard and D. Forsyth, “Learning the semantics of words and pictures,” *Int’l Conf. Computer Vision*, vol. 2, pp. 408-415, 2001.

[50] P. Carbonetto and N. de Freitas, “Why can’t José read? The problem of learning semantic associations in a robot environment,” *Human Language Technology Conf. Wkshp. Learning Word Meaning from Non-Linguistic Data*, June 2003.

[51] B. Georgescu, and C.M. Christoudias, “Edge detection and image segmentation (EDISON) system,” available from:
http://www.caip.rutgers.edu/riul/research/code.html

[52] J. Matas, R. Marik, and J. Kittler, “On representation and matching of multi-colored objects,” *Fifth Int’l Conf. Computer Vision*, pp. 726-732, 1995.

[53] J. Matas, R. Marik and J. Kittler, “Illumination invariant colour recognition,” *5th British Machine Vision Conf.*, pp. 469-479, 1994.

[54] B.K.P. Horn, “Determining lightness from an image,” *Computer Vision, Graphics and Image Processing*, vol. 3, pp. 277-299, 1974.

[55] C. Carson, M. Thomas, S. Belongie, J.M. Hellerstein, and J. Malik, “Blobworld: A system for region-based image indexing and retrieval,” *Third Int’l Conf. Visual Information Systems*, pp. 509-516, 1999.

[56] C. Frankel, M. J. Swain, and V. Athitsos, “Webseer: An image search engine for the world wide web,” Tech. Rep. TR-96-14, University of Chicago Department of Computer Science, July 1996.

[57] M. La Cascia, S. Sethi, and S. Sclaroff, “Combining textual and visual cues for content-based image retrieval on the world wide web,” *IEEE Wkshp. Content-based Access of Image and Video Libraries*, pp. 24-28, Santa Barbara, CA, June 1998.

[58] R. Srihari, “Extracting visual information from text: Using captions to label human faces in newspaper photographs,” SUNY at Buffalo, Ph.D. thesis, 1991.

[59] R. Srihari and D. T. Burhans, “Visual semantics: Extracting visual information from text accompanying pictures,” *Proc. AAAI-94*, pp. 793-798, Seattle, 1994.

[60] R.W.G Hunt, *The Reproduction of Colour in Photography*. Printing and Television, Fourth edition, Fountain Press, England, 1987.

[61] G. Wyszecki and W.S. Styles, *Color Science: Concepts and Methods, Quantitative Data and Formulae*. Second edition, John Wiley & Sons, 1982.