Image Classification Using Color Cues and Texture Orientation

by

Elaine C. Yiu

Submitted to

the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

Massachusetts Institute of Technology

May 28, 1996

Signature of author [illegible]
Department of Electrical Engineering and Computer Science
May 28, 1996

Certified by [illegible]
Dr. Federico Girosi
Thesis Supervisor

Accepted by
. R. Morgenthaler
Chairman, Department Committee on Graduate Theses

Image Classification Using Color Cues and Texture Orientation

by

Elaine C. Yiu

Submitted to the
Department of Electrical Engineering and Computer Science

May 28, 1996

in partial fulfillment of the requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Image classification is a classical computer vision problem with many potential applications. As greatly increasing amounts of image and video are stored in computers, it becomes more difficult for human to categorize, store, and retrieve quickly for a particular scene or video clips. This project attempt to ease the process of sorting various scene into indoor or outdoor categories. We hope that the methodology we developed can also applied to more general sense of image classification problems.

Thesis Supervisor: Dr. Federico Girosi
Department of Brain and Cognitive Sciences

Acknowledgments

I would like to express my deepest gratitude towards my thesis advisor, Dr. Federico Girosi, who has patiently guilded me through every stages of this thesis. I am truely grateful for his continuous source of ideas and his help on editing the drafts of this thesis. His friendly support had made my research experience truly enjoyable.

Thanks to the following people who have made this thesis possible: to Professor Tomaso Poggio, for providing me a chance to work in the CBCL at MIT; to Kah-Kay Sung, for his advice and insights on image classification problems in the early stage of my research and also his source code for K-means algorithm; to Monika Gorkani for providing source code for the texture orientation module; to Edgar Osuna for providing guidance and source code to the Support Vector Machines.

To all my brothers and sisters in the Hong Kong Student Bible Study Group and Boston Chinese Evangelical Church, for their prayer support and friendship. Especially, Katherine, Jenny, Vivian, and Phoebe, who have not only shared my burdens, but have also challenged me in my daily journey of Christian life.

To Kin Hong, for his love and patience, his guidance and caring on my daily life, and his sense of humor and singing have given me the greatest joy.

To June, my little sister, for her troubles and "emergency" phone calls. Though you have added a little more crises to my life, I always love you. :)

To my parents, who introduced me the love of God and demostrated it in their love for me. Their phone calls from miles away and their unconditional support through all my endeavors are my greatest blessing.

To my Lord Jesus Christ, who is my refuge and strength, an ever-present help in trouble.

"Be still, and know that I am God;
I will be exalted among the nations,
I will be exalted in the earth."
Ps 46:10

Contents

Chapter 1

Introduction

Most of the information accessed everyday by millions of users is in the form of video, audio, pictures, or text. Except for text, that in many cases is accessible in a machine readable form, in all the other cases the problem of classifying, parsing or indexing this information is still a task performed by human operators, and it is mostly done using text information. While the value and the amount of the information available on-line is in principle immense, it is in practice limited by the limited amount of human resources that can process it. As the image databases are rapidly growing in size, the labor involved with sorting, indexing and classifying images by hand can be extremely costly. Searching a large database of unlabeled images for an image of a certain type or containing a certain pattern is still a very challenging task to automate, and the number of research groups around the world and private companies investing efforts in research topics of this type is growing.

In this thesis we take a concrete problem which has been proposed to us by Eastman Kodak Company: to classify images into indoor and outdoor categories. We have been provided by Eastman Kodak with a databases of several hundreds of color images of indoor and outdoor scenes, and our goal is, given a new image, that does not belong to the database, to decide whether it is an indoor or an outdoor scene. This can be seen as one of the many steps of an indexing process, that can be used, for example, to retrieve pictures with certain characteristics. Once we know that an image is outdoor, for instance, we could apply similar techniques to the ones presented in this thesis to decide whether the image contains a coastal scene or an alpine scene, or whether it contains people or not. One of the goal of this thesis is to investigate the use of global properties of the image, such as, color and texture information, for scene classification. The use of multiple cues poses a number of problems, one which being the problem of how to combine different classifiers, built using different input representations. As we will see in this thesis, this problem does not have just one solution, and in this work we will investigate two possible approaches. We hope, however, that the methodology we developed can also applied to a wider class of image classification problems.

Many applications in the field of image classification or scene analysis have been developed. These applications are mostly in the computer vision technologies. There are image recognition systems developed to diagnose patients in the medical environments. There are also computer vision system to analyse landscape and to identify weapons and nuclear warhead for military purposes. Recently, auto-pilot cars or driver's assistant systems have been tested in a lot of research laboratories. Many of these approaches are based on the computation of some statistical properties of the images, such a color histogram, that are approximately invariant across one class. So, for example, the amount of "blue" in the upper portion of an image could be assumed to be more or less invariant in all the pictures that contain sky, and could be used as one of many cues in an indexing process. Color is only one the many cues that can be used for indexing purposes, and it has been used successfully , for example in the work of described in [31] and [32]. Texture orientation is another potentially very useful cue as shown in [33] and [34]. Both color and texture orientation cues produce very positive result. In this project, we attempt to combine the two approaches in a sensible way in order to achieve better performance in scene analysis and thereby sorting the images correctly.

Among other projects in this area we can mention the MIT Media Lab Photobook [2], the Image and Advanced Television Laboratory of Columbia University [5], the CANDID Project, Los Alamos National Laboratory [3], the Jacob Project at the University of Palermo, Italy [4], Virage Incorporated [7], the QBIC project at IBM's

Almaden Research Center [6], the Image Science Group, Communications Technology Lab., at the Swiss Federal Institute of Technology [8], the Institute of Systems Science at the National University of Singapore, [1].

1.1 Overview

This thesis contains 6 chapters. In chapter 2 we describe our approach to scene classification, which is based on the use on multiple classifiers, that use color or texture information either independently or in a combined fashion. In chapter 3 we describe the classification techniques that we use in this thesis, that is the K-nearest neighbor method and the Support Vector Machines. We also describe two ways to linearly combine different classifiers in order to obtain a new classifier, whose performance are expected to be better than the best of the individual classifiers. In chapter 4 we present some preliminary data analysis, that motivated the choice of a non-trivial representations for the images in the texture feature space. In chapter 5 we discuss the experimental results. We present results for several different classifiers and for their linear combinations. In chapter 6 we draw some conclusions and present future work.

Chapter 2

Approach

As described earlier, there are many cues that we can use to classify images. In this thesis, we considered only two kinds of cues, color and texture orientation, that have been proved to be good descriptors of images in the past[35, 33], and that extract both spatial (i.e.. texture) and spectral (i.e.. color) attributes separately from a color image. In this thesis, we used a color histogram technique similar to the one used by Swain and Ballard [35] in order to represent the color information, and use the work on dominant orientation detection by Gorkani and Picard [33] for our texture representation. Our work is similar in spirit to the color/texture classifier developed by Tan and Kittler [31], that uses a parallel system to extract textural and color attributes separately in order to remove highly correlated information. The system for classifying a given image into indoor and outdoor categories is shown in Figure 2.

The input images are in PPM format, and their size is either 256x384 or 384x256. Each pixel of the image is represented by a red, green, and blue component, which has a value between 0 to 255. The input image will be processed into 2 different formats before feeding into the dominant orientation detector and the color histogram module. The dominant orientation detector requires the input images to be in grey-scale and the color histogram module requires the input images to be in L, C1, and C2 representation. Detail description of the modules and the image formats will be discussed in the following section.

After passing through the dominant orientation detector, the texture feature vec-

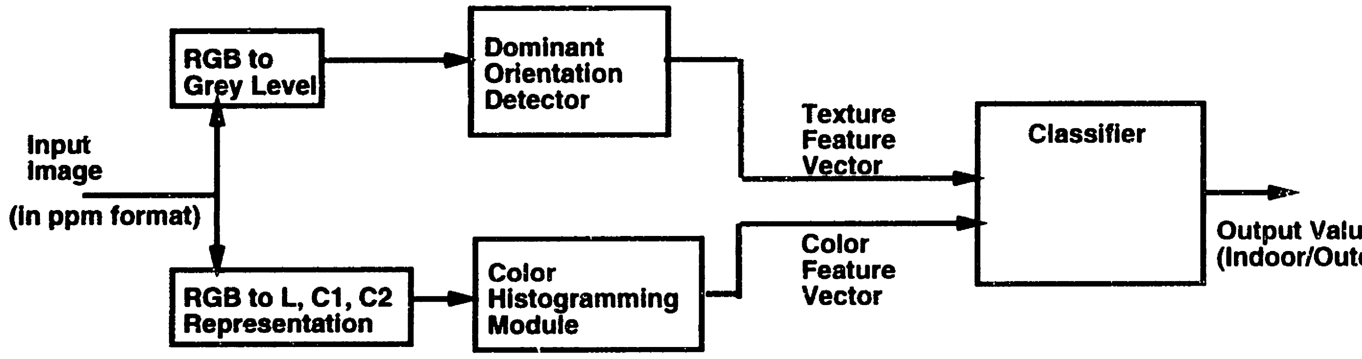


Figure 2-1: The Indoor/Outdoor Classifier

tor, a vector with 16 components, is produced. In parallel, the color histogram module, produces the color feature vector which is a vector with 34 components.

The classifier of this system contains a database of 400+ images. Their color feature vectors and the texture feature vectors were precomputed and stored. The Classifier uses Nearest-Neighbor [38], K-Nearest Neighbor, the Support Vector Machine [27], and the Boosting Algorithm [19, 20, 26] to classify the input image into indoor or outdoor categories. Detail description of the classifier and the classification algorithms will be discussed later.

2.1 Color Information

Color information has long been used as a recognition cue, and it is potentially extremely useful in variety of tasks. One reason for its importance is that it is a local surface property that is view invariant and largely independent of resolution [35]. In this thesis we used color histogram to be our color representation.

A color histogram is obtained by mapping the colors in the image array to a discrete color space containing n colors on a given color axes. A color histogram HM is a vector $(h_1, h_2, \ldots, h_n)$ in a n-dimensional vector space, where each element h_j represents the number of pixels of color j in the image M. All images have been scaled to contain the same number of pixels N before histogram. These histograms are the feature vectors to be stored as the index of the image database. In the next section we describe in details how each image in the database has been represented by mean of a color histogram.

2.1.1 Color Histogram

Color Axes and the Problem of Color Constancy

The effectiveness of the color histogram algorithm depends very heavily of the intensity and color of the scene illumination. This problem is related to the well known problem of color constancy, and comes from the fact that what is measured by the camera (for example the R, G, and B values) depends both on the color properties of the physical surfaces and the illumination of the scene. Therefore, changing the illumination will change the appearance of the colors in the image. While humans can be very good at discounting the illumination component, in machine vision this problem has been not fully solved yet, although many solution have been proposed.

In this thesis we are interested in using very simple and fast pre-processing techniques, that can be applied to a wide range of images, and therefore we cannot afford to use any of the available color constancy algorithms. However, it is well known that a certain degree of color constancy can be usually achieved by an appropriate choice of the color space.

The choice of the "best" color space is a problem that attracted a lot of attention in the machine vision community. Conventional color spaces [39] were defined either for colorimetry, such as the CIE tristimulus values and the CIE(x,y) chromaticity diagram, or for perceptual comparison and graphic arts, such as the Munsell system, the Natural Color System, and the Optical Society of America System. Although theses color systems provide good specifications of color information, they are not necessary suitable for machine vision and image processing applications [29].

In this thesis we adopted the physics-based color encoding model developed by Hsien-Che Lee at Kodak Inc. [29]. This model is based on the human visual systems, and provides a physical explanation of the similarity between the eigenvector color representation and the human opponent-color encoding. There are four ideas behind this effective color encoding:

1. An ideal sensor response function should match the signal statistics. The integral log-normal curve is chosen for this reason.
2. The color signals should be decomposed into one lightness component that represents the overall log intensity variation across all color channels, and two chroma components that represent the relative intensity variation between color channels.
3. The chroma components should be insensitive to change in surface orientation and illumination gradient.
4. One of the chroma components should be oriented along the direction of natural daylight variations.

This particular model is used because it presented a fairly simple transformation from the original color axes (RGB). We are particular concerned about the simplicity of the algorithm because we would like to be able to classify the images in a speedy manner. In additional, this model can also achieve illumination invariance which improve the effectiveness of the color histogram indexing algorithm.

Lee [29] showed that a "good" color space can be obtained by a 45° rotation of the RGB space. The new color variables are named L, C1, and C2, and are defined below:

$$\begin{aligned} L &= \frac{(\log R + \log G + \log B)}{\sqrt{3}} \\ C1 &= \frac{(\log R - \log B)}{\sqrt{2}} \\ C2 &= \frac{(\log R - 2\log G + \log B)}{\sqrt{6}} \end{aligned}$$

The L component of this new color vector represents the illumination, leaving the other 2 components to represent the chromatic information. Therefore, a simple way to achieve a certain degree of color constancy consist in encoding the color with the C1 and C2 components, disregarding the L variable.

The Histogram Space

After choosing the color axes, we need to define the histogram space H. Since all the images were scaled to contain the same number N of pixels, the histogram space H is the following subset of an n-dimensional vector space:

$$\mathcal{H} = \{(h_1, h_2, \ldots, h_n) \mid h_i \geq 0 \quad (1 \leq i \leq n), \ \sum_{i=1}^{n} h_i = N\}$$

where each element h_j represents the number of pixels of color j in the image that has to be encoded. Since the space of colors is clearly huge, some quantization is in order. We used the K-means clustering algorithm [24] in order to find a "reasonable" number of prototypical colors, that would represent well the color of the images in our database. We applied the K-means algorithm to our database of 400+ images in the (C1, C2) color space, and obtained 34 clusters. The center of the clusters are the "prototypical" colors, and have been used as histogram bins. Each color image is therefore represented by a vector of 34 elements, each element representing the number of pixels whose color belongs to a certain color cluster.

2.2 Texture Information

2.2.1 Dominant Orientation Detection

Texture information has been used extensively in computer vision, for recognition and discrimination purposes. In this work we concentrate on one specific kind of texture information, that is the texture dominant orientation. We believe that the distribution of the texture orientation in an image can be used to discriminate between outdoor and indoor image. This kind of feature has been already successfully used by Gorkani and Picard [33] to distinguish "city/suburb" scenes from "country" scenes. The basic idea in their work is that the city scenes have strong vertical and horizontal orientation since there is more man-made structure in a city/suburb scene, e.g. car, road, sign posts, buildings, windows, etc. We expect a similar phenomenon to be at least partially true for indoor/outdoor classification, e.g. we expect to see more regular patterns, with well-defined orientations, in indoor scenes than in outdoor scenes.

A number of approaches have been proposed to estimate texture orientation [23, 21, 22]. Most of them use estimates computed on the basis of the outputs of several directional filters applied to each pixel in the image. In this thesis we follow the approach of Gorkani and Picard [33] to extract texture dominant orientation. Unlike other approaches, they base their technique on the steerable pyramid of Freeman and Adelson [36] and therefore use information extracted at different scales. One interesting aspect of their technique is that they estimate some of the parameters of the technique by trying to match the behavior of their algorithm with the behavior of 40 human subjects. Interestingly enough, the parameters estimated in this way applied to different classes of problems with little or no modification.

In this work we applied their algorithm, which is described in details in [33], to estimate texture orientation in different portions of the image. More precisely, we divided each image I^j in 16 sub-images of equal size, and for each of the sub-images we used Gorkani and Picard's algorithm to derive two number: the direction of the dominant orientation and its strength. Each image is therefore represented as a 32-dimensional vector:

$$I^j = (O_1^j, S_1^j, O_2^j, S_2^j, \ldots, O_{16}^j, S_{16}^j) \tag{2.1}$$

where O_p^j and S_p^j are respectively the orientation and the strength of the p-th sub-image in image I^j. In chapter 4 we will argue that a more compact representation can be used, and in this thesis we will actually work with the one proposed in that chapter. However representation (2.1) leaves space to many different choices, and a priori information on the spatial distribution of texture orientation could be used if conjunction with it. For example, if the upper portions of the image is mostly blue and has very little texture, with no strong dominant orientations, it is very likely to contain sky, and therefore to be outdoor. If the upper portion were blue, but it contained some strong vertical and horizontal lines it is more likely to be an indoor scene, in which the ceiling happens to be blue and doors and windows generate strong dominant orientations.

2.3 Combining Color and Texture Information

Now that we have two different representations for the images in our database, we are left with the difficult task of combine them in a sensible way. There are two main approaches we can follow, that we briefly discuss here. The exact implementation will be discussed in the Chapter 5.

2.3.1 Single classifier with color and texture

An obvious solution consists in combining the color and texture representation in one single vector and train one classifiers using this input representation. There are some potential drawbacks of this technique:

1. The database is quite small, of the order of 400 images. This means that we should try to work in the least number of dimensions possible. Combining color and texture representation increases largely the number of variables involved, and therefore the number of examples needed in order to achieve a certain accuracy. It could be the case that the amount of information gained in using the combined representation is lost because of the increased complexity of the problem due to the larger number of variables.

2. If we combine color and texture information in one large vector we have to deal with the fact that the components of the vector have now different units of measurements. We already have this problem in the texture representation, that combines orientation and strength in the same vector. Of course, a sphering of the data could be performed, but having a large number of variables with

so few data points could lead to large errors. Therefore, methods based on the computation of distances, like nearest neighbors or Radial Basis Functions, are not easy to use in this representation. Techniques like polynomial classifiers are likely to be less sensitive to this problem. In the next section we will show how to use the method of Support Vectors to train a polynomial classifier that combines texture and color information.

2.3.2 Combination of color and texture based classifiers

An alternative to the idea of combining color and texture information in one single feature vector is to train two different classifiers and then combine the output of those. The problem of combining the output of different classifiers has received renewed attention in the last few years, and there is no easy solution. In this thesis we have the opportunity of dealing with classifier which differ in more than one sense. In fact we could choose on specific classifier, say nearest neighbor, and use it in the color and texture representation in order to obtain two different classifiers. But we can also train N different classifiers, using both color and texture information, and obtain $2N$ new classifiers. We will explore this approach in this thesis, where we will combine two different classifiers, nearest neighbors and Support Vector Machines, each trained separately using the color and texture representation.

There are many ways to combine different classifiers, and we will review some of them in the next chapter. Here we briefly describe the main issues arising with this problem. We discuss the case in which we actually have function estimators rather than classifiers, because it makes the discussion a little simpler, but conceptually there are no differences. Let $\phi_1(\mathbf{x}), \ldots, \phi_n(\mathbf{x})$ be n estimators for the same function. A common approach consists in taking a linear combination of them, and build a classifier of the form:

$$\phi(\mathbf{x}) = \sum_{i=1}^{n} \lambda_i \phi_i(\mathbf{x}) \tag{2.2}$$

A number of techniques of this type have been proposed, and they differ in several aspects:

1. the choice of the classifiers ϕ_i. In techniques like "bagging" [16] the classifiers ϕ are all of the same kind (say Radial Basis Functions with k centers), while in techniques like "stacked regression" they must be different (MLP, RBF, nearest neighbor ...). In other techniques, like "boosting" [26], it does not matter what the choice of the ϕ_i is.

2. the classifiers ϕ_i can be trained either on the same data set, or on different ones. In techniques like "stacked regression" the data set is unchanged, while the essence of "boosting' and "bagging" is in changing the data set.

A compelling reason for which combining different classifiers should lead to a classifier that perform better than the best of the single ones has not been given yet. If the estimators ϕ_i are unbiased, we can think of them as random variables whose average value is the "true" function. If the ϕ_i were all independent we could expect that taking a "weighted average", like in eq. (2.2), will reduce the variance, and therefore lead to an improvement in the generalization performances. But in practical cases the estimators ϕ_i will be far from being independent, and taking a linear combination could lead to a decrease of performances, rather than to an increase.

An alternative to the choice of eq. (2.2) is the following way of combining the ϕ_i:

$$\phi(\mathbf{x}) = \sum_{i=1}^{n} \lambda_i(\mathbf{x})\phi_i(\mathbf{x}) \tag{2.3}$$

The interpretation of a model of this type is the following: each of the estimators $\phi)_i(\mathbf{x})$ is fairly simple, but it is an "expert" in a certain region of the input space, where the function λ_i is significantly different from zero. The model (2.3) is therefore a smooth combination of experts, that assigns high confidence to one, or few, experts at a time, depending on the location of the input. This idea, that has been studied in a number of papers [10], is certainly interesting, but is also has some drawbacks. In particular, since the functions λ_i have to be learned from the examples as well as the "experts" ϕ_i, it is not clear whether there is enough information to do that or whether it would be a better idea to train just one, more complex, classifier rather than many simple ones.

Although convincing theoretical results that prove the validity of these approaches over standard techniques are still missing, there is now enough experimental evidence that techniques of type (2.2) or (2.3) can be very useful in practice, and we will experiment with two of these, that we will describe in more details in the next chapter.

Chapter 3

Classification Techniques

3.1 Nearest Neighbor

One of the oldest and simplest classification techniques is the *nearest neighbor* rule [9]. This decision rule assigns to an unclassified sample point the class label of the nearest point in a set of previously classified points [38]. This rule is based on the assumption that observations which are close together (in some appropriate metric) will have the same classification, or at least will have almost the same posterior probability distributions on their respective classifications. The nearest-neighbor rule is known to be sub-optimal, in the sense that its use will always lead to an error rate that is greater than the minimum possible, that is the Bayes rate. However, very well known results [38] showed that, when the number of samples is unlimited, the error rate is never worse than twice the Bayes rate. In fact, if P_n is the error rate on n samples, and if we define

$$P = \lim_{n \to \infty} P_n$$

then it can be shown that, in the case of binary classification,

$$P^* \leq P \leq P^*(2 - P^*)$$

where P^* is the Bayes error rate.

Of course, this result is only asymptotic, and should be taken with extreme care. However, the nearest neighbor decision rule has been proven to be effective in a number of cases, and the pattern recognition community has taken renovated interest in techniques of this type. Moreover, the nearest neighbor method can be used as a general indicator of the performances of other, more sophisticated techniques. In this thesis we decided to use the nearest neighbors rule, which is a very simple technique, and the Support Vector Machine, which is a far more sophisticated classifier, in order to explore the spectrum of the possible classification techniques. It is worth mentioning that Lipman and Yang are building a VLSI chip [11] that retrieves the k closest points in a large data base (up to 10^7 points in 256 dimensions).

A crucial choice in the nearest neighbor technique is the choice of the distance function. In this thesis we deal we a data set of images that is very often ambigous or can contain outliers, that is scenes that are classified as indoor but are actually outdoor or vice versa. For this reason the use of the standard Euclidean norm to compute the distance is probably not a very good choice, since it is well known not to lead to robust estimators [25]. A more stable measure of distance, less sensitive to outliers, is known to be the L_1 norm, which is the one we adopt in this work. If $\mathbf{f}$ and $\mathbf{g}$ are two d-dimensional feature vectors corresponding to two images, we therefore compute the distance as

$$d_{L_1}(\mathbf{f},\mathbf{g}) = \frac{1}{d}\|\mathbf{f}-\mathbf{g}\|_{L_1} = \frac{1}{d}\sum_{i=1}^{d}|f_i - g_i|$$

where $\mathbf{f} = (f_1, \ldots, f_d)$ and $\mathbf{g} = (g_1, \ldots, g_d)$.

3.2 K-Nearest Neighbor

The nearest neighbor rule can often be unstable and sensitive to outliers. A simple extension is the *k-nearest neighbor* rule, that consists in assigning to an unclassified point the class label that is most heavily represented among its k nearest neighbor. (k is odd to avoid ties). In the limit of a large number of data points, it can be shown that a large value of k will lead to consistent improvements over the standard

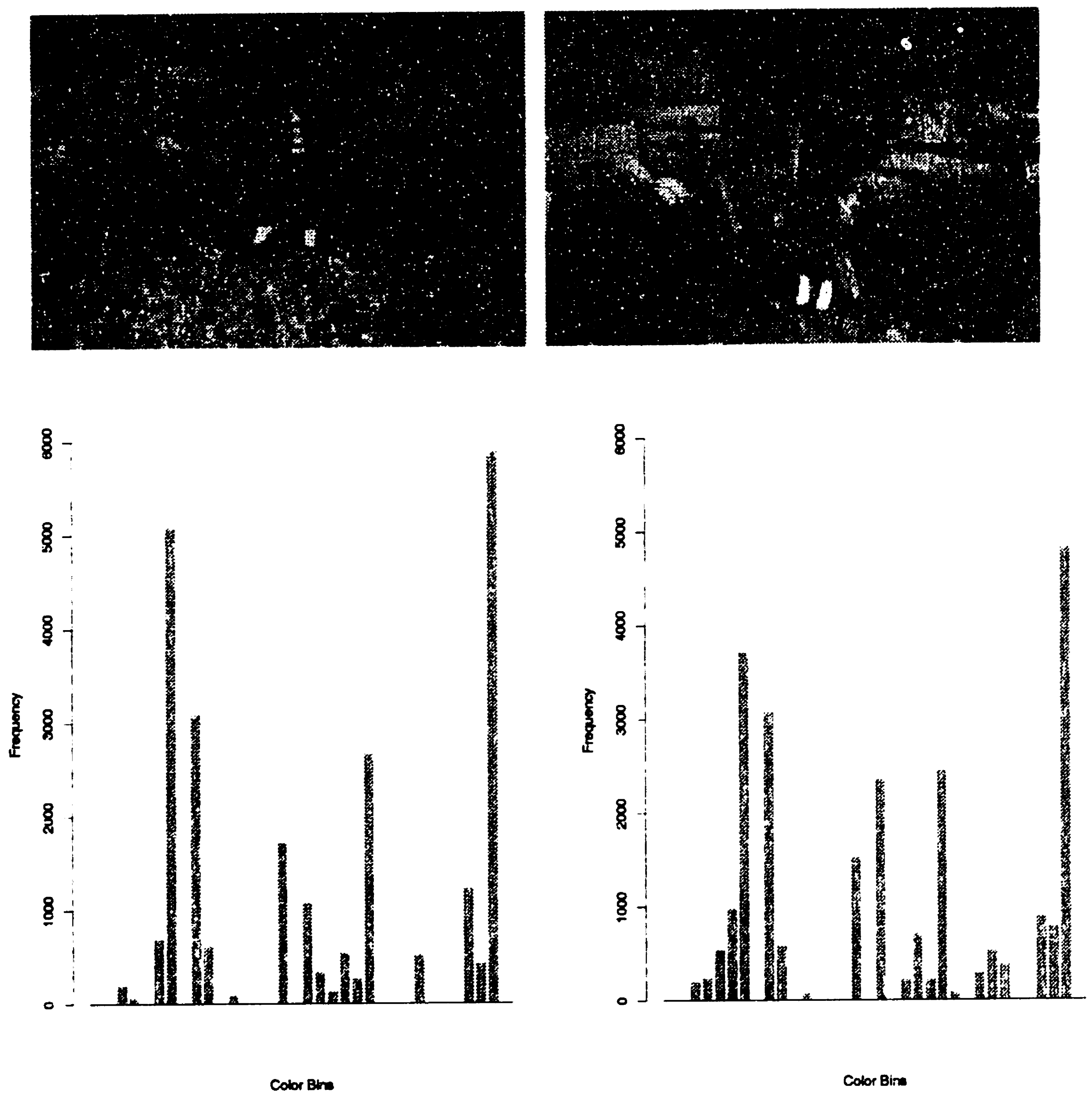


Figure 3-1: An example of a similar images which has similar color distribution (Distance=213.705).

nearest neighbor rule. However, for finite sample size, there is a trade off between the optimal value of k and the size of the data set. For small values of k an improvement can be usually obtained over the case $k = 1$, but when k is very large the size of the neighbor spanned by the k closest points becomes comparable to the size of the data set, and therefore the k-nearest neighbor rule will tend to classify points with the label of the most represented class in the data set, loosing its significance. In this thesis, we chosen $k = 3$.

3.3 Support Vector Machine

The Support Vector (SV) algorithm is a new and very promising classification technique due to Vapnik and his group at AT&T Bell Labs (Boser, Guyon and Vapnik, 1992; Cortes and Vapnik, 1995; Vapnik, 1995). The algorithm can be seen as a new training technique for polynomial, RBF or MLP classifiers. Vapnik and his group demonstrated excellent performances of this technique on an image classification task in the OCR domain. This technique is very new, and it has not been applied to many concrete problems yet. The only other application we are aware of is to the problem of detecting faces in images [30], where the SVM machine is used to classify between face and non-face patterns. While this technique seems very promising, it is still not clear what is the class of problems on which it performs well, and how it compares with other techniques, In this thesis we will compare it with the k-nearest neighbor technique, which is a much simpler classifier. We review here some of the key aspects of SVM machines.

For the case of two–class pattern recognition, the task of *learning from examples* can be formulated in the following way: given a set of functions

$$\{f_\alpha : \alpha \in \Lambda\}, \quad f_\alpha : R^N \rightarrow \{-1, +1\}$$

and a set of examples

$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l), \quad \mathbf{x}_i \in R^N, y_i \in \{-1, +1\},$$

each one generated from an unknown probability distribution $P(\mathbf{x}, y)$, we want to find a function f_{α^*} which provides the smallest possible value for the *expected risk*:

$$R(\alpha) = \int |f_\alpha(\mathbf{x}) - y| \, dP(\mathbf{x}, y).$$

The set of functions f_α could be for example the set of Radial Basis Functions or MLP with a certain number of nodes, and in that case the set Λ is the set of weights of the network. The problem is that $R(\alpha)$ is unknown, since $P(\mathbf{x}, y)$ is unknown, and therefore it cannot be minimized. The usual strategy consists of minimizing a stochastic approximation of $R(\alpha)$, the so called *empirical risk*:

$$R_{emp}(\alpha) = \frac{1}{l} \sum_{i=1}^{l} |f_\alpha(\mathbf{x}_i) - y_i|$$

It is well known, however, that minimizing the empirical risk does not necessarily mean minimizing the expected risk, especially if the number l of training examples is limited. Therefore, the technique of *Structural Risk Minimization* has been developed (Vapnik, 1982) to overcome this problem. The technique is based on the fact that for the above learning problem, for any $\alpha \in \Lambda$ with a probability of at least $1 - \eta$, the following bound holds:

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\frac{h\left(\log \frac{2l}{h} + 1\right) - \log(\eta/4)}{l}}. \tag{3.1}$$

where h is the *VC-dimension* of the set of functions f_α, that is a positive integer that describes the capacity of the learning algorithm (Vapnik, 1982). Equation (3.1) makes clear that, if the expected risk has to be minimized, *both* the empirical risk and the VC-dimension have to be made small, and this statement is usually called the *Structural Risk Minimization principle.*

The simplest version of the SV algorithm applies the Structural Risk Minimization principle to a linear classifier, that is the case in which the set of functions f_α is a set of linear decision surfaces

$$f_{\mathbf{w},b} = \text{sign}\,(\mathbf{w}\cdot\mathbf{x}+b)\,. \tag{3.2}$$

where we set $\alpha = (\mathbf{w}, b)$. Notice that if the parameters $\mathbf{w}$ and b are scaled by the same quantity the decision surface (3.2) is unchanged. In order to remove this redundancy, and to make each linear decision surface to correspond to one unique pair $(\mathbf{w}, b)$, we impose the constraint:

$$\min_{i=1,\ldots,r} |\mathbf{w}\cdot\mathbf{x}_i + b| = 1, \tag{3.3}$$

where $\mathbf{x}_1,\ldots,\mathbf{x}_r$ are the points in the data set, and call the set of hyperplanes which satisfy this condition *canonical hyperplanes*[1].

If no further constraints are imposed on the pair $(\mathbf{w}, b)$ the VC-dimension of the canonical hyperplanes is equal to $N+1$, that is the total number of free parameters. However, the VC-dimension can be made arbitrarily small by controlling the norm of the weight vector $\mathbf{w}$. In fact, assuming that all the points $\mathbf{x}_1,\ldots,\mathbf{x}_r$ lie in the unit N-dimensional sphere, the set $\{f_{\mathbf{w},b} : \|\mathbf{w}\| \leq A\}$ has a VC-dimension h satisfying the following bound (Vapnik, 1995):

$$h \leq A^2. \tag{3.4}$$

Assuming that the data set is linearly separable, the idea of the SV algorithm is therefore to find, among the canonical hyperplanes that correctly classify the data, the one with minimum norm $\|\mathbf{w}\|^2$, because if the norm is small the VC-dimension is also small. If a pair $(\mathbf{w}, b)$ separates the data, then the canonical condition (3.3) implies that

$$y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1, \qquad i = 1,\ldots,l. \tag{3.5}$$

and the SV algorithm consists in minimizing $\|\mathbf{w}\|^2$ under the inequality constraint

[1]Notice that all the linear decision surfaces can be represented by canonical hyperplanes. The constraint (3.3) acts simply as a "normalization". An alternative choice could have been to select $b = 1$, but that would have not generated any interesting result.

(3.5).

In many practical applications, a separating hyperplane does not exist. To allow for the possibility of examples violating (3.5), Cortes & Vapnik (1995) introduce slack variables

$$\xi_i \geq 0, \quad i = 1, \dots, l, \tag{3.6}$$

to get

$$y_i((\mathbf{w} \cdot \mathbf{x}_i) + b) \geq 1 - \xi_i, \quad i = 1, \dots, l. \tag{3.7}$$

In this case, the SV algorithm consists in minimizing:

$$\Phi(\mathbf{w}, \xi) = (\mathbf{w} \cdot \mathbf{w}) + \gamma \sum_{i=1}^{l} \xi_i \tag{3.8}$$

subject to the constraints (3.6) and (3.7). According to (3.4), minimizing the first term amounts to minimizing the VC–dimension of the learning machine, thereby minimizing the second term of the bound (3.1). The term $\sum_{i=1}^{l} \xi_i$, on the other hand, is an upper bound on the number of misclassifications on the training set — this controls the empirical risk term in (3.1). For a suitable positive constant γ, this approach therefore constitutes a practical implementation of Structural Risk Minimization on the given set of functions.

Introducing Lagrange multipliers α_i and using the Kuhn–Tucker theorem of optimization theory one can show that the solution has an expansion

$$\mathbf{w} = \sum_{i=1}^{l} y_i \alpha_i \mathbf{x}_i, \tag{3.9}$$

with nonzero coefficients α_i only for the cases where the corresponding example $(\mathbf{x}_i, y_i)$ precisely meets the constraint (3.7). These $\mathbf{x}_i$ are called *Support Vectors*, and (3.9) is the *Support Vector Expansion.* All the remaining examples $\mathbf{x}_j$ of the training set are irrelevant: their constraint (3.7) is satisfied automatically (with $\xi_j = 0$), and they do not appear in the support vector expansion. Although the solution $\mathbf{w}$ is unique, the coefficients α_i are not. They can be found by solving the following quadratic programming problem: maximize

$$W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j \mathbf{x}_i \cdot \mathbf{x}_j \tag{3.10}$$

subject to

$$0 \leq \alpha_i \leq \gamma, \quad i = 1, \ldots, l, \quad \text{and} \quad \sum_{i=1}^{l} \alpha_i y_i = 0. \tag{3.11}$$

By linearity of the dot product, the decision function (3.2) can thus be written as

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{l} y_i \alpha_i \mathbf{x} \cdot \mathbf{x}_i + b\right). \tag{3.12}$$

So far, we have described linear decision surfaces. These are not appropriate for all tasks. To allow for much more general decision surfaces, one first nonlinearly transforms the input vectors into a high–dimensional feature space by a map ϕ:

$$\mathbf{x} \rightarrow \phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \ldots, \phi_n(\mathbf{x})) .$$

Now the linear algorithm can be used in the feature space, using the features $\phi(\mathbf{x})$. Maximizing (3.10) then requires the computation of dot products $\phi(\mathbf{x}) \cdot \phi(\mathbf{x}_i)$. Under certain conditions (Boser, Guyon and Vapnik, 1992; Vapnik, 1995) these expensive calculations can be reduced significantly, since it is possible to show that there exists a function K such that

$$\phi(\mathbf{x}) \cdot \phi(\mathbf{x}_i) = K(\mathbf{x}, \mathbf{x}_i).$$

With this form of scalar product, the decision function (3.12) assumes the form:

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{l} y_i \alpha_i K(\mathbf{x}, \mathbf{x}_i) + b\right). \tag{3.13}$$

Therefore, rather than choosing the feature map $\phi(\mathbf{x})$ we only have to choose the the kernel K. Interestingly enough, some of the choices for the kernel K lead to decision

Kernel	classifier
$K(\mathbf{x},\mathbf{x}_i) = \exp(-\|\mathbf{x}-\mathbf{x}_i\|^2)$	RBF
$K(\mathbf{x},\mathbf{x}_i) = (\mathbf{x}\cdot\mathbf{x}_i)^k$	polynomial of degree k
$K(\mathbf{x},\mathbf{x}_i) = \tanh(\mathbf{x}\cdot\mathbf{x}_i - \Theta)$	MLP

Table 3.1:

surfaces that are Radial Basis Functions, Multilayer Perceptron or polynomials, as shown in table (3.1):
To find the decision function (3.13), we have to maximize

$$W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{3.14}$$

under the constraint (3.11). To find the threshold b, one takes into account that due to (3.7), for support vectors $\mathbf{x}_j$ for which $\xi_j = 0$ we have

$$\sum_{i=1}^{l} y_i \alpha_i K(\mathbf{x}_j, \mathbf{x}_i) + b = y_j.$$

This technique has been very successfully tested on an OCR problem at AT&T Bell Labs. by V. Vapnik and his collaborators. The reason for its success is that rather than minimizing the usual empirical risk, that is the training error, *it minimizes a bound on the generalization error.* Notice that the final decision surface can be a Radial Basis Functions or a Multilayer Perceptron network, which we know can usually model very well complex surfaces. The difference between this technique and the conventional RBF and MLP techniques is the way the parameters are found, and their interpretation.
The implementation of a SVM machine is not trivial, because it involves the solution of a quadratic programming problem, with linear constraints, in a number of variables equal to the number of data points. In this thesis we used the system developed by E. Osuna [30] at MIT, that is based on a commercially available software package called MINOS 5.4. MINOS 5.4 solves nonlinear problems with linear constraints using Wolfe's Reduced Gradient algorithm in conjunction with Davidson's quasi-Newton

method.

3.4 Linear Combinations of Elementary Classifiers

The idea of combining several different classifiers, trained to solve the same problem, in order to obtain a classifier which performs better than the single ones, is not a new one. However, a number of new ideas appeared recently in the literature, and some of these techniques begin to be used in practice, with very encouraging results (Breiman, 1993; Drucker, Schapire and Simard, 1993; Freund and Schapire, 1995). Here we review briefly some of the techniques that have been proposed.

3.4.1 Stacked Regression

Suppose we train k different predictors, such as Radial Basis Functions, Multilayer Perceptrons, kernel regression, linear models, on the same data set. A common procedure consists in choosing the "best" model by means of some cross-validation technique, that is estimating the generalization error for each model and then selecting the one corresponding to the best estimate. An alternative, proposed by Breiman (Breiman, 1993), consists in using the predictors $\{\phi_\alpha(\mathbf{x})\}_{\alpha=1}^k$ to produce a new predictor, which is a linear combination of the ϕ_α:

$$\phi(\mathbf{x}) = \sum_{\alpha=1}^{k} c_\alpha \phi_\alpha(\mathbf{x}) \tag{3.15}$$

In the case in which the problem to solve is a classification task a natural way to modify eq. (3.15) is simply to threshold it with an Heaviside function. In the technique proposed by Breiman (1993) the coefficients c_α form a convex combination, and are estimated by cross-validation. The convex combination guarantees that the generalization error of the new predictor ϕ is a weighted average of the generalization errors of the single predictors, but this is not enough to guarantee that it will perform better than the best of them. However, this technique performed well in a number of cases.

This technique has a even simpler implementation, in which the linear combination is simply an average rather than a weighted average. We notice that in the classification case this simplification is equivalent to take a majority vote of the different classifiers: a point is classified as class 1 if the majority of the classifiers agree that it is of class 1. While it is not clear why this should work better or worse than the original stacked regression idea, there is experimental evidence that hints that taking the average of different classifiers never hurts, and it can lead to improvements. This is the option that we explore in this thesis, where we will combine classifiers that differ for the kind of input representation (color/texture) and for architecture (support vectors versus k-nearest neighbor).

3.4.2 Bagging Predictors

Suppose we have k different data sets of a function:

$$D_\alpha = \{(\mathbf{x}_i^\alpha, y_i^\alpha)\}_{i=1}^l \ , \quad \alpha = 1, \dots, k \ ,$$

and let us denote by $\phi_\alpha(\mathbf{x})$ the predictors we obtained as the result of training a certain learning algorithm on the data set D_α. An obvious way to combine the different predictors ϕ_α is to take their average:

$$\phi(\mathbf{x}) = \frac{1}{k} \sum_{\alpha=1}^{k} \phi_\alpha(\mathbf{x}) \tag{3.16}$$

It is possible to show (Breiman, 1995) that, in the limit of k going to infinity, the predictor $\phi(\mathbf{x})$ will lead, on the average, to a generalization error which is smaller than the average generalization error of the single predictors $\phi_\alpha(\mathbf{x})$. The improvement in the generalization error is measured by the *variance* of the estimators $\phi_\alpha(\mathbf{x})$, that is by the quantity:

$$V = E_D[\phi_\alpha^2] - (E_D[\phi_\alpha])^2$$

where E_D denotes the average with respect all the data sets of size l. This technique therefore seems to work when the predictors ϕ_α are *unstable*, that is when they are very different when computed on different data sets. Examples of unstable predictors are neural networks and CART, while an example of *stable* predictors is nearest neighbor. In practice, of course, one does not have access to k independent data sets, but this idea can be still used if the data sets are generated by *bootstrapping*, that is if they generated by resampling k times, with replacement, the same data set. While this is not the only possibility, and other techniques of the type of *leave-n-out* type could be used, this seemed to be effective in a number of practical cases (Breiman, 95). Notice how bagging can be considered as a particular case of stacked regression.

Chapter 4

Data analysis and pre-processing techniques

It is often advantageous to apply pre-processing transformations to the input data before it is presented to a network [14]. Some simple form of pre-processing involve normalizing the input data. More complex pre-processing involved reduction of the dimensionality of the input data using principal component analysis. Some prior knowledge is also incorporated into the input data. This knowledge is some relevant information which might be used to develop better solution.

4.1 In-class and out-of-class average distances

The nearest neighbor decision rule classifies the test image into the category of its closest point in the data set. This technique will work well if there is a good margin between the two classes. In order to get an idea of how well the two classes are separated we computed the average in-class and out-of-class average distances. Let $\mathbf{x}^j$ denote a point of class j, with probability distribution $P_j(\mathbf{x}^j)$, then we define

$$d_{i,j} = E[\|\mathbf{x}^i - \mathbf{x}^j\|]$$

where $E[\cdot]$ denotes the expectation with respect to the joint probability $P(\mathbf{x}^i)P(\mathbf{x}^j)$.

Class 1	Class 2	Average distance	Standard Deviation
Indoor	Indoor	963.4371	83.63
Outdoor	Outdoor	1029.55	86.832
Indoor	Outdoor	1050.591	86.00

Table 4.1: Average Distance Using Color Feature Vectors

Class 1	Class 2	Average distance	Standard Deviation
Indoor	Indoor	33.98	12.95
Outdoor	Outdoor	23.64	9.9
Indoor	Outdoor	24.05	10.39

Table 4.2: Average Distance Using Texture Feature Vectors.

If the out-of-class distance $d_{1,2}$ were much larger than the in-class distances $d_{1,1}$ and $d_{2,2}$ we knew that the two classes were probably well separated, and we would expect a technique like nearest neighbor to work well. In table 4.1 and 4.1 we report the average and standard deviation of the in-class nad out-of-class distances, that makes clear that the indoor and outdoor images are not so well separated. In the texture domain there seems to be a larger degree of overlapping (the variances are much higher), so we will expect that any technique will do worse in the texture representation than in the color representation.

4.2 Extracting relevant features in texture space

In chapter 2 we discussed the representation of images in texture space. Each image was divided in 16 subimages and for each subimage two features were extracted: the dominant orientation and its strength. Therefore each image I^j was represented as a vector:

$$I^j = (O_1^j, S_1^j, O_2^j, S_2^j, \ldots, O_{16}^j, S_{16}^j) \tag{4.1}$$

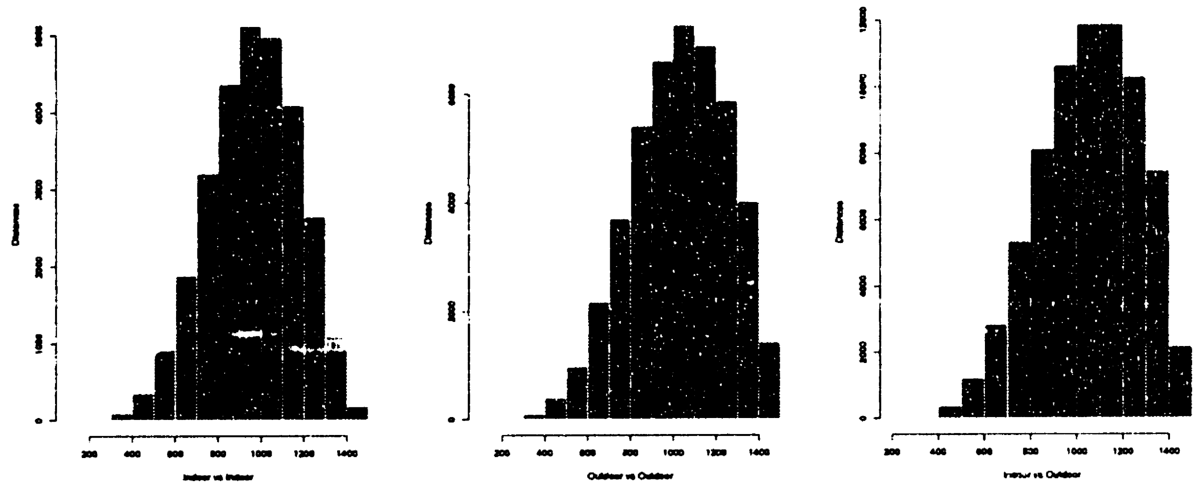


Figure 4-1: Histogram of the in-class and out-of-class distances in the color representation.

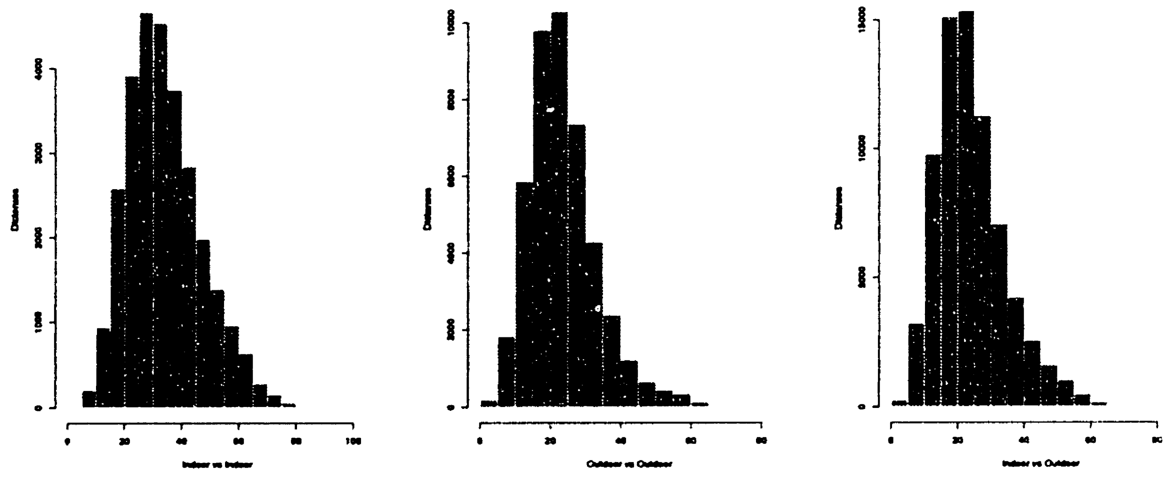


Figure 4-2: Histogram of the in-class and out-of-class distances in the texture representation.

Type	Average Number of Vertical Lines	Standard Deviation
Indoor	5.04	2.7246
Outdoor	2.73	2.32

Table 4.3: Average Number of Vertical Lines Counted

where O_p^j and S_p^j are respectively the orientation and the strength of the p-th subimage in image I^j.

Some preliminary experiments with this kind of representation convinced us that it was using too many variables, and a more compact representation would have been more appropriate. Moreover, we wanted to use some a priori knowledge on the problem, that could be more easily used in a different representation than the one of eq. (4.1). In the following sections we describe how the texture representation has been modified from the original of (4.1).

4.2.1 Quantifying vertical structures

Our prior knowledge consists in the fact that indoor environments usually contains more "strong" vertical lines than outdoor environment. This is due to the fact that indoor environment contains artifact and artifact tends to have well-defined vertical lines (e.g. wall, doors, book shelves, furniture, etc.). We purposely did not use horizontal lines as a cue since they are largely view dependent (i.e. horizontal lines may look diagonal if looking at a different angle). However, assuming the pictures are taken in upright position, vertical lines usually remain vertical. Therefore, rather than using the dominant texture orientation of all the 16 subimages we just compute the number of of subimages with dominant orientation to be "vertical", where "vertical" means with dominant orientation between $-5°$ and $+5°$ ($0°$ is the vertical direction).

As shown in Table 4.3, we indeed observed there are generally more vertical lines in the indoor environment than in the outdoor environment. It is also evident from the histogram of figure (4-3) that this quantity has different distributions in indoor and outdoor scenes, and it is therefore likely to bring relevant information. Therefore,

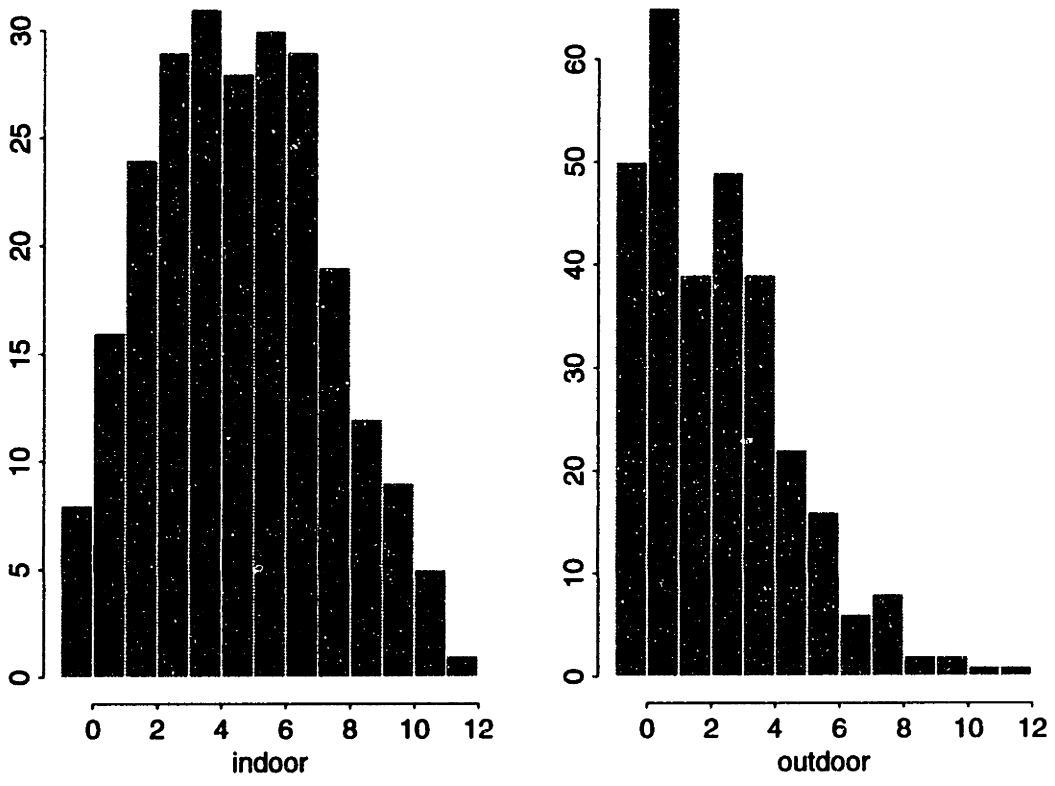


Figure 4-3: The distribution of the number of subimages with dominant vertical orientation in the indoor and outdoor scenes.

Type	Average Strength	Standard Deviation
Indoor	19.48	16.03
Outdoor	13.14	12.37

Table 4.4: Average Strength

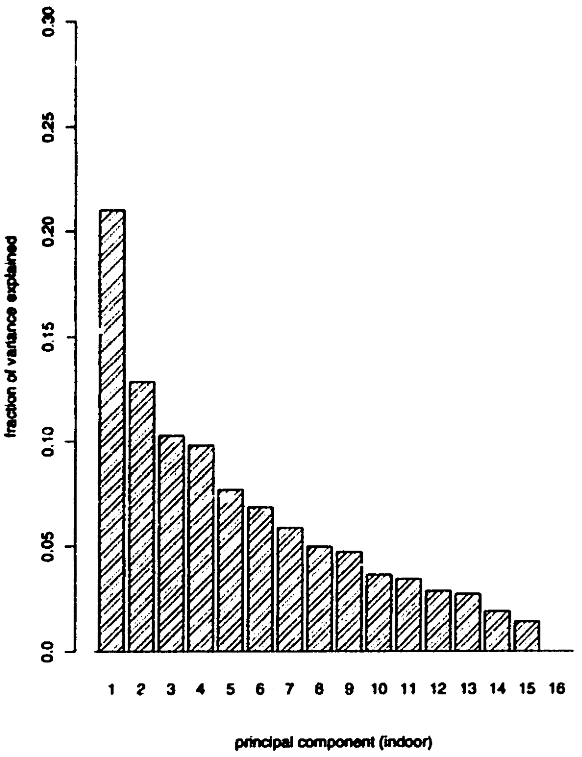
the orientation component of the texture feature vectors is reduced from 16 "generic" dimensions to only one dimension, that we believe to be particularly informative.

4.3 Texture strength and Principal Components Analysis

The relationship between the strength of a texture of an image and whether that image represents an indoor or an outdoor scene is not as clear as in the case of vertical orientation. Indoor scenes have a tendency to have more artifacts, and these are often associated to strong lines and contours. However it is also true that in many indoor scenes there are many walls or flat surfaces with very little texture. It appears that, at least in our database, indoor scenes have an average strength that is higher than the one in outdoor scenes. However, the difference is very small, as it can be seen from the high variances reported in table (4.3).

Since the difference in the average strength between indoor and outdoor scenes is so small, we expect that using 16 numbers to represent strength information, as in representation (4.1), is not a good approach. We therefore tried to compress some of the strength information in a lower dimensional vector, using Principal Component Analysis (PCA).

Principal component analysis (PCA) is a common method from statistics for dimension reduction. Linsker (1988) notes that performing principal component analysis is equivalent to maximizing the information content of the output signal in situations where that has a Gaussian distribution. The aim is to find a set of M orthogonal vectors in data space that account for as much as possible of the data's variance.



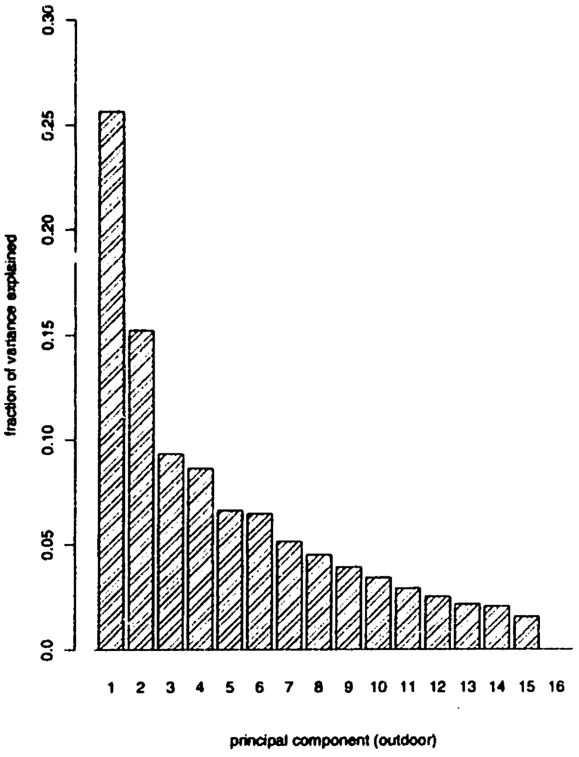


Figure 4-4: The 16 eigenvalues of the correlation matrix of the strength feature vectors ofr indoor and outdoor images.

Projecting the data from their original N-dimensional space onto the M-dimensional subspace spanned by these vectors then performs a dimensionality reduction that often retains most of the intrinsic information in the data. Typically $M \ll N$, making the reduced data much easier to handle. Specifically the first principal component is taken to be along the direction with the maximum variance. The second principal component is constrained to lie in the subspace perpendicular to the first. Within that subspace it is taken along the direction with the maximum variance. Then the third principal component is taken in the maximum variance direction in the subspace perpendicular to the first two, and so on.

The principal component analysis of the 16 strength features of indoor images and outdoor images are calculated and plotted in Figure 4-4. The 16 dimensional strength vectors were first translated in such a way that they had zero mean.

Class 1	Class 2	Average distance	Standard Deviation
Indoor	Indoor	36.47	30.43
Outdoor	Outdoor	27.48	26.67
Indoor	Outdoor	33.04	29.6

Table 4.5: Average Distance Using New Texture Feature Vectors

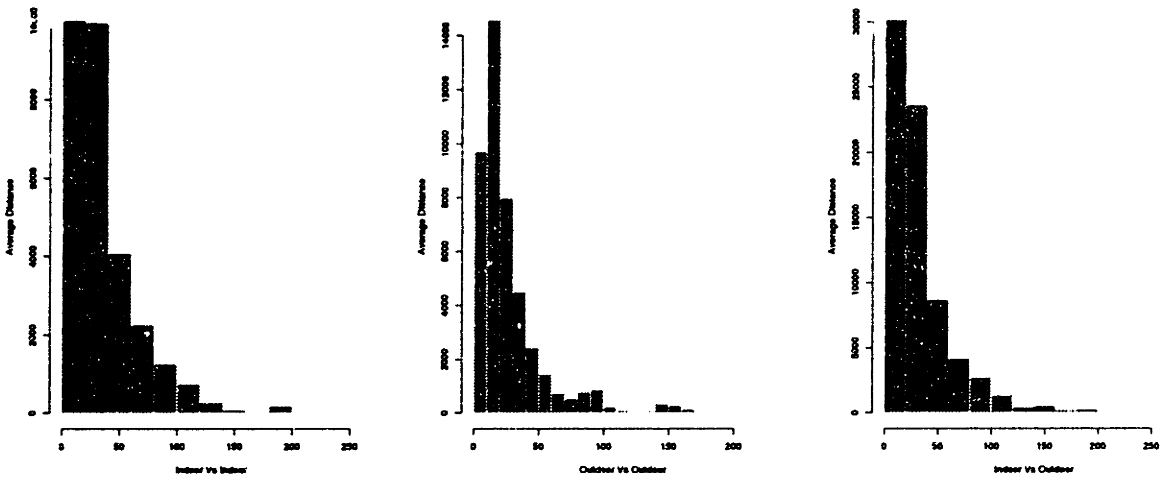


Figure 4-5: Histogram of the in-class and out-of-class distances in the new texture representation.

From Figure (4-4) it is evident that the first 5 components capture about 60% of the variance, and this proved to be enough in our experiments. Therefore, in our texture representation, an image I^j is going to be represented as

$$I^j = (N_j, PS_1, \ldots, PS_5) \tag{4.2}$$

where N_j is the number of subimages of image I^j with dominant vertical orientation and $PS_1, \ldots, PS_5$ are the first 5 principal components of the 16 dimensional strength feature vector.

The average distance of these new texture feature vectors is calculated and listed in Table 4.3.

Chapter 5

Experiment

In this section we discuss the experiments we performed and report their results. In all our experiments we used a data base of 192 indoor images and 249 outdoor image. the test set consisted of 49 indoor images and 51 outdoor images. About 90 % of the images were provided by Eastman Kodak Company and consisted of random pictures that they developed for customers, from whom permission to do this research was granted. The remaining part of the data set was collected from some commercially available CD ROM and from some public World Wide Web site. All the images are real life images (i.e. they are not from cartoon or generated by computers), and we had no control on their quality, that was in many cases quite poor.

5.1 Nearest Neighbor and K-Nearest Neighbor Experiment

Nearest neighbor is a very simple classification technique, in which the only choice that has to be made is the distance function to be used. In both the color and texture cases we used the distance induced by the L_1 norm, because it is more robust against noise and outliers than the standard, euclidean L_2 norm. We remind the reader that the L_1 norm of a vector $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ is defined as

$$\|\mathbf{x}\|_{L_1} = \sum_{\alpha=1}^{d} |x_\alpha| \, .$$

In the case of texture the image is represented as

$$I^j = (N_j, PS_1, \ldots, PS_5)$$

where N_j is the number of subimages of image I^j with dominant vertical orientation and $PS_1, \ldots, PS_5$ are the first 5 principal components of the 16 dimensional strength feature vector. Since the components of this vector have different meanings, a sphering of the data has been performed before applying the nearest neighbor algorithm.

We did experiments with both the nearest neighbor and the K-nearest neighbor technique. In the case of K-nearest neighbor, the closest k points are retrieved (with k odd in order to avoid ties), and then a majority rule is applied. We did experiments with $k = 3, 5, 7$ and run some cross-validation technique to estimate which is the best value for k, that turned out to be $k = 3$.

We first applied the nearest neighbor algorithm to our data base, which is supposed to give us an idea of the level of performance we can expect also from other techniques. We performed independent tests, using color or texture information, and the results are reported in Table (5.1) and (5.2). Notice how the texture representation seems to carry less information, and how in both cases the classifier has a tendency to misclassify indoor scenes more often that outdoor scenes. It seems therefore that there are many indoor scenes that look like outdoor scenes, but not too many outdoor scenes that can be misclassified as indoor.

A similar scenario appears when we use K-nearest neighbor, with $k = 3$. However, there is a network improvement in the performances, as shown in tables (5.3) and (5.4).

In order to achieve a better understanding of how well K-nearest neighbor works we report here also some results on specific images. In particular we consider the following:

- A typical indoor scene (5-1a);

Type	Number of Misclassified	Total Number of Test Images	% Correct
Indoor	14	49	71.4%
Outdoor	8	51	84.3%
Total	22	100	78.0%

Table 5.1: Nearest Neighbor Experiment on Color Feature Vectors

Type	Number of Misclassified	Total Number of Test Images	%Correct
Indoor	20	49	59.2%
Outdoor	8	51	84.3%
Total	28	100	72.0%

Table 5.2: Nearest Neighbor Experiment on Texture Feature Vectors.

Type	Number of Misclassified	Total Number of Test Images	% Correct
Indoor	13	49	73.5%
Outdoor	4	51	92.2%
Total	17	100	83.0%

Table 5.3: K-Nearest Neighbor Experiment on Color Feature Vectors, with $k = 3$.

Type	Number of Misclassified	Total Number of Test Images	%Correct
Indoor	16	49	67.4%
Outdoor	6	51	88.2%
Total	22	100	78.0%

Table 5.4: K-Nearest Neighbor Experiment on Texture Feature Vectors, with $k = 3$.

- An indoor scene that was incorrectly classified using color information but correctly classified using texture information (5-3a);
- A typical outdoor scene (5-5a);
- An outdoor scene that was incorrectly classified using both color and texture information (5-7a);

For each of these images we also report in Fig. (5-1b) and (5-1c), (5-3b) and (5-3c), (5-5b) and (5-5c), (5-7b) and (5-7c), the closest images, respectively in the color and texture domain. For each of these images we also report, in Figure (5-2) (5-4) (5-6) (5-8), the plot of the distances of the image from the closest 20 indoor and outdoor images, using color and texture.

In Figure (5-2), which correspond to the typical indoor scene of Fig. (5-1a), we notice that both color and texture feature space have a good margin to classify correctly the scene as indoor: in color space, the first 4 closest images are indoor, and in the texture space the closest 12 images are indoor.

Figure (5-4) corresponds to the image of Fig. (5-3a), which is an indoor scene that was incorrectly classified using color information but correctly classified using texture information. Although this is a typical indoor scene, its color is predominately brown, which is a typical outdoor color (e.g. wood, soil, etc). The texture classifier was able to work effectively with this type of indoor scene since this image has well-defined vertical lines. As shown in Figure (5-4), the texture information provided a quite confident result, since the first 4 closest images fall in the indoor catagory.

Figure (5-6) corresponds to the image of Fig. (5-5a), which is a typical outdoor scene. On this scene texture information is the one that has more confidence, since its closest 14 images are of the same class. In the color space there seem to be more indoor images with a similar color distribution.

Figure (5-8) corresponds to the image of Fig. (5-5a), which is an example of outdoor scene which was incorrectly classified both in the color and texture space. The system often fails on close-up pictures, because they do not provide a lot of spatial and spectral information. The dominant color in this image is red (the jacket on). Both of these features are unfavorable to classify this image into the outdoor catagory. For humans it is easy to classify the scene as outdoor, due to the presence of the tree and of the sky in the background, but this is clearly not enough for our system.

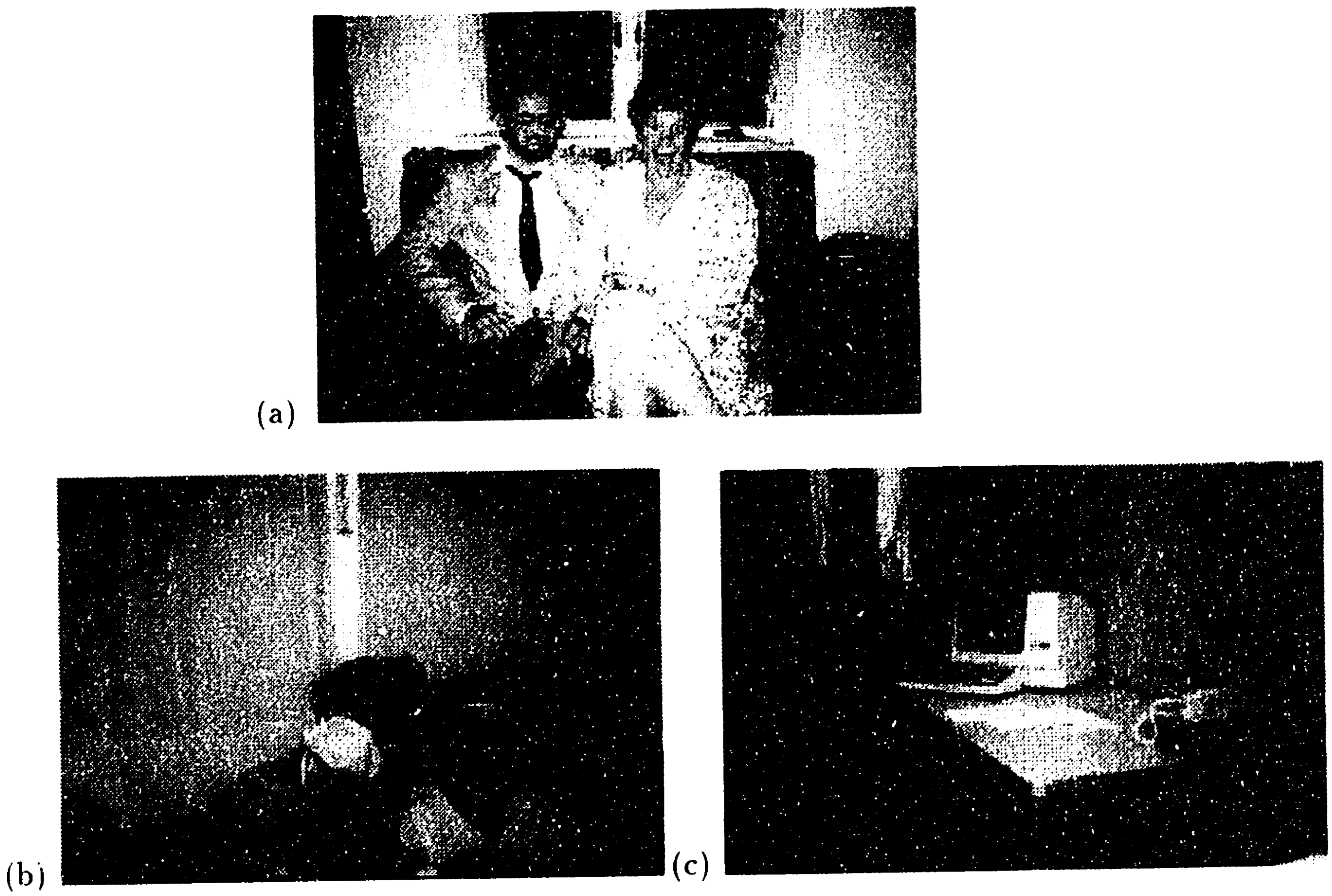
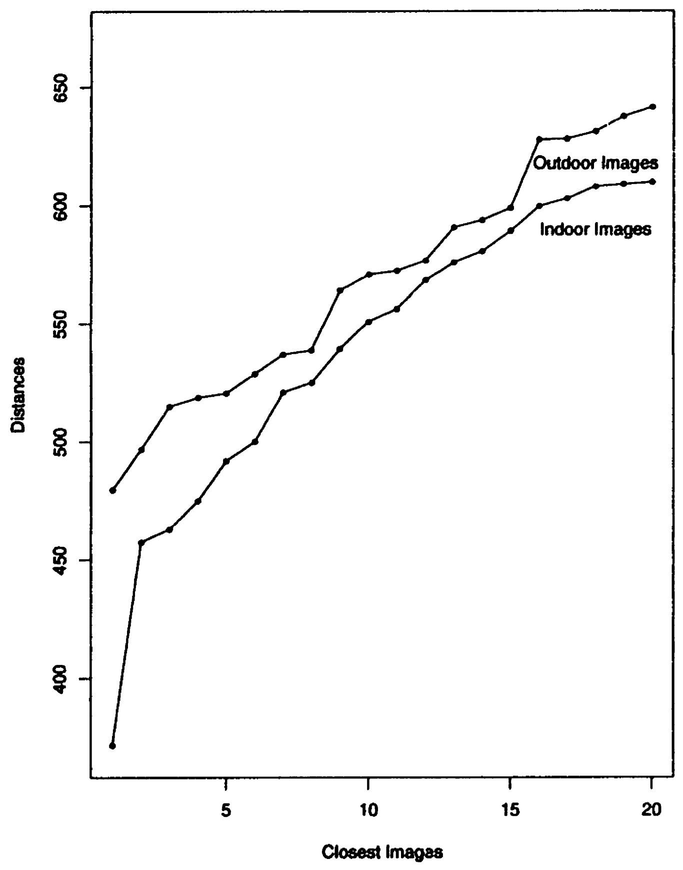


Figure 5-1: a) A typical indoor scene and the nearest images retrieved using color (b) and texture (c) information



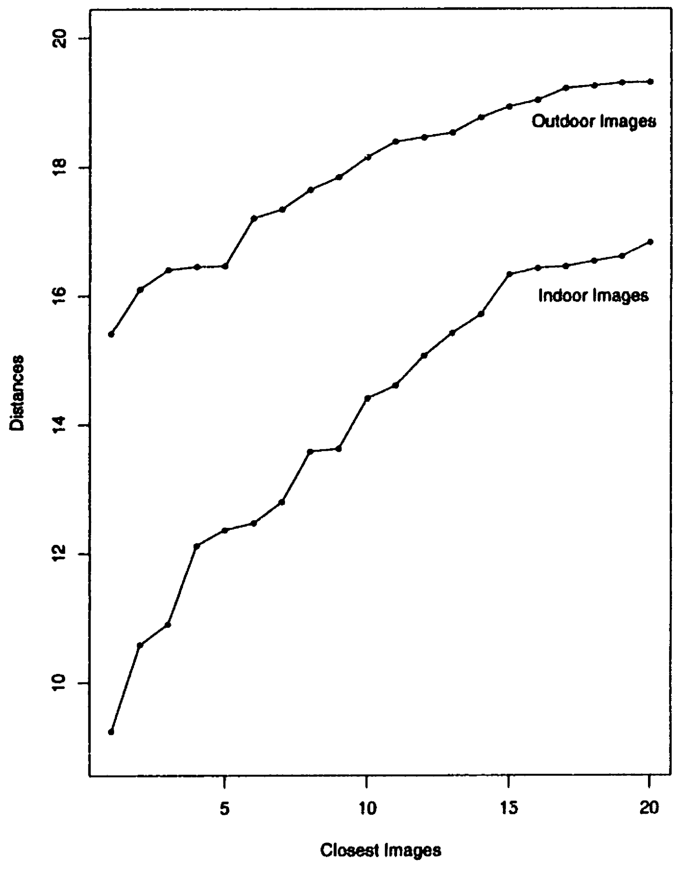


Figure 5-2: Plot of the distances of image 5-1(a) from the closest 20 indoor and outdoor images, using color (left) and texture (right)

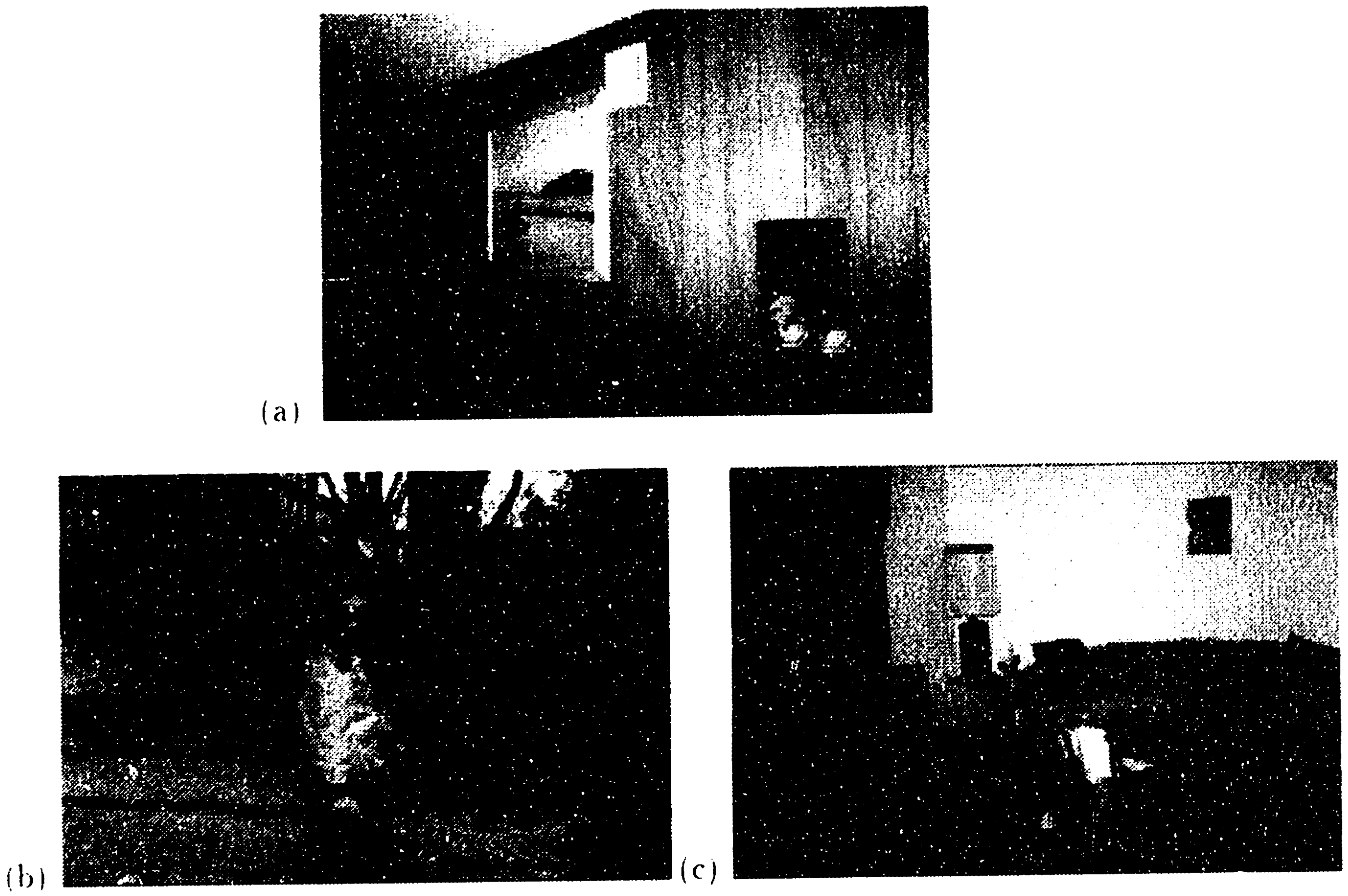


Figure 5-3: a) An indoor scene which was incorrectly classified using color information and the nearest images retrieved using color (b) and texture (c) information

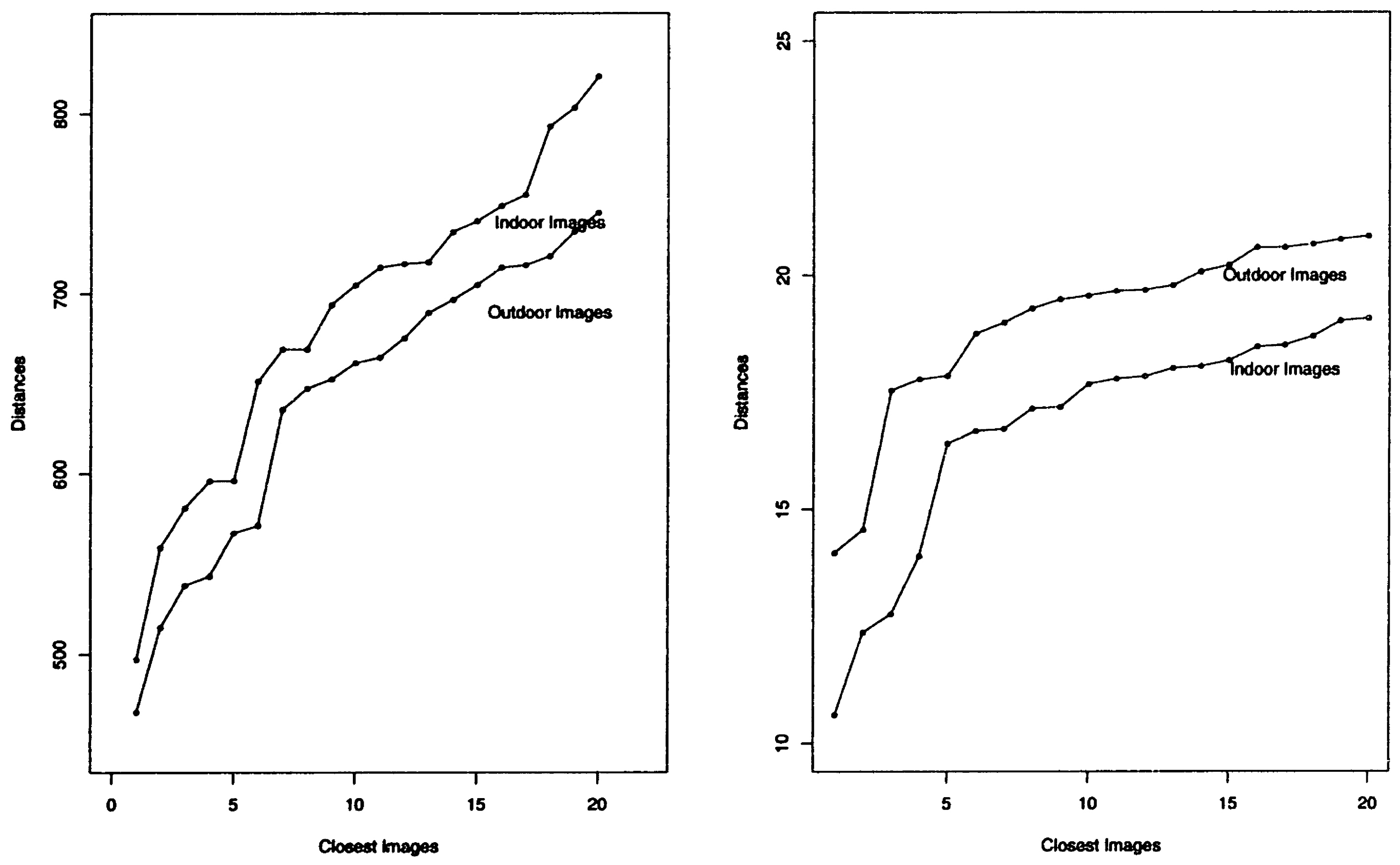
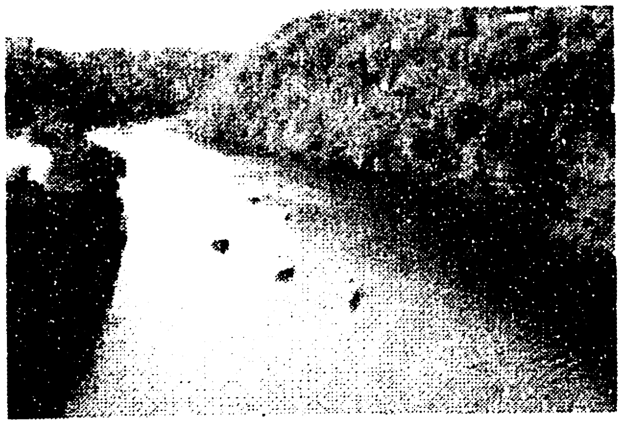
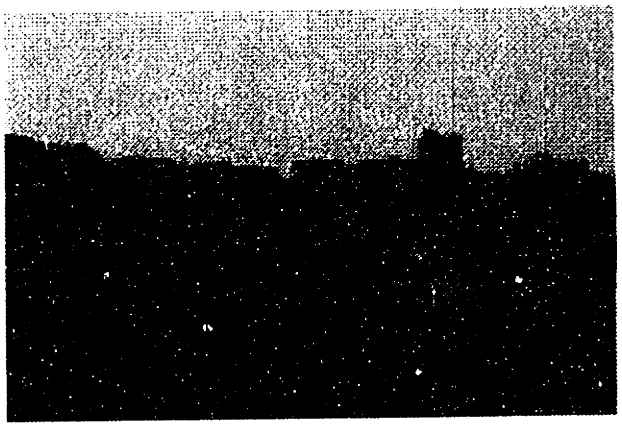


Figure 5-4: Plot of the distances of image 5-3(a) from the closest 20 indoor and outdoor images, using color (left) and texture (right)


(a)


(b)

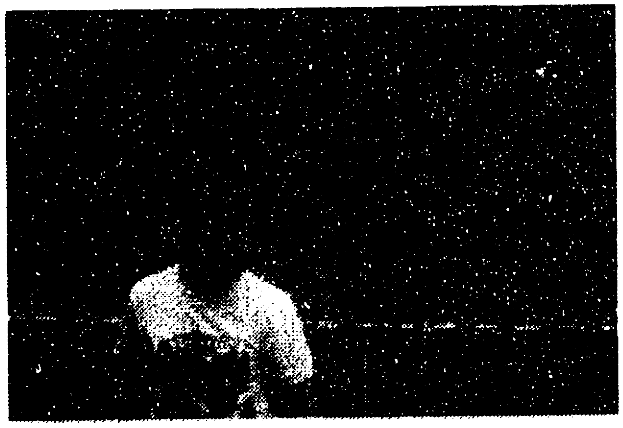

(c)

Figure 5-5: A typical outdoor scene and the nearest images retrieved using color (b) and texture (c) information

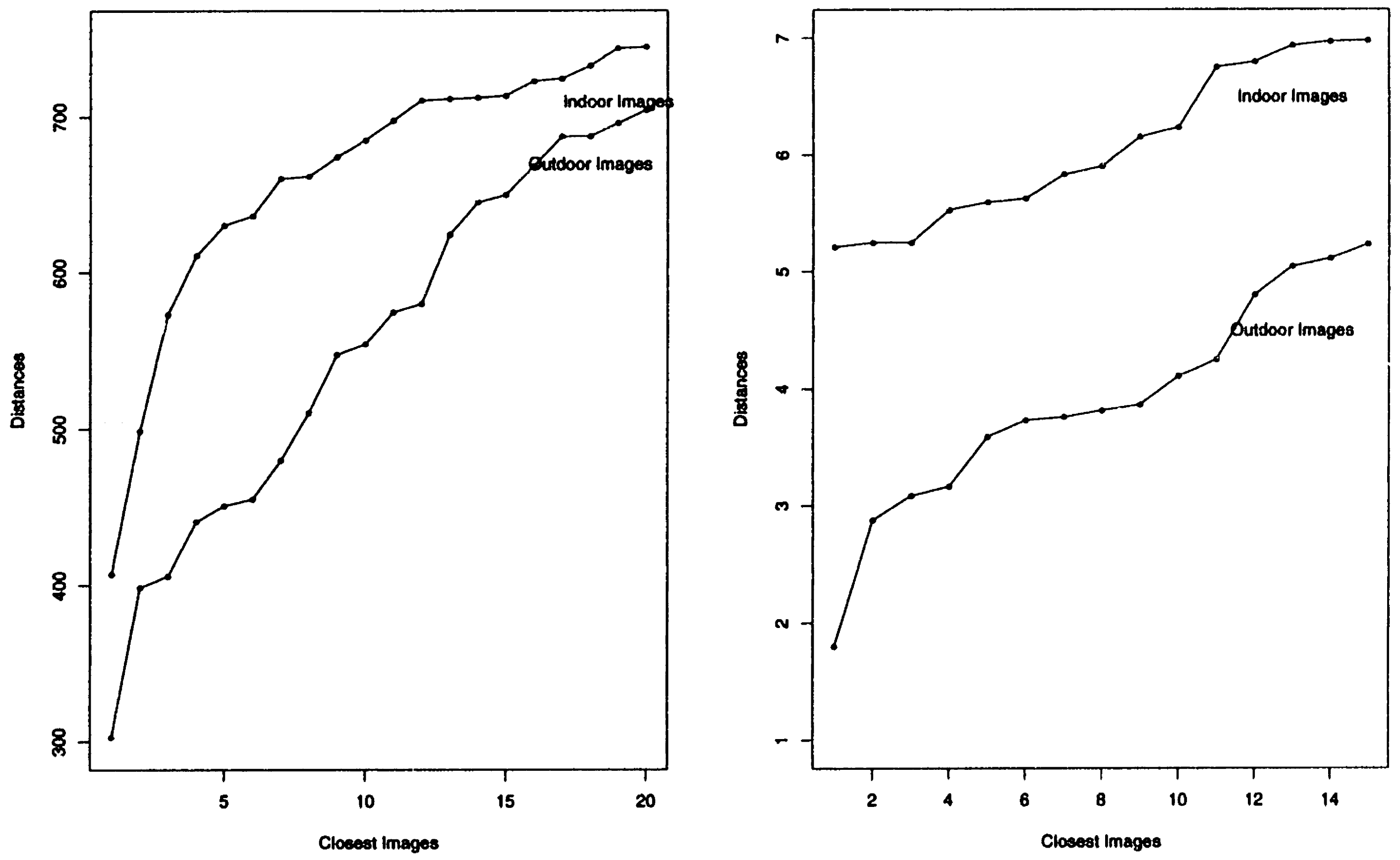


Figure 5-6: Plot of the distances of image 5-5(a) from the closest 20 indoor and outdoor images, using color (left) and texture (right)

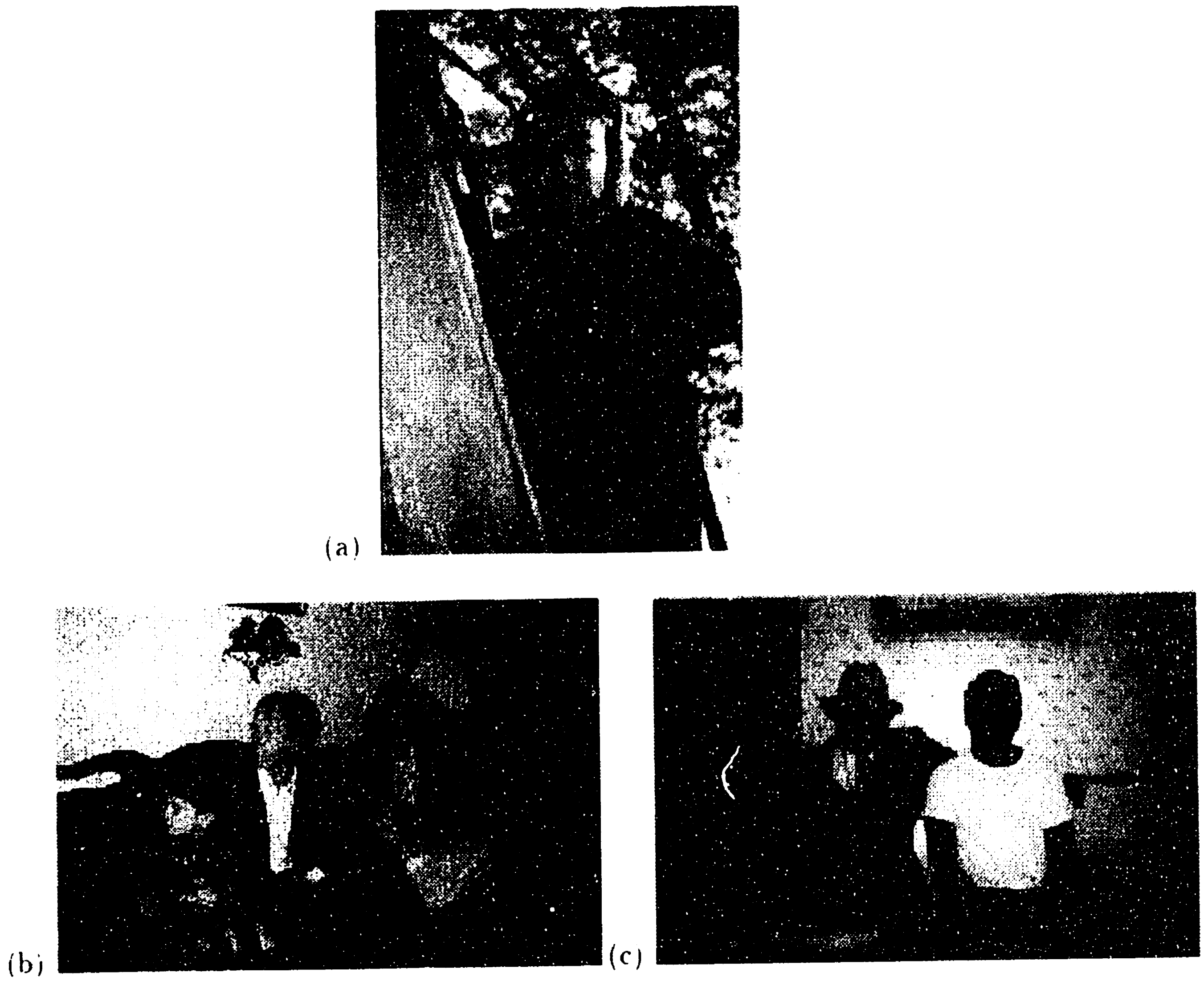


Figure 5-7: a) An outdoor scene which was fincorrectly classified using both color and texture information and the nearest image retrieved using color (b) and texture (c) information.

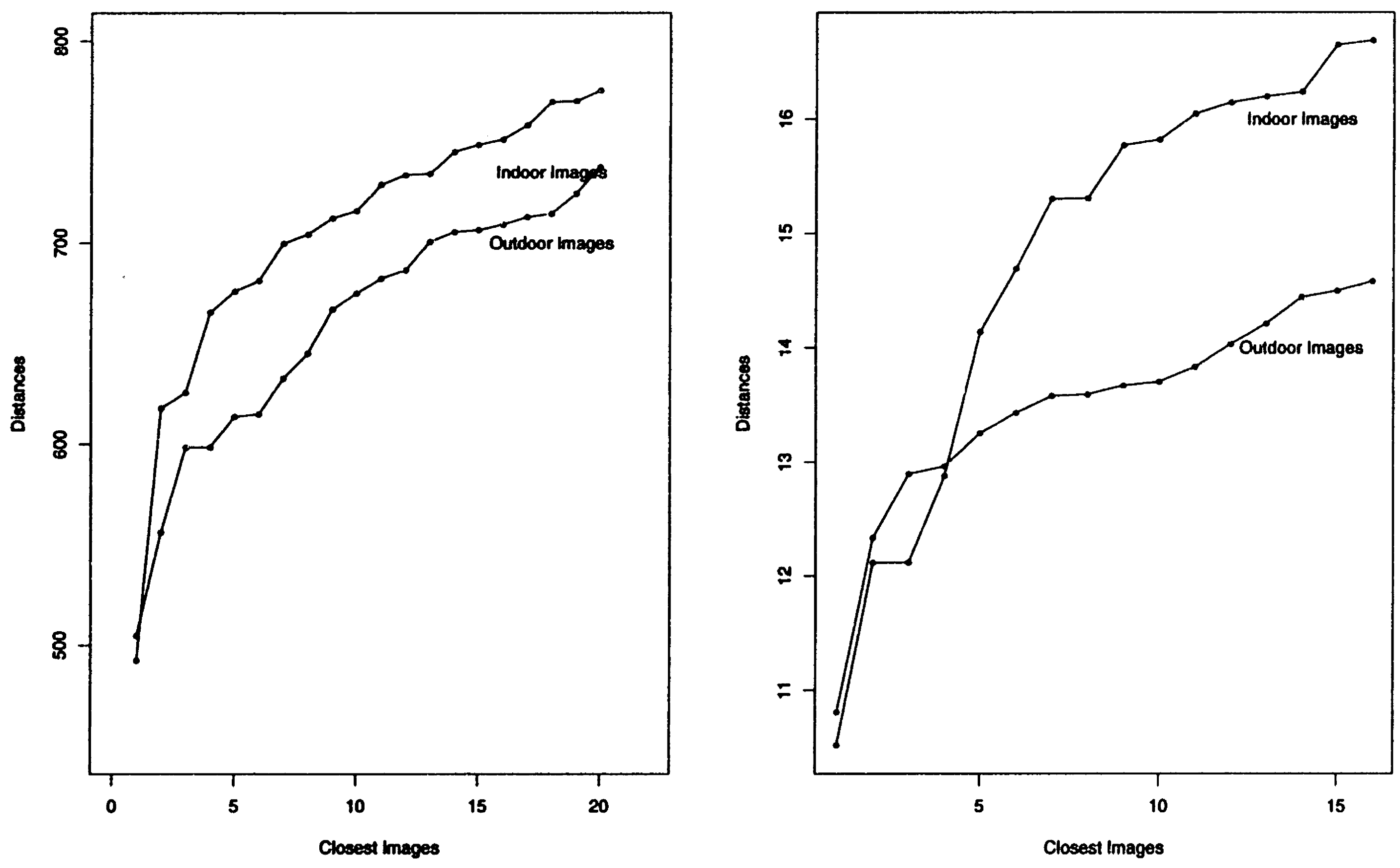


Figure 5-8: Plot of the distances of image 5-7(a) from the closest 20 indoor and outdoor images, using color (left) and texture (right)

5.2 Support Vector Machine

In this section we report the results of our experiments with Support Vector Machines. We considered the same data base used in the previous section, consisting of 192 indoor images and 249 outdoor images, and the same test set, of 49 indoor images and 51 outdoor images. We used three different representations for the images: color (36 dimensions), texture (6 dimensions), and the combination of color and texture in one single feature vector (40 dimensions). We trained the Support Vector Machine with a polynomial kernel function. There are only 2 free parameters to set in the Support Vector Machine algorithm: the degree of the polynomial and the constant γ of equation, which is basically the penalty associated to a misclassified point. We used some simple cross validation techniques to estimate these two free parameters, and determined that the best degree for the polynomial is equal to three, and that a value of $\gamma = 500$ is satisfactory. We noticed however that the algorithm is not very sensitive to the choice of the value of γ, nor to the degree of the polynomial.

Type	Number of Misclassified	Total Number of Test Images	% Correct
Indoor	16	49	67.4%
Outdoor	6	51	88.2%
Total	22	100	78.0 %

Table 5.5: Support Vector Experiment on Color Feature Vectors

Type	Number of Misclassified	Total Number of Test Images	% Correct
Indoor	16	49	67.4%
Outdoor	1	51	98.0%
Total	17	100	83.0 %

Table 5.6: Support Vector Experiment on Texture Feature Vectors

Type	Number of Misclassified	Total Number of Test Images	% Correct
Indoor	11	49	77.6%
Outdoor	7	51	86.3%
Total	18	100	82.0 %

Table 5.7: Support Vector Experiment on Both Color and Texture Feature Vectors

5.3 Stacked Classification

In this thesis we implemented the simplest version of stacked regression, that consists in building a new estimator by taking the average of already trained estimators. In the case of classification we can think of the classifiers as "experts", and stacking is achieved by taking a majority vote among the experts (always in an odd number). We considered 7 classifiers: the support vector machine trained on texture, color and their combination, the nearest neighbor on texture and color, and the k-nearest neighbor on texture and color. The result of our experiment is reported in table (5.8). Interestingly, tha stacking technique performs much better than the best of the individual classifiers, achieving a 92% correct classification rate, which is a large improvement over the 83% achieved by k-nearest neighbor in color space and SVM in texture space. The reason for this very good result appears to be the fact that the individual classifiers disagree "often enough", because if they always agreed we could not expect any improvement. It seems therefore that they are also fairly statistical independent. While we find surprising to find statistical independency between classifiers like nearest neighbor and k-nearest neighbor, it is not surprising that classifiers based on color and texture are independent, because we do not expect too much correlation between color and texture.

Type	Number of Misclassified	Total Number of Test Images	% Correct
Indoor	7	49	85.7%
Outdoor	1	51	98.0%
Total	8	100	92.0 %

Table 5.8: Stacking Experiment

5.4 Bagging Predictors

We tried another way of combining different classifiers, which is the "bagging" technique proposed by Breiman, and explained in section 3.4.2. Since Breiman reported that bagging does not work very well for "stable" techniques like nearest neighbor we applied it to SVM, which is still not yet clear whether is stable or unstable. We artificially constructed 49 data sets out of our original data set, by resampling with replacement. Then 49 support vector machines have been trained on these data sets, and a final classifier has been built ba taking a majority vote of the 49 classifiers. The results are reported in Table (5.9), for the color, texture and combined representation.

Type	Indoor Mis.	Outdoor Mis.	Total Mis	% Correct	% Correct (no Bagging)
Color	14	5	19	81%	78%
Texture	15	3	18	82%	83%
Combined	10	6	18	84%	82%

Table 5.9: Bagging Experiment on Color, Texture and their Combination

It appears that Support Vector Machine with polynomials of degree three is a quite stable classifier, because bagging only leads to minor improvements in two cases, and to a degeneration of the performances in one case. We attribute this phenomenon to the fact that polynomials of degree three lead probably to a model with low variance, but higher bias, because thay do not model very complex surfaces. Therefore all the "bagged" version of the support vector machine are very similar to each other, and highly correlated.

A summary of some of the experiments described in this section is reported in table (5.10), and will be discussed in the next chapter.

Classifier	Indoor Mis.	Outdoor Mis.	Total Mis	% Correct
NN Color	14	8	22	78%
NN Texture	20	8	28	72%
K-NN Color	13	4	17	83%
K-NN Texture	16	6	22	78%
SV Color	16	6	22	78%
SV Texture	16	1	17	83%
SV Combined	11	7	18	82%
Stacking	7	1	8	92%
Bagging on Color	14	5	19	81%
Bagging on Texture	15	3	18	82%
Bagging on Combined	10	6	16	84%

Table 5.10: Summary of the results from all experiments

Chapter 6

Conclusion

A lot of the existing image database systems used verbal descriptions to perform retrieval operations. However, this paradigm is inadequate since the majority of pictorial information in an image cannot be fully captured by text and numbers due to the essential limitations in expressive power of language [37], and problems like "find and image that looks like this" cannot be solved within this approach. This observation is at the basis of this thesis, that was an attempt to derive some more useful paradigm for image indexing and retrieval. We concentrated on the specific problem of discriminating indoor scenes from outdoor scenes, but we believe that the techniques we used can be applied to other classes of problems. In this work we explored the use of global statistical properties of the image, related to color and texture information. Below we present some observations that derive from our work, and that we think will be useful for future work.

- One of the problems encountered in using color information is color constancy. This problem can be addressed at different levels of complexity. For indexing purposes one needs to have computationally fast solutions, that require no prior knowledge on the image. One simple way to achieve a certain degree of color constancy consists in choosing an appropriate color space, in which one of the three components takes in account the illumination effects, and therefore can be discarded. This is the approach we took in this thesis, where we adopted the $(L, C1, C2)$ color space proposed by Lee, which is a $45°$ rotation of the RGB color space. Experimenting with similar images taken under different points of view or illumination we did find that a good degree of color constancy can be achieved by disregarding the L component. Given the complexity of the data set, we believe that result obtained with k-nearest neighbor (83% correct classification) is quite good, and we attribute it to the correct choice of the color space.

- Texture information can be extremely useful but it is often not trivial to decide what a good representation for this information is. We originally divided every image in 16 sub-images, and for every sub-image we extracted the dominant texture orientation and its strength, so that every image was represented by a 34 dimensional vector. Experiments with this representation did not provide good results, apparently because of too much redundancy in the 34 variables. We then summarized that information in a 6 dimensional vector, whose first component is the number of sub-images with dominant vertical orientation, and the other 5 variables are the first 5 principal components of the strength features. This representation proved to be much more effective, providing a result that is as good as the k-nearest neighbor when used in conjunction with Support Vector Machines. One of the reasons for this improvement is that, with this choice of the first variable, we are actually embedding some prior knowledge into the system: we are using what we know about indoor and outdoor scenes to assert that the presence of vertical lines is a discriminating factor. We believe that much more can be done on this aspect of the problem, and different kind of prior knowledge can be used, by an appropriate choice of the variables.

- Combining different cues, such color and texture, is a difficult task, Intuitively, if the cues are statistically independent one would expect that combining them in an appropriate way, one could get a very small classification error, but it is not clear what is the best way of doing it. We trained different classifiers, such as k-nearest neighbor and Support Vector Machines on different input representations, and combined in a two simple ways. The most effective is a voting scheme, which is equivalent to the stacked regression technique proposed by Breiman. When the voting scheme is applied to 7 different classifiers, the combined classifier achieves a 92% correct classification, which is much better than the best of the single classifiers, that is 83%. The reason for this results seeme to lie in the fact that these classifiers disagree often enough, and are fairly independent. We also tried the "bagging" scheme proposed by Breiman, which is a voting scheme based on training the same classifier on bootstrapped version of the same database. This scheme led to small improvement in two of the three cases we tried it on (Support Vector Machines on color, texture and combined color/texture features). The apparent reason for the small improvement is that the Support Vector Machine provided very similar classifiers on the bootstrapped data bases, proving to be what Breiman a "stable" technique.

We think that the achieved result, 92% correct classification, is a very good indication that difficult, real world problems, can be attacked. We have made very simple use of color and texture cues in this thesis, concentrating on global properties of the image. We believe that including more prior knowledge, for example on spatial relationships between color or texture patterns, could lead to great improvements, and we think it is an interesting direction for future work. We also think that much more could be done in the combination of different classifiers: simple voting schemes are not necessarily the best options, and alternative techniques, such as the "boosting" algorithm recently proposed by Schapire and Freund [26], are worth exploring in the future.

Bibliography

[1] http://holodeck.iss.nus.sg:80/rnd/projects.html.

[2] http://www-white.media.mit.edu/~tpminka/photobook/.

[3] http://www.c3.lanl.gov/~kelly/candid/main.shtml.

[4] http://wwwcsai.diepa.unipa.it/research/projects/jacob/default.html.

[5] http://www.ctr.columbia.edu/advent/.

[6] http://wwwqbic.almaden.ibm.com/~qbic/qbic.html/.

[7] http://www.virage.com.

[8] http://www.vision.ee.ethz.ch/.

[9] R.O. Duda and P.E. Hart. Pattern Classification and Scene Analysis. Wiley, New York, 1973.

[10] M. Jordan and R. Jacobs Hierarchical Mixtures of Experts and the EM Algorithm. In *Neural Computation*, Vol 6, pp 181–214, 1994.

[11] A. Lipman and W. Yang VLSI Hardware for Example-Based Learning (personal communication)

[12] B. Schölkopf and C. Burges and V. Vapnik. Extracting support data for a given task. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, Menlo Park, CA, 1995. AAAI Press.

[13] B.E. Boser, I.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifier. In *Proceedings of the fifth annual ACM workshop on computational learning theory*, pp 144–152, Pittsburgh, PA, July 1992.

[14] C.M. Bishop. Neural Networks for Pattern Recognition Clarendon Press, pp 295–329, Oxford, UK, 1995.

[15] L. Breiman. Bagging predictors. (unpublished manuscript).

[16] L. Breiman. Stacked regression. Technical report, University of California, Berkeley, 1992.

[17] C. Burges, F. Girosi, P. Niyogi, T. Poggio, B. Schölkopf, K.K. Sung, and V. Vapnik. Choosing rbf centers with the support vector algorithm. A.I. Memo, MIT Artificial Intelligence Lab., Cambridge, MA, 1995.

[18] C Cortez and V. Vapnik. Support vector networks. *Machine Learning*, 1995. (to appear).

[19] H. Drucker, R. Schapire, and P. Simard. Boosting performance in neural networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(4):705–719, 1993.

[20] Y. Freund and R. Schapire. (personal communication).

[21] R. Rao and B.G. Schunck. Computing Oriented Patterns. *CVGIP Graphical Models and Image Processing*, 53(2):157-185, 1991

[22] J. Bigün and G.H. Granlund. Optimal Orientation Detection of Linear Symmetry. In Proc. 1st Int. Conf. Comp. Vis., pp 433-438, London, England, 1987.

[23] M. Kass and A. Witkin. Analyzing Oriented Patterns. In M. A. Fishchler and M. Kaufman, editors, *Readings in Computer Vision*, pp 268-276.

[24] J.S. Lim. Two-Dimensional Signal and Image Processing Prentice Hall. K-means algorithm, pp 607-609.

[25] P.J. Huber. Robust Statistics. John Wiley and Sons, New York, 1981.

[26] R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[27] V. Vapnik. The Nature of Statistical Learning Theory. Springer, New York, 1995.

[28] V. N. Vapnik. Estimation of Dependences Based on Empirical Data. Springer-Verlag, Berlin, 1982.

[29] H. C. Lee. A Phsics-Based Color Encoding Model for Images of Natural Scenes. Image Processing Algorithm Laboratory, Eastman Kodak Company.

[30] E. E. Osuna Support Vector Machines MIT CBCL Paper, 1996.

[31] T. S. C. Tan and J. Kittler Colour Texture Classification using Features from Colour Histogram. Department of Electronic and Electrical Engineering University of Surrey

[32] E. De Micheli, R. Prevete, G. Piccioli, M. Campani. Color cues for traffic scene analysis. Technical paper, Istituto di Cibernetica e Biofisica - C.N.R -Genova

[33] R. W. Picard and M. M. Gorkani. Texture Orientation for Sorting Photos "at a Glance". In *IEEE Conference on Pattern Recognition*, Oct 1994.

[34] R. W. Picard and T. Kabir. Finding Similar Patterns In Large Image Databases Proc. IEEE Conf. on Acoustics, Speech, and Signal Processing Minneapolis, MN, April 1993.

[35] M. J. Swain and D. H. Ballard. Indexing Via Color Histograms. Image Understanding Workshop, pp 623-630, Sept, 1990

[36] W. T. Freeman and E. H. Adelson. The Design and Use of Steerable Filters. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp 891-906, 1991.

[37] T. Gevers and A.W.M. Smeulders. Color-metric Historgram Matching for Viewpoint Invariant Image Retrieval. Technical Paper, University of Amsterdam.

[38] T.M.Cover and P.E. Hart. Nearest Neighbor Pattern Classification. In *IEEE Transactions on Information Theory*, Vol IT-13, pp 21-27, 1967

[39] R.W.G. Hunt. Measuring Colour. Chichester, England: Ellis Horwood Limited, 1987.