FUZZY SEMANTIC LABELING OF NATURAL IMAGES

Margarita Carmen S. Paterno

A THESIS SUBMITTED
FOR THE DEGREE OF MASTER OF SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
SCHOOL OF COMPUTING
NATIONAL UNIVERSITY of SINGAPORE
2004

CONTENTS

List of Figures

List of Tables

Summary

The rapid development of technologies for digital imaging and storage has led to the creation of large image databases that are time consuming to search using traditional methods. As a consequence, content-based image organization and retrieval emerged to address this problem. Most content-based image retrieval systems rely on low-level features of images that, however, do not fully reflect how users of image retrieval systems perceive images since users tend to recognize high-level image semantics. An approach to bridge this gap between the low-level image features and high-level image semantics involves assigning semantic labels to an entire image or to image blocks. Crisp semantic labeling methods assign a single semantic label to each image region. This labeling method has so far been shown by several previous studies to work for a small number of semantic classes. On the other hand fuzzy semantic labeling, which assigns multiple semantic labels together with a confidence measure to an image region, has not been investigated as extensively as crisp labeling.

This thesis proposes a fuzzy semantic labeling method that uses confidence measures based on the orthogonal distance of an image block's feature vector to the hyperplane constructed by a Support Vector Machine (SVM). Fuzzy semantic labeling is done by first training *m* one-vs-rest SVM classifiers using training samples. Then using another set of known samples, a *confidence curve* is constructed

for each SVM to represent the relationship between the distance of an image block to the hyperplane and the likelihood that the image block is correctly classified. Confidence measures are derived using the confidence curves and gathered to form the *fuzzy label* or *signature* of an image block.

To perform region matching, prototype signatures have to be obtained to represent each semantic class. This is carried out by performing clustering on the signatures of the same set of samples used to derive the confidence curves and taking the centroids of the resulting clusters. The multiple prototype signatures obtained through clustering is expected to capture the large variation of objects that can occur within a semantic class. Region matching is carried out by computing the Euclidean distance between the signature of an image block and each prototype signature.

Experimental tests were carried out to assess the performance of the proposed fuzzy semantic labeling method as well as to compare it with crisp labeling methods. Tests results show that the proposed fuzzy labeling method yields higher classification accuracy than crisp labeling methods. This is especially true when the fuzzy labeling method is applied to a set of image blocks obtained by partitioning images into overlapping fixed-size regions. In this case, fuzzy labeling more than doubled the classification accuracy achieved by crisp labeling methods.

Based on these tests results, we can conclude that the proposed fuzzy semantic labeling method performs better than crisp labeling methods. Thus, we can expect that these results will carry over to image retrieval.

CHAPTER 1

Introduction

1.1 Background

The rapid development of technologies involved in digital imaging and storage has greatly encouraged efforts to digitize massive archives of images and documents. Such efforts have resulted in considerably large databases of digital images which users will naturally want to access to find a particular image or group of images for use in various applications. An obstacle to finding images in these databases however is that searching for a specific image or group of images in such a large collection in a linear manner can be very time consuming. One straightforward approach to facilitate searching involves sorting similar or related images into groups and searching for target images within these groups. An alternative approach involves creating an index of keywords of objects contained in the images and then performing a search on the index. Either method however requires manually inspecting each image and then sorting the images or assigning keywords by hand. These methods are extremely labor intensive and time consuming due to the mere size of the databases.

Content-based image organization and retrieval has emerged as a result of the need for *automated* retrieval systems to more effectively and efficiently search such

large image databases. Various systems that have been proposed for content-based image retrieval include QBIC [HSE+95], Virage [GJ97], ImageRover [STC97], Photobook [PPS96] and VisualSEEK [SC95]. These image retrieval systems make direct use of low-level features such as color, texture, shape and layout as a basis for matching a query image with those in the database. Studies proposing such systems have so far shown that this general approach to image retrieval is effective for retrieving simple images or images that contain a single object of a certain type. However, many images actually depict complex scenes that contain multiple objects and regions.

To address this problem, some researches have turned their attention to methods that segment images into regions or fixed-sized blocks and then extract features from these regions instead of from the whole images. These features are then used to match the region or block features in a query image to perform image retrieval. Netra [MM97], Blobworld [CBG+97] and SIMPLIcity [WLW01] are examples of region-based and content-based image retrieval systems.

However, low-level features may not correspond well to high-level semantics that are more naturally perceived by the users of image retrieval systems. Hence, there is a growing trend among recent studies to investigate the correlation that may exist between high-level semantics and low-level features and formulate methods to obtain high-level semantics from low-level features. A popular approach to this problem involves assigning semantic labels to the entire image or to image regions. Semantic labeling of image regions thus is an important step in high-level image organization and retrieval.

1.2 Objective

There have so far been three approaches to assigning semantic labels to images or image regions. One, known as crisp labeling, classifies an image or image region into a single semantic class. The second, often referred to as auto-annotation, predicts groups of words corresponding to images. Finally, the third approach, which is the focus of this thesis, is called fuzzy semantic labeling.

This thesis aims to develop an approach for performing fuzzy semantic labeling on natural images by assigning multiple labels and associated confidence measures to fixed-sized blocks of images. More specifically, this thesis addresses the following problem:

> Given an image block R characterized by a set of features F_t, $t = 1, \ldots, n$ and m semantic classes C_i, $i = 1, \ldots, m$, compute for each i the confidence $Q_i(R)$ that the image region R belongs to class C_i.

Here, the confidence measure $Q_i(R)$ may be interpreted as an estimate of the confidence of classifying image block R into class C_i. Then, the fuzzy semantic label of block R, which contains the confidence measures, can be represented as the vector $\mathbf{v} = (Q_1(R), \ldots, Q_m(R))^T$.

Hence, with this study, we intend to make the following contributions:

- We develop a method that uses multi-class SVM outputs to produce fuzzy semantic labels for image regions.
- We demonstrate the proposed fuzzy semantic labeling method for a large number of semantic classes.
- The method we propose adopts an approach that uses all the confidence measures associated with the assigned multiple semantic labels when performing region matching.

- Furthermore, we also compare the performance of our proposed fuzzy semantic labeling method with those of two crisp labeling methods using multi-class support vector machine classifiers.

CHAPTER 2

Related Work

In this chapter, we review similar studies that present methods to associate image or image regions with words. First we cover studies that perform crisp semantic labeling, which involves classifying an entire image or part of an image into exactly one semantic class. This essentially results in assigning a single semantic label to an image. Then, we follow this with some representative studies that perform auto-annotation of images where multiple words, often called captions or annotations, are assigned to an image or image region. Finally, we review studies that propose methods that perform fuzzy semantic labeling where, similar to auto-annotation, several words are also assigned to an image or image region. But this time, a confidence measure is attached to each label.

2.1 Crisp Semantic Labeling

Early studies on content-based image retrieval initially focused on implementing various methods to assign crisp labels to whole images or image regions. Furthermore, these studies have also explored labeling methods based on a variety of extracted image features, sometimes separately and occasionally in combination.

In [SP98], Szummer and Picard classified whole images as indoor or outdoor scene using a multi-stage classification approach. Features were first computed for individual image blocks or regions and then classified using a k-nearest neighbor classifier as either indoor or outdoor. The classification results of the blocks were then combined by majority vote to classify the entire image. This method was found to result in 90.3% correct classification when evaluated on a database of over 1300 consumer images of diverse scenes collected and labeled by Kodak.

Vailaya et al. [VJZ98] evaluated how simple low-level features can be used to solve the problem of classifying images into either city scene or landscape scene. Considered in the study were the following features: color histogram, color coherence vector, DCT coefficient, edge direction histogram and edge direction coherence vector. Edge direction-based features were found to be best for discriminating between city images and landscape images. A weighted k-nearest neighborhood classifier was used for the classification resulting in an accuracy of 93.9% when evaluated on a database of 2716 images using the leave-one-out method. This method was also extended to further classify 528 landscape images into forest, mountain and sunset or sunrise scene. In order to do this, the landscape images were first classified as either sunset/sunrise or forest and mountain scene for which an accuracy of 94.5% was achieved. The forest and mountain images were then classified into either forest or mountain scene with an accuracy of 91.7%.

A hierarchical strategy similar to that used by Vailaya et al. was employed in another study carried out by Ciocca et al. [CCS$^+$03]. Images were first classified into either pornographic or non-pornographic. Then, the non-pornographic images were further classified as indoor, outdoor or close-up images. Classification was performed using tree classifiers built according to the classification and regression trees (CART).

This was demonstrated on a database of over 9000 images using color, texture and edge features. Color features included color distribution in terms of moments of inertia of color channels and main color region composition, and skin color distribution using chromaticity statistics taken from various sources of skin color data. Texture and edge features included statistics on wavelet decomposition and on edge and texture distributions.

Goh et al. [GCC01] investigated the use of margin boosting and error reduction methods to improve class prediction accuracy of different SVM binary classifier ensemble schemes such as one-vs-rest, one-vs-one and the error-correcting output coding (ECOC) method. To boost the output of accurate classifiers with a weak influence on making a class prediction, used a fixed sigmoid function to map posterior probabilities to the SVM outputs. In their error reduction method that uses what they call *correcting classifiers* (CC), they train, for each classifier separating class *i* from *j*, another classifier to separate class *i* and *j* from the other classes. Their proposed methods were applied to classify 1,920 images into one of fifteen categories. Color features extracted from an entire image included color histograms, color mean and variance, elongation and spreadness while texture features included vertical, horizontal and diagonal orientations. Using the fixed sigmoid function produced an average classification error rate of about 12 to 13% for the different SVM binary classifier ensemble schemes. Their correcting classifiers error reduction method further improved error rate by another 3 to 10%.

Then Wu et al. [WCL02] compared the performance of an ensemble of one-vs-rest SVM binary classifiers to that of an ensemble of one-vs-rest Bayes point machines when carrying out image classification. Using the same data set and image features in [GCC01], they found that the classification error rate for the ensemble

Bayes point machines of 0.5% to as 25.1% for the different categories considered did not vary much from that for the one-vs-rest SVM ensemble which ranged from 0.5% to 25.3%. Furthermore, they reported that the average error rate for the ensemble of Bayes point machines was lower than that of the one-vs-rest SVMs by just a margin of 1.6%.

Fung and Loe [FL99] presented an approach by defining image semantics at two levels, namely primitive semantics based on low-level features extracted from image patches or blocks and scene semantics. Learning of primitive semantics was performed via a two-staged supervised clustering where image blocks were grouped into elementary clusters that were further grouped into conglomerate clusters. Semantic classes were then approximated using the conglomerate clusters. Image patches were assigned to the clusters using k-nearest neighbor algorithm and then assigned the semantic labels of the majority clusters. The study however did not give quantitative classification results.

Town and Sinclair [TS00] showed how a set of neural network classifiers can be trained to map image regions to 11 semantic classes. The neural network classifiers—one for each semantic class—were trained on region properties including area and boundary length, color center and color covariance matrix, texture feature orientation and density descriptors and gross region shape descriptors. This method produced classification accuracies for the different semantic classes ranging from 86% to 98%.

Similar to [TS00], a neural network was trained as a pattern classifier in [CMT+97] by Campbell et al. But instead of using fixed-size blocks as image regions, images were divided into coherent regions using the k-means segmentation method. A total of 28 features representing color, texture, shape, size, rotation and centroid formed the basis for classifying the regions into one of 11 categories such as

sky, vegetation, road marking, road, pavement, building, fence or wall, road sign, signs or poles, shadows and mobile objects. When evaluated on a test set of 3751 regions, their method produced an overall accuracy of 82.9% on the regions.

Belongie et al. [BCGM97] also chose to divide an image into regions of coherent color and texture which they called *blobs*. Color and texture features were extracted and the resulting feature space was grouped into blobs using an Expectation-Maximization algorithm. A naïve Bayes classifier was then used to classify the images into one of twelve categories based on the presence or absence of region blobs in an image. Classification accuracy for the different categories ranged from as low as 19% to as high as 89%.

2.2 Auto-annotation

One of the earlier works on automatic annotation of images is that by Mori et al [MTO99] which employs a co-occurrence model. In their proposed method, images with key words are used for learning. Then when an image is divided into fixed-size image blocks, all image blocks inherit all words associated with the entire image. A total of 96 features, consisting of a 4×4×4 RGB color histogram and an 8-directions × 4-resolutions histogram of intensity after Sobel filtering, were calculated from each image block and then clustered by vector quantization. The estimated likelihood for each word is calculated based on the accumulated frequencies of all image blocks in each cluster. Then given an unknown image, the image is divided into image blocks from which features are extracted. Using these features, the nearest centroids for each image block are determined and the average of the likelihoods of the nearest centroids is calculated. Then words with the largest average likelihood are output. When applied on a database of 9,681 images with a total of 1,585 associated words, this

method achieved an average "hit rate" of 35%. "Hit rate" here is defined as the rate at which originally attached words appear among the top output words. Additional tests carried out and described in [MTO00] using varying vocabulary size showed that "hit rate" for the top ten words ranged from 25% when using 1,585 words to 70% when using 24 words. The "hit rate" for the top three words, on the other hand, ranged from 40% when using 1,585 words to 77% when using 24 words.

Barnard and Forsyth [BF01] use a generative hierarchical model to organize image collection and enable users to browse through images at different levels. In the hierarchical model, each node in the tree has a probability of generating each word and an image segment with given features: higher-level nodes emit larger image regions and associated words (such as sea and sky) while lower-level nodes emit smaller image segments and their associated words (such as waves, sun and clouds). Leaves thus correspond to individual clusters of similar or closely-related images. Taking blobs such as those in [BCGM97] as image segments, they train the model using the Expectation Maximization algorithm. Although they gave no specifics regarding the number of images and words used in their experiments, Barnard and Forsyth report that, on the average, an associated word would appear in the top seven output words.

In [BDF01], Barnard et al. further demonstrated the system proposed in [BF01] using 8,405 images of work from the Fine Arts Museum of San Francisco as training data and using 1,504 from the same group as their test set. When 15 naïve human observers were shown 16 clusters of images and were instructed to write down keywords that captured the sense of each cluster, about half of the observers on the average used a word that was originally used to describe each cluster.

In Duygulu et al. [DBF+02], image annotation is defined as a task of translating blobs to words in what is known as the translation model. Here, images are first segmented into regions using Normalized Cuts. Then only those regions larger than a threshold size are classified into region types (blobs) using *k*-means based on features such as region color and standard deviation, region average orientation energy, region size, location, convexity, first moment and ratio of region are to boundary length squared. Then the mapping between region types and keywords associated with the images is learned using a method built on Expectation Maximization (EM). Experiments were conducted using 4,500 Corel images as training data. A total of 371 words were included in the vocabulary where 4-5 words were associated with each image. In the evaluation tests, only the performance of the words that achieved a recall rate of at least 40% and a precision of at least 15% were presented. When no threshold on the region size was set, test results using a test set of 500 images reveal that the proposed method achieves an average precision is around 28% and average recall rate is 63%. The given average precision however includes an outlier value of 100% achieved for one word with an average precision of 21% for the remaining 13 words. Because only 80 out of the 371 words could be predicted, the authors considered re-running the EM algorithm using the reduced vocabulary. But this did not produce any significant improvement on the annotation performance in terms of precision and recall.

Jeon et al. [JLM03] use a similar approach by first assuming that objects in an image can be described using a small vocabulary of blobs generated from image features using clustering. They then apply a cross-media relevance model (CMRM) to derive the probability of generating a word given the blobs in an image. Similar to [DBF+02], experiments were conducted on 5,000 images which yielded 371 words

and 500 blobs. Test results show that with a mean precision of 33% and a mean recall rate of 37%, the annotation performance of CMRM is almost six times better than the co-occurrence model proposed in [MTO99] and twice better than the translation model of [DBF+02] in terms of precision and recall.

Blei and Jordan [BJ03] extended the Latent Dirichlet Allocation (LDA) Model and proposed a correspondence LDA model which finds conditional relationships between latent variable representations of sets of image regions and sets of words. The model first generates representative features for image regions obtained using Normalized Cuts and subsequently generates caption words based on these features. Tests were performed on a test set of 1,750 images from the Corel database using 5,250 images from the same database to estimate the model's parameters. Each image was segmented into 6-10 regions and associated with 2-4 words for a total of 168 words in the vocabulary. By calculating the per-image average negative log likelihood of the test set to assess the fit of the model, Blei and Jordan showed that their proposed Corr-LDA model provided at least as good a fit as the Gaussian-multinomial mixture and the Gaussian-multinomial LDA models. To assess annotation performance, the authors computed the perplexity of the outputted captions. They define perplexity as equivalent algebraically to the inverse of the geometric mean per-word likelihood. Based on this metric, Corr-LDA was shown to find much better predictive distributions of words than either of the two other models considered.

Similar to the models in [JLM03] and [BJ03], [LMJ03] presents a model called the continuous-space relevance model (CRM). Their approach aims to model a joint probability for observing a set of regions together with a set of annotation words rather than create a one-to-one correspondence between objects in an image and

words in a vocabulary. The authors stress that a joint probability captures more effectively the fact that certain objects (e.g., tigers) tend to be found in the same image more often with a specific group of objects (e.g. grass and water) than with other objects (e.g. airplane). With the same dataset provided in [DBF+02], CRM achieved an annotation recall of 19% and an annotation precision of 16% on the set of 260 words occurring in the test set; and an annotation recall of 70% and an annotation precision of 59% on the subset of 49 best words.

2.3 Fuzzy Semantic Labeling

Labeling methods using fuzzy region labels have been proposed in an attempt to overcome the limitations and difficulties encountered when labeling more complex images with crisp labels. Fuzzy region labels are primarily multiple semantic labels assigned to image regions.

A study by Mulhem, Leow and Lee [MLL01] recognized the difficulty of accurately classifying regions into semantic classes and so explored the approach of representing each image region with multiple semantic labels instead of single semantic labels. Disambiguation of the fuzzy region labels was performed during image matching where image structures were used to constrain the matching between the query example and the images.

The only study so far that has focused on fuzzy semantic labeling is that by Li and Leow in [LL03]. They further explored fuzzy labeling by introducing a framework that assigns probabilistic labels to image regions using multiple types of features such as adaptive color histograms, Gabor features, MRSAR and edge-direction and magnitude histograms. The different feature types were combined through a probabilistic approach and the best feature combinations were derived using feature-

based clustering using appropriate dissimilarity measures. The subset of features obtained was then used to label a region. Because feature combinations were used to label a region, this method could assign multiple semantic classes to a region together with the corresponding confidence measures. To evaluate the accuracy of the fuzzy labeling method, the image regions were classified into the class with the largest corresponding confidence measure. Using this criterion and without setting a threshold on the minimum acceptable confidence measure, a classification accuracy of 70% was achieved on a test set of fixed-size image blocks cropped from whole images.

2.4 Summary

The studies as reviewed in Section 2.1 have shown that a relatively high classification accuracy can be achieved using the crisp labeling methods that they proposed. But since these methods have been demonstrated on labeling at most 15 classes, the good classification performance may not necessarily be extendable to labeling a much larger number of semantic classes that commonly occur in a database of complex images. It is unlikely that very accurate classifiers can be derived in such a case because of the noise and ambiguity that are present in more complex images. Crisp labeling methods therefore may not be very practical when used for the labeling and retrieval of complex images.

In the auto-annotation methods, a much larger word vocabulary size, that is, number of classes in the context of the reviewed crisp labeling methods, was considered. However, the good evaluation test results reported can be deceiving as they cannot be directly compared with the results obtained for crisp labeling. The “hit rates”, for instance, in [MTO99] and [MTO00] reflect how often output words

actually include the words originally associated with the image. Naturally, "hit rates" will be higher because the group of output words is already considered correct if at least one of the original associated words appears in the output words. On the other hand, accuracy values reported in crisp labeling are based on how often a single word assigned to or associated with an image matches the single word originally associated with the image or image region. This is analogous to considering only the top one output word in auto-annotation. The same can be fairly said of accuracy values reported on region classification tests performed to assess the performance of fuzzy semantic labeling method in [LL03]. Thus, a "hit rate" of 70% obtained for the top three output words, for instance, may actually translate to a "hit rate" of roughly just 23% for the top one output word.

In [LL03] on fuzzy semantic labeling, aside from the high classification accuracy achieved, the probabilistic approach taken has the following advantages:

- It makes use of only those dissimilarity measures appropriate for the feature types considered.
- It adopts a learning approach that can easily adapt incrementally to the inclusion of additional training samples, feature types and semantic classes.

Although [LL03] presented a novel approach using fuzzy labeling and demonstrated it for 30 classes, a number larger than those used in the studies of crisp semantic labeling, it had not demonstrated the advantage of fuzzy semantic labeling over crisp labeling. Moreover, in the performance evaluation, only a single confidence measure (the one with the largest value) of a fuzzy label was used. Potentially useful information contained in the other confidence measures was omitted. We intend to address these shortcomings with the contributions made by our proposed fuzzy semantic labeling method as outlined in Section 1.2.

CHAPTER 3

Semantic Labeling

This chapter first discusses crisp semantic labeling to lay the foundation for our proposed fuzzy semantic labeling.

3.1. Crisp Semantic Labeling

Crisp semantic labeling is essentially a classification problem where an image or image region is classified into one of m semantic classes C_i where $i = 1, 2, \ldots, m$. As discussed in Chapter 2, crisp labeling involves assigning a single semantic label to the image or image region and can be carried out using a variety of methods based on various image features.

In this section, we discuss how crisp semantic labeling can be performed using multi-class classifiers based on Support Vector Machines (SVMs) [Vap95, CV95]. While several methods have been used to perform crisp labeling, we choose to use SVM for classification due to its advantages over other learning methods. SVM is guaranteed to find the optimal hyperplane separating samples of two classes given a specific kernel function and the corresponding kernel parameter values. This aspect leads to considerably better empirical results compared to other learning methods such as neural networks [Vap95]. Wu et al. [WCL02] in particular pointed out that

although SVMs achieved a slightly lower classification accuracy compared to Bayes point machines, SVMs are more attractive for image classification because they require a much lesser time to train. Chappelle et al. in [Cha99] also obtained good results when they tested SVM for histogram-based image classification.

3.1.1. Support Vector Machines

Support Vector Machines [Vap95, CV95] are learning machines designed to solve problems concerning binary classification (pattern recognition) and real-valued function approximation (regression). Since the problem of semantic labeling is essentially a classification problem, we focus solely on how SVMs perform classification. First, we describe how an SVM tackles the basic problem of binary classification.

In order to present the underlying idea behind SVMs, we first assume that the samples in one class are linearly separable from those in the other class. Within this context, binary classification using SVM is carried out by constructing a hyperplane

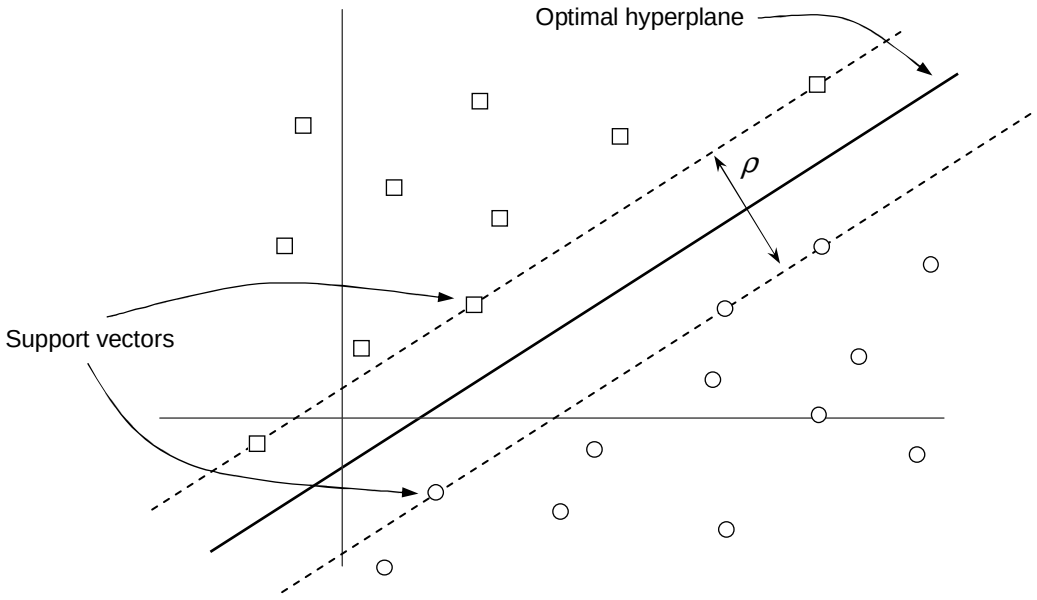


Figure 3.1. *An optimal hyperplane for the linearly separable case.*

that separates samples of one class from the other in the input space. The hyperplane is constructed such that the margin of separation between the two classes of samples is maximized while the upper bound of the classification error is minimized. Under this condition, the *optimal hyperplane* is defined by

$$\mathbf{w}^T\mathbf{x} + b = 0 \tag{3.1}$$

and the *margin of separation* ρ to be maximized is given by (Figure 3.1)

$$\rho = \frac{2}{\|\mathbf{w}\|}. \tag{3.2}$$

We can likewise define the following decision function

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b. \tag{3.3}$$

Given any sample represented by the input vector $\mathbf{x}$, the sign of the decision function $f(\mathbf{x})$ in Eq. 3.3 indicates on which side of the optimal hyperplane the sample $\mathbf{x}$ falls. When $f(\mathbf{x})$ is positive, the sample falls on the positive side of the hyperplane and is classified as class 1. On the other hand, when $f(\mathbf{x})$ is negative, the sample falls on the negative side of the hyperplane and is classified as class 2. Furthermore, the magnitude of the decision function, $|f(\mathbf{x})|$, indicates the sample's distance from the optimal hyperplane. In particular, when $|f(\mathbf{x})| \approx 0$, the sample falls near the optimal hyperplane and is most likely an ambiguous case. We may extend this observation by assuming that the nearer $\mathbf{x}$ is to the optimal hyperplane, the more likely is there an error in its classification by the SVM.

In practice, samples in binary classification problems are rarely linearly separable. In this case, SVM carries out binary classification by first projecting the feature vectors of the nonlinearly separable samples into a high-dimensional feature space using a set of nonlinear transformations $\mathbf{\Phi}(\mathbf{x})$. According to Cover's theorem, the samples become linearly separable with high probability when transformed into this

new feature space as long as the mapping is nonlinear and the dimensionality of the feature space is high enough. This enables the SVM to construct an optimal hyperplane in the new feature space to separate the samples. Then, the optimal hyperplane in the high-dimensional feature space is given by:

$$\mathbf{w}^T \mathbf{\Phi}(\mathbf{x}) + b = 0 \tag{3.4}$$

The nonlinear function $\mathbf{\Phi}(\mathbf{x})$ is a kernel function of the form $K(\mathbf{x}, \mathbf{x}_i)$ where $\mathbf{x}_i$ is a support vector. The decision function now is

$$f(\mathbf{x}) = \sum_i \mathbf{w}^T K(\mathbf{x}, \mathbf{x}_i) + b \tag{3.5}$$

Commonly used kernel functions $K(\mathbf{x}, \mathbf{x}_i)$ include linear function, polynomial function, radial base function or Gaussian and hyperbolic tangent (Table 3.1).

Although SVMs are originally designed to solve binary classification problems, multi-class SVM classifiers have been developed since most practical classification problems involve more than two classes. The main approach for SVM-based multi-class classification is to combine several binary SVM classifiers into a single ensemble. Generally, the class that is ultimately assigned to a sample arises from consolidating the different outputs of the binary classifiers that make up the ensemble. These methods include one-vs-one [KPD90], one-vs-rest [Vap98], Directed Acyclic Graph (DAG) SVM [PCS00], SVM with error-correcting output code (ECOC)

Table 3.1. *Commonly used SVM kernel functions*

Type	**Kernel Function**
Linear	$\mathbf{x}^T \mathbf{x}_i + 1$
Polynomial	$(\mathbf{x}^T \mathbf{x}_i + 1)^p$, $p > 1$
Gaussian	$\exp\left(-\frac{1}{2\sigma^2} \left\| \mathbf{x} - \mathbf{x}_i \right\|^2 \right)$
Hyperbolic tangent	$\tanh (\beta_0 \mathbf{x}^T \mathbf{x}_i + \beta_1)$

[DB91] and binary tree [Sal01]. Of these methods, only the one-vs-rest implementation and DAG SVM will be discussed in more detail because they are used in this study.

One-vs-rest SVM. One-vs-rest implementation [Vap98] is the simplest and most straightforward of the existing implementations of a multi-class SVM classifier. It requires the construction of m binary SVM classifiers where the uth classifier is trained using class u samples as positive samples and the remaining samples as negative samples. The class assigned to a sample is then the class corresponding to the binary classifier that classifies the sample positively and returns the largest distance to the optimal separating hyperplane.

An advantage of this method is that it uses a small number of m binary SVMs. However, since only m binary classifiers are used, there is a limit to the complexity of the resulting decision boundary. Moreover, when a large training set is used, training a one-vs-rest SVM can be time consuming since all training samples are needed in training each binary SVM.

Directed Acyclic Graph (DAG) SVM. Another implementation of a multi-class SVM classifier is the Directed Acyclic Graph (DAG) SVM developed by Platt et al. [PCS00]. A DAG SVM uses $m(m-1)/2$ binary classifiers arranged as internal nodes of a directed acyclic graph (Figure 3.2) with m leaves. Unlike the one-vs-rest implementation, each binary classifier in the DAG implementation is trained only to classify samples into either class u or class v. Evaluation of an input starts at the root and moves down to the next level to either the left or right child depending on the outcome of the classification at the root. The same process is repeated down the rest of the tree until a leaf is reached and the sample is finally assigned a class.

One advantage of DAG SVM is that it only needs to perform m-1 evaluations to

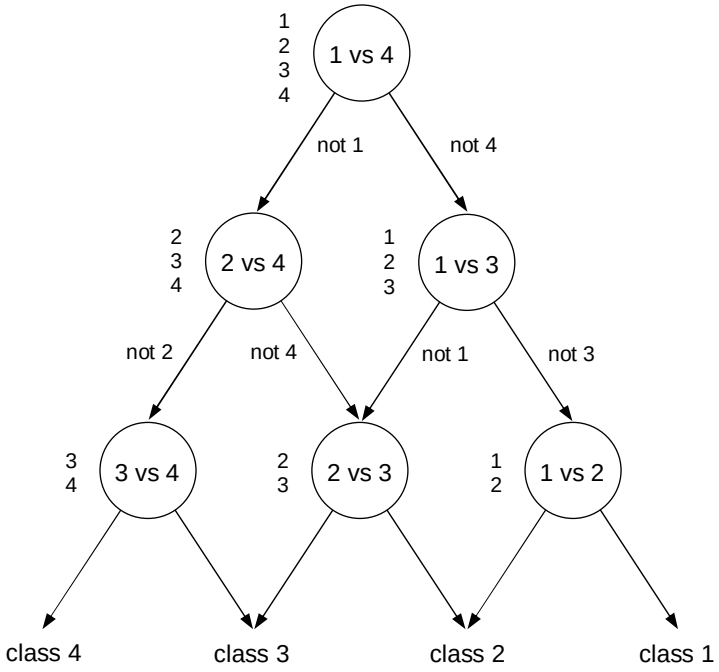


Figure 3.2. *A directed acyclic graph decision tree for the classification task with four classes.*

classify a sample. On the other hand, besides requiring the construction of $m(m\text{-}1)/2$ binary classifiers, DAG SVM has a stability problem: if just one binary misclassification occurs, the sample will ultimately be misclassified. Despite this problem, the performance of the DAG SVM is slightly better or at least comparable to other implementations of multi-class SVM classifier as demonstrated in [PCS00, HL02, Wid02].

3.1.2. Crisp Labeling Using SVMs

Using the multi-class SVM classifier implementations discussed in Section 3.1.1, we can assign crisp labels of m semantic classes to image regions in two ways as described below.

First crisp labeling method. The one-vs-rest implementation of the multi-class SVM classifier is used for labeling image regions with crisp labels. The j^{th} one-vs-rest binary SVM is trained to classify regions into either class j or non-class j. After

training, a region *i* is classified using all the *m* one-vs-rest binary classifiers. Then region *i* is assigned the crisp label *c* if among the SVMs that classify region *i* positively, the c^{th} SVM returns the largest distance between region *i*'s feature vector and its hyperplane. If no SVM classifies region *i* as positive, then region *i* would be labeled as "unknown".

Second crisp labeling method. The second crisp labeling method is to classify a region *i* using the DAG SVM into one of *m* semantic classes, say, class *c*. The crisp label of the region *i* would then be *c*.

3.2. Fuzzy Semantic Labeling

As stated previously, fuzzy semantic labeling is carried out by assigning multiple semantic labels along with associated confidence measures to an image or image region. Our proposed method assigns a *fuzzy label* or *signature* in the form of vector

$$\mathbf{v} = [v_1\ v_2 \ldots\ v_m]^T \tag{3.6}$$

where v_j is the confidence that the image or image region belongs to class *j*.

The fuzzy labeling algorithm mainly consists of two phases: the training phase (Section 3.2.1) and the labeling phase (Section 3.2.3). During image retrieval, fuzzy labels or signatures are matched and compared. The procedure we use in region matching is described in Section 3.2.4.

3.2.1 Training Phase

The training phase of the fuzzy labeling algorithm consists of two main steps: (1) train *m* one-vs-rest SVMs and (2) construct a confidence curve for each of the trained SVMs.

Step 1. *Train m one-vs-rest binary SVMs.*

The j^{th} SVM is trained using training samples to classify image regions into either class *j* or non-class *j*.

Step 2. *Construct confidence curves.*

A confidence curve is constructed for each SVM to approximate the relationship between a sample's distance to the optimal hyperplane and the confidence of the SVM's classification of the sample.

To obtain the confidence measures, we may examine the relationship between the distance $f(\mathbf{x})$ of a sample $\mathbf{x}$ from the hyperplane constructed by the SVM and the confidence of classification of the sample by the SVM. As stated earlier, the distance $f(\mathbf{x})$ of a sample $\mathbf{x}$ to the hyperplane is computed using the decision function given in Eq. 3.5.

Given the positions of samples in the feature space used by an SVM, an error in classification is more likely to occur for samples that fall near the optimal hyperplane. Samples that lie far away from the optimal hyperplane are more likely to be correctly classified than those that lie near the optimal hyperplane. This relationship between distance to hyperplane and likelihood of correct classification can be represented by a mapping or *confidence curve*. The confidence curve is obtained using a set of samples other than that used to train the SVMs and whose classes are known. This set of samples will be referred to as the set of generating samples or the *generating set* for the remainder of this thesis.

To obtain the confidence curve, the generating samples are first classified using each of the *m* SVMs trained in the training phase. For each SVM, the distance of each sample in the generating set to the hyperplanes is computed. The samples in the

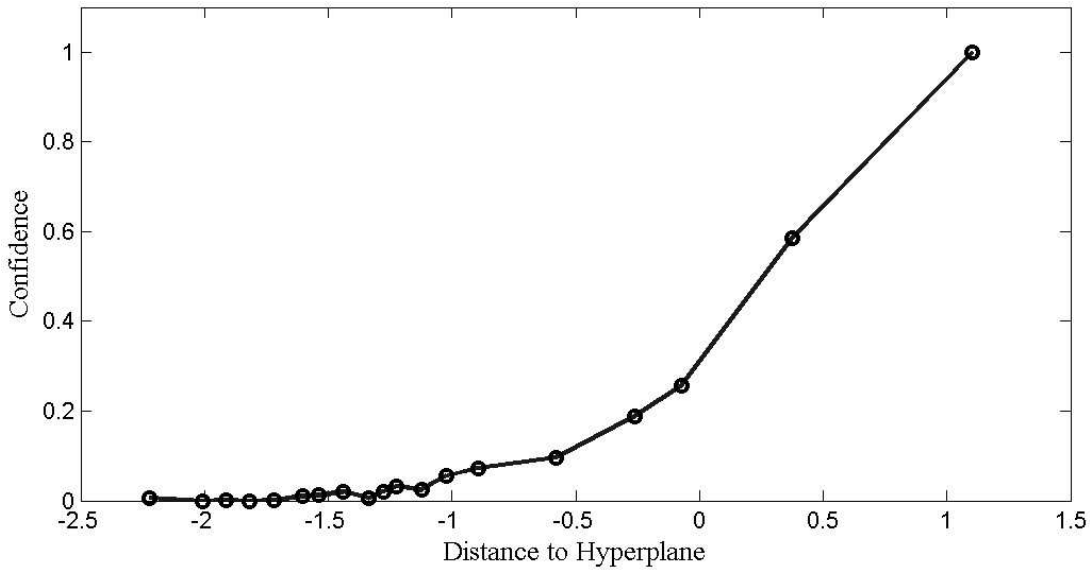


Figure 3.3. *A sample confidence curve.*

generating set are then sorted in increasing order of distance. A recursive algorithm, described in Section 3.2.2, is applied to recursively partition the range of distances into intervals such that the classification accuracy within each interval can be measured and the accuracy changes smoothly from one interval to the next. This results in a confidence curve such as that shown in Figure 3.3. We choose to obtain the confidence curve in this manner since we would like a confidence measure to be based on the classification accuracies of the samples in the generating set rather than be an arbitrary function of the distance d of a sample $\mathbf{x}$ to the hyperplane, such as the logistic function $(1 + \exp(d(\mathbf{x})))^{-1}$. Also note that while the resulting confidence curve is considerably smooth, it need not be monotonically increasing even if, ideally, confidence is expected to increase as distance from the hyperplane increases. Furthermore, since the classification accuracy is bounded between 0 and 1, the confidence curves of the SVMs also provide nonlinear normalizations of distance ranges of different SVMs to confidence measure within the [0,1] range.

3.2.2 Construction of Confidence Curve

The algorithm that constructs the confidence curve recursively partitions the range of distances of the samples into intervals such that the classification accuracy within each interval can be measured and the accuracy changes smoothly from one interval to the next. Imposing these two requirements essentially results in a smooth confidence curve.

Since the main goal now is to obtain a smooth curve, we can use the following rationale for the construction algorithm. In a smooth curve, the angles formed by line segments that define the curve are large whereas those in a jagged curve are small. Since we want to obtain a smooth curve, the algorithm aims to eliminate these small angles by merging intervals until all angles are greater than or equal to a pre-defined threshold.

Let us define a confidence curve $C = \{Z, E\}$ as consisting of a series of vertices $Z = \{z_0, z_1, z_2 \ldots z_n\}$ connected by n edges $E = \{e_1, e_2, \ldots, e_n\}$. Each edge is defined as $e_i = (z_{i-1}, z_i)$ for $i = 1, 2, \ldots, n$, i.e., the edge e_i has z_{i-1} and z_i as its endpoints. It follows that adjacent edges e_i and e_{i+1} form an angle θ_i with its vertex at z_i. In the context of our problem, the vertex z_i is the point with coordinates (μ_i, p_i) where μ_i defines the midpoint of the interval $[a_i, b_i]$ and p_i is the percentage of samples in the interval $[a_i, b_i]$ that belong to class c. The algorithm that constructs the smooth curve is shown as Figure 3.4.

The algorithm examines all angles θ_i and takes note of the smallest angle θ_{min}. Given that this angle has its vertex at point z_{min}, we look at the intervals corresponding to the two vertices adjacent to z_{min} and take the interval containing fewer samples. This interval $[a_x, b_x]$ is then merged with $[a_{min}, b_{min}]$. The result of merging the two intervals is illustrated in Figure 3.5. Merging is repeated until all θ_i are greater than

```
Repeat until θ_min ≥ θ*
   Find the smallest angle θ_min with vertex at z_min
      corresponding to interval [a_min, b_min].
   If θ_min < θ*
      Take interval [a_x, b_x] whose corresponding
         vertex z_x is adjacent to z_min and contains
         fewer samples.
      Merge interval [a_x, b_x] with interval [a_min, b_min]
```

Figure 3.4. *Algorithm for obtaining a smooth confidence curve.*

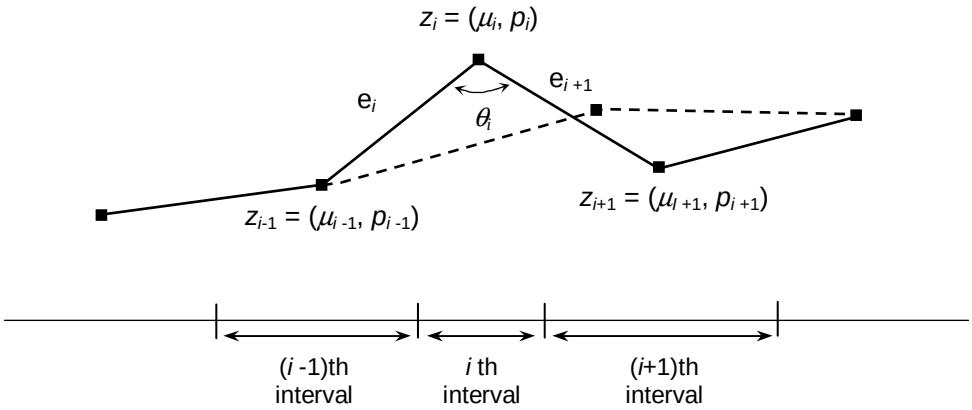
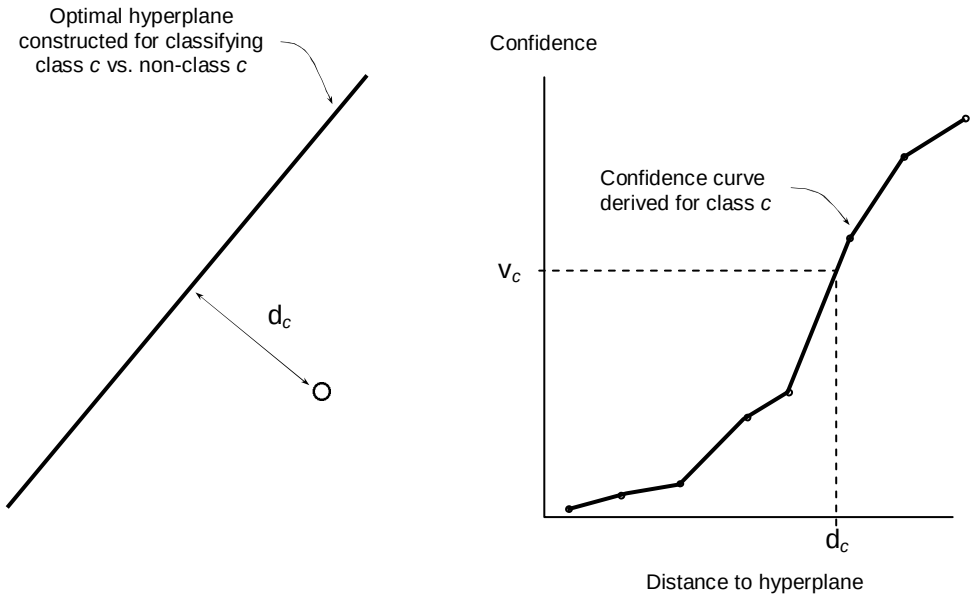


Figure 3.5. *A sample segment of a confidence curve showing angle θ_i defined by edges e_i and e_{i+1} that connect the vertices z_{i-1}, z_i and z_{i+1}. Dotted lines show the updated line segments after merging the ith interval with the (i+1)th interval.*

or equal to the given threshold θ^*. At this point, the resulting curve is now smooth since all angles on the curve are large.

Initially, all intervals contain a single sample such that $\mu_m = d_m$, the distance of the single sample in the interval to the hyperplane, and $p_m = 1$ if the sample was correctly classified and $p_m = 0$, otherwise.



(a) (b)

Figure 3.6. *Given (a) the distance d_c of a sample to the hyperplane, the expected confidence v_c of the sample can be estimated from (b) the confidence curve using linear interpolation.*

3.2.3 Labeling Phase

In the labeling phase, a sample is first classified using the SVMs trained in the training phase. The distances of the sample to the SVMs' hyperplanes are computed. The confidence measure v_c with respect to each SVM c is then obtained from the confidence curve using linear interpolation (Figure 3.6). This expected classification accuracy v_c can be regarded as the confidence measure for SVM c. Now the sample can be assigned a *fuzzy label* or *signature* $\mathbf{v} = [v_1\ v_2 \ldots\ v_m]^T$.

Note that with the first crisp labeling method using m one-vs-rest SVMs described in Section 3.1.3, a sample's signature would be $\mathbf{v}$ such that at most one of the v_j's is 1 if at least one of the binary classifiers classifies the sample positively. In the case where none of the binary classifiers in the one-vs-rest SVM implementation classifies the sample positively, the sample's signature would be a null vector. With the second crisp labeling method using DAG SVMs, exactly one of the v_j's is 1 and the rest are 0.

3.2.4 Region Matching

To perform region matching, we need to first obtain the prototype signatures of known samples. This requires two steps.

Step 1. *Obtain signatures of known samples.*

First, we take the same set of samples used to generate the confidence curves and obtain their signatures by following the steps discussed in the labeling phase. These signatures are needed in the next step where prototype signatures are obtained.

Step 2. *Obtain prototype signatures for each semantic class.*

A simple way to obtain prototype signatures is to take the average of the signatures $\mathbf{v}_{ci}$ of the n_c generating set samples belonging to semantic class c. That is,

$$\mathbf{p}_c = \frac{1}{n_c}\sum_{i}^{n_c}\mathbf{v}_{ci} \tag{3.7}$$

This clearly results in a single prototype signature $\mathbf{p}_c$ for each semantic class c.

However, a large variation of signatures can occur within a single semantic class due to the large variation of objects even within a semantic class. Thus we should obtain more than one prototype signature for each semantic class to capture the diversity of objects within a single semantic class. In order to obtain multiple prototype signatures, we perform clustering on those samples in the generating set belonging to class c according to their signatures. Two clustering methods were considered: k-means clustering and adaptive clustering proposed in [LL01]. In k-means clustering, the appropriate number of clusters k is chosen with the aide of silhouette values that measure how well the samples are clustered. Silhouette values are discussed in Section 3.2.5. For adaptive clustering [LL01], the maximum radius of the clusters, nominal separation between clusters and the minimum number of

samples per cluster were set. This enabled adaptive clustering to generate the most appropriate number of clusters given these restrictions.

After having obtained the clusters for each semantic class, the cluster centroids, i.e., the mean signatures of the samples in each of the k clusters belonging to a semantic class, are computed and taken as the prototype signatures $\mathbf{p}_{ci}$ of the semantic class c:

$$\mathbf{p}_{ci} = \frac{1}{n_{ci}} \sum_{j}^{n_{ci}} \mathbf{v}_{cij} \quad i = 1,...,k \tag{3.8}$$

where n_{ci} is the number of samples in the ith cluster for semantic class c. Since $k \geq 1$ a semantic class can therefore have more than one prototype signature.

In empirical tests, it was found for some prototype signatures $\mathbf{p}_{ci} = [\ p_{ci1}\ p_{ci2}\ \ldots\ p_{cim}\ ]^T$, $i = 1, \ldots, k$ of a semantic class c, that

$$\max_j \{p_{cij}\} \neq p_{cic}\,. \tag{3.9}$$

That is, the prototype signature of class c indicates that the confidence of belonging to class c is actually lower than those of other classes. Hence, these prototype signatures are misleading and are thus regarded as *unreliable* and may not be used in region matching.

Given the signature $\mathbf{v}$ of a sample region r and prototype signatures $\mathbf{p}_{ci}$ of a class c, the distance d between the region and the class c is simply the minimum Euclidean distance between $\mathbf{v}$ and $\mathbf{p}_{ci}$:

$$d(r,c) = \min_k\ d(\mathbf{v}, \mathbf{p}_{ci})\,. \tag{3.10}$$

The computation of Eq. 3.10 may include only the reliable prototype signatures or both the reliable and unreliable prototype signatures. In Chapter 4, we will show that

region matching performance is poorer when unreliable prototype signatures are used together with reliable prototype signatures.

3.2.5 Clustering Algorithms

K-means clustering and silhouette plots. K-means clustering is a well-known approach to generating a specific number of disjoint clusters. In the k-means clustering algorithm, each object is assigned to one of *k* clusters so that a given measure of dispersion among the clusters is minimized. Often this measure of dispersion is the sum of distances or sum of squared Euclidean distances of each sample from the mean or centroid of its cluster. Even though the algorithm is efficient, among its disadvantages is the difficulty in predicting what number of clusters will produce optimal clustering. One way to determine the optimal number of clusters is to run the algorithm over a range of values for *k* that are near the number of clusters one expects from the data. Then one can observe how the sum of distances reduces with increasing values of *k*. This procedure, however, can be tedious and inaccurate since it is often difficult in the first place to know what range of values for *k* to use. Many other criteria that can be used to solve the problem of selecting the optimal value for *k* are discussed by Milligan and Cooper in [MC85].

One proposed solution to this problem uses silhouette plots developed by Rousseeuw [Rou87]. Silhouette plots are graphical displays that can be used to aid in the interpretation and validation of cluster analysis results. Given the clusters generated by a clustering algorithm, a silhouette can be constructed for each cluster in order to show which samples lie well within the cluster and which do not. The silhouette of samples in a cluster is constructed by plotting the silhouette value of each sample in the cluster in decreasing order. The silhouette value of a sample measures how similar that sample is to other samples in its own cluster compared to

samples in other clusters. The silhouettes of all clusters generated are displayed in a single diagram, such as the three shown in Figure 3.7, in order to create an overall graphical representation of the clustering results. This allows the user to visually compare the quality of the clusters.

A silhouette value of a sample i, $s(i)$, is obtained as follows. Given a sample i that has been assigned to some cluster A, let $d_A(i)$ denote average dissimilarity of i to all other samples assigned to cluster A. Now consider another cluster C that is different from cluster A so that we have $d_C(i)$, the average dissimilarity of i to all samples assigned to cluster C. After computing for $d_C(i)$ for all other cluster $C \neq A$, determine cluster B for which $d_B(i)= \min d_C(i)$ for all $C \neq A$. Given these average dissimilarities, the silhouette value $s(i)$ is computed as:

$$s(i) = \frac{d_B(i) - d_A(i)}{\max\{d_A(i), d_B(i)\}}. \tag{3.11}$$

One can see that $-1 \leq s(i) \leq 1$. Moreover, when $s(i)$ is close to 1, this indicates that the sample i has been assigned to the most appropriate cluster, A, rather than to the closest second-best choice which is cluster B. An $s(i)$ that is about zero occurs when $a(i)$ and $b(i)$ are approximately equal suggesting that it is not too clear if sample i should have been assigned to either cluster A or cluster B. On the other hand, a silhouette value $s(i)$ that is close to –1 indicates that the sample i actually lies closer to cluster B than to cluster A. This indicates that sample i should have been assigned to cluster B rather than to A. Therefore, having been assigned to cluster A by the clustering algorithm, sample i in this case has possibly been assigned to the wrong cluster.

Rousseeuw further suggests that one may take the average silhouette value over all objects for a given clustering. This summary value can also be interpreted as the

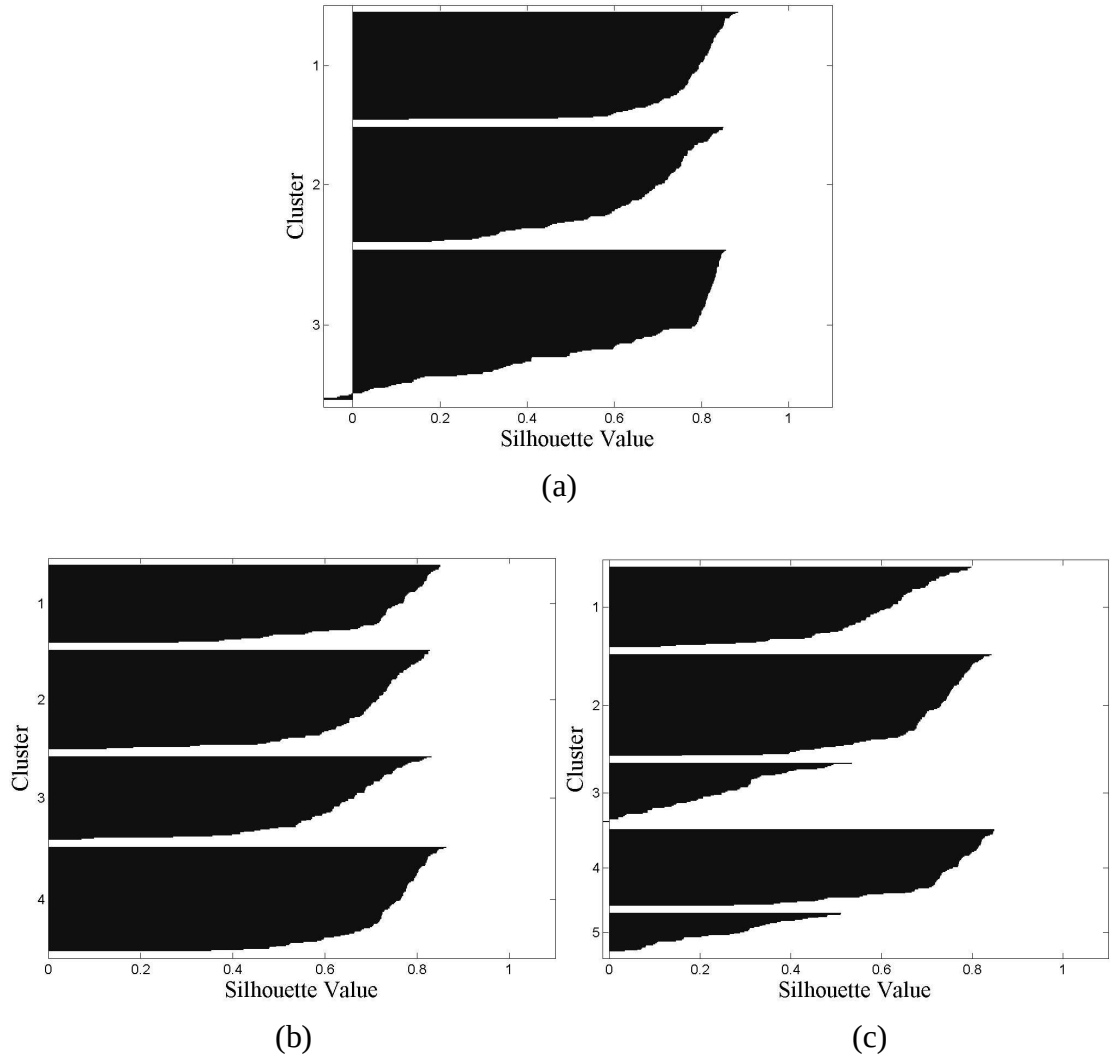


Figure 3.7. *Three silhouette plots obtained for the same data set. (a) k = 3. All three clusters have wide uniform silhouettes but some samples in cluster 3 have negative silhouette values indicating that this may not be the right number of clusters for the data set. (b) k = 4. All four clusters have wide uniform silhouettes and no sample has negative silhouette values. This indicates that this may be the best number of clusters for the data set. (c) k = 5. Two clusters (3 and 4) have samples with low silhouette values and cluster 3 has a sample with a negative silhouette value indicating that this may not be the right number of clusters for the data set either.*

```
Repeat
   For each sample p
      Find the nearest cluster k to sample p.
      If no cluster is found or distance d_kp ≥ S
         create a new cluster containing sample p.
      Else if d_kp ≤ R
         add sample p to cluster k.
   For each cluster i
      If cluster i has at least N_m samples
         update centroid c_i of cluster i.
      Else remove cluster i.
```

Figure 3.8. *Adaptive clustering algorithm.*

average silhouette plot width for the entire data set. More importantly, it also can be used for the selection of the optimal value of *k* by choosing the *k* for which the average silhouette value is as high as possible. Thus, the optimal number of clusters *k* is that for which the overall average silhouette value or overall average silhouette width is the largest as illustrated in Figure 3.7.

Adaptive clustering. The adaptive clustering algorithm proposed in [LL01] overcomes the problem of finding the appropriate number of clusters encountered with ordinary k-means clustering. By fixing the maximum cluster radius *R* and the nominal separation *S* between clusters, the algorithm generates only those clusters that meet these criteria. The adaptive clustering algorithm is described in Figure 3.8.

The adaptive clustering algorithm assigns a sample *p* to the nearest cluster *A* if it is near enough to cluster *A*. Else, if it is too far away from the nearest cluster, a new cluster is created containing the sample *p*.

This clustering algorithm ensures that each cluster has a maximum radius of *R* and that clusters are separated by a distance of approximately *S*. Moreover, it also ensures

that each cluster contains a significant number of samples because small clusters are removed. Hence it can also automatically determine the appropriate number of clusters. Adaptive clustering has been shown to be effective in creating adaptive color histograms for both image [LL01] and texture [LL02] retrieval and classification.

CHAPTER 4

Evaluation Tests

A series of evaluation tests were performed to quantitatively assess the performance of the proposed fuzzy semantic labeling method as well as compare it with the crisp labeling methods.

4.1. Image Data Sets

A variety of 31 semantic classes were identified. Descriptions of typical image blocks for each semantic class are given in Table 4.1 while sample image blocks are shown in Figure 4.1

For each semantic class, a total of 550 image blocks of size 64×64 pixels were cropped from images in the Corel library of 50,000 photos. The image blocks were cropped such that each block contained objects of only *one* semantic class. Each set of 550 well-cropped image blocks was further divided into three sets:

1) The *training set* contains 375 image blocks chosen at random to be used to train the Support Vector Machines (SVMs).
2) The *generating set* contains 125 image blocks chosen at random to be used for constructing the confidence curves and for obtaining the prototype signatures of each semantic class.

Table 4.1 *Descriptions of image blocks for the 31 selected semantic classes.*

Class name	Description
big building	buildings either singly or in groups as in cityscapes
brick wall	manmade stonework such as brick walls and stone walls
calm water	water surfaces featuring little or no surface structure (i.e. reflective surfaces)
choppy water	water surfaces featuring more prominent surface structure (waves and white water) such as that occurring on stormy sea surfaces and water falls
clear day sky	sky regions that are of relatively uniform color including that during dawn and twilight
clouds	sky regions that are covered partially or completely by clouds
dome	architectural domes, towers and steeples
fence	fences mainly featuring vertical structures such as picket fences, metal fences and ceramic banisters and excludes privacy-type fences such as stone walls
fireworks	various firework displays generally displayed against a night sky
flames	fire, lava flows, candle flames
flowers	single blooms or compound flowers
foliage	leaves, shrubbery and tree foliage (include summer and autumn foliage)
fur	animal fur
grass	grass-like vegetation including lawns and grasslands
house	small houses either singly or in groups.
human face	human faces in various views ranging from full-frontal to three-quarters
mountain	unobscured mountain peaks
night sky	featureless (moonless and starless) regions of sky during nighttime
paved road	road surfaces such as concrete or cement roads and cobblestone roads.
pebbles	pebbles and gravel
pillars	pillars, posts and columns
rock face	naturally occurring single rocks, rock faces and rocky mountain sides
roof	metal, tiled or thatched roofs.
sand	sandy surfaces such as that which occurs on beaches and desserts
scales	scale covering on reptiles, amphibians and fish
snow	snow covered surfaces
soil	ground surfaces
staircase	stairways and stepped structures
tree trunks	tree bark and trunks of trees appearing singly or in groups
window	windows occurring singly or in groups.
wooden surface	bare and painted or stained wood surfaces

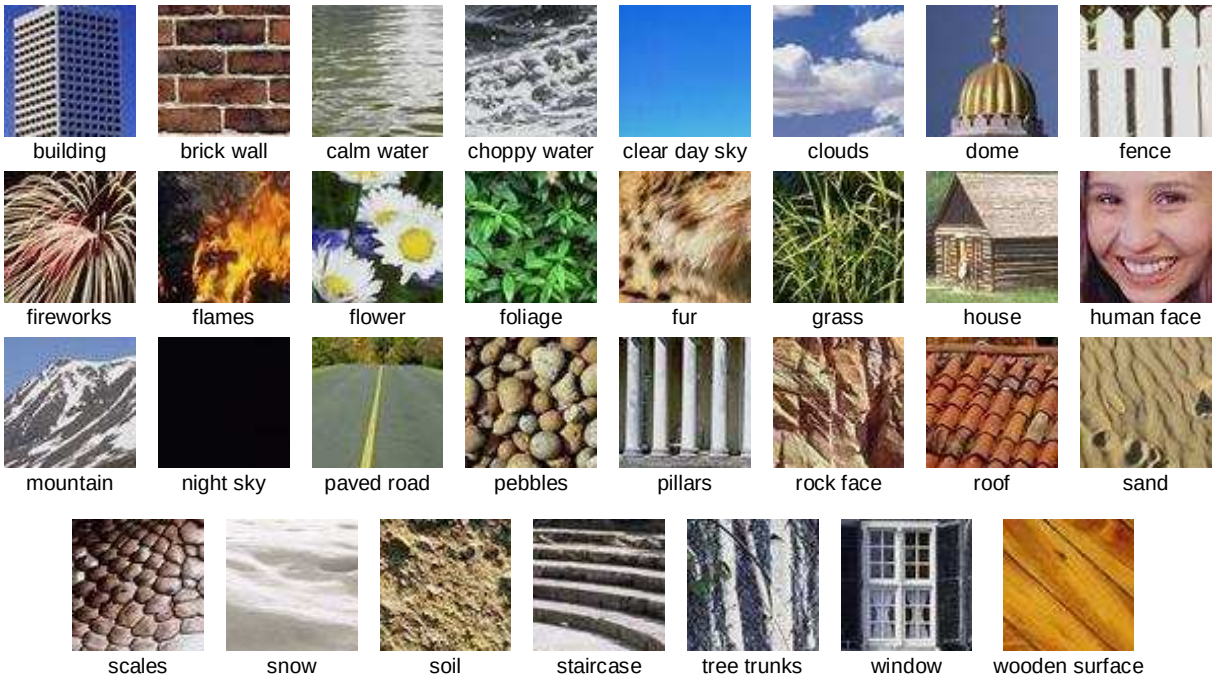


Figure 4.1. *Sample images of 31 semantic classes used.*

3) The *testing set* contains 50 image blocks chosen at random for evaluating the performance of fuzzy semantic labeling.

In total, there were 11,625 image blocks in the training set, 3875 image blocks in the generating set and 1,550 image blocks in the testing set.

In practice, however, semantic labeling almost always has to be performed on image regions that do not necessarily contain objects of a single semantic class. Moreover, the object of interest may not be centered in the image block, i.e., the image blocks are not always well-cropped. Hence, to evaluate how well the labeling method can generalize to image blocks that are not well-cropped, an additional 800 images were selected from the Corel photo library to form a *general test set.* The selection of images was made to ensure that among these images, at least 25 contain regions of big buildings, at least 25 contain brick walls, and so on. Each image was partitioned at regular intervals into 77 overlapping image blocks of size 64×64 pixels. Each image block was manually assigned a label to denote the ground truth

which was one of the following:

- one of the 31 semantic classes if the image block contained objects belonging to exactly one semantic class;
- "unknown" if the image block contained objects that did not belong to any of the 31 of the semantic classes; or
- "ambiguous" if the image block contained objects in more than one semantic class.

As a result of the manual assignment of labels, a total of 26179 (42.5%) image blocks were labeled with one of the known semantic classes, 4588 (7.4%) were labeled as "unknown" and 30833 (50.1%) were labeled as "ambiguous". The set of image blocks that were labeled with one of the known semantic classes was used in the evaluation tests in addition to the test set of well-cropped image blocks.

4.2 Low-Level Image Features

Four different types of low-level features, namely, fixed color histograms, Gabor features, multiresolution simultaneous autoregressive features (MRSAR) and edge histograms, were extracted from each image block *i*. These features are generally known to be good features for image classification and retrieval and have been used singly or in combination in existing methods. In this study we chose to use all four features together. The different features were concatenated into a single feature vector of 274 dimensions of which 165 were for color histogram, 30 for Gabor features, 15 for the MRSAR features and the remaining 64 for edge histogram.

Since the different features were of different scales but were assumed to be equally important in training the support vector machines, the data were normalized across different feature types. Principle component analysis (PCA) was used to

determine the normalization factor. The data were normalized so that the largest eigenvalues for the four feature types were the same. This prevents accidental biasing of the one-vs-rest support vector machines towards those feature types with large values.

4.2.1 Fixed Color Histogram

The colors in an image or image block play a major role in distinguishing objects among different semantic classes. For example, image blocks containing objects belonging to the semantic classes grass and foliage are generally green, those of the class clear day sky is almost always blue and those of snow are usually white. Color histograms are often used to represent the distribution of color features in an image or image block. This is often preferred to a single feature since a distribution can more fully describe the variation of colors that occurs throughout the image or image block. In the fixed-binning method for obtaining a color histogram, the color space is partitioned into rectangular bins [SB91]. The method is called "fixed-binning" because once the different bins are derived, the same binning scheme is applied to all images or image blocks.

4.2.2 Gabor Feature

Gabor texture features [MM96] are features for measuring texture differences. This is especially useful for structured and oriented features which occur in some semantic classes considered in this study such as pillars, brick walls and staircases.

The Gabor wavelet transform of an image $I(x, y)$ can be defined as:

$$W_{mn}(x,y) = \int I(x_1,y_1) g_{mn} * (x - x_1, y - y_1) dx_1 dy_1 \qquad (4.1)$$

where * indicates the complex conjugate and $g_{mn}(x, y)$ is the generating function used to obtain the class of self-similar functions called Gabor wavelets. It is assumed that

the local texture regions are spatially homogenous. The mean and standard deviation of the magnitude of the transform coefficients defined below are used to represent the region for classification and retrieval purposes:

$$\mu_{mn} = \iint |W_{mn}(xy)|\, dxdy \quad \text{and} \quad \sigma_{mn} = \sqrt{\iint \left(|W_{mn}(x,y)| - \mu_{mn}\right)^2 dxdy}\,. \quad (4.2)$$

In this study, we set the number of scales to five and the number of orientations to six resulting in the feature vector

$$\bar{f} = [\mu_{00}\ \sigma_{00}\ \mu_{01}\ \sigma_{01} \ldots \mu_{30}\ \sigma_{30}]. \quad (4.3)$$

4.2.3 Multi-resolution Simultaneous Autoregressive (MRSAR) Feature

Other natural images can contain random textures such as foliage and fireworks instead of structured textures. In order to capture the characteristics of random textures, multi-resolution simultaneous autoregressive (MRSAR) features [MJ92] were also extracted from the image block samples.

The MRSAR model is a second-order model described by five parameters at each resolution level. In this study, a symmetric MRSAR was applied to the L^* component of the $L^*u^*v^*$ image data. The pixel value $L^*(\mathbf{x})$ at a certain location $\mathbf{x}$ was modeled as a linearly combination of the pixel values $L^*(\mathbf{y})$ of the neighboring pixels $\mathbf{y}$ and a zero-mean additive independent Gaussian noise term $\varepsilon(\mathbf{x})$ as shown in the following formula:

$$L^*(\mathbf{x}) = \mu + \sum_{y \in u} \theta(\mathbf{y}) L^*(\mathbf{y}) + \varepsilon(\mathbf{x}) \quad (4.4)$$

where μ is the bias that is dependent on the mean value of L^*, u is the set of neighbors of the pixel at location $\mathbf{x}$, and $\theta(\mathbf{y})$ are the model parameters. The neighbors were defined for the three different window sizes used: 5×5, 7×7 and 9×9. These window sizes are said to provide the best overall retrieval performance over the entire Brodatz

database according to [PKL93] and [LP96].

Five parameters were used to represent the MRSAR models. These parameters include the bias μ and the four model parameters $\theta(\mathbf{y})$, one at each neighboring position $\mathbf{y}$. These parameters were estimated using the least squares technique. This procedure was repeated for each of the three window sizes considered to form a 15-dimensional feature vector.

In order to extract the MRSAR feature of a given image block, a 21×21 overlapping window was moved over the image at increments of two pixels in both the horizontal and vertical directions, obtaining a multi-resolution feature vector each time. The mean vector $\mathbf{t}$ over all windows in a given image block are the MRSAR features associated with that image block.

4.2.4 Edge Direction and Magnitude Histogram

Normalized edge direction and magnitude histograms [Bra99, BLO00] were extracted from the images in the following manner:

The image block was first transformed to the HSI (hue, saturation, intensity) space. The hue channel was neglected while other two channels were convolved with the eight Sobel operators [Bra99], one for each of eight quantized directions. For each pixel, its gradient magnitude was taken as the largest magnitude of the responses of the Sobel operators, and its directions was taken as the quantized direction of the corresponding operator. Then the pixels with low gradient magnitudes were discarded. Next, the gradient magnitudes of the remaining pixels were quantized into eight levels. This set of pixels with eight quantized directions and eight quantized magnitudes form the 8×8 edge histograms.

4.3 Parameter Settings

4.3.1 SVM Kernel and Regularizing Parameters

The choice of the regularizing parameter *C* and the kernel of a support vector machine affect the SVM's performance and thus have to be chosen with care. One method that can be used to help make the selection is to measure the performance of the SVM on testing samples under various parameter values. Another is an analytical approach that requires estimating the bounds on the generalization performance. In this study, we chose to use the former method for practical reasons.

The binary one-vs-rest SVMs were first trained and evaluated using five representative semantic classes with different kernels and regularizing parameter values. Only five semantic classes were used in these preliminary tests because training for all 31 semantic classes was very time consuming. The five classes were selected based on initial tests performed to identify those classes for which the SVM yielded the best, average and worst performance.

When a polynomial kernel was used, no convergence of training occurred. Moreover, no significant difference in performance was observed for different values of *C*. Tests were further carried out using the Gaussian kernel with various values of σ and regularizing parameter *C* set to 100.

Tables 4.2 and 4.3 show the classification precision and the classification accuracy achieved for each of the five selected classes for varying values of σ. The average precision and average accuracy computed over the five selected classes are plotted against σ in Figure 4.2. Precision and accuracy in this context are defined as follows:

Let A_c be the set of testing samples that actually belong to class *c* and let S_c be the

Table 4.2 *Precision achieved for selected five classes in preliminary tests using different values for Gaussian kernel parameter σ.*

		Value for kernel parameter σ						
	Class	**0.100**	**0.125**	**0.250**	**0.375**	**0.500**	**0.625**	**0.750**
1	grass	81.8%	82.4%	86.4%	82.9%	85.2%	87.0%	90.0%
2	foliage	64.0%	65.5%	77.8%	83.1%	89.1%	92.3%	93.3%
3	flowers	80.2%	84.1%	85.3%	87.4%	90.7%	92.2%	96.9%
6	rocks	40.9%	43.8%	50.8%	65.8%	82.6%	94.1%	100.0%
18	brick	55.8%	58.9%	64.5%	73.0%	77.2%	82.2%	93.8%
	Average	**69.6%**	**71.7%**	**76.9%**	**81.5%**	**87.1%**	**91.0%**	**95.3%**

Table 4.3 *Accuracy achieved for selected five classes in preliminary tests using different values for Gaussian kernel parameter σ.*

		Value for kernel parameter σ						
	Class	**0.100**	**0.125**	**0.250**	**0.375**	**0.500**	**0.625**	**0.750**
1	grass	72.0%	71.2%	60.8%	50.4%	41.6%	32.0%	28.8%
2	foliage	45.6%	44.0%	39.2%	39.2%	32.8%	28.8%	22.4%
3	flowers	71.2%	72.0%	69.6%	66.4%	62.4%	56.8%	49.6%
6	rocks	30.4%	28.0%	24.0%	20.0%	15.2%	12.8%	8.0%
18	brick	42.4%	42.4%	39.2%	36.8%	35.2%	29.6%	24.0%
	Average	**58.9%**	**58.1%**	**53.1%**	**48.8%**	**44.1%**	**39.1%**	**34.3%**

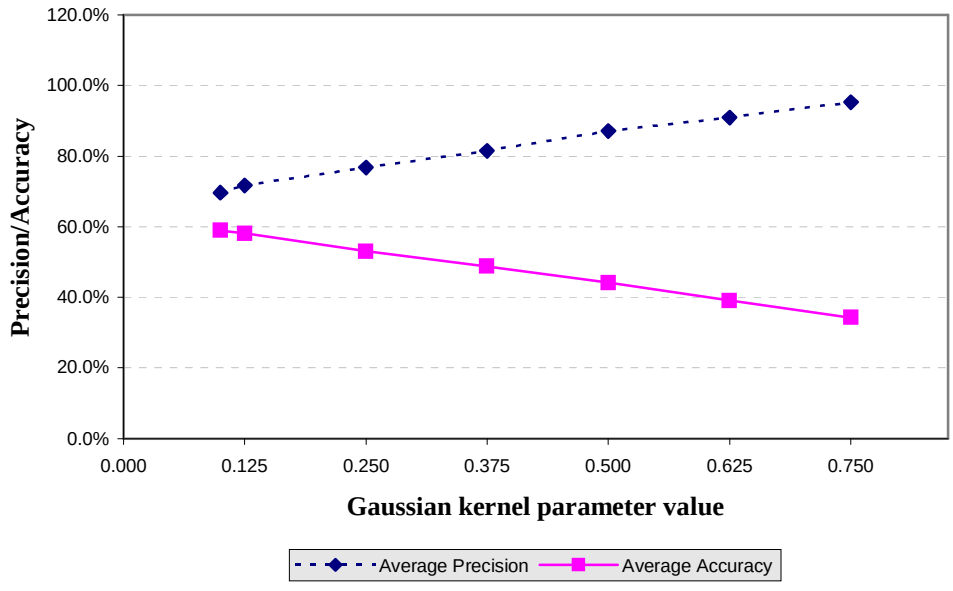


Figure 4.2 *Average precision and average accuracy achieved for different values of Gaussian kernel parameter σ.*

set of testing samples that the SVM labels as class *c*. Then,

$$precision = \frac{|A_c \cap S_c|}{|S_c|} \quad \text{and} \quad accuracy = \frac{|A_c \cap S_c|}{|A_c|}.$$

Precision was also considered as a criterion in addition to accuracy in selecting the best parameter settings in view of eventually using the proposed fuzzy semantic labeling method for image retrieval.

From the results in Tables 4.2 and 4.3 and Figure 4.1, it is understandable that there was a trade off between precision and accuracy: precision increased as σ decreased while the reverse occurred with accuracy. It may be desirable to choose σ = 0.75 because the average precision was at its highest at 95.3% at this point but then the corresponding average accuracy of 34.3% was considered too low. The value for σ where the most acceptable balance between precision and accuracy was thought to have been achieved was 0.125 where precision was still of an acceptable level at 71.7% and accuracy was more acceptable at 58.1%.

Thus, the Gaussian kernel was used in the final evaluation tests with kernel parameter σ set at 0.125 and regularizing parameter *C* at 100 since these settings yielded the most balanced results based on precision and accuracy.

4.3.2 Adaptive Clustering

Adaptive clustering discussed in Section 3.2.3 requires setting maximum cluster radius *R*, nominal separation *S* among the clusters and the minimum number of samples per cluster. Similar to the empirical approach used in selecting the kernel and regularizing parameters for the SVM, the best radius was selected by generating clusters over a range of values for *R* and measuring the performance of the fuzzy semantic labeling algorithm on the testing set. At the same time, nominal separation *S*

Table 4.4 *The average, maximum and minimum number of clusters for different values of cluster radius R.*

Number of Clusters	Radius *R*								
	0.025	**0.050**	**0.075**	**0.100**	**0.125**	**0.150**	**0.175**	**0.200**	**0.225**
Average	0.13	0.45	0.74	1.32	2.03	2.71	2.52	2.71	2.74
Max	2	3	3	2	3	5	4	5	6
Min	0	0	0	0	0	1	1	1	1

Number of Clusters	Radius *R*								
	0.250	**0.275**	**0.300**	**0.325**	**0.350**	**0.375**	**0.400**	**0.425**	**0.450**
Average	2.74	2.19	2.13	2.03	2.00	2.03	1.81	1.84	1.74
Max	4	4	3	3	3	4	3	3	3
Min	1	1	1	1	1	1	1	1	1

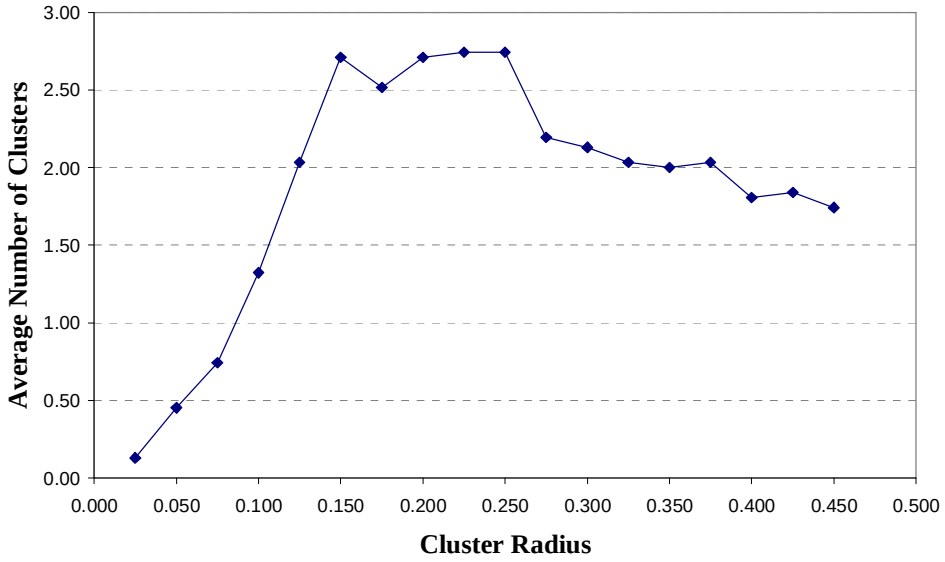


Figure 4.3 *Average number of clusters over all classes for different values of cluster radius R.*

Table 4.5 *Average accuracy achieved for selected values of cluster radius R.*

	Radius *R*			
	0.200	**0.225**	**0.250**	**0.275**
Accuracy	32.2%	33.0%	36.2%	41.7%

was set such that no overlap occurs among the clusters. Clusters with less than ten samples were also discarded.

Figure 4.3 shows the average number of clusters taken over all 31 semantic classes for the different values of radius tested. We can see that the average number of clusters peaks when the cluster radius was set to 0.225 and 0.250. The average, maximum and minimum number of clusters for the different values of cluster radius R tested are shown in Table 4.4. These results on the number of clusters show that when the cluster radius is small, no clusters were generated at all for at least one semantic class. Naturally this is an undesirable situation since we want all the classes to have clusters to enable us to obtain prototype signatures for all classes. The largest number of clusters generated for a class was six when cluster radius is 0.225.

Since the average number of clusters peaks at around $R = 0.225$ and $R = 0.25$, performance tests were carried out in this small neighborhood of cluster radius values Results of these performance tests on a testing set shown in Table 4.5 imply that the accuracy is highest when $R = 0.275$ despite a lower average number of clusters compared to the other cases. Therefore, a cluster radius of 0.275 was chosen.

4.3.3 Prototype signatures

As part of the region matching phase of the proposed fuzzy labeling method, prototype signatures were obtained for each semantic class via k-means clustering and adaptive clustering. The application of the clustering algorithms on the signatures of the samples of a semantic class was expected to identify groups of homogenous signatures from which the prototype signatures for the semantic class can be derived. Since a prototype signature was taken for each cluster that results from clustering, the number of prototype signatures may be taken as a measure of the variation occurring within each semantic class.

For the case where k-means clustering was used, the clustering algorithm was performed for $k = 2,\ldots,15$ clusters. The number of clusters with the highest overall average silhouette value was selected to be the best value for k. The Matlab Statistical Toolbox functions `kmeans` and `silhouette` were used for this purpose.

A single cluster had to be used for the class of night sky image blocks. When applied to the signatures of the night sky image blocks, the `kmeans` function produced an error for all the values of k tested when a cluster became empty during reassignment of samples among clusters. Hence, the prototype signature for night sky was obtained by taking the average of all signatures of the construction samples for the class.

Table 4.6 shows the number of prototype signatures obtained for each semantic class using the two clustering methods considered in this study. From Table 4.6 we can see that with k-means clustering, most of the semantic classes produced just two or three clusters. The classes that produced the most clusters was big buildings (13 clusters) followed by clouds (10 clusters). The largest number of clusters that yielded reliable prototype signatures for a single class was four for the classes cloud, calm water and soil. The rest of the classes had either one or two clusters that produced reliable prototype signatures.

There was a smaller variation in the number of clusters resulting from applying adaptive clustering. Here, the largest number of clusters was four for class choppy water. Not all samples were included in the resulting clusters because only those clusters with at least ten samples were considered. The largest number of clusters that produced reliable prototypes was three for the classes human face, window and staircase.

Table 4.6. *Results of k-means clustering and adaptive clustering on signatures of known samples. The table shows the number of clusters and the number of samples included in these clusters both for all clusters generated (All) and for only those clusters that contained reliable prototype signatures (Reliable).*

		K-means clustering				Adaptive clustering			
		# of clusters		# of samples included		# of clusters		# of samples included	
	Class name	All	Reliable	All	Reliable	All	Reliable	All	Reliable
1	grass	2	2	125	125	2	2	104	104
2	foliage	3	3	125	112	3	1	111	55
3	flowers	2	2	125	125	2	1	116	87
4	clouds	10	4	125	73	3	1	103	57
5	clear sky	2	2	125	125	1	1	92	92
6	rocks	8	2	125	75	2	2	102	102
7	mountain	2	2	125	125	2	2	104	104
8	sand	4	1	125	62	2	1	90	53
9	calm water	8	4	125	91	2	2	75	75
10	choppy water	5	2	125	98	4	1	105	52
11	fur	3	1	125	114	2	1	83	73
12	human face	2	2	125	125	3	3	117	117
13	pebbles	2	2	125	125	2	2	121	121
14	snow	3	1	125	84	2	1	101	82
15	roof	4	2	125	114	2	2	106	106
16	paved road	3	2	125	63	2	1	93	37
17	dome	2	2	125	125	2	2	92	92
18	brick wall	2	2	125	125	3	2	101	59
19	tree trunk	2	2	125	125	2	2	104	104
20	wooden surface	2	1	125	79	2	1	94	74
21	window	3	2	125	120	3	3	104	104
22	fences	2	2	125	125	2	2	112	112
23	flames	2	2	125	125	2	2	121	121
24	fireworks	2	2	125	125	3	1	102	80
25	night sky	1	1	125	125	1	1	113	113
26	big building	13	2	125	38	1	1	96	96
27	house	2	2	125	125	2	2	110	110
28	soil	9	4	125	79	2	2	96	96
29	scales	2	2	125	125	2	1	92	61
30	pillars	2	2	125	125	2	2	102	102
31	staircase	2	2	125	125	3	3	115	115

If we compare the number of samples that were included in those clusters that yielded reliable prototype signatures, a larger number of samples were ultimately included for nearly all classes when k-means clustering was used. The only exceptions were rocks, big buildings and soil. Another interesting difference between the results produced by the two clustering algorithms is the number of clusters for the class big buildings. Here, k-means clustering produced 13 clusters out of which only two yielded reliable prototype signatures accounting for just 38 samples. On the other hand, adaptive clustering produced a single cluster containing 96 samples which yielded a reliable prototype signature.

4.3.4 Confidence Curve

The choice for threshold angle θ^* in the recursive algorithm for producing the confidence curve is highly subjective. A few values of θ^* were tested and that which produced the smoothest confidence curves for *all* classes was chosen. Thus, θ^* was set to $97\pi/120$ or approximately 155°.

The sample confidence curve shown in Figure 3.3 resulting from the above setting is typical of the confidence curves obtained for the different semantic classes. It is interesting to note that beyond a distance of –1, that is, one unit beyond the margin of separation on the negative side of the optimal hyperplane, the percentage of image blocks in the generating set belonging to class *c* practically drops to near zero.

4.4 Semantic Labeling Tests

4.4.1 Experiment Set-Up

To assess the accuracy of fuzzy semantic labeling quantitatively, a region classification test was performed on both the well-cropped image blocks and the

general test image blocks. In carrying out these region classification tests, our main goal, as stated at the beginning of this chapter, is to compare the performance of our proposed fuzzy semantic labeling method with that of crisp labeling methods based on SVMs. We also aim to compare the performance of fuzzy labeling when using single prototype signatures per class to that when using several prototype signatures obtained through clustering, as well as to determine if excluding unreliable prototype signatures will affect performance. The following labeling methods were compared:

- crisp labeling using DAG SVM trained with the training set only,
- crisp labeling using DAG SVM trained with a combination of both the training set and the generating set,
- crisp labeling using one-vs-rest SVMs trained with the training set only,
- crisp labeling using one-vs-rest SVMs trained with a combination of both the training set and the generating set
- fuzzy labeling using a single prototype signature per class,
- fuzzy labeling using all prototype signatures obtained by k-means clustering,
- fuzzy labeling using all prototype signatures obtained by adaptive clustering,
- fuzzy labeling using reliable prototype signatures obtained by k-means clustering,
- fuzzy labeling using reliable prototype signatures obtained by adaptive clustering.

The training and the generating sets used to training the DAG SVM and the one-vs-rest SVMs are those described in Section 4.1. Normally, the DAG SVM and the one-vs-rest SVMs would be trained using the training set only. But since our fuzzy labeling method uses additional information provided by the generating set through the construction of the confidence curves and computation of prototype signatures, we

thought that this might give the fuzzy labeling method an unfair advantage over the crisp labeling methods trained using only the training set. Consequently, we also considered training the DAG SVM and the one-vs-rest SVMs using a combination of the training and the generating sets.

Region classification, carried out only for evaluation purposes, was performed by computing the distance $d(r, c)$ between an image block r and each of the classes $c = 1, \ldots, 31$ using Eq. 3.10. Then, the image block was assigned the class for which the distance was the smallest over all 31 classes.

4.4.2 Overall experimental results

Tables 4.7 and 4.8 show the experimental results on the set of well-cropped image blocks and on the set of general test image blocks. Given in these tables are classification accuracy (ClsAcc) and labeling effectiveness (LabEff) which were computed to measure performance. These are defined as:

$$\text{ClsAcc} = N_{\text{correct}} / T_{\text{known}}$$
$$\text{LabEff} = N_{\text{correct}} / T_{\text{class}}$$

where

T_{known} = total number of image blocks manually labeled with one of the 31 classes
T_{class} = total number of image blocks classified
N_{correct} = number of correctly classified image blocks.

First of all, Table 4.7 shows that all the methods with the exception of crisp labeling with one-vs-rest SVMs and fuzzy labeling using prototype signatures from adaptive clustering can assign a known label to all test image blocks. Crisp labeling with one-vs-rest SVMs can label only around 67% to 70% of the well-cropped image blocks. Not all image blocks are assigned known labels using the one-vs-rest SVM approach because some image blocks may not be classified positively by any of the m one-vs-rest SVMs. Hence the image block receives a zero vector as a fuzzy label.

Table 4.7 *Experimental results on well-cropped image blocks.*

	Crisp Labeling				**Fuzzy Labeling**				
	DAG		**One vs Rest**			**k-means clustering**		**Adaptive clustering**	
	Training only	**Training + Generating**	**Training only**	**Training + Generating**	**Single signature**	**All signatures**	**Reliable signatures**	**All signatures**	**Reliable signatures**
ClsAcc	58.3%	61.7%	51.2%	55.2%	59.6%	52.5%	60.8%	41.7%	41.9%
LabEff	100.0%	100.0%	67.6%	70.9%	100.0%	100.0%	100.0%	68.6%	68.5%

Table 4.8 *Experimental results on general test image blocks.*

	Crisp Labeling				**Fuzzy Labeling**				
	DAG		**One vs Rest**			**k-means clustering**		**Adaptive clustering**	
	Training only	**Training + Generating**	**Training only**	**Training + Generating**	**Single signature**	**All signatures**	**Reliable signatures**	**All signatures**	**Reliable signatures**
ClsAcc	10.4%	9.7%	10.5%	11.1%	21.6%	19.8%	24.7%	17.8%	17.3%
LabEff	100.0%	100.0%	19.1%	21.6%	100.0%	100.0%	100.0%	84.2%	83.6%

Similarly, fuzzy labeling using prototype signatures obtained by adaptive clustering manages to label only around 68% of the well-cropped image blocks. In this case, not all image blocks are assigned known labels because image blocks whose signatures lie outside the set cluster radius of 0.275 are not assigned labels. In other words, such image blocks are said to belong to some "unknown" semantic class.

Table 4.7 further shows that not all fuzzy labeling methods performed better than crisp labeling on the well-cropped test samples. The highest classification accuracy of 61.7% was actually achieved with DAG SVM trained with a combination of the training and the generating sets. But this was followed very closely by fuzzy labeling using reliable signatures obtained through k-means clustering (60.8%), as well as by fuzzy labeling using single prototype signatures (59.6%). Furthermore, fuzzy labeling using adaptive clustering performed even worse than crisp labeling in terms of classification accuracy.

On the other hand, different results were obtained when region matching was performed on the image blocks in the general test set (Table 4.8). Both crisp labeling using one-vs-rest SVMs and fuzzy labeling using adaptive clustering were unable to label some image blocks in the general test set. However, labeling effectiveness for crisp labeling using one-vs-rest SVMs was much worse, being only 21.6% when trained with the training and the generating sets combined, and 19.1% when trained with the training set only. For fuzzy labeling using adaptive clustering, there was a marked improvement when labeling image blocks in the general test set with labeling effectiveness increasing to around 84%.

More importantly, we can also observe that fuzzy labeling clearly outperforms crisp labeling in terms of classification accuracy when labeling image blocks in the general test set. While both crisp labeling methods have a classification accuracy of

just around 10%, fuzzy semantic labeling actually almost doubles this figure in most cases. With a classification accuracy of 24.7%, using only reliable signatures obtained through k-means clustering more than doubles the classification accuracy of crisp labeling. This translates to correctly labeling 19 out of the 77 image blocks that make up an entire image. Correctly labeling 19 image blocks in an image should be sufficient to perform image retrieval.

Another observation that can be made is that among crisp labeling methods, DAG SVM generally performs labeling better than one-vs-rest. This is actually consistent with the results in past comparative studies [PCS00, HL02, Wid02]. More notably, DAG SVM can label *all* image blocks while one-vs-rest can label only *some* of the image blocks.

Multiple prototype signatures obtained with k-means clustering, but not with adaptive clustering, also yielded better results than single prototype signatures. This may be taken as a confirmation of the large variation occurring within a single semantic class that is not captured by single prototype signatures.

In Section 3.2.3, we stated that some prototype signatures of a class *c* were believed to be misleading because they indicated a higher confidence for some class *k* other than the class they were supposed to represent. Our choice of using only the reliable prototype signatures is justified by the results since generally better classification accuracy was obtained using only the selected (reliable) prototype signatures. This holds true whether or not the test was performed on the set of well-cropped image blocks or on the general test set.

In summary, we can make the following observations:

- Fuzzy labeling generally performed better than crisp labeling.

- Among the crisp labeling methods considered, DAG SVM generally performed better than one-vs-rest SVMs.
- Fuzzy labeling methods using multiple prototype signatures per class was better than using a single prototype signature for each class. This however is only true for prototype signatures obtained using k-means clustering.
- Prototype signatures obtained through k-means clustering provided better overall labeling performance compared to prototype signatures obtained through adaptive clustering.
- Reliable prototype signatures generally produced higher classification accuracy.
- Fuzzy semantic labeling using reliable prototype signatures from k-means clustering most consistently performed well for both the set of well-cropped image blocks and the image blocks in the general test set.

4.4.3 Experimental Results on Individual Classes

Since fuzzy semantic labeling using reliable prototype signatures from k-means clustering consistently performed well among all semantic labeling methods compared in this study, we now focus on the performance of this labeling method on the individual semantic classes. The confusion matrices resulting from performing region classification on the image blocks in both the set of well-cropped image blocks and the general test set are shown in Tables 4.9 and 4.10. Both tables also show the individual accuracy and precision achieved for each semantic class.

Accuracy was relatively high for well-cropped image blocks in the test set, ranging from a high of 94% to a low of 34% (Table 4.9). The highest accuracies were achieved for flames and night sky (94%), followed closely by clear day sky (90%), human faces (90%) and pebbles (88%). The lowest accuracy was achieved for big buildings.

On the other hand, precision achieved for the test set of well-cropped samples had a high of 98% (night sky) and a low of 24% (rocks). Understandably, image blocks belonging to the semantic class of night sky were the most homogenous among the 31 classes considered in this study. Other semantic classes with high precision were roof (85%), grass (83%), choppy water (82%) and flame (81%).

It is interesting to note that many image blocks of calm water were mislabeled as clouds possibly because some image blocks of calm water included reflective water surfaces that actually reflected the sky above. The mislabeling of big buildings as domes is also easy to explain since most images of single buildings shown at a distance resemble steeples and domes.

As revealed by the relative overall performance obtained on the set of well-cropped image blocks and on the general test set, accuracy drops significantly when labeling is performed on general test set (Table 4.10). While the highest accuracy of 85.8% achieved with clear day sky is comparable to the highest accuracy achieved with the test set of well-cropped image blocks, the worst accuracy achieved is extremely low at 0.5% for roofs, followed closely by 0.7% for human faces. In fact, out of the more than 26,000 image blocks manually labeled with one of the 31 semantic classes, only 15 were classified as human faces and 28 were classified as roofs. This result for human faces in particular on the general test set is in complete contrast to that achieved with image blocks of human faces in the set of well-cropped image blocks. This further confirms that labeling is much more difficult when human faces are not well-centered in image blocks in the general test set. This observation may well be generalized to the other classes where more confusion occurs when an image block contains objects from multiple semantic classes or when the object of interest is not centered in the image block.

As in the case of accuracies achieved on the general test set, precisions for individual classes for the general test image blocks are lower than those for the well-cropped image blocks. The highest precisions were achieved with flowers (75%), foliage (71%), night sky (69%) and grass (67%) although foliage was often mislabeled as grass. Each of the classes for rocks, domes and houses achieved the lowest precision of only 6%. A very large number of image blocks were in fact mislabeled as rocks: 6265 of which only 384 were actually rocks. Also among those with low precision were the classes for roofs (7%), paved roads (8%) and fences (9%). Many of the image blocks labeled as paved roads and fences actually belonged to the class of big buildings. This is so possibly because single buildings taken at an upward angle resembled images of paved road shown in perspective and images of big buildings, particularly those of cityscapes, resembled fences. Similarly, most image blocks of roofs were mislabeled as brick walls, rocks and mountains possibly because images of roofs shown up close resembled brick walls and images of roof gables resembled rock outcroppings and mountain peaks.

Similar to the overall classification results (Section 4.4.2), classification results of individual classes for well-cropped samples are better than those for general test samples. Nevertheless, fuzzy labeling still performs better than crisp labeling particularly for image blocks in the general test set. This affirms the strength of fuzzy labeling over crisp labeling. Furthermore, since image blocks in the general test set resemble those typically encountered in image retrieval, it is expected that fuzzy semantic labeling should perform better than crisp labeling when applied to image retrieval in real-world situations.

Table 4.9 *Confusion matrix for region classification performed on well-cropped image blocks. Fuzzy labeling method using only reliable prototype signatures obtained using k-mean clustering.*

		Assigned Label																																
	Actual Label	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	**Total**	**Recall**
1	grass	**33**	4	1	0	0	1	0	0	1	0	2	0	0	0	0	0	0	0	1	0	0	1	0	1	0	0	0	3	2	0	0	50	66.0%
2	foliage	5	**33**	0	0	0	3	0	1	0	0	0	0	0	0	0	0	1	1	2	0	0	0	0	0	0	0	0	1	2	0	1	50	66.0%
3	flowers	1	3	**32**	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	3	3	0	2	3	0	1	0	0	50	64.0%
4	clouds	0	0	0	**23**	6	1	0	1	8	1	1	0	0	3	0	3	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	50	46.0%
5	clear sky	0	0	0	1	**46**	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	92.0%
6	rocks	0	0	0	0	0	**16**	1	0	0	0	2	1	4	0	0	1	1	2	6	0	1	1	1	0	0	2	0	4	2	1	4	50	32.0%
7	mountain	0	0	0	0	3	1	**29**	0	1	2	0	0	1	1	1	2	1	0	2	0	1	1	0	0	0	1	2	0	0	0	1	50	58.0%
8	sand	0	0	0	0	0	6	0	**27**	0	0	4	0	0	1	1	0	0	1	0	1	0	0	0	0	0	1	0	7	0	0	1	50	54.0%
9	calm water	0	1	0	1	1	0	1	1	**31**	1	0	0	0	4	0	3	1	0	0	2	0	2	0	0	0	0	0	1	0	0	0	50	62.0%
10	choppy water	0	0	0	4	0	0	5	0	4	**27**	0	1	0	5	0	2	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	50	54.0%
11	fur	1	0	0	1	0	3	1	0	0	0	**23**	1	0	2	2	2	0	1	4	0	2	0	0	3	0	0	0	2	0	1	1	50	46.0%
12	human face	0	0	0	0	0	0	0	0	0	0	0	**45**	0	0	0	0	0	0	0	0	1	0	2	0	0	0	2	0	0	0	0	50	90.0%
13	pebbles	0	0	0	0	0	0	0	0	0	0	0	0	**44**	0	0	0	1	0	1	0	1	1	0	0	0	0	0	1	0	0	1	50	88.0%
14	snow	0	0	0	5	1	2	3	2	3	0	0	0	0	**34**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	68.0%
15	roof	0	0	0	0	0	1	0	0	0	0	0	1	0	0	**29**	3	1	2	1	0	0	1	1	0	0	1	0	1	0	0	8	50	58.0%
16	paved road	0	0	1	0	0	1	0	6	2	0	0	2	0	0	1	**23**	0	0	0	0	1	1	0	0	1	1	1	4	0	0	5	50	46.0%
17	dome	0	0	0	2	1	0	2	0	1	0	0	0	0	0	0	0	**33**	1	1	0	2	0	0	0	0	3	2	0	1	0	1	50	66.0%
18	brick wall	0	0	0	0	0	2	0	2	1	1	2	0	3	0	0	0	0	**27**	0	3	1	0	0	0	0	0	0	6	0	0	2	50	54.0%
19	tree trunk	0	0	1	0	0	5	0	0	1	0	0	0	3	0	0	1	0	2	**20**	1	2	2	0	4	0	0	3	2	0	1	2	50	40.0%
20	wood	0	0	1	1	0	4	0	1	1	0	4	0	0	2	0	4	0	0	3	**22**	2	1	0	0	0	0	0	1	2	0	1	50	44.0%
21	window	0	0	1	0	0	1	0	0	0	0	1	0	2	0	0	1	2	0	2	0	**30**	3	0	0	0	0	0	1	0	6	0	50	60.0%
22	fences	0	0	1	0	0	4	0	1	0	0	0	0	0	0	0	3	0	1	2	0	3	**25**	0	0	0	0	1	1	3	2	3	50	50.0%
23	flames	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	**47**	0	0	0	0	0	1	0	0	50	94.0%
24	fireworks	0	0	3	0	0	3	0	0	0	0	2	1	0	0	0	0	0	0	1	0	1	0	2	**37**	0	0	0	0	0	0	0	50	74.0%
25	night sky	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**47**	0	0	0	0	0	0	50	94.0%
26	big building	0	1	0	0	1	1	1	0	0	0	1	3	0	0	0	1	8	0	2	0	3	1	0	0	0	**17**	8	0	1	1	0	50	34.0%
27	house	0	0	1	0	0	0	2	0	0	0	0	0	2	0	0	1	0	0	1	0	3	1	0	0	0	4	**33**	0	0	0	2	50	66.0%
28	soil	0	0	0	0	0	5	0	7	0	1	1	0	2	0	0	2	0	1	2	2	0	0	0	0	0	0	1	**24**	0	0	2	50	48.0%
29	scales	0	1	1	0	0	1	2	0	0	0	0	0	2	1	0	1	0	0	2	0	1	5	1	3	0	0	1	0	**26**	2	0	50	52.0%
30	pillars	0	0	0	1	0	0	0	0	0	0	2	3	0	0	0	0	0	0	0	3	5	0	0	0	0	2	1	0	1	**32**	0	50	64.0%
31	staircase	0	0	0	0	0	3	0	1	1	0	2	0	1	0	0	3	0	0	1	1	2	3	0	0	0	2	0	3	0	0	**27**	50	54.0%
	Total	40	43	43	39	60	66	47	51	57	33	47	58	65	54	34	56	51	39	55	36	62	50	58	51	48	36	59	62	42	46	62		
	Precision	83%	77%	74%	59%	77%	24%	62%	53%	54%	82%	49%	78%	68%	63%	85%	41%	65%	69%	36%	61%	48%	50%	81%	73%	98%	47%	56%	39%	62%	70%	44%		

Table 4.10 *Confusion matrix for region classification performed on the general test image blocks. Fuzzy labeling method using only reliable prototype signatures obtained using k-mean clustering.*

		Assigned Label																															
	Actual Label	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	**Total**	**Recall**
1	grass	**367**	34	0	6	87	254	0	22	245	0	4	0	1	0	0	12	0	3	53	10	0	8	0	0	0	3	1	24	32	0	6	1172	31.3%
2	foliage	137	**471**	11	21	29	969	10	8	248	2	6	1	6	4	0	31	70	7	283	0	42	66	2	36	32	113	9	30	55	39	39	2777	17.0%
3	flowers	2	25	**42**	3	19	480	0	1	42	2	1	0	2	3	0	6	37	1	38	0	5	11	69	4	0	35	1	7	18	2	0	856	4.9%
4	clouds	0	1	0	**512**	1117	82	67	20	214	8	4	0	0	300	1	63	36	0	7	10	1	3	25	0	27	1	0	2	3	0	8	2512	20.4%
5	clear sky	0	0	0	28	**1540**	18	9	4	30	0	0	0	0	75	0	4	3	0	1	4	0	0	0	0	75	0	0	2	1	0	0	1794	85.8%
6	rocks	1	12	0	19	34	**384**	3	12	76	0	5	1	12	17	4	25	3	62	35	2	39	31	0	6	4	50	2	66	4	32	32	973	39.5%
7	mountain	0	4	0	97	59	137	**84**	1	77	8	1	0	0	55	4	51	22	0	5	0	9	30	0	2	10	23	4	2	2	1	4	692	12.1%
8	sand	4	0	0	14	102	77	3	**101**	91	0	7	0	0	8	2	28	4	19	15	19	3	9	0	0	1	5	0	60	3	0	4	579	17.4%
9	calm water	3	3	2	59	173	73	13	4	**307**	2	1	0	2	70	3	38	5	3	7	3	12	12	3	9	53	1	0	9	11	7	13	901	34.1%
10	choppy water	0	0	0	103	45	36	9	1	90	**25**	0	0	0	109	0	31	2	0	3	1	4	0	1	0	0	6	0	4	1	0	4	475	5.3%
11	fur	0	0	0	8	19	99	0	8	48	0	**22**	0	4	1	0	12	1	7	29	6	7	1	0	2	0	1	0	9	0	1	2	287	7.7%
12	human face	0	1	0	2	1	66	0	0	6	0	1	**3**	2	1	0	3	4	0	7	0	2	7	10	1	0	8	0	0	0	0	1	126	2.4%
13	pebbles	0	2	0	7	0	255	0	18	106	1	0	0	**163**	1	0	51	0	1	27	0	72	70	0	3	0	52	0	4	31	14	26	904	18.0%
14	snow	0	0	0	129	332	44	26	5	84	6	1	0	0	**283**	0	26	17	0	5	0	0	3	2	0	0	18	0	1	1	0	1	984	28.8%
15	roof	0	0	0	8	10	133	0	5	107	0	0	0	3	2	**2**	17	3	5	14	0	11	33	1	0	3	8	0	7	3	7	24	406	0.5%
16	paved road	1	5	0	42	21	86	4	21	146	3	0	0	2	33	0	**58**	5	9	7	2	2	22	0	0	3	19	0	9	0	1	24	525	11.0%
17	dome	0	2	0	24	60	66	6	1	30	1	4	0	0	19	0	28	**22**	0	7	0	3	10	0	3	7	10	0	0	0	0	4	307	7.2%
18	brick wall	0	2	0	0	11	205	0	7	47	0	2	0	20	0	5	5	1	**30**	21	0	3	4	0	0	0	37	0	63	2	3	17	485	6.2%
19	tree trunk	5	29	1	9	15	421	0	3	79	1	0	2	8	14	0	21	18	2	**215**	4	98	29	2	18	3	97	2	4	15	178	18	1311	16.4%
20	wood	0	0	0	13	37	166	2	16	122	0	12	0	3	13	0	44	7	9	46	**70**	15	25	5	0	0	6	0	34	0	7	20	672	10.4%
21	window	0	5	0	15	5	259	2	1	34	0	0	0	5	21	0	19	2	4	17	1	**101**	43	2	0	0	26	0	4	7	56	6	635	15.9%
22	fences	22	18	0	17	6	192	2	0	30	0	1	1	1	17	0	18	20	8	20	0	61	**61**	1	11	0	31	1	5	2	23	9	578	10.6%
23	flames	0	0	0	3	69	84	1	0	40	0	0	0	0	1	0	4	13	0	7	4	2	1	**174**	20	0	6	0	0	5	3	8	445	39.1%
24	fireworks	1	3	0	1	6	317	0	0	25	0	3	0	2	0	3	8	7	0	16	0	9	5	11	**168**	0	9	0	3	22	16	17	652	25.8%
25	night sky	0	0	0	0	176	9	5	0	19	0	0	0	0	0	0	3	0	0	0	0	0	0	0	8	**576**	0	0	0	1	0	13	810	71.1%
26	big building	1	27	0	56	28	532	9	2	184	3	1	4	12	39	1	65	47	0	52	0	109	57	12	5	40	**138**	11	7	37	64	73	1616	8.5%
27	house	0	0	0	6	4	120	0	1	17	1	1	0	2	0	0	17	10	0	5	0	12	35	2	1	0	43	**2**	1	5	1	6	292	0.7%
28	soil	3	0	0	8	24	226	0	22	107	0	4	1	0	0	1	11	3	39	23	6	1	2	0	0	0	7	0	**198**	11	0	7	704	28.1%
29	scales	2	3	0	4	6	148	0	0	45	0	1	1	4	2	2	3	2	0	11	0	8	8	12	2	0	22	0	1	**65**	5	0	357	18.2%
30	pillars	0	1	0	26	43	188	0	0	49	0	6	1	1	9	0	29	6	0	61	7	77	21	9	5	0	7	0	2	1	**132**	8	689	19.2%
31	staircase	1	20	0	30	6	139	0	1	104	0	7	0	2	30	0	25	1	2	37	0	36	42	0	0	3	21	1	9	0	2	**144**	663	21.7%
	Total	550	668	56	1270	4084	6265	255	285	2849	63	95	15	257	1127	28	756	371	211	1074	149	744	649	343	304	837	803	34	567	338	594	538		
	Precision	67%	71%	75%	40%	38%	6%	33%	35%	11%	40%	23%	20%	63%	25%	7%	8%	6%	14%	20%	47%	14%	9%	51%	55%	69%	17%	6%	35%	19%	22%	27%		

CHAPTER 5

Conclusion

In this study, we have developed a fuzzy semantic labeling method for image blocks that assigns multiple semantic labels and associated confidence measures to each image block. In order to obtain these confidence measures, we first trained a one-vs-rest binary Support Vector Machine (SVM) for each semantic class and approximated a confidence curve using the orthogonal distances of an extra set of image samples to the constructed hyperplanes. Given the distance of a test sample to a hyperplane, we obtained the expected confidence of its classification into a semantic class from the derived confidence curve using linear interpolation.

Using an image region's fuzzy labels, region matching was then carried out by classifying the image region into the class with the prototype signature nearest to the signature of the image region. Here, the Euclidean distance was used as a dissimilarity measure. This method for region matching, in contrast to that in [LL02] as pointed out in Chapter 2, takes into consideration each and every confidence measure in an image region's fuzzy label.

We performed the proposed fuzzy semantic labeling on 31 semantic classes. This number of classes was as large as that used in [LL02] and much larger than those used in existing crisp labeling methods. Furthermore, our evaluation tests included

comparisons of the classification performance of the proposed fuzzy semantic labeling method with that of crisp labeling methods based on two multi-class SVM classifiers: one-vs-rest SVMs and Directed Acyclic Graph (DAG) SVMs. Tests were performed on both a set of well-cropped image blocks and on a general test set consisting of fixed-size partitions of whole images.

Test results show that the proposed fuzzy semantic labeling method generally performs better than the two crisp labeling methods. The disparity in performance is even more prominent when classification is performed on the general test set despite an overall drop in performance. This, however, is expected since noise and ambiguity in the general test image blocks are more pronounced than those in the well-cropped image blocks.

Different methods for obtaining these prototype signatures for each semantic class were also explored. One method simply required computing the average of the signatures of image regions in a semantic class and using this single "average" signature as a prototype signature for that semantic class. The other method for obtaining prototype signatures involved performing clustering algorithms on image samples in a semantic class and taking the centroids of the resulting clusters as the prototype signatures of that semantic class. K-means clustering and the adaptive clustering algorithms were used. Evaluation tests show that multiple prototype signatures achieved higher classification accuracy than single prototype signatures. In particular, prototype signatures obtained through k-means clustering yielded better test results than the prototype signatures obtained through adaptive clustering.

We also recognized that some of the prototype signatures obtained were actually unreliable and may not be used for region matching. Test results indeed confirm that discarding these unreliable prototype signatures does improve labeling accuracy. On

the whole, fuzzy semantic labeling using reliable prototype signatures obtained through k-means clustering produced the best overall performance.

Given the outcome of the evaluation tests, we can conclude that our proposed fuzzy semantic labeling method performs better than crisp labeling methods on a relatively large number of semantic classes. We expect that this advantage over crisp labeling methods will carry over to semantic based image retrieval.

CHAPTER 6

Future Work

Evaluation tests performed in this study focused on how the proposed fuzzy semantic labeling method performs when used in image region classification. Since image region classification is merely a step in image retrieval, it is important that additional tests be conducted to investigate how the proposed fuzzy semantic labeling method performs when applied to image retrieval.

Instead of using fixed-size image blocks of an image, the proposed fuzzy semantic labeling method may be applied in combination with other image segmentation methods such as those employed by Campbell et al. [CMT+97] and Belongie et al. [BCG+97].

Features used as a basis for carrying out fuzzy labeling in this study were fixed color histograms, Gabor features, multiresolution simultaneous autoregressive features (MRSAR) and edge histograms. Perhaps other features introduced in other studies may also be used to contribute other image information needed to improve labeling performance. One of these is a structure feature proposed by Zhou et al. [ZRH99] that the authors describe as more general than texture or shape because it is a combination of texture and shape. Consequently, it can capture some information that may not be captured by texture or shape features alone and is effective on non-

uniform natural images. In addition to this, a color-spatial feature such as the color coherence histogram [PZM96] or the color correlogram [HKM+97] may contribute spatial information not provided by any of the four features considered in this study.

The fuzzy semantic labeling method in this study involved approximating a confidence curve from which confidence measures of image blocks are obtained. Other methods that map the output of a Support Vector Machine (SVM) to class classification probabilities may be used in place of the recursive algorithm described in section 3.2.2. Such methods include that by Platt [Pla99] where SVM outputs are mapped into probabilities by training an SVM and then fitting the SVM classification results to a two-parameter sigmoid model. The two parameters in the sigmoid model are estimated via the maximum likelihood estimation method using a new set of training samples.

Another similar method that may also be used is that by Goh, et al. [GCC01] as applied on the output of ensembles of SVM binary classifiers to enhance their accuracy. Their method, a proposed improvement on Platt's sigmoid fitting method, uses a fixed sigmoid function to boost the output of accurate classifiers with a weak influence on making a class prediction. They also apply an error reduction procedure to reduce the effect of noise from inaccurate classifiers in an ensemble.

Bibliography

[BDF01] K. Barnard, P. Duygulu, and D. Forsyth. Clustering art. *Computer Vision and Pattern Recognition*, 2:435-439, 2001.

[BF01] K. Barnard and D. Forsyth. Learning the semantics of words and pictures. *International Conference on Computer Vision*, 2: 408-415, 2001.

[BCG+97] S. Belongie, C. Carson, H. Greenspan, and J. Malik. Recognition of images in large databases using a learning framework. Technical Report 97-939, Computer Science Division, University of California at Berkeley, 1997.

[BJ03] D. Blei and M. Jordan. Modeling annotated data. *In Proc. ACM SIGIR Conf. on Research and Devt. in Information Retrieval*, 127-134, 2003.

[Bra99] S. Brandt. *Use of shape features in content-based image retrieval*. Unpublished masters thesis, Helsinki University of Technology, Finland, 1997.

[BLO00] S. Brandt, J. Laaksonen and E. Oja. Statistical shape features in content-based image retrieval. In *Proc. Intl Conf on Pattern Recognition*, 2: 1066-1069, 2000.

[CMT+97] N. W. Campbell, W. P. J. Mackeown, B. T. Thomas, and T. Troscianko. (1997) Interpreting image databases by region classification. *Pattern Recognition*, 30(4): 555-563, 1997.

[CBG+97] C. Carson, S. Belongie, H. Greenspan and J. Malik. Region-based image querying. In *Proc. CVPR Workshop on Content-Based Access of Image and Video Libraries*, 1997.

[CHV99] O. Chapelle, P. Haffner and V. Vapnik. Support vector machines for histogram-based image classification. *IEEE Trans. on Neural Networks*, 10:1055-1064, 1999.

[CCS+03] G. Ciocca, C. Cusano, R. Schettini, and C. Brambilla. Semantic labeling of digital photos by classification. In *Proc. Internet Imaging Conference*, 5018, 2003.

[CS99] G. Ciocca and R. Schettini. A relevance feedback mechanism for content-based image retrieval. *Infor. Proc. and Management*, 35: 605-632, 1999.

[CV95] C. Cortes and V. N. Vapnik. Support vector networks. *Machine Learning*, 20: 273-297, 1995.

[DB91] T. G. Dietterichand G. Bakiri. Error-correcting output codes: A general method for improving multiclass inductive learning programs. In *Proc. of the 9th AAAI National Conference on Artificial Intelligence*, 572-577, 1991.

[DBF+02] P. Duygulu, K. Barnard, N. de Freitas, and D. Forsyth. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. *In 7th European Conf. on Computer Vision*, 97-112, 2002.

[FL99] C. Y. Fung and K. F. Loe. Learning primitive and scene semantics of images for classification and retrieval. In *Proc. ACM Multimedia*, II: 9-12, 1999.

[GCC01] K. Goh, E. Chang and K. Cheng. SVM binary classifier ensembles for image classification. *ACM International Conference on Information and Knowledge Management*, 395-402, 2001.

[GJ97] A. Gupta and R. Jain. Visual information retrieval. *Comm. of the ACM*, 40(5), 1997.

[HSE+95] J. Hafner, H. S. Sawhney, W. Esquitz, M. Flickner, and W. Niblack. Efficient color histogram indexing for quadratic form distance functions. *IEEE Trans. PAMI*, 17: 729-736, 1995.

[Hay99] S. Haykin. *Neural Networks: Comprehensive Foundation* (2nd ed.). Upper Saddle River : Prentice Hall , 1999.

[HL02] C. W. Hsu and C. J. Lin. A comparison of methods for multi-class support vector machines. *IEEE Trans. on Neural Networks*, 13: 415-425, 2002.

[HKM+97] J. Huang, S. Kuma, M. Mitra, W-J. Zhu and R. Zabih. Image indexing using color correlogram. In *Proc. of Computer Vision and Pattern Recognition*, 762-768, 1997.

[JLM03] J. Jeon, V. Lavrenko and R. Manmatha. Automatic image annotation and retrieval using cross-media relevance models. *Proc. of the 26th Annual Intl. ACM SIGIR Conf. on Research and Devt. in Information Retrieval*, 119-126, 2003.

[KR90] L. Kaufman, and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. New York: Wiley, 1990.

[KPD90] S. Knerr, L. Personnaz. and G. Dreyfus. Single-layer learning revisited: A stepwise procedure for building and training a neural network. In F.

Fogelman-Soulie and J. Herault (Eds.), *Neurocomputing: Algorithms, Architectures and Applications* (pp. 41-50). New York: Springer-Verlag, 1990.

[LMJ03] V. Lavrenko, R. Manmatha and J. Jeon. A model for learning the semantics of pictures. *In Proc. Neural Information Processing Systems,* 2003.

[LL01] W. K. Leow and R. Li. Adaptive binning and dissimilarity measure for image retrieval and classification. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* II: 234-239, 2001.

[LL03] R. Li and W. K. Leow. From region features to semantic labels: A probabilistic approach. In *Proc. Int. Conf. on Multimedia Modeling,* 402-420, 2003.

[LL02] F. S. Lim and W. K. Leow. Adaptive histograms and dissimilarity measure for texture retrieval and classification. In *Proc. Int. Conf. on Image Processing,* II: 825-828, 2002.

[LP96] F. Liu and R. W. Picard. Periodicity, directionality and randomness: Wold features for image modeling and retrieval. *IEEE Trans. PAMI,* 18(7):722-733, 1996.

[MM97] W. Y. Ma and B. S. Manjunath. NeTra: A toolbox for navigating large image databases. In *Proc. IEEE Int. Conf. On Image Processing,* 568-571, 1997.

[MM96] B. Manjunath and W. Y. Ma. Texture features for browsing and retrieval of image data. *IEEE Trans. PAMI,* 8(18): 837-842, 1996.

[MJ92] J. C. Mao and A. K. Jain. Texture classification and segmentation using multiresolution simultaneous autoregressive models. *Pattern Recognition,* 25: 173-188, 1992.

[MKD+95] B. M. Mehtre, M. S. Kankanhalli, A. Desai, and G. C. Man. Color matching for image retrieval. *Pattern Recognition Letters,* 16: 325-331, 1995.

[MC85] G. W. Milligan and M. C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika,* 50, 159–179, 1985.

[MTO99] Y. Mori, H. Takahashi, and R. Oka. Image-to-word transformation based on dividing and vector quantizing images with words. *In MISRM'99 1st Intl. Workshop on Multimedia Intelligent Storage and Retrieval Mgt,* 1999.

[MTO00] Y. Mori, H. Takahashi, and R. Oka. Automatic words assignment to images based on image division and vector quantization. *In Proc. of RIAO,* 2000.

[MLL01] P. Mulhem, W. K. Leow, and Y. K. Lee. Fuzzy conceptual graph for matching images of natural scenes. In *Proc. Int. Joint Conf. On Artificial Intelligence*, 1397-1402, 2001.

[NBE+93] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, et al. The QBIC project: Querying images by content using color, texture and shape. In *Proc. SPIE Conf. On Storage and Retrieval for Image and Video Database*, 1908: 173-181, 1993.

[PZM96] G. Pass, R. Zabih and J. Miller. Comparing images using color coherence vectors. In *Proc. of the 4th ACM Multimedia Intl. Conf*., 65-73, 1996.

[PPS96] A. Pentland, R. W. Picard, and W. Sclaroff. Photobook: Tools for content-based manipulation of image databases. *Int. Journal Computer Vision*, 18(3): 233_254, 1996.

[PKL93] R. W. Picard, T. Kabir and F. Liu. Real-time recognition with the entire Brodatz texture database. In *Proc. IEEE CVPR*, 1993.

[Pla99] J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*. A. Smola, P. Bartlett, B. Scholkopf, D. Schuurmans, eds., MIT Press, 1999.

[PCT00] J. Platt, N. Cristianini and J. Shawe-Taylor. Large margin DAGs for multiclass classification. *Advances in Neural Information Processing Systems*, 12: 547-553, 2000.

[Rou87] P. J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Comp. And Applied Mathematics*, 20: 53-65, 1987.

[Sal01] J. Salomon. *Support Vector Machines for Phenome Classification*. Unpublished masters thesis. University of Edinburgh, 2001.

[STC97] W. Sclaroff, L. Taycher, and M. La Cascia. Image-Rover: A content-based image browser for the world wide web. In *Proc. IEEE Workshop on Content-Based Access of Image and Video Libraries*, 1997.

[SWS+00] A. W. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Trans. On PAMI*, 22(12): 1349-1379, 2000.

[SBI+01] J. R. Smith, S. Basu, G. Iyengar, C. Y. Lin, M. Naphade, B. Tseng., et al. Integrating features, models and semantic for trec video retrieval. In *Proc. of the Tentht TREC*, 2001.

[SC95] J. R. Smith and S. F. Chang. Single color extraction and image query. In *Proc. IEEE Int. Conf. On Image Processing*, 1995.

[SB91] M. Swain and D. Ballard. Indexing via Color Histograms. *International Journal of Computer Vision*, 7:11-32, 1991.

[SP98] M. Szummer and R. W. Picard. Indoor-outdoor image classification. In *Proc. ICCV Workshop on Content-based Access of Image and Video databases*, 42-51, 1998.

[TS00] C. Town and D. Sinclair. Content-based image retrieval using semantic visual categories. Technical Report 2000.14, AT&T Laboratories Cambridge, 2000.

[VJZ98] A. Vailaya, A. Jain, and H. J. Zhang. On image classification: city images vs. landscapes. *Pattern Recognition*, 31: 1921-1935, 1998.

[Vap95] V. N. Vapnik. *The Nature of Statistical Learning Theory*. New York: Springer, 1995.

[Vap98] V. N. Vapnik. *Statistical Learning Theory*. New York: Wiley, 1998.

[WLW01] J. Z. Wang, J. Li, and G. Wiederhold. SIMPLIcity: Semantics sensitive integrated matching for picture libraries. *IEEE Trans. on PAMI*, 23(9): 947-963, 2001.

[Wid02] I. Widjaja. *Identifying painters from color profiles of skin patches in painting images.* Unpublished masters thesis. National University of Singapore, Singapore, 2002.

[WCL02] G. Wu, E. Chang, and C. S. Li. BPMs vs. SVMs for Image Classification. In *Proc. IEEE International Conference on Multimedia*, 2002.

[ZRH99] X. S. Zhou, Y. Rui. and T. S. Huang. Water-filling: A Novel Way for Image Structural Feature Extraction. In *Proc. IEEE Int. Conf. On Image Processing*, 570-574, 1999.