IMAGE CLASSIFICATION WITH BAGS OF LOCAL FEATURES

A Dissertation Presented

by

DIMITRI A. LISIN

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2006

Department of Computer Science

IMAGE CLASSIFICATION WITH BAGS OF LOCAL FEATURES

A Dissertation Presented

by

DIMITRI A. LISIN

Approved as to style and content by:

Erik G. Learned-Miller, Chair

Allen Hanson, Member

Paul E. Utgoff, Member

Walter A. Carrington, Member

W. Bruce Croft, Department Chair
Department of Computer Science

ACKNOWLEDGMENTS

First of all I thank my parents for bringing me into this world and for urging me to make something of myself. My dad taught me discipline and rationality, and my mom inspired me to be creative. Together with my grandparents they instilled in me the drive to succeed that has helped me to get through seven years of grad school.

I thank my lab-mates for their help and support. Without them this dissertation, which by the way is part of a project supported by the National Science Foundation under grant number ATM-0325167, would have been impossible. I particularly thank Frank Stolle, Marwan Mattar, Pla Silapachote, Vidit Jain, and Matthew Blaschko. Special thanks to Frank, Pla, and Vidit for helping me prepare for my defense, and to Marwan for helping me collect the signatures. I also thank Stella X. Yu, David Martin, and Gang Tan of Boston College for listening to my practice talk.

I thank the faculty of the Computer Science department at UMASS Amherst for teaching and guiding me, particularly Ed Riseman, Allen Hanson, and Howard Schultz. Special thanks to Ed Riseman for nominating me for a fellowship from Eastman-Kodak Corporation Research labs and convincing me to join the plankton classification project. I also thank Eastman-Kodak Corporation Research Labs for supporting me for four years through a graduate fellowship.

I thank my committee members for taking the time to read my dissertation multiple times, and for their insightful comments. Special thanks to Erik Learned-Miller, the chairman of my committee. This dissertation would not have been finished, without Erik's expertise, energy, and candor.

I also thank the staff members, particularly Sharon Mallory and Laurie Downey for helping me deal with the administrative issues. I would also like to thank the technical

staff, particularly Robert Heller for keeping the computers in the lab running, and Victor Danilchenko for being my gym buddy.

Last, but by no means least, I thank my wife Tania for being there for me and for renewing my faith in the bright future.

ABSTRACT

IMAGE CLASSIFICATION WITH BAGS OF LOCAL FEATURES

MAY 2006

DIMITRI A. LISIN

B.S., WORCESTER POLYTECHNIC INSTITUTE

M.S., WORCESTER POLYTECHNIC INSTITUTE

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Erik G. Learned-Miller

Many classification techniques expect class instances to be represented as *feature vectors*, i.e. points in a feature space. In computer vision classification problems, it is often possible to generate an informative feature vector representation of an image, for example using global texture or shape descriptors. However, in other cases, it may be beneficial to treat images as variable size unordered sets or *bags of features*, in which each feature represents a localized salient image structure or patch. These local features do not require a segmentation, and can be useful for object recognition in the presence of occlusion and clutter.

The local features are often used to find point correspondences between images to be later used for 3D reconstruction, object recognition, detection, or image retrieval. However, there are many cases when exact correspondences are difficult or even impossible to compute. Furthermore, point correspondences may not be necessary, unless

one is interested in recovering the 3D shape of an object. If the correspondences are not computed, then this representation indeed constitutes an unordered set of local features.

In this dissertation we present methods for object class recognition using bags of features without relying on point correspondences. We also show that using bags of features and more traditional feature vector representation of images together can improve classification accuracy. We then propose and evaluate several methods of combining the two representations. The proposed techniques are applied to a challenging marine science domain.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

Object recognition is a fundamental problem in computer vision. Early attempts to solve this problem relied on CAD-like models of objects designed by people [74]. Using these models typically involved finding a transformation that would optimally align the model with the image. The disadvantage of such approaches is that they are not scalable to large sets of objects.

In the late 1990's algorithms have been developed that can learn the appearance of particular objects from example images labeled by people (e. g. [47]), and then recognize these same objects in new unlabeled images. Methods of this type do not require models of objects to be carefully constructed by hand. However, applying them to classifying images of objects into broad categories such as "chairs" or "lamps" has proved to be more difficult. This type of object recognition is usually called *object class recognition* or *object categorization*.

In this dissertation we focus on algorithms for automatically labeling images of plankton. Each image typically depicts one organism, which needs to be classified into a correct taxonomic category. This is an object categorization problem, because each image shows a different organism, rather than different views of the same organism. In our attempt to solve this problem we explore different image representations and machine learning algorithms. In particular we investigate representing images with unordered sets or *bags* of local features.

1.1 Using Local Features Without Correspondences

Representing images as sets of local features has been used in many computer vision problems. More precisely, by local features we denote descriptors of localized image neighborhoods, centered around certain points of interest. The origins of local features are in the problems of stereo matching [82] and 3D reconstruction [72], where correspondences between points in two or more images are required. Some applications, such as generating digital elevation maps, require correspondences between all pixels in two or more images [27]. This is often called "dense stereo", and it is computationally expensive. For other applications, such as mosaicking [7], or vision-based robotic manipulation [57], it is sufficient only to compute correspondences for some of the pixels, reducing the amount of computation. A subset of pixels, usually called *interest points*, is chosen in each image, and some descriptor of the image patch around each point is computed. The descriptors are used to assess the similarity of features, which is used, in conjunction with other constraints, for determining the correspondences.

In recent years local image features have also been used for the problems of object recognition [47, 22, 40, 56], object detection [47], and image retrieval [49]. These problems are often more challenging, because many of the assumptions used in stereo matching are no longer applicable. In stereo matching one is typically dealing with two images of the same scene taken from similar points of view. In the case of general object recognition, complications such as changes in viewpoint, scale, and lighting or presence of clutter and occlusion are to be expected. In object categorization we are even looking at images of the same object, but rather at instances of a broad class of objects.

In the face of these complications, local features offer a number of advantages over many other techniques, such as appearance-based approaches using global descriptors, e.g. eigenspace methods [73], or many approaches that use shape [4] or texture [61].

Local features usually yield a highly redundant representation of an object, which is robust with respect to occlusion, when some of the features are not visible. Most object recognition methods that use local features are also robust to the presence of spurious features corresponding to the background clutter, and thus do not require figure-ground segmentation.

Approaches using local features usually consist of three distinct parts: the *interest point detector*, the *feature descriptor*, and the *feature matching algorithm*, that assigns the correspondences between features. A plethora of methods of interest point detection and methods of defining the feature descriptors is described in the literature and is discussed in Sections 2.3 and 2.4. There are also a number of feature matching approaches discussed in Section 2.5, which utilize various heuristics.

Even though local features have been used for problems other than stereo matching, the paradigm has remained the same: point correspondences are computed between images. However, there are many cases for which exact correspondences are difficult or even impossible to compute. Examples include images of objects with high in-class variability, objects that are non-rigid, or images with repeating patterns. Furthermore, point correspondences may not be necessary, unless one is interested in recovering the 3D shape of an object.

In this dissertation we discuss several algorithms for object categorization that use local images features without computing explicit point correspondences between images. One particularly important contribution of this dissertation is the analysis of an embedding technique called the dissimilarity space [54], which can map stets of local features onto a Euclidean space. The analysis lead to the development of an efficient algorithm that has yielded promising results (Section 5.4.2).

1.2 Application Domain

In this dissertation we propose and analyze several different methods for object categorization using local image features without explicitly having to establish point correspondences between query images and the training images. The methods are applied to two image sets from a challenging marine science domain (Chapter 3). The images depict different types of plankton, and are collected *in situ*. The task is to classify each image into a correct taxon.

Studying plankton is an important part of ecological research. In particular, the marine biologists are interested in relative abundances of different plankton species in various areas of the ocean. They have tools at their disposal that are capable of taking images of plankton *in situ*. Thousands of images can be produced in a very short time, which currently need to be classified by experts. This work is part of an on-going project to automate this time-consuming task, which is a collaboration between the Computer Vision Research Laboratory at University of Massachusetts Amherst, the Machine Learning Laboratory also at UMASS Amherst, the Bigelow Laboratory for Ocean Sciences and the Department of Oceanography at Louisiana State University.

Automatically labeling plankton images is difficult because of high in-class variation. Recall that each image depicts a different organism rather than a different view of the same object. This problem is made more difficult by the fact that the organisms are photographed at arbitrary 3D orientations, under varying lighting conditions, and by the fact that many of them are capable of articulated motion. For these reasons computing meaningful point correspondences is very difficult.

An example of this is shown in Figure 1.1. The figure shows two views of the same organism, a copepod, taken at the same time at different angles. A state-of-the-art local feature matching algorithm [47] has produced 9 correspondences, indicated by dashed lines, out of approximately 400 features detected in each image. Only two of

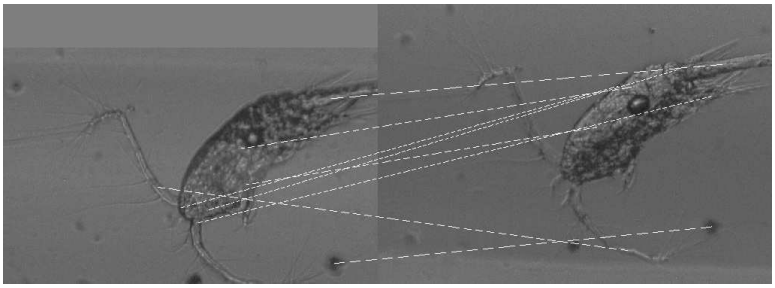


Figure 1.1. An example illustrating that computing correspondences even between images of exactly the same organism is very difficult. Dashed lines connect points considered to be corresponding by a state-of-the art matching algorithm. Only 2 out of 9 matches are correct.

these matches are correct. In our data sets each image shows a different organism, which makes finding correspondences even more difficult.

1.3 Combining Bags of Local Features with Feature Vectors

While local features have certain advantages for object class recognition, other cues such as shape and global texture can also be helpful. These object characteristics are usually represented by feature vectors, and require a segmentation of the object from the background. In most images in our data sets only one organism is present, and the background is generally uniform. This means that a reasonable segmentation can be computed (Section 6.1.1).

We thus have two representations for our images: the bags of local features and the feature vectors. We show in this dissertation that these representations capture different aspects of the objects of interest, and that using both in conjunction can increase classification accuracy. However, because one represents an image as an unordered set of vectors and the other as a single vector, using them together in a single classification system is not trivial. We present and analyze several methods for combining the bags of features with feature vectors.

The rest of the dissertation is organized as follows: in Chapter 2 we present the background on local features; in Chapter 3 we describe the application domain in greater detail; in Chapter 4 we describe a maximum likelihood classifier for bags of local features; in Chapter 5 we present several methods for classifying images that rely on pairwise comparisons between bags of features; in Chapter 6 we present the methods for combining bags of features and feature vectors, and, finally, our conclusions are presented in Chapter 7

CHAPTER 2

BACKGROUND

In this chapter we discuss various aspects of solving computer vision problems with local features. We explore the difference between representing images with bags of local features and feature vectors, present the background on multi-scale image representation, and attempt to construct a taxonomy of the existing local feature methods.

2.1 Feature Vectors vs. Bags of Local Features

Many classification techniques expect instances to be represented as *feature vectors*, i.e. as points in a feature space [19]. For example, if the task were to categorize people as high school students or college students, then age would be a reasonable feature, and the feature space would be one-dimensional. When feature vector representations are used, the classifier's task reduces to finding decision boundaries in feature space to separate classes of objects. In computer vision classification problems, it is often possible to generate an informative feature vector representation of an image, for example using global texture or shape descriptors. It is then straightforward to apply any of a large number of techniques such as perceptron learning, logistic regression, or support vector machines to image classification.

For example, eigenspace approaches [73]. usually normalize the images to be the same size, and treat the resulting arrays of pixels as feature vectors. The dimensionality of the resulting space is very high, making classification prohibitively expensive.

Therefore standard dimensionality reduction techniques, such as the principal components analysis, are often used to project the data onto a lower dimensional space.

Other approaches utilize histograms as feature vectors to represent images of different size, without the need for rescaling [61]. Functions of the intensity surface, such as its derivatives or combinations of derivatives, which are essentially texture descriptors, can be used instead of the raw pixel values [61].

A feature vector usually describes the image as a whole. As a result, methods using this representation typically expect an image to contain a single object with little background and little or no clutter. In this case it may be possible to perform figure-ground segmentation resulting in a labeling of pixels as belonging to the object of interest or the background. This allows for a variety of shape features can be computed. These include area, perimeter, compactness, moment-based features, convexity measures, granulometric features, and many others [4]. The values of these descriptors can also become elements of a feature vector describing the image.

While a feature vector is a compact representation of each image, the fact that its elements are usually global descriptors can be a weakness. While providing an overall description, they do not describe the various constituent parts of the image. For example, a global texture descriptor computed over an entire image [61], or even over the segmented object of interest, will poorly convey the fact that the object may contain several distinct types of texture. This is analogous to the Fourier transform, which describes the global distribution of frequencies in an image, but does not tell how these frequencies are distributed spatially in the image. By the same reasoning, a global shape descriptor will not be useful if objects of similar shape are distinguished by localized interior structures. The information associated with particular locations in the image, discarded by the global features, can be very important for representing classes of images, especially in the case of high in-class variability.

Indeed, there are methods using global texture descriptors that attempt to represent some of the local information by dividing images into regions. For example, a face recognition approach using histograms of differential features, proposed by Ravela [61], divides an image of a face into three regions in the vertical direction. The histograms are then computed for each region separately, and concatenated into a feature vector. This significantly increased the recognition accuracy compared to using a single histogram computed over the entire image. Of course, fixed partitioning works only for highly restricted domains, such as cropped-out faces, where images can be partitioned in a consistent and meaningful fashion. This will not work for a more diverse domain, such as images of phytoplankton, where a consistent fixed partitioning scheme cannot be applied.

Local features take the idea of representing different regions of an image to the extreme. They represent very small neighborhoods around certain image locations, i. e. the interest points. Representing an image by a set of such local features emphasizes the aspects of the image's appearance overlooked by the global features. However, local features yield a much less compact representation of images. Images are no longer described by single feature vectors, but rather by sets of different numbers of vectors. This makes it difficult to use standard classification and clustering techniques, which usually assume that each instance is representable as a single point in space.

2.2 Scale-Space Representation of Images

The concept of scale is extremely important in computer vision. It is illustrated by the following example: suppose we have taken several images of the same scene by successively moving the camera farther and farther away from it. If we then examine the resulting images, we will find that as the distance from the scene increases, the smaller structures in the image become less distinct, and eventually disappear en-

tirely. In each image, structures of a certain size are still clearly visible, while smaller structures have already vanished.

The process of taking multiple images of a scene from increasing distances is modeled mathematically by convolving a single image of the scene with a number of Gaussian kernels of increasing σ. The result of each convolution is called a "scale plane." The scale planes form a 3D volume, called the *scale-space* of the image [79]. Here we can observe the same phenomenon: the smaller structures become less pronounced at coarser scales, and eventually vanish completely. The scale space provides a convenient framework for obtaining a robust representation of the image structures of different sizes that is stable with respect to scale changes.

The Gaussian function is used for generating the scale-space of an image because it satisfies a number of important conditions. These are *causality*, meaning that coarse-scale structures are caused by fine-scale structures, and spurious artifacts are not introduced; *isotropy*, meaning that the Gaussian operator does not have a preferred direction; *homogeneity*, meaning that the operator is a translational invariant; *scale-similarity*, meaning that the Gaussian function does not change shape with scale [61]. The Gaussian and its derivatives form a unique family of functions for constructing the scale-space that satisfy these conditions.

The mathematical formulation of scale-space was originally proposed by Koenderink [41], and then further expanded by Lindeberg [43]. An in-depth discussion of scale-space, including implementation details is presented by Ravela [61]. In this dissertation we discuss local features computed at multiple scales, which make our approaches robust with respect to scale changes.

2.3 Interest Point Detection

In theory one can define a local feature at every pixel in the image. However, this generates a very large number of features, most of which are not necessary for

classification. To keep the number of features manageable, and at the same time to preserve most of the information contained in the image, local features are computed at an appropriate subset of pixels, called interest points. At a very high level this is analogous to using dimensionality reduction in eigenspace methods [73].

The interest points are usually defined as local extrema of some function of the image, and are designed to correspond to image structures that are deemed important. Examples of such structures include edges, which are points in the image where the signal changes abruptly; corners, which are points where the signal changes in two directions; and blobs, which are patches of relatively constant intensity, distinct from the background.

2.3.1 Evaluation of Interest Point Detectors

Schmid et al. [64] propose criteria for evaluating different interest point detectors. These are *repeatability* and *information content*. An interest point is repeated if it is detected at corresponding locations in two images. The notion of *information content* of an interest point has to do with how distinct it is, i. e. how likely it is to be mismatched.

Repeatability is defined as the ratio of the number of repeated interest points to the total number of interest points detected in an image [64]. Repeatability of an interest point detector is estimated by detecting the interest points in pairs of images related by a known transformation [64], so that the true correspondences are known.

It should be noted, however, that interest points are more likely to be repeated between two images if their density in the image is high. In other words, this definition rewards detectors that produce more interest points. This defeats the purpose of using interest points in the first place, which is to reduce the amount of information to be processed. A better definition of repeatability would take into account the density

of the interest points in the image, which is the prior probability of encountering an interest point at a particular pixel.

To define the *information content* of interest points [64], a feature vector of differential invariants is computed at its location in the image. The information connected is then defined as the entropy of the resulting set of features. Again, this definition is clearly limited. It only measures the information content with respect to a particular feature descriptor, namely the differential invariants. There does not seem to be any reason to believe that if one type of interest points has higher information content than another with respect to the differential invariants, it will still have higher information content with respect to some other feature descriptor.

Despite their shortcomings the definitions of repeatability and information content [64] discussed above provide an important basis of comparison for interest point detectors. Proposing these measures is a step in the right direction toward understanding and formalizing approaches that use local features.

2.3.2 Examples of Interest Point Detectors

In this section we describe several interest point detectors. These can be broadly categorized into "blobs", "corners", and "wavelet-based".

2.3.2.1 Blobs

Lindeberg [43] has formalized the notion of a "blob" in the image. A blob is a roughly circular image region containing similar intensity values. Centers of blobs are often used as interest points. Specifically, Lindeberg's interest points are local maxima in x, y, and scale of the Laplacian of the image. The Laplacian is computed at several scales using Gaussian derivative filters. The σ of the scale plane of a particular interest point corresponds to the radius of that particular blob.

Many approaches, including Scale Invariant Feature Transfrom (SIFT) [46], use the Difference-of-Gaussians (DOG) interest point detector, which approximates the

(a) Downtown Seattle (b) Difference-of-Gaussians (c) Harris-Laplacian

Figure 2.1. Examples of interest point detectors.

Laplacian. The DOG function of the image is computed by subtracting adjacent scale planes generated by convolution with Gaussian filters. Again, local maxima of the DOG function in x, y, and σ are used as interest points. These points correspond to the same blob-like structures as the maxima of the Laplacian, but the DOG function requires less computation. For the Laplacian I_{xx} and I_{yy} must be computed at every scale (at least 2 convolutions), while DOG only requires the scale planes themselves (1 convolution). An example of the interest points produced by the DOG detector is shown in Figure 2.1(b). Every interest point is located at the base of an arrow. The length of an arrow indicates the scale at which the interest point was detected, and the direction corresponds to the dominant gradient orientation around the interest point.

2.3.2.2 Corners

Corners are another type of image feature used as interest points. Formally, corners are defined as points of high curvature of the intensity surface of the image [82] [56], or as points, where the signal changes in two directions [64]. For example,

the corner detector used by Piater [56] returns the local maxima of the isophote curvature of the intensity surface [1]:

$$isophote(I) = I_{xx}I_y^2 + 2I_xI_yI_{xy} + I_{yy}I_x^2, \tag{2.1}$$

where I is the image. The curvature is computed at multiple scales using Gaussian derivative filters.

Arguably the most popular corner detector is the one proposed by Harris and Stevens [29]. The Harris detector, as it is usually called, is an improvement of an earlier technique developed by Moravec [51], which is based on the auto-correlation function of the signal. The Harris detector computes the auto-correlation matrix at each pixel in the image, and considers it an interest point if both eigenvalues of the matrix are high. Schmid et al. [64] describe the Harris features along with several other related corner detectors in more detail.

One of the disadvantages of the Harris detector is that it has no inherent notion of scale. In other words it only finds interest points that correspond to the finest scale plane of the image. To address this Mikolajczyk and Schmid [49] have proposed an extension to the Harris detector known as Harris-Laplacian. This method generates the Gaussian scale-space, and detects the Harris features at every scale plane. After that, the interest points that do not correspond to local maxima of the Laplacian in the scale direction are discarded. An example of Harris-Laplacian corners is shown in Figure 2.1(c). The centers of circles corresponds to the interest points, and their radii indicate scale.

---

[1] The isphote curvature is the curvature of the level curve, which is the intersection between the intensity surface of the image and a horizontal plane.

2.3.2.3 Wavelet-based Detectors

Another family of interest point detectors uses wavelets. Shokoufandeh et. al [69] propose a system in which a multi-scale decomposition of an image is computed using a 1D wavelet at a range of orientations. The local maxima of the sum of the wavelet responses are used as interest points.

An approach by Sebe et. al [67] uses a 2D wavelet transform of the image. The wavelet coefficients are tracked across scales from the coarsest scale to the original pixels, and the "saliency value" of a pixel is set to the sum of all the coefficients in its track. The resulting map is then thresholded to obtain the interest points. For images of natural scenes this detector returns points that are more uniformly distributed throughout an image than points produced by corner detectors. According to the authors this results in a "more complete representation" [67]. The performance of Sebe's detector is shown to be superior to that of Harris corners using repeatability and information content metrics proposed by Schmid et al. [64].

2.3.3 The Meaning of Saliency

The interest points are often called "salient" points. The word "salient" is defined by the Webster's Dictionary as "prominent", "conspicuous", "projecting or pointing outward." In other words, it is something that stands out from its surroundings.

The interest points defined by the above mentioned detectors are not necessarily salient in this intuitive sense. Saliency depends on the context, which these feature detectors do not really take into account. For example, a corner would not at all seem salient in an image of a grid pattern. Similarly, a stop sign, which is designed to stand out (Figure 2.2(a)), does not appear salient in Figure 2.2(b).

Walker et al. [77] have proposed an interest point detector that corresponds more closely to the Webster's definition of saliency. A local feature descriptor consisting of differential invariants is computed at every pixel of the image forming a large feature

(a) (b)

Figure 2.2. Saliency depends on context.

space. The saliency of a feature point is represented by the density of the feature space, which is computed using kernel estimation. Features in the low density areas of the space are considered to be more salient. This approach determines the saliency of points from the data, rather than using a generic notion of which points "should" be salient.

Interestingly, this approach produces interest points that by construction have high information content with respect to the feature descriptor used. Unfortunately, high repeatability for these points cannot be guaranteed. Consider two images of large scenes containing the same object on very different backgrounds. The densities of the feature spaces corresponding to the two images are likely to be very different. As a result the points corresponding to the object may easily end up in areas of different densities and may be considered salient in one image but not the other.

A similar approach is presented by Lisin et al.[44]. Instead of estimating the density non-parametrically over all the features from every pixel, this method estimates the density of the feature at each pixel locally, using the features at the neighboring pixels. It then uses an outlier detection method to determine the saliency of each feature.

This method is based on the same notion of computing the saliency of the features from the data. Its complexity is $O(n)$, as opposed to the $O(n^2)$ complexity of the Walker's approach [77], and the repeatability of the resulting interest points is likely to be higher. The latter, however, can also be due to the fact that a very large number of interest points is detected. Also, because the saliency of the points is defined using only their immediate neighbors in the image, they are likely to have lower information content. However, the saliency of these points should be affected little by the background or clutter.

Another attempt to formalize the intuitive notion of saliency was proposed by Kadir and Brady [37]. They define an image region to be salient if the entropy of some value computed at every pixel in the region is high. For example, the value can be pixel intensity or color. In other words, image patches containing highly varying signal have high entropy and are considered salient. Intensity distributions of image patches containing uniform or slowly varying signal will exhibit one or more strong peaks resulting in low entropy.

The entropy is computed for image regions of different sizes, resulting in a multi-scale interest point detector. This original detector used circular windows [37]. An anisotropic version of the detector using ellipses has also been proposed [38], and a fully affine-invariant version has been developed [39].

Kadir and Brady's idea of using entropy of a region as a measure of saliency resulted in a high quality interest point detector. However, it also does not entirely correspond to the dictionary definition of "salient." When the detector was applied to an image of a leopard [39], almost every one of the leopard's spots was considered salient. However, intuitively we know that the individual spots are insignificant. It is the texture pattern they form, that can make the whole leopard salient depending on the background. In fact, trying to match individual spots does not seem to be very useful at all, since they all are essentially the same.

We conclude the discussion of saliency by stating that not all interest points are salient in the intuitive sense. Whether or not a point or a patch in an image is salient heavily depends on the context, i. e. the rest of the image. Because of that, points or patches that are truly salient may not be repeatable, and thus may not be good interest points. However, saliency can be useful as a focus-of-attention mechanism [33], which is very different from finding interest points for matching.

The term *salient points* does not quite capture the properties that interest points are expected to have. For example, to be effective interest points need to be repeatable, which may contradict the dictionary definition of "salient". For this reason throughout the rest of this dissertation we shall use the term *interest points*.

2.4 Local Feature Descriptors

The feature descriptor is an essential part of all methods using local features. After interest points have been detected, descriptors need to be computed to represent the image patches around those points. The simplest feature descriptor can be defined using the pixel values around an interest point. For example, many stereo matching systems simply define a window around an interest point and use cross-correlation to detect it in another image [82]. Unfortunately, in many situations, using raw pixel values will not yield the best performance, because this feature descriptor is not robust to rotation and changes in scale, lighting, and viewpoint.

2.4.1 Differential Feature Descriptors

More sophisticated feature descriptors based on image derivatives have been proposed. Taylor series expansion allows us to approximate the intensity surface of the image to an arbitrary precision using image derivatives. Therefore, derivatives computed at a pixel describe the image patch around it.

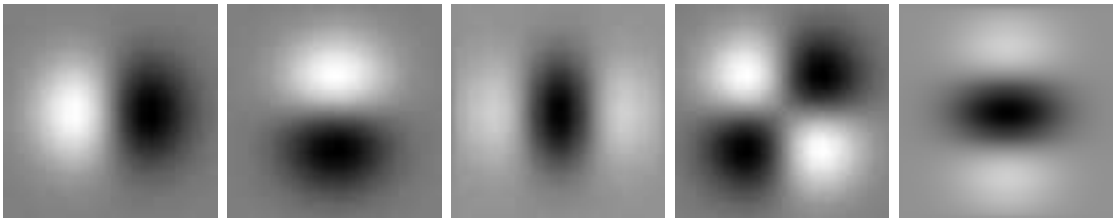


Figure 2.3. Gaussian derivative filters up to the 2nd order.

The simplest differential feature representation at a pixel can be defined by computing the derivatives up to some order at that pixel. This is typically done using Gaussian derivative filters, shown in Figure 2.3, which means that the derivatives can be computed at a range of scales. This descriptor is known as the *local jet* [41].

There are two methods for making a differential feature descriptor invariant to in-plane rotation. The first one is to normalize the derivative responses for rotation using steerable filters [23]. The second one is to compute rotationally invariant combinations of the derivatives, such as the gradient magnitude, the Laplacian, and the isophote and flowline curvatures [61]. Interestingly, the feature descriptors using steered derivatives seem to yield higher matching accuracy than those using differential invariants. According to Mikolajczyk and Schmid [50] this is probably due to the accumulation of errors when the combinations of derivatives are computed.

Differential descriptors provide a compact low-dimensional feature representation that can be computed relatively inexpensively. They have been used in many methods for object recognition from gray scale images [77] [44] [56], and they have also been extended to color images [25].

2.4.2 Region-based Feature Descriptors

Very effective feature descriptors can be computed over image patches around interest points, unlike the differential descriptors computed at the interest points. The most famous example is the SIFT (Scale Invariant Feature Transform) features

[46]. Each SIFT descriptor is a three-dimensional histogram of gradient orientations (x, y and θ), computed over a Gaussian-weighted window around an interest point, which is represented as a vector of 128 elements. This descriptor has been shown to perform exceptionally well, compared to differential descriptors computed at interest points [50].

Two variations to the SIFT descriptor have been proposed. The first, known as PCA-SIFT [40], also uses the gradient around an interest point, but does not compute a histogram. Instead this approach represents an image patch around an interest point as a long vector of the values of the gradient orientation and magnitude, and then uses the principal components analysis to reduce the dimensionality. The second variation, known as the Gradient Location and Orientation Histogram (GLOH) improves the SIFT descriptor by using a log-polar histogram in x and y [50].

2.4.3 Evaluation of Feature Descriptors

Mikolajczyk and Schmid [50] present a very thorough evaluation of the feature descriptors mentioned above, as well as several others. The performance evaluation of the descriptors was done in the context of finding point correspondences between images of the same object or scene under various distortions, such as scale change, in-plane rotation, image blur, illumination change, and even JPEG compression. The distorted images, aside from the case of the JPEG compression, were not simulated. Rather, they were obtained by varying the camera zoom, angle, and other settings. Descriptor performance was evaluated and compared using the recall-precision method. The results of the evaluation show that higher-dimensional descriptors computed over a larger support area, namely GLOH and SIFT, significantly outperform the lower-dimensional differential descriptors.

2.5 Matching Local Features

In this section we discuss methods for establishing correspondences between sets of local features. We describe criteria used to determine if a pair of local features from different images are a match and methods for identifying and discarding matches that are ambiguous.

2.5.1 Similarity-Based Matching Strategies

Most methods use similarity or dissimilarity measure between two feature descriptors as the primary cue for matching. By similarity we mean a function of two descriptors that returns a larger value when they are more similar. Conversely, a dissimilarity measure returns a smaller value when the two descriptors are more similar. An example of a similarity measure is a normalized inner product between two descriptors [56]. Examples of a dissimilarity measure include the Euclidean distance [46] and the Mahalanobis distance [77, 50, 25].

The Euclidean distance can be used when all components of the descriptor are expressed in the same units. This, for example, is the case for SIFT, whose descriptor is a histogram. On the other hand, differential features are usually compared using the Mahalanobis distance [77, 50, 25], because the ranges of values of their components differ by orders of magnitude.

Regardless of how the descriptors are compared, there are several common strategies for matching. We describe these strategies in term of a dissimilarity measure, but they can be easily adapted for a similarity measure. Let $d(f_1, f_2)$ be the dissimilarity between feature descriptors f_1 and f_2. Let F_1 and F_2 be the sets of features extracted from images I_1 and I_2 respectively. One approach to define a match between $f_1 \in F_1$ and $f_2 \in F_2$ is to require that

$$d(f_1, f_2) = \min_{f \in F_2} d(f_1, f). \tag{2.2}$$

In other words f_1 matches f_2 iff f_2 is its nearest neighbor in F_2. This strategy, called *nearest neighbor* (NN) [50] is used by Piater [56]. It can be more restrictive by requiring that f_1 also has to be the nearest neighbor of f_2 in F_1 [44]. Of course neither version of the match definition can guarantee that the resulting match is indeed an actual correspondence.

A different definition of a match simply requires that the distance between two feature descriptors be less than some threshold t. Under this definition, called *threshold-based matching* [50] there may be several matches for feature f_1 in the set F_2. One advantage of using this method is that features from different images can be organized into an indexing structure such as a k-d-b tree [63] to find the matches faster. An important disadvantage is the fact that choosing the right threshold is always difficult.

Lowe [46] proposed a heuristic for defining a match, called *nearest neighbor distance ratio* (NNDR)[50]. To find a match for a feature f in F, which is a set of features extracted from a particular image, we consider f_1, the nearest neighbor of f in F, and f_2, its second nearest neighbor in F. We define f_1 to be a match of f if

$$\frac{d(f, f_1)}{d(f, f_2)} < t, \tag{2.3}$$

where t is a threshold. Thus a feature is only considered to be a match to its nearest neighbor if its distance to the second nearest neighbor is significantly larger. The purpose of this heuristic is to detect cases in which a feature f is similar to many features in set F, and thus avoid ambiguous matches. One negative side effect of this is that local features corresponding to repeating patterns in images will not be matched.

Mikolajczyk and Schmid [50] present recall-precision graphs comparing performance of various feature descriptors under three matching strategies: NN, threshold-based, and NNDR. There was little or no change for all descriptors between NN and

NNDR, and all descriptors performed worse with the threshold-based matching than with the other two.

2.5.2 Matching Constraints

The matching strategies using the distance between feature descriptors are designed to decrease the probability of spurious matches. However, incorrect matches still occur, even with the most appropriate interest point detector and the most robust feature descriptor. Other constraints are often employed to increase the matching accuracy.

In stereo matching systems the epipolar geometry is the primary matching constraint [82]. Usually stereo systems consist of two cameras. The epipolar plane is defined by a point in the 3D space and its projections onto the images in each of the cameras. The intersection of the epipolar plane and an image plane is called the epipolar line. If the cameras are calibrated, meaning that their relative positions and orientations are known, then for each pixel in one camera a corresponding epipolar line in the other camera can be easily computed. Thus one only needs to search for matches along the epipolar lines. This reduces the search space from two dimensions to one, and increases matching accuracy. With uncalibrated cameras a small number of feature matches deemed reliable can be used to infer the unknown epipolar geometry [82].

In tasks such as general object recognition, and especially object class recognition, the epipolar constraint is not applicable. Instead the geometric arrangement of features in the image is often used as a matching constraint. Of course, this assumes that the objects of interest are rigid.

Piater [56] utilizes spatial relationships among the local features by combining them into compound features using their relative distances and orientations. The order of the constituent primitive features is fixed: each feature's match is used as an

anchor to locate the match of the subsequent feature. Unfortunately, this means that if any of the features in this chain is matched incorrectly, the remaining ones will be wrong as well. While combining local features into compound features seems to be a good idea, this particular method for matching compound features is not likely to perform very well in the presence of noise or image distortions.

A more robust method of taking into account the spatial relationships among local features is the generalized Hough transform [1]. It assumes a transformation between two sets of points in two images, such as the similarity or the affine transformation. Given the candidate correspondences, the transformation parameters can be computed from triples of matches. The parameters are then placed binned into a hash table, and the bin with the highest number of hits corresponds to the dominant transformation. After that, matches that do not fit the dominant transformation can be discarded.

The original formulation of the generalized Hough transform has a time complexity of $O(mn^3)$, where m is the number of the transformation parameters, and n is number of matches. Fortunately, its performance can be significantly improved if the location, scale, and orientation of each local feature is known, as is the case with the SIFT features [47]. In this case the similarity transformation parameters can be computed from each single match rather that a triple, and used for the initial binning of matches. The resulting complexity is $O(mn)$.

An interesting method to represent the spatial relationships among local features is to incorporate them into the feature descriptor, as proposed by Belongie et al. [2]. For every interest point the descriptor, called the *shape context*, is defined as the log-polar histogram of occurrences of the other interest points around it.

Many approaches use probabilistic models to represent the spatial relationships as well as feature matching in general. These are discussed separately in Section 2.5.3.

2.5.3 Probabilistic Matching

Matching strategies and constraints discussed above are designed to reduce the probability of incorrect or spurious matches. However, these methods are mainly heuristics, and do not guarantee correctness. An attractive alternative is to use probabilistic matching methods that maximize the probability of correctness directly. While there are still no guarantees, at least such methods quantify the uncertainty to let us make more intelligent decisions.

A general probabilistic framework for matching local features is presented by Pope and Lowe [59]. Their paradigm is to learn the model of a particular object from a set of training images, and then to use it for recognition. The model is composed of local features that are not restricted to descriptors of the intensity surface. In fact the features used by Pope and Lowe [59] are mainly edge based, including line segments, curve segments, parallel line or curve segments, and others. However, in principle, the framework is decoupled from the feature definition, and should be applicable to any type of feature.

The method performs probabilistic alignment of local features taking into account the feature types such as lines and curves, and feature attributes, represented by vectors of values specific to particular feature types. The attributes are analogous to the feature descriptors discussed above. The alignment involves estimating the transformation parameters between the likely matches of features from the model and the test image, iteratively adding matches consistent with the transformation, and updating the parameter estimation. The goal of this process is to maximize the likelihood of the matches given the transformation parameters. The uncertainty of the transformed feature locations is modeled by a Gaussian distribution, and the distributions of feature attribute values are represented non-parametrically. The details of the treatment of the attributes are explained in Pope's Ph. D. thesis [58]. To reduce the computational complexity, the features are assumed to be independent.

Fergus et al. [22] attempt to solve the problem of object class recognition, rather than that of recognizing a specific object, by using a similar probabilistic approach. The task is to decide whether or not an image contains an object of a particular class, such as cars, motorcycles, or airplanes. This method uses Kadir and Brady's interest points [37], and the feature descriptors are patches of intensity values around them, normalized for brightness and size. The patches are represented as vectors, and the principal components analysis (PCA) is used to reduce their dimensionality.

The probabilistic model of a class of objects is similar to the one proposed by Pope and Lowe [59]. It also incorporates similarity between feature descriptors and the spatial relationships among the interest points. In this case the feature appearance, i. e. the descriptors, is modeled by a Gaussian distribution, rather than non-parametrically. In addition, this approach models the relative scale of the features separately. The likelihoods of feature descriptors and their relative scales are assumed to be independent. The parameters of the model are learned using the Expectation Maximization (EM) algorithm.

The spatial relationships in Fergus's approach are also represented differently. Instead of estimating the transformation parameters by solving the normal equations [59], translation, rotation, and scale are eliminated by transforming the features in an image into a *shape space* [17]. First two reference features f_1 and f_2 are chosen. The feature set is then translated so that f_1 is at the origin, and then scaled and rotated so that feature f_2 is at $(0, 1)$. The shape of the object in the image is then represented as a vector consisting of the coordinates of the local features in this new shape space. The distribution of these shape representations for objects of a particular category is estimated using a mixture of Gaussians. Each possible choice of f_1 and f_2 is treated as a hypothesis, whose likelihood is computed from the estimated distribution [17] [10].

Moreels and Perona [52] use a probabilistic matching method together with the SIFT features [47] in the context of specific object recognition. The features are assumed to be independent and their appearance, position, orientation, and scale are modeled by separate distributions. The spatial relationships among features are represented in an interesting fashion. Locations, orientations, and scales of features are computed with respect to a *reference frame* defined by the image from which they were extracted. When two images are compared, a list of candidate matches is compiled using only the feature's appearance. For each candidate match the reference frames corresponding to the two images are aligned using the difference in location, scale, and orientation between the matched features. As each additional match hypothesis is considered, the frame alignment is recomputed accordingly to minimize the mean squared error in the location of the features. The likelihoods of the features' geometric properties are then computed with respect to the aligned frame.

2.6 Summary

In this chapter we have presented a taxonomy of methods using local features. We have discussed existing approaches for interest point detection, defining local feature descriptors, and establishing correspondences. In the next chapter we describe in detail our application domain, for which computing point correspondences does not seem to be the best approach. In subsequent chapters we present a number of object class recognition algorithms that use local features without explicit correspondences.

CHAPTER 3

APPLICATION DOMAIN

This work is a part of an on-going project in collaboration with marine scientists, who are interested in studying plankton. They have a variety of image acquisition tools capable of producing an enormous number of images in a very short time. The objective of this project is to develop software tools to automate the process of labeling these images. In this chapter we describe this application domain and, specifically, the data sets that we used.

3.1 Why Study Plankton?

The Earth's oceans serve as major sources and sinks of bio-active elements that naturally cycle through the biosphere. The ocean water is a soup of living (plankton) and non-living (detrital) particles. The importance of plankton for the global ecosystem cannot be overestimated. Microscopic algae, phytoplankton, are sometimes called the grasses of the sea. Just like land plants, they consume carbon dioxide and produce oxygen through photosynthesis. Phytoplankton are an integral component of the global carbon cycle, which is responsible for regulating the temperature of the planet [18]. They are also the first link in the food chain for all marine creatures. Their primary consumers are the zooplankton, who in turn become food for larger animals.

Studying plankton is important to ecological research. For example, understanding the carbon cycle is necessary to be able to predict global climate changes. On a smaller scale, studying plankton can allow marine biologists to create early warn-

ing systems for detecting harmful algal blooms in coastal waters [4]. Applications in other fields could include ship ballast water treatment, drinking water treatment, public health, bio-terrorism defense, and industrial chemical processing.

3.2 Previous Work

Research to automate the task of labeling plankton specimens has been going on for many years [36]. However, many systems have only been shown to work under controlled laboratory conditions with a cultured population. The specimens are usually carefully prepared and imaged using high magnification, yielding high quality images. For example, the ADIAC project [18] resulted in the development of a system that has been very successful in classification of diatoms from very high-resolution images. The system uses a wider variety of shape and texture features along with a range of classification techniques.

Work has also been done attempting to classify field-collected specimens. For example, Culverhouse et al. have developed a system known as DiCANN for classification of dinoflagellates [12], which has been tested on images from video-cameras used underwater, as well as images obtained by laboratory-based instruments. DiCANN uses several types of texture and edge-based image features as inputs for a neural network.

Classifying images collected *in situ* is a very difficult problem. In many cases even the human experts disagree. Poor image quality, articulated motion, partial visibility, high in-class variability, and even fatigue from looking at thousands of images result in differing opinions among experts on how to classify a particular organism. A study has been done attempting to evaluate the consistency rate of human experts labeling images of dinoflagellates [13]. Its conclusion was that on that particular data set marine scientists agree on image labels only 75-80% of the time.

Figure 3.1. The FlowCAM.

One interpretation of this result is that classifying plankton images is a difficult problem even for human experts. Therefore we should not expect the accuracy of automatic classifiers to be perfect. Another interpretation is that the labels of the training images provided by human experts are not necessarily correct, which is likely to hinder the performance of an automated system.

3.3 FlowCAM Images

One type of plankton image used in this thesis has been acquired by an instrument for monitoring the abundance of phytoplankton and small zooplankton, known as the FlowCAM (Flow Cytometer And Microscope) [70] (Figure 3.1). The FlowCAM detects and takes images of micro-organisms from a stream of water siphoned directly from the ocean. It even has a rudimentary image segmentation capability, used to crop out individual organisms. The instrument is used by marine biologists to estimate the population sizes of different plankton species. In particular, scientists are interested in detecting an influx of potentially harmful organisms.

The FlowCAM is capable of generating thousands of images in a matter of hours. Currently, manually identifying the organisms appearing in all these images is a daunting task for the marine biologists, requiring a huge number of man-hours. Clearly, there is a strong motivation for trying to automate this process.

Classifying FlowCAM images automatically is difficult due to ambiguity resulting from high in-class variability. In our experiments we have used a data set acquired by the FlowCAM, containing 980 images of 13 taxonomic categories (Table 3.1). The images have been labeled by consensus of a committee of experts at Bigelow Laboratory for Ocean Sciences, in Boothbay Harbor, Maine. The organisms depicted in the images flow by the camera in a stream of water turning and tumbling, resulting in arbitrary 3D orientations. Furthermore, some organisms are capable of articulated motion, changing their shape significantly, while others form chains of cells that can bend and twist.

Another difficulty peculiar to FlowCAM images is low magnification. The goal of the marine biologists is to detect and identify as many organisms in a sample of water as possible. In order to achieve that, the magnification of the microscope has to be small (4x) to increase the field of view. Because of the low magnification much of the potentially descriminative high-frequency texture information is lost. Also, it is more likely than not that the organism is out of focus, losing even more high-frequency detail.

To illustrate the quality of FlowCAM images, Figure 3.2 shows one of them next to an image used in the ADIAC project [18]. We have also compared the quality of FlowCAM images to those used in ADIAC more objectively, by computing the power spectra of 100 images of each type, and averaging them. Figure 3.3 shows the average spectra for the ADIAC and the FlowCAM images. Clearly we can see that the FlowCAM images contain very little high-frequency information as compared to the ADIAC ones, which indicates a general lack of detail.

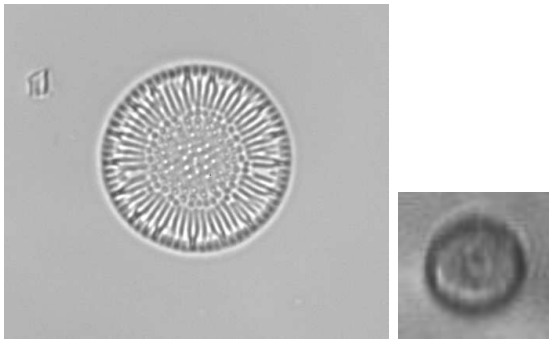


Figure 3.2. Left: image from ADIAC project. Right: image from the FlowCAM

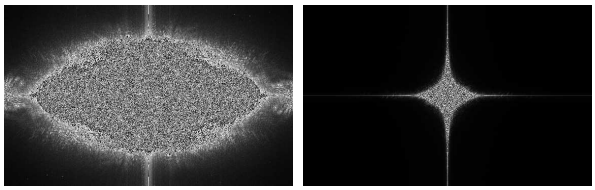


Figure 3.3. Left: average power spectrum of ADIAC images. Right: average power spectrum of FlowCAM images

Table 3.1. Taxonomic Categories of Images in the FlowCAM Data Set

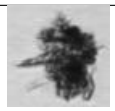
Category Name	images	example
Unknown	39	
Centric diatoms	26	
Pennate diatoms	124	
Dinoflagellates	29	
Ciliates	179	
Unidentified cell	32	
Non-cell	113	
Mesodium	71	
Laboea	30	
Skeletonema	169	
Thallasiosira	86	
Thallasionema cf.	23	
Pseudo-nitzschia	61	

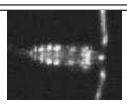
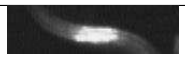
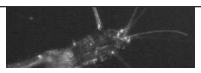
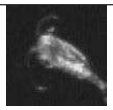
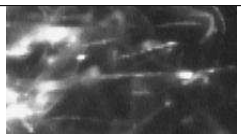
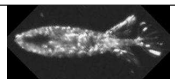
Figure 3.4. The Video Plankton Recorder.

3.4 Video Plankton Recorder Images

Another type of image, that typically feature larger zooplankton, is produced by an instrument called the Video Plankton Recorder (VPR) [3] (Figure 3.4). The VPR consists of a single video camera and synchronized strobe that captures images of the contents of a small volume of water at a rate of 60 Hz. In this study the volume was 5.1 milliliters per image, and the field of view was 17.5 mm wide x 11.7 mm tall x 25 mm deep. Images were transmitted to the surface via fiber-optic cable. Then time-code from a GPS system was added, and the data were archived on S-VHS videotape. In the lab, the video signal was routed through a PC-based image processing system (Imaging Technologies) that digitized each image and located objects meeting user-defined criteria for size, brightness, and focus. Objects meeting these criteria (termed regions of interest or ROIs) were cropped and written to disk as individual TIFF files. A subset of images was manually classified into categories that ranged from individual species to broader groups, depending on how many characteristics indicative of a particular category, such as eyes or appendages, were present.

For our experiments we have used a data set of 1826 images of 14 taxonomic categories acquired by VPR. The categories are listed in Table 3.2. The images were labeled by a single expert, Dr. Mark Benfield at Louisiana State University.

Table 3.2. Taxonomic Categories of Images in the VPR Data Set

Category Name	Taxonomic Group	images	example
Calanus finmarchicus	copepod species	132	
Chaetognaths	zooplankton phylum	86	
Conchoecia Ostracods	ostracod genus	100	
Ctenophores	zooplankton phylum	34	
Euphausiids	zooplankton order	131	
Hyperiid Amphipods	zooplankton suborder	68	
Pteropods	zooplankton order	142	
Diatom Rods	phytoplankton class	97	
Larvaceans	zooplankton class	133	
Small Copepods	zooplankton class	433	
Unidentified Cladocerans	zooplankton order	108	
Siphonophores	zooplankton suborder	202	
Euchaeta norvegica	copepod species	81	
Siphonulae	developmental stage of zooplankton suborder	78	

Just as the FlowCAM, the VPR photographs organisms *in situ*, which means that they can be at any orientation relative to the camera. Also, the larger zooplankton, which are a primary target of the VPR, are often capable of a wider range of articulated motion than the microscopic plants imaged by the FlowCAM, resulting in greater variation in shape. On the other hand, because the organisms in the VPR images are larger, more texture detail is often visible.

An issue peculiar to the data produced by the VPR is that the video frames are interlaced, i. e. each frame only contains half of the scan lines, either the even or the odd ones. Because both the camera and the organisms are moving rapidly, reconstituting a complete scene from adjacent frames is impossible. Therefore the images have to be interpolated to recover the proper aspect ratio.

3.5 Summary

In this chapter we have described our application domain. We have discussed the importance of studying plankton, and we have described the two data sets that we used in detail. In the subsequent chapters we present algorithms for classifying the images in these data sets using local features without explicit correspondences.

CHAPTER 4

MODELING DISTRIBUTIONS OF LOCAL FEATURES

In this chapter we discuss a method of classifying images represented by unordered sets or bags of local features by estimating the probability distribution of features in each class. We estimate the probability density function over the space of real-valued local feature non-parametrically for each category, and construct a maximum likelihood classifier. We first review non-parametric density estimation, and then discuss how this technique is adapted for bags of features.

4.1 Non-parametric Density Estimation

The maximum likelihood classifier is a very well-understood generative classification technique [19]. If the instances are points in a space, and the probability distribution for each class over this space is known, then one can simply look up the likelihood of the instance given each class, and output the label corresponding to the maximum likelihood.

Unfortunately the probability distributions are usually not known in advance, and need to be estimated from sample data. The methods for doing that can be categorized as parametric or non-parametric. A parametric method assumes that the distribution has a particular shape, described by a parametric family of functions such as Gaussian or Poisson, and estimates its parameters from data. This is a reasonable approach when one knows in advance the shape that the distribution may have.

In our case we know very little about the shape that a distribution of the local features may have. In such cases the probability density function (PDF) of a set of

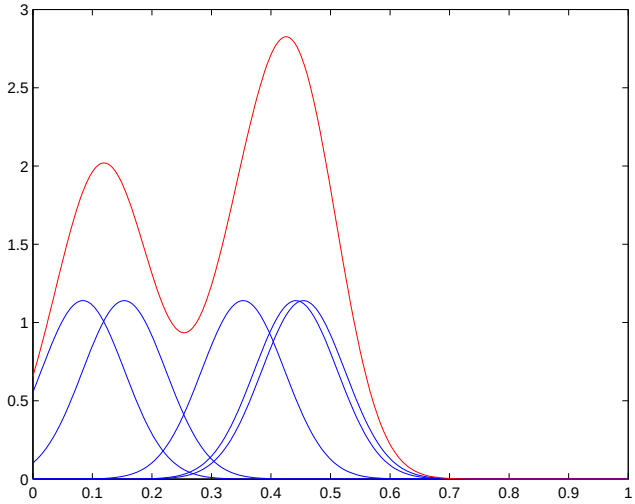


Figure 4.1. Example of non-parametric density estimation.

samples can be estimated non-parametrically by a normalized sum of kernels placed at every sample. The most common choice of a kernel is a Gaussian centered at the sample, but other functions such as the Laplacian are sometimes used. An example is shown in Figure 4.1. This method is also called *kernel density estimation* (KDE).

An important question to ask at this point is: "What should the bandwidth of each of the kernels be?" Figure 4.2 shows how the bandwidth of the kernels affects the resulting PDF. There are several approaches for dealing with this issue. Below we describe three of them.

In the first method, called Parzen windows [19], the kernels are restricted to be isotropic and to have identical bandwidth. Thus, there is only one parameter, the bandwidth σ, which is set to maximize the leave-one-out mean log likelihood of every sample.

The second method, called k-nearest neighbor estimation, sets the bandwidth of every kernel to be the distance to its kth nearest neighbor. This approach is much faster than Parzen windows because no optimization is required. Also, under this

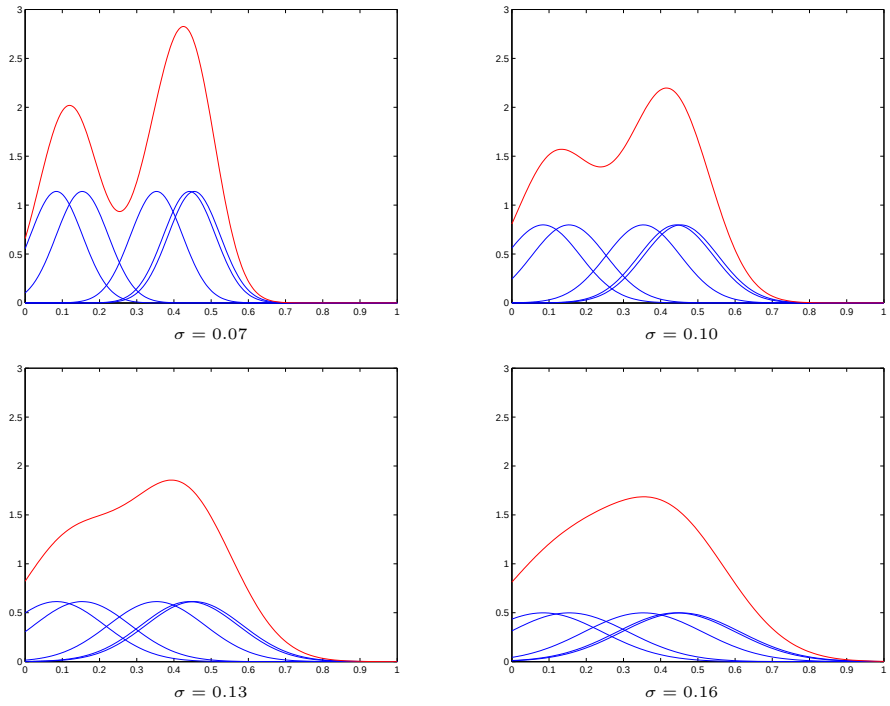


Figure 4.2. What should the bandwidth be?

scheme every kernel will have a different bandwidth, which may be advantageous if the space has areas of high and low density. However, while the Parzen windows method is fully automatic, for the KNN approach the parameter k must be set manually.

The third approach is the simplest of the three, but often does not perform as well. It is called the "rule-of-thumb" [66]. In this approach all of the kernels are identical, but not necessarily isotropic. The bandwidth h_k of each kernel along a dimension k is defined as

$$h_k = \sigma_k n^{-1/(d+4)}, \tag{4.1}$$

where σ_k is the variance of the data along the dimension k, n is the number of samples, and d is the number of dimensions.

4.2 Adapting Kernel Density Estimation to Bags of Features

Our task is to classify images that are represented by sets of vectors, rather than individual points. In order to do that we make a simplifying assumption that an instance is a set of local features drawn independently from a distribution specific to each class. This assumption allows us to compute the likelihood of an image as the product of the likelihoods of its constituent local features, which we can get from the kernel density estimate. Equivalently, the log likelihood of an image is the sum of the log likelihoods of its local features.

We start by pooling local features from training images of a particular class [45]. Then the kernel density estimate for each class is computed using Parzen windows. Let $C = \{C_1, C_2, ..., C_n\}$ be the set of image classes. Let $Q = \{q_1, q_2, ..., q_m\}$ be a query image to be classified, and $q_i \in Q$ be its constituent local feature. Then the log likelihood of the query given each class is given by

$$\log p(Q|C_i) = \sum_{j=1}^{m} \log p(q_j|C_i), \tag{4.2}$$

where $p(q_j|C_i)$ is given by the PDF of C_i. Then the posterior probabilities for each class $p(C_i|Q)$ can be easily computed by normalizing the likelihoods:

$$p(C_i|Q) = \frac{p(Q|C_i)}{\sum_{j=1}^{n} p(Q|C_j)}. \tag{4.3}$$

Then the maximum likelihood label of Q is given by

$$\arg\max_i p(C_i|Q) \tag{4.4}$$

This approach essentially estimates the marginal distribution of images over the local features. This is advantageous because we have many more local features than we have images, i. e. the distribution is estimated from a larger number of samples. The disadvantage is the feature independence assumption that allows us to do that

does not hold in reality. One reason for this comes from the very nature of multiscale local features. Finer scale features give rise to those at the coarser scale, and thus the occurrence of the latter can be predicted from the occurrence of the former. Another reason is that local features may correspond to parts of the organism that are not statistically independent, such as a head and a tail.

4.3 Experimental Results

The Maximum likelihood classifier has been tested on the FlowCAM and the VPR image sets. In both cases the images were represented as sets of SIFT features. One problem with computing a kernel density estimate over the space of SIFT descriptors is the fact that the space has 128 dimensions. For the number of samples that we have, this dimensionality is extremely high. This is especially evident in the case of the FlowCAM data set, which consists of only 982 images, and each image yields very few SIFT features (often fewer than 10). In order to be able to compute a better estimate we first reduce the dimensionality of the local features using principal components analysis (PCA). This preprocessing step may not be necessary for a lower-dimensional feature descriptor. It also may not be need for data sets that either consist of many more images than either FlowCAM or VPR, or contain larger images that produce many more local features.

We present classification accuracy obtained using 10-fold cross-validation for 2, 4, 8, and 16 principal components. The error bars depict one standard deviation of the accuracies of individual folds. Results for the FlowCAM data set are shown in Figure 4.3, and results for the VPR data set are presented in Figure 4.4. Notice that on average the best performance of the classifier on both data sets was achieved when the dimensionality of the SIFT features was reduced from 128 to 8. These results are compared to those of other classifiers for bags of features in Section 5.5.

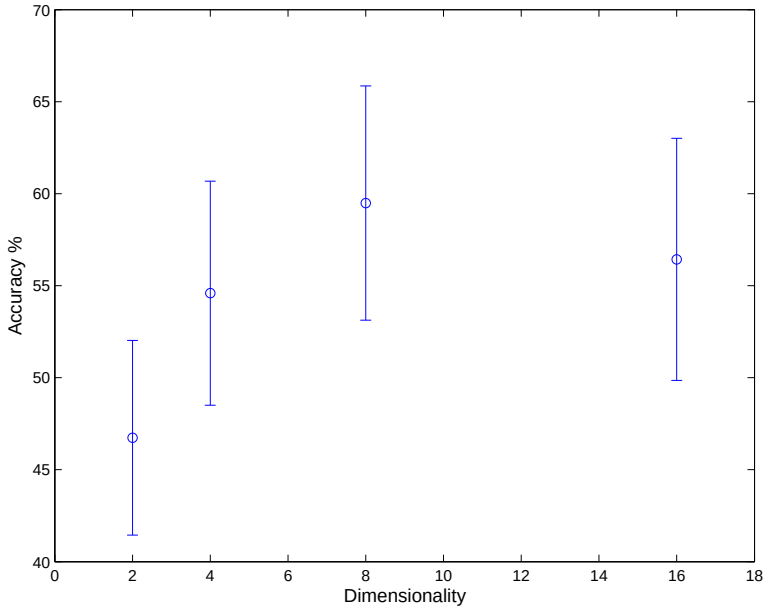


Figure 4.3. Accuracy for the Maximum likelihood classifier on the FlowCAM data set.

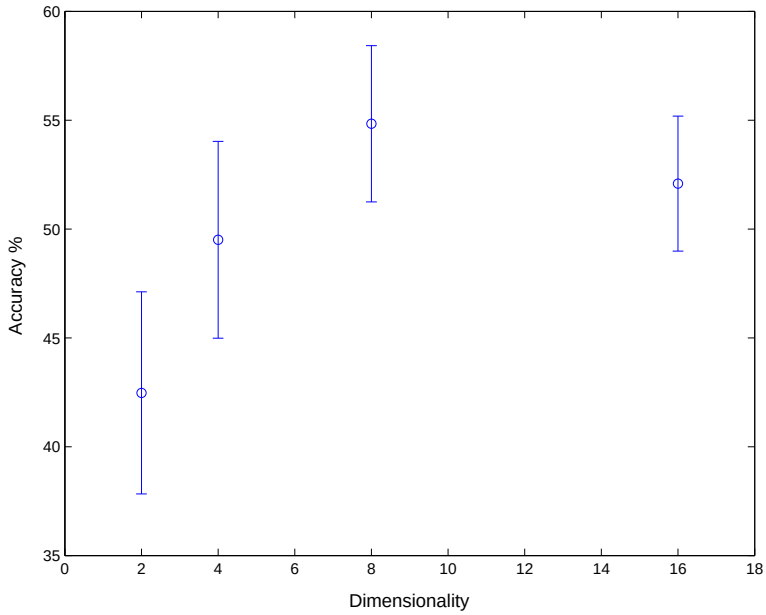


Figure 4.4. Accuracy for the Maximum likelihood classifier on the VPR data set.

CHAPTER 5

CLASSIFICATION USING PAIRWISE IMAGE COMPARISONS

In Chapter 4 we have described a maximum likelihood classifier for bags of features that estimates the distribution of local features in a class of images. Alternatively, one could devise a distance measure between a pair of images and use that for classification (e. g. by using a nearest neighbor classifier). Unlike the Maximum Likelihod classifier, which assumes statistical independence of the local features in an image, a classifier using pairwise comparisons between images would take into account feature co-occurrence. Such a classifier would be implicitly estimating the joint distribution of images over the local features. The disadvantage here is that the number of images is orders of magnitude fewer than the number of local features in them. In other words, the trade-off here is between estimating the marginal distribution from more samples and estimating the joint distribution from fewer samples.

In this chapter we introduce the Hausdorff distance [32], which we use to compare pairs of images (Section 5.1). We then discuss four classification techniques that are based on pairwise comparisons between instances and have been adapted for bags of features. The first is a simple k-nearest neighbor classifier [19] (Section 5.2). The second is a support vector machine classifier that uses a kernel designed to operate on bags of features [5] (Section 5.3). The last two techniques embed bags of features into a space using pairwise distances between them, after which many standard classifiers can be used (Section 5.4). One of them is Multidimensional scaling [11], a well-known embedding method. The other is a less known but a more efficient method called Dissimilarity Space [54].

5.1 Hausdorff Distance

The one-sided Hausdorff distance [32] between two sets of points in a space is defined as

$$h(A, B) = \max_{a \in A} \min_{b \in B} ||a - b||, \tag{5.1}$$

where A and B are the two sets of points, and $|| \cdot ||$ is a norm for points in the sets.

In general, under this formulation $h(A, B) \neq h(B, A)$. To address this, the bi-directional Hausdorff distance [32] is defined as

$$\hat{h}(A, B) = \max(h(A, B), h(B, A)). \tag{5.2}$$

The Hausdorff distance is often used for object detection, where an image is represented by a set of edge points. In our case, we use the Hausdorff distance to compare sets of points in a high-dimensional feature space, rather than in the image plane. Specifically, we use a variation of the Hausdorff distance, known as the Hausdorff average, defined as

$$ha(A, B) = \frac{\sum_{a \in A} \min_{b \in B} \|a - b\|}{|A|}, \tag{5.3}$$

where $|A|$ is the cardinality of A. It can also be made bi-directional by taking the maximum of $ha(A, B)$ and $ha(B, A)$ as in Equation 5.2. The Hausdorff average has been shown to be the most stable variation of the Hausdorff distance under image distortions [68]. However, unlike the original Hausdorff distance, the Hausdorff average is not guaranteed to satisfy the triangle inequality, and therefore is not a metric. This fact becomes relevant in Section 5.4, where we use the Hausdorff distance to embed bags of features into a Euclidean space.

5.2 K-Nearest Neighbor Classifier

Given a distance measure between two bags of features, the simplest classification technique that can be used is k-nearest neighbor (KNN), which is Bayes-optimal in the limit. One disadvantage of this technique is that a query instance needs to be compared to every training instance. This is especially problematic in the case of bags of features, because computing the Hausdorff distance between two sets is expensive. The complexity is $O(nm)$ where n and m are the cardinalities of the two sets of points respectively.

5.2.1 KNN Performance

Performance of the KNN classifier on the FlowCAM and VPR data sets is shown in the graphs in Figures 5.1 and 5.2 respectively. We have tested setting k to 1, 5, 10, 15, and 20 for both data sets. The graphs show results for $k = 10$ for the FlowCAM data set, and $k = 15$ for the VPR data set, which on average have performed the best respectively.

The dimensionality of the SIFT descriptors has been reduced using PCA to 2, 4, 8, 16, 32, and 64 dimensions for the FlowCAM data set, and to 2, 4, 8, and 16 dimensions for the VPR data set. The accuracy of the KNN classifier for each of these cases has been computed using 10-fold cross-validation. The error bars depict one standard deviation of the accuracies of the individual folds. In both graphs the X-axes are scaled for the purpose of display.

Notice that for the FlowCAM data set the difference in the accuracy of the KNN classifier between the original and reduce SIFT descriptors is not significant. However, for the VPR data set reducing the dimensionality of the descriptors actually improves the accuracy. The accuracy of the KNN classifier using the original 128-dimensional SIFT descriptors is $44.40 \pm 2.43\%$. Reducing the dimensionality of the descriptors to

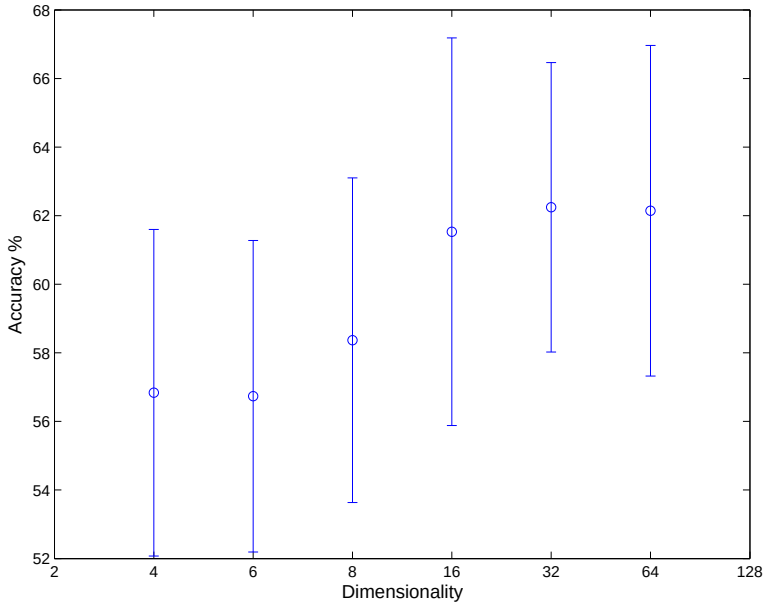


Figure 5.1. Accuracy for the KNN classifier using the Hausdorff Average on the FlowCAM data set with $k = 10$. Results are shown for the original 128-dimensional SIFT features, and for features whose dimensionality has been reduced to 64, 32, 16, 8, 4, and 2 dimensions via PCA.

8 via PCA resulted in the accuracy of $49.07 \pm 3.14\%$. The difference between these two results is statistically significant according to a two-sample t-test ($p < 0.0016$).

5.3 Support Vector Machines for Bags of Features

In this section we discuss classifying images represented by bags of features with support vector machine (SVM) classifiers. To use an SVM one needs to define a kernel, which is a function of two instances that is equivalent to an inner product in some high-dimensional space. While a kernel is not the same as a distance, it is still a pairwise comparison, which in the case of bags of features takes into account feature co-occurrence. Therefore the discussion of SVM classifiers for bags of features is included here. Furthermore, the kernels that are discussed are related to the Hausdorff distance (Section 5.1).

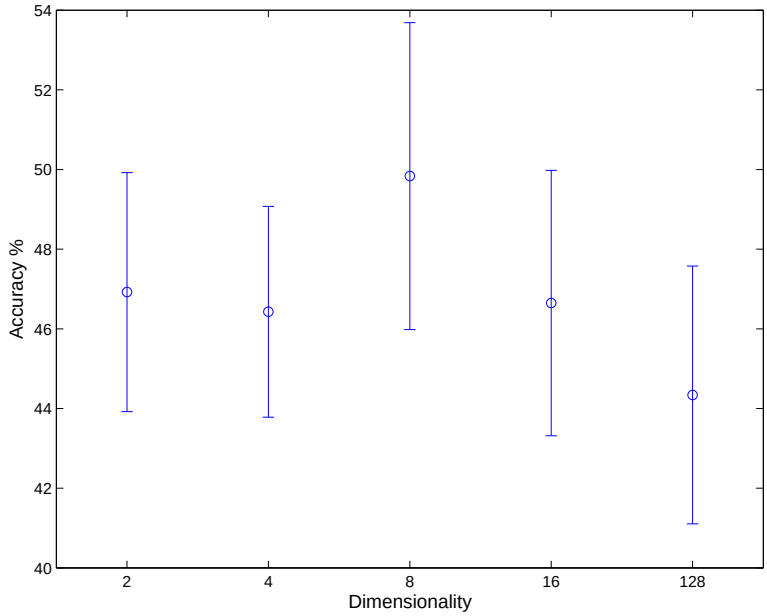


Figure 5.2. Accuracy for the KNN classifier using the Hausdorff Average on the VPR data set with $k = 15$. Results are shown for the original 128-dimensional SIFT features, and for features whose dimensionality has been reduced to 16, 8, 4, and 2 dimensions via PCA.

We start with a brief overview of SVMs. Then we present several support vector machine kernels for bags of features investigated by Blaschko et al. [5]. Specifically, Blaschko et al. [5] present experiments on the VPR data set for three different kernels: the matching kernel, the expected likelihood kernel, and the hybrid kernel combining several of the characteristics of the other two. We describe these kernels in some detail. We also present the experimental results for SVM classifiers using these kernels on the FlowCAM data set.

5.3.1 Support Vector Machines

Support Vector Machines (SVMs) are classifiers that separate two class problems[1] with a maximum margin hyperplane [65]. In the case that the data are not separable, we introduce slack variables, ξ_i, to allow for some incorrectly classified exemplars. The procedure for computing the maximizing hyperplane defined by

$$\langle w, x \rangle + b = 0 \tag{5.4}$$

where $w \in \mathbb{R}^D$, $b \in \mathbb{R}$, is given below. We define $x_1, x_2, \ldots x_m$ to be the exemplars, and $y_i \in \{-1, 1\}$ the class labels, and w is given in terms of its expansion

$$w = \sum_{i=1}^{m} \alpha_i y_i \Phi(x_i) \tag{5.5}$$

where $\Phi(x_i)$ is the projection of x_i via the kernel trick [2]. The value of w is obtained by solving the following quadratic programming problem:

$$\text{minimize}_{\xi,w,b} \qquad \tfrac{1}{2}\langle w, w \rangle + C\tfrac{1}{m}\textstyle\sum_{i=1}^{m} \xi_i \tag{5.6}$$

$$\text{subject to} \quad y_i(\langle w, \Phi(x_i) \rangle + b) \geq 1 - \xi_i, \quad i = 1, \ldots, m \tag{5.7}$$

$$\text{and} \qquad \xi_i \geq 0, \quad i = 1, \ldots, m \tag{5.8}$$

We can in fact maximize the Lagrangian dual, in which the ξ_i disappear:

---

[1] The extension to multi-class problems is explored in many works, many of which are independent of the Support Vector Framework. Some approaches include one vs. rest classification [62], pairwise classification [42], and error correcting output codes [16].

[2] The kernel is a function defined on the original space, which is equivalent to an inner product in a higher dimensional space. The kernel trick refers to computing the inner product in a higher dimensional space implicitly, by evaluating the kernel in the original space.

$$\text{maximize}_{\alpha \in \mathbb{R}^m} \quad \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \langle \Phi(x_i), \Phi(x_j) \rangle \tag{5.9}$$

$$\text{subject to} \qquad 0 \leq \alpha_i \leq \frac{C}{m} \text{for all } i = 1, \ldots, m, \tag{5.10}$$

$$\text{and} \qquad \sum_{i=1}^{m} \alpha_i y_i = 0. \tag{5.11}$$

where each α_i is a Lagrangian multiplier. The decision function itself can then be written as a weighted sum of inner products where the weights correspond to the Lagrangian multipliers

$$f(x) = sgn(\sum_{i=1}^{m} y_i \alpha_i \langle \Phi(x), \Phi(x_i) \rangle + b) \tag{5.12}$$

For additional details on the formulation of the quadratic programming problem, the reader is referred to the book by Schölkopf and Smola [65]. Burges also provides an accessible tutorial introduction to Support Vector Machines [9].

5.3.2 Matching Kernel

The Matching Kernel [78] was proposed to handle images represented by sets of vectors resulting from computing local image descriptors at interest points. In other words it was designed for the bags of features representation. The kernel consists of a *minor kernel*, which is computed between individual vectors, and a function for combining the results of the minor kernel evaluations for the entire set. The function that computes the overall result takes the form

$$k(I, I') = \frac{1}{2}[\hat{k}(I, I') + \hat{k}(I', I)] \tag{5.13}$$

$$\hat{k}(I, I') = \frac{1}{N} \sum_{i=1}^{N} \max_{j=1,\ldots,N'} \phi(x_i, x'_j) \tag{5.14}$$

where I and I' are sets of vectors corresponding to objects, x_i and x'_j are individual vectors in those sets, respectively, N is the number of vectors in I, and $\phi(x_i, x'_j)$ is

the minor kernel. It turns out, however, that the kernel in Equation (5.14) is not positive definite due to the max operation [20], and so it is not a Mercer kernel [76] despite the claim in the original paper. Nevertheless, reported results in [78] and [20] indicate that this technique can be successfully applied to simple object recognition tasks.

Because the matching kernel is not a Mercer kernel, the quadratic optimization is not guaranteed to converge. However, its successful application to object recognition provides empirical evidence that in many cases it does converge. This has led Boughorbel et al. [6] to propose a probabilistic extension to the Mercer property. Given a kernel K with a parameter σ that satisfies certain conditions, Boughorbel et al. derive a bound on the probability that K is positive definite as a function of σ. Specifically they show that if K is the matching kernel and σ is the bandwidth of the minor kernel, then the probability that K is positive definite is higher for smaller σ. Thus the probability that K is positive definite can be bounded arbitrarily by setting an appropriate threshold for σ. The fact that the matching kernel satisfies this relaxed Mercer condition explains why it often works in practice.

Thus far we have not discussed what the choice of a minor kernel should be. This choice is not unconstrained, as can be seen by simply choosing the dot product. For a fixed query point $\mathbf{x} = \{x_1, \ldots, x_n\}$, $\phi(\mathbf{x}, \mathbf{y})$, a function of $\mathbf{y}$, is a hyperplane that passes through the origin. Intuitively, we would like $\max_{j=1,\ldots,N'} \phi(x_i, x'_j)$ to select a value for j that corresponds to a vector close to x_i in that space. Use of a simple dot product, however, will favor points that are infinitely far from x_i.

If we instead modify Equation (5.14) to be

$$\hat{k}(I, I') = \frac{1}{N} \sum_{i=1}^{N} \phi(x_i, x'_{j_i^*}) \tag{5.15}$$

where

$$j_i^* = arg \min_{j=1,\ldots,N'} \|x_i - x'_j\| \tag{5.16}$$

we constrain ourselves to matching points that are similar in the sense that they are close together (we can in fact use a metric in an induced space [8]). However, this technique is implicitly dependent on the choice of the origin, and the result of averaging dot products is difficult to interpret from the perspective of spatial similarity of a set of vectors . Therefore, the minor kernel itself must act as a similarity measure in Equation (5.14). Choosing a radially symmetric kernel that is monotonically decreasing as $\|x_i - x'_j\|$ results in Equations (5.14) and (5.15) being equivalent. One can think of the result of its evaluation as the likelihood that the two vectors match each other in that space. Notice also that there is a clear similarity between Equations (5.15) and (5.16) and the Hausdorff average (Section 5.1).

A simple dot product is an appropriate candidate for the minor kernel in the special case where the data are normalized, which is indeed true for the SIFT features [47]. In this case, the data are constrained to lie on a hyper-sphere and the dot product is the cosine of the angle between two vectors. The cosine of the angle is radially symmetric with respect to a point on the manifold, and we arrive at a radially symmetric, monotonically decreasing similarity metric, though we have little control over the scale at which we compare the data. Without such a geometric constraint, we must take greater care with our choice of kernel.

To interpret explicitly the minor kernel as representing the likelihood of a match between vectors, the kernel takes the shape of a density over the space. If we rely on the feature space, or a transformation of that space, to separate the data, then we require that the kernel have density inversely proportional to a distance metric in the space. A Gaussian RBF kernel is, to a constant factor, a density over the original feature space that has density inversely proportional to distance.

5.3.3 Expected Likelihood Kernel

A general approach for generating a kernel between distributions over observations was outlined by Jebara, Kondor, and Howard [35]. They propose *probability product* kernels of the form

$$k(p, p') = \int p(x)^\rho p'(x)^\rho dx \tag{5.17}$$

where p and p' are distributions that represent the two objects and ρ is a parameter of the family of kernels.

In this section we focus on a special case called the *expected likelihood kernel* [34, 35]

$$k(p, p') = \int p(x)p'(x)dx = E_p[p'(x)] = E_{p'}[p(x)] \tag{5.18}$$

This kernel is unbounded, and favors distributions with low entropy.

Jebara, Kondor, and Howard derive closed form solutions for many parametric forms for $p(x)$ [35]. Of particular interest is that of the Gaussian distribution.

$$\int_{\mathbb{R}^D} p(x)^\rho p'(x)^\rho dx = \frac{1}{((2\pi)^{(2\rho-1)}\rho)^{D/2}} \frac{|\Sigma^\dagger|^{1/2}}{|\Sigma|^{\rho/2}|\Sigma'|^{\rho/2}} e^{-\frac{\rho}{2}(\mu^T\Sigma^{-1}\mu + \mu'^T\Sigma'^{-1}\mu' - \mu^{\dagger T}\Sigma^\dagger\mu^\dagger)} \tag{5.19}$$

where

$$\Sigma^\dagger = (\Sigma^{-1} + \Sigma'^{-1})^{-1} \tag{5.20}$$

and

$$\mu^\dagger = \Sigma^{-1}\mu + \Sigma'^{-1}\mu' \tag{5.21}$$

The expected likelihood kernel applied to two spherical Gaussians of equal variance is equal to

$$k(p, p') = \frac{1}{(4\pi\sigma^2)^{D/2}} e^{-\|\mu'-\mu\|^2/(4\sigma^2)}, \tag{5.22}$$

which is equivalent to the Gaussian RBF kernel to a constant factor [35]. Although one might argue that the expressive capacity of single Gaussians is overly restrictive[3], we see below that the kernel between Gaussians is important in the derivation of more sophisticated results.

Recall non-parametric density estimation from Chapter 4. If we estimate $p(x)$ using the Parzen window method and a Gaussian kernel, and then apply the expected likelihood kernel to it, we achieve the result

$$k(p, p') = \int \left(\frac{1}{N} \sum_{i=1}^{N} \phi(x_i, x) \right) \cdot \left(\frac{1}{N'} \sum_{j=1}^{N'} \phi(x'_j, x) \right) dx \tag{5.23}$$

where $\phi(x_i, x)$ is the Gaussian kernel. Rearranging terms in Equation (5.23) we arrive at

$$k(p, p') = \frac{1}{N} \frac{1}{N'} \sum_{i=1}^{N} \sum_{j=1}^{N'} \int \phi(x_i, x) \cdot \phi(x'_j, x) dx \tag{5.24}$$

Since the inner integral is simply the expected likelihood kernel between Gaussians, the end result is

$$k(p, p') = \frac{1}{N} \frac{1}{N'} \frac{1}{(4\pi\sigma^2)^{D/2}} \sum_{i=1}^{N} \sum_{j=1}^{N'} e^{-\|x'_j - x_i\|^2/(4\sigma^2)} \tag{5.25}$$

when the Gaussians are isotropic and of equal variance [5].

5.3.4 Hybrid Kernel

The expected likelihood kernel between kernel density estimations has a very similar form to the matching kernel with a Gaussian RBF as the minor kernel (Equations (5.13) and (5.25)). Aside from a constant factor, the only difference is that the matching kernel sums the contribution only from the closest match via the max operation, while the expected likelihood kernel between kernel density estimations

[3]For the more restrictive formulation in Equation (5.22) we have no more information than were we to represent the data by their centroid.

sums over every contribution. Density estimation uses a lower variance statistic than the matching kernel and there are no discontinuities introduced as a result of the max operation.

The similarity between the matching kernel and the expected likelihood kernel gives rise to the question of what statistics are appropriate to allow maximum discrimination while maintaining robustness. Robust estimators are those that make use of some subset or weighting of the data to reduce the effect of outliers [26, 30]. One of the most simple techniques for selecting a subset of the data is via order statistics. By including only a certain quantile of data, outliers will fall in the excluded range and the estimator will not be affected. The max operation in the matching kernel is an order statistic that excludes every data point except the closest match. This is an optimal choice in the event that it is assumed that each local feature in one image matches exactly one feature in the other. Alternative statistics include hybrid approaches in which the tradeoff between an estimator based on all the data, or on just a portion of the data, are controlled by a parameter of the estimator [30].

In terms of discrimination, we wish to select an estimator that is robust to values that give little or misleading information about class membership. In general, we wish to choose a statistic that gives a higher similarity value to objects of the same class and a lower similarity value to objects of different classes. This is a data-dependent choice, and without further assumptions about the distribution from which the data are drawn, the statistic used must be chosen experimentally.

There is a certain amount of robustness built into any system that calculates statistics over Gaussian kernel evaluations. Because

$$\lim_{|x_i - x'_j| \to \infty} \phi(x_i, x'_j) = 0, \tag{5.26}$$

outliers will tend to have a limited effect on the summation. However, experimental results show that the choice of estimator does have a significant effect on classification performance.

Because of the superior performance of the matching kernel over the expected likelihood kernel, we define a class of kernels parametrized by the order statistics as follows

$$k_\beta(I, I') = \frac{1}{2}[\hat{k}_\beta(I, I') + \hat{k}_\beta(I', I)] \tag{5.27}$$

$$\hat{k}_\beta(I, I') = \frac{1}{N}\frac{1}{\lceil \beta N' \rceil}\sum_{i=1}^{N}\sum_{\{x'_j | \phi(x_i, x'_j) \geq \phi(x_i, I'_\beta)\}} \phi(x_i, x'_j) \tag{5.28}$$

where $\phi(x_i, I'_\beta)$ is the $\lceil \beta N' \rceil$th largest kernel evaluation ranging over the set of vectors, I'. β represents the fraction of kernel evaluations that will be averaged in the inner loop of the double summation. In the case that $\beta = \frac{1}{N'}$ this is equivalent to the matching kernel, and in the case that $\beta = 1$ the kernel becomes equivalent to the expected likelihood kernel [5].

5.3.5 Performance of SVMs on Bags of Features

The accuracy of the SVM classifiers using the matching kernel and the expected likelihood kernel on the FlowCAM data set is presented in Figure 5.3. Results of the same classifiers on the VPR data sets [5] are shown in Figure 5.4. Notice that on the VPR data set the expected likelihood kernel has performed significantly worse than the matching kernel. On the other hand, both kernels have yielded statistically the same results for a wide range of bandwidth on the FlowCAM data.

This phenomenon can be explained by the fact that VPR images are larger and contain orders of magnitude more SIFT features than the FlowCAM images. Recall the discussion in Section 5.3.4 about the differences between the matching kernel and the expected likelihood kernel. The matching kernel sums the contributions from the closest match via the max operation, while the expected likelihood sums over every contribution. When the number of local features in an image is very small, however,

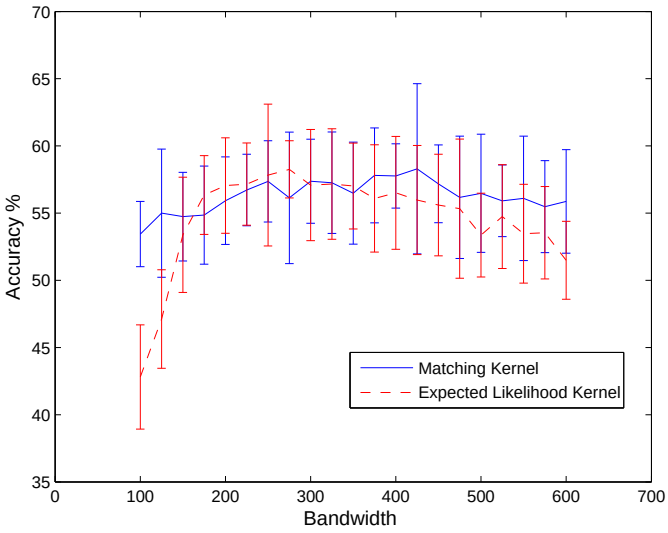


Figure 5.3. Bandwidth, σ, is plotted vs. accuracy on the FlowCAM data set. Results are shown for the matching kernel, and for the expected likelihood kernel.

summing up all the contributions does not differ significantly from taking only the closest match. The difference in accuracy for the VPR and the FlowCAM data sets reflect precisely that.

The results for the hybrid kernel on the VPR data set [5] are shown in Figure 5.5. Notice that it outperforms the matching kernel. On the other hand it does not make sense to test the hybrid kernel on the FlowCAM images because the number of local features per image is so small.

5.4 Embedding Bags of Features

In Sections 5.2 and 5.3 we have discussed KNN and SVM classifiers that operate on bags of features directly. However both of those techniques have a significant drawback: evaluating a kernel or a distance between a pair of bags of features is very expensive computationally. The complexity is quadratic in the number of local features in an image, which needs to be large for accurate classification. The

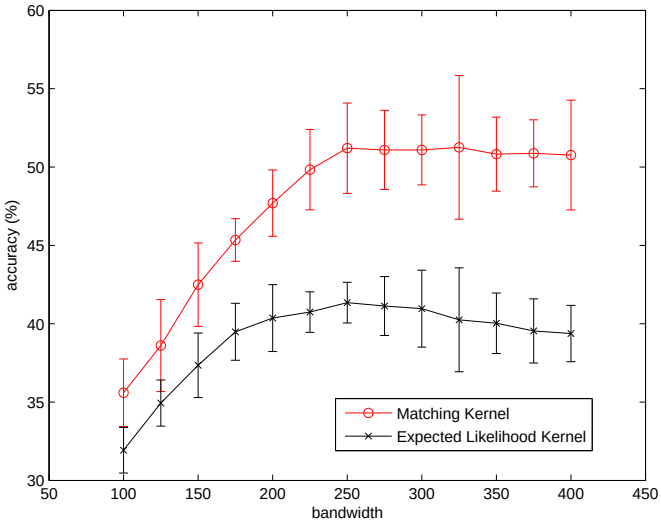


Figure 5.4. Bandwidth, σ, is plotted vs. accuracy on the VPR data set. Results are shown for the matching kernel, and for the expected likelihood kernel [5].

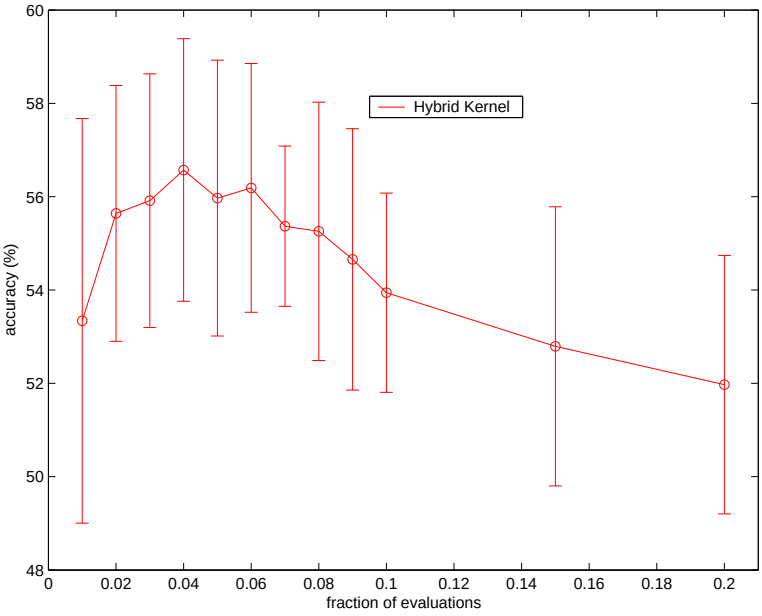


Figure 5.5. Fraction of minor kernel evaluations, β, is plotted vs. accuracy on the VPR data set. The kernel is computed as in equation 5.27.

KNN classifier requires the query to be compared to every training instance, and the SVM requires pairwise comparisons between all training instances during the training phase.

We have mentioned before that method relying on pairwise comparisons between bags of features are implicitly modeling their joint distribution over the space of local features. However, the dimensionality of this space is unknown, and is likely to be very high. Recall also that the number of features is much larger than the number of images. In other words, the number of samples is likely too small relative to the dimensionality of the space to compute an accurate estimate.

An alternative that addresses both of these issues is to use an embedding to map bags of features onto points in a space. First of all, this would give us a larger set of classifiers from which to choose, instead of only KNN and SVM. There are many classification techniques (e.g. decision trees) that operate on vectors, but cannot be adapted easily to using only pairwise distances. Secondly, comparing vectors is significantly faster than comparing bags of features. Finally, an embedding can be used to reduce the dimensionality of the problem, which may give us a better estimate of the joint distribution and improve accuracy. In the following sections we discuss two embedding methods: Multidimensional scaling and Dissimilarity spaces.

5.4.1 Multidimensional Scaling

Multidimensional Scaling (MDS) [11] is a family of algorithms for embedding data represented by pairwise distances or dissimilarities. Given a set of objects, the matrix of pairwise distances or dissimilarities is computed, and used as input for an MDS algorithm. The algorithm assigns coordinates to every object in a space of given dimensionality, trying to preserve the pairwise distances between them.

There are two main approaches to MDS. One is called Classical MDS, which treats the pairwise dissimilarities as Euclidean distances, and solves for the coordinates

analytically. This is done using the eigen decomposition of the inner product matrix, which is in turn computed from the dissimilarity matrix. Classical MDS can also handle non-metric dissimilarities to some extent. The other approach, called Non-classical MDS or Non-metric MDS generates a configuration of points corresponding to the objects by iteratively minimizing a "stress" function. In both cases, however, dissimilarities between all pairs of instances must be computed beforehand, which is computationally expensive, especially in the case the Hausdorff distance.

Non-classical MDS is slower, but it can be extended to produce a configuration of points that preserves the rank orderings rather than the actual distances by defining an appropriate stress function. This can sometimes be useful in cases when the dissimilarity measure is non-metric and the dimensionality of the embedding space is low. Non-classical MDS is often used for the purpose of data visualization, which restricts the dimensionality of the embedding space to 2 or 3.

In our case, the dissimilarity measure is the Hausdorff average, which is non-metric. However, the embedding space is by no means restricted to 2 or 3 dimensions. For our experiments we have used classical MDS, which we describe in greater detail below.

Let O be the set of n objects. Let δ_{rs} be the dissimilarity between objects r and s, where $r, s \in O$. The objective of Classical MDS is to map the objects in O onto points in p-dimensional Euclidean space, such that

$$d_{rs} \approx \delta_{rs}, \tag{5.29}$$

where d_{rs} is the Euclidean distance between the points representing the objects r and s.

Let $\mathbf{x}_r$ be the coordinates of a point in p-dimensional Euclidean space representing object r. Then the Euclidean distance between points r and s is given by

$$d_{rs}^2 = (\mathbf{x}_r - \mathbf{x}_s)^T(\mathbf{x}_r - \mathbf{x}_s) \tag{5.30}$$

It can also be expressed as

$$d_{rs}^2 = \mathbf{x}_r^T\mathbf{x}_r + \mathbf{x}_s^T\mathbf{x}_s + 2\mathbf{x}_r^T\mathbf{x}_s \tag{5.31}$$

Equation 5.31 leads to the following expression for $\mathbf{B}$, the matrix of inner products:

$$\mathbf{B} = -\frac{1}{2}\mathbf{HDH}, \tag{5.32}$$

where $\mathbf{D}$ is a matrix of squared Euclidean distances d_{rs}^2, and $\mathbf{H}$ is the centering matrix,

$$\mathbf{H} = \mathbf{I} - n^{-1}\mathbf{1}\mathbf{1}^T, \tag{5.33}$$

with $\mathbf{1}$ being a column vector of n ones [11]. Matrix $\mathbf{B}$ is necessary, because

$$\mathbf{X}\mathbf{X}^T = \mathbf{B} \tag{5.34}$$

where $\mathbf{X}$ is the $p \times n$ coordinate matrix, which is what we are trying to compute. To recover $\mathbf{X}$, we first write $\mathbf{B}$ in terms of its eigen decomposition:

$$\mathbf{B} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T, \tag{5.35}$$

where $\mathbf{\Lambda}$ is the diagonal matrix of eigenvalues and $\mathbf{V}$ is the matrix of corresponding eigenvectors. The inner product matrix $\mathbf{B}$ is positive semi-definite and of rank p, which means that it has p positive eigenvalues. Then the coordinate matrix $\mathbf{X}$ is given by

$$\mathbf{X} = \mathbf{V}_p\mathbf{\Lambda}_p^{1/2}, \tag{5.36}$$

where $\mathbf{\Lambda}_p^{1/2} = \text{diag}(\lambda_1^{1/2}, ..., \lambda_p^{1/2})$, is the diagonal matrix of the p positive eigenvalues, and $\mathbf{V}_p$ is the matrix of the corresponding p eigenvectors. [11].

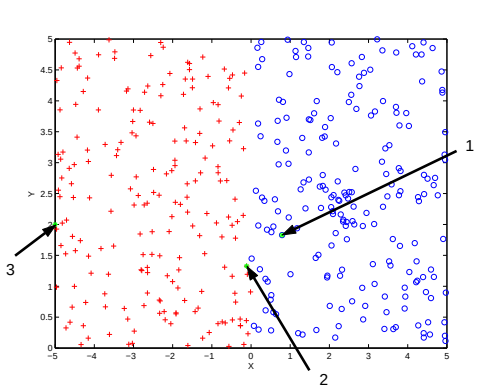
If the dissimilarities are indeed metric distances, then the other $n - p$ eigenvalues are all equal to zero. If the dissimilarities are non-metric, then some of the eigenvalues will be negative. For the purpose of the embedding one can simply ignore them [11]. Furthermore, instead of using all p positive eigenvalues one could use only the first k, resulting in a lower dimensional embedding. Notice that the dimensions of the embedding are decorrelated, and if the dissimilarities are Euclidean then classical MDS is equivalent to the principal components analysis.

5.4.2 Dissimilarity Spaces

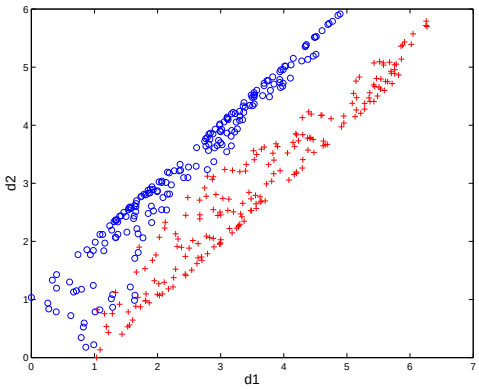
Dissimilarity space embedding [54, 55] is a technique in which each object in a set is represented by a vector of distances to k prototype or reference objects from the same set. Thus, each object is projected into a k-dimensional space. This method is useful for dealing with a set of objects for which we can compute pairwise distances, but have no notion of a coordinate system.

This representation can be viewed as a special case of the Lipshitz embedding [31] in which an element a set S is represented by a vector of distances to k subsets of S. A distance between an object o and a set $A \subset S$ is defined as the distance between o and its nearest neighbor in A. Thus, a dissimilarity space representation is a Lipshitz embedding in which the subsets are singletons. Related variants of the Lipshitz embedding have been proposed [21, 75] that deal exclusively with the problem of efficient nearest neighbor search, rather than with general classification.

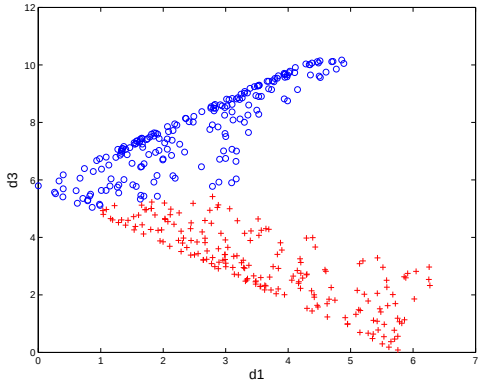
In our actual problem we do not know what the dimensionality of our original space is, and the distance measure we use, the Hausdorff average, is a non-metric. However, for the purpose of illustration let us consider the idealized case when the original space is a 2D Euclidean plane (Figure 5.6(a)). It is easy to see that in that case any point can be localized exactly given its distances to three prototype points, whose coordinates are known, and which are not co-linear. Let P_1, P_2, and P_3 be



(a) Original 2D Space. Two classes are perfectly separable



(b) Dissimilarity Space Using Prototypes 1 and 2. The two classes are still largely separable, but there is a small area of overlap.



(c) Dissimilarity Space Using Prototypes 1 and 3. The two classes are almost perfectly separable.

Figure 5.6. 2D Example.

prototype points. Let the coordinates of P_i be (x_i, y_i). Let Q be a point in the plane with coordinates (x, y). Then if Q is projected into a dissimilarity space defined by P_1, P_2, and P_3, its ith coordinate in the dissimilarity space is computed as follows:

$$\begin{aligned} d_i^2 &= (x - x_i)^2 + (y - y_i)^2 && (5.37) \\ &= x^2 - 2x_i x + y^2 - 2y_i y + x_i^2 + y_i^2, && (5.38) \end{aligned}$$

where $i = 1, 2, 3$. In other words, projecting points from a Euclidean space into a dissimilarity space is a one-to-one quadratic mapping. For instance, this means that linearly separable classes become quadratically separable. This analysis can be generalized for a $(k-1)$-dimensional Euclidean space, for which the number of prototypes must be k.

Notice that any set of k prototypes not lying in the same hyper-plane of dimensionality less than or equal to $k-2$ will uniquely identify the position of any other point up to a translation and a rotation. On the other hand, if a smaller number of prototypes is chosen, then the mapping from the original space into the dissimilarity space will not be one-to-one. This means that some points that are far apart in the original space, may end up close together in the dissimilarity space. It also means that some sets of prototypes may preserve separability better than others (Figure 5.6(b) and 5.6(c)).

Let us illustrate how the choice of prototypes affects separability if their number is less than k. In Figure 5.6(a) we see two classes on a 2D plane, perfectly separable by the straight line $x = 0$. Figure 5.6(b) shows the resulting dissimilarity space, if points 1 and 2 are chosen as prototypes. Most of the points are still separable, but there is a small region region of overlap. In Figure 5.6(c) the dissimilarity space was defined using points 1 and 3. In this case the classes are almost completely separable. In fact, they would have been perfectly separable if the line connecting the two prototypes

in the original space were perpendicular to the linear decision boundary. Of course, they would also have been perfectly separable if we used three prototypes instead of two.

Unfortunately, in our case we do not know the dimensionality of the space in which the bags of features reside. Furthermore, that space is not metric, because the Hausdorff average does not obey the triangle inequality. Therefore, we do not know ahead of time how many prototypes we need. In practice, the number of prototypes represents a trade-off between the classification accuracy and the computational cost. For example, it can be optimized using the "wrapper method" [4]. This is similar to using many other dimensionality reduction techniques, such as the principal components analysis, for which it is often not clear how many dimensions to keep.

An important difference between MDS and dissimilarity space the computational efficiency. MDS requires computing the full dissimilarity matrix, which takes time $O(n^2)$ in the number of instances. Dissimilarity space only requires every instance to be compared to a fixed number of prototypes, which is significantly smaller than the total number of instance. In other words, computing the dissimilarity space representation takes linear time in the number of instances, which is a significant advantage considering that computing the Hausdorff average between a pair of bags of features is quite expensive.

Another advantage of dissimilarity space embedding over MDS is that using MDS for classification is problematic. MDS would need to embed both training and test instances simultaneously, which means that all the test instances need to be available at the time of training. It also means that the training set and the test set would not be independent. Extensions to MDS have been proposed that can incrementally

[4]The wrapper method refers to optimizing the parameters of a classification algorithm using cross-validation on the training set.

add points to a set that has been previously embedded [11]. However, using such a scheme would require additional computation during testing.

The Dissimilarity space embedding, on the other hand is inherently incremental. Once the set of prototypes is selected new instances can be easily embedded at any moment, in constant time. Furthermore, dissimilarity space embedding can be used to devise an any-time classification algorithm. One can first train several classifiers using different size sets of prototypes, thus creating several dissimilarity spaces with different dimensionalities. Then, depending on how much time is available, a query instance is embedded into a dissimilarity space of an appropriate dimensionality, and then classified by the corresponding classifier thus trading accuracy for speed.

5.4.3 Classification Performance in Dissimilarity Space

We have performed classification experiments in dissimilarity space. An SVM classifier was used with a linear kernel. The results for the FlowCAM images are shown in Figure 5.7, and the results for the VPR images are shown in Figure 5.8. We chose 10 random sets of 2, 8, 16, 32, 64, 128, and 256 prototypes, and ran 10-fold cross-validation for each of the sets. The accuracies shown in the graph are the averages over all 10 folds for all 10 sets of prototypes. The error bars, which depict one standard deviation of the individual folds for all 10 sets of prototypes, indicate that the accuracy of this method is not very sensitive to the choice of prototypes.

5.5 Comparison of Classifiers for Bags of Features

5.5.1 Accuracy

Table 5.1 and Table 5.2 summarize performance of the classifiers for bags of features that we have discussed on the FlowCAM and the VPR data sets respectively. Recall that the accuracy of these algorithms has been sampled using different parameter settings, such as the number of dimensions to which the SIFT descriptors were

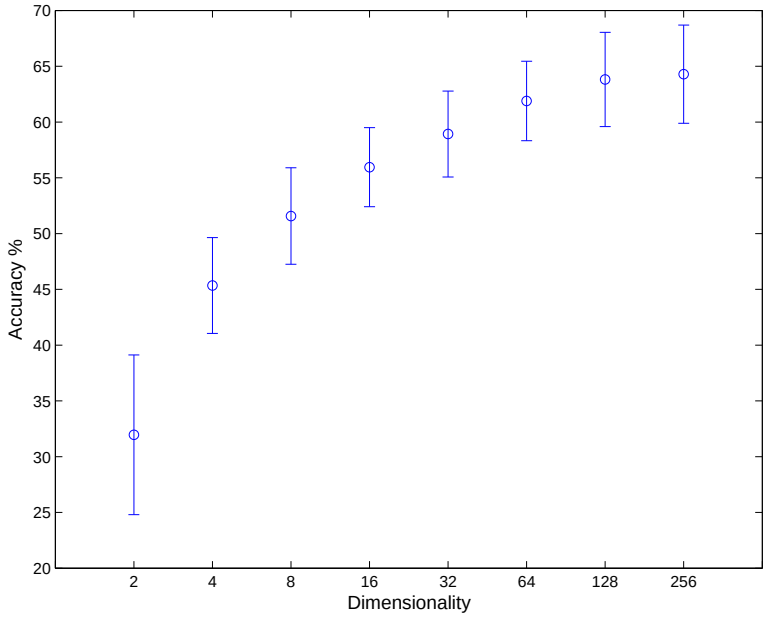


Figure 5.7. Accuracy for the SVM classifier in dissimilarity space on the FlowCAM data set.

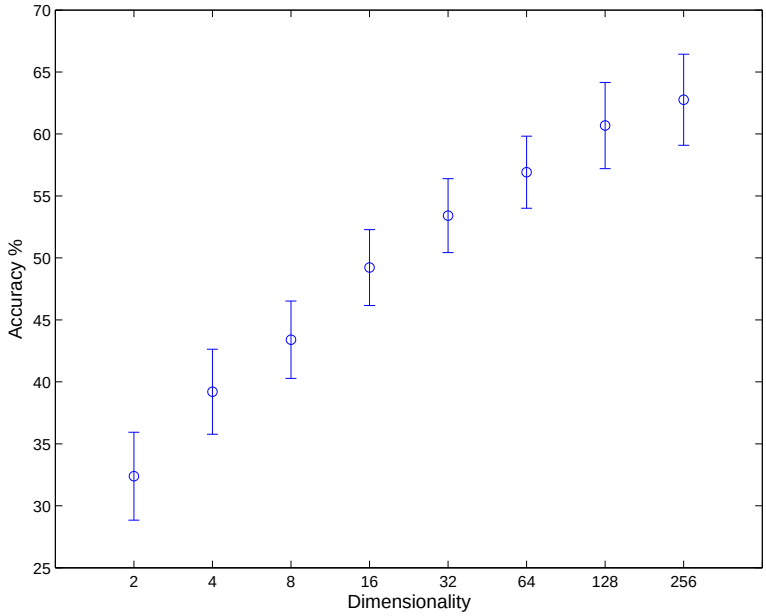


Figure 5.8. Accuracy for the SVM classifier in dissimilarity space on the VPR data set.

Table 5.1. Accuracy of Classifiers for Bags of Features for FlowCAM Data Set

Method	Accuracy %
Maximum Likelihood	59.49 ± 6.36
KNN	62.45 ± 4.27
SVM with Matching Kernel	58.26 ± 2.13
SVM in Dissimilarity Space	64.29 ± 4.40

reduced via PCA or the number of prototypes used to construct a dissimilarity space. For each method, the tables show its best accuracy for the parameters that have been tried.

On the FlowCAM data set an SVM operating on the dissimilarity space has performed better than an SVM using the matching kernel operating on bags of features directly. The statistical significance in this and all subsequent cases was determined by using a two sample t-test. In this case it resulted in $p < 0.0219$. However there was no significant difference in performance between an SVM in a dissimilarity space and a simple KNN classifier ($p < 0.1874$).

The situation with the VPR data set is very different. The two approaches that use embedding have performed significantly better than all the others. The statistical significance of the difference in performance between an SVM classifier operating on the dissimilarity space and an SVM using the hybrid kernel operating on the bags of features directly was also determined by performing a two sample t-test resulting in $p < 0.0041$. In this case an SVM classifier in an embedding space produced by Multidimensional scaling has performed the best, with an SVM in Dissimilarity space a close second. The difference in accuracy between the two winners is not statistically significant ($p < 0.1971$).

5.5.2 Computational Complexity

Aside from classification accuracy, the computational efficiency of the methods is an important consideration. From this point of view using the dissimilarity space

Table 5.2. Accuracy of Classifiers for Bags of Features for VPR Data Set

Method	Accuracy %
Maximum Likelihood	54.84 ± 3.59
KNN	49.84 ± 3.85
SVM with Hybrid Kernel	56.60 ± 2.80
SVM in Dissimilarity Space	62.77 ± 3.67

embedding is the best approach. We now examine the computational complexity of every method to show why that is the case.

- The Maximum likelihood classifier requires a very expensive training phase, in which the bandwidth of the kernel for the non-parametric density estimate is optimized. The complexity of that is quadratic in the total number of local features in all the images in the training set. In addition, during classification, every feature in the query needs to be compared to every feature in the training set.

- The k-nearest neighbor classifier requires no training at all. However, during classification, the query must be compared to every training instance. In other words, the complexity of the classification is the same as that for the Maximum likelihood classifier.

- The training of SVM classifier using a kernel operating directly on bags of features, such as the matching kernel, is also expensive. Its complexity is quadratic in the number of training instances. Also computing a kernel between two instances is quadratic in the number of features in each of the instances. However, the classification phase of this classifier is more efficient than that of the Maximum likelihood classifier, because the kernel only needs to be computed between the query and the support vectors, rather than between the query and all training instances.

Table 5.3. Computational Complexity of Classifiers for Bags of Features

Method	Training Complexity in the Size of Training Set	Classification Complexity in the Size of Training Set
Maximum Likelihood	Quadratic	Linear
KNN	Constant	Linear
SVM on Bags of Features	Quadratic	Linear or better
Dissimilarity Space	Linear	Constant

- Dissimilarity Space embedding offers the same advantage of converting the bags of features into vectors as MDS. However, as we have already mentioned, it is significantly more efficient. The number of prototypes is chosen ahead of time, and is therefore constant. Thus the complexity of embedding the training instances is linear. Furthermore, embedding each query instance takes a constant time in the size of the training set, because it only needs to be compared to the prototypes.

The computational complexity of the training and classification phases of the classifiers is summarized in Table 5.3. Note that the classification complexity of an SVM operating directly on the bags of features is listed as "Linear or better", because the kernel only needs to be computed between the query and the support vectors, whose number is usually smaller than the total number of training instances. However, for some kernels the number of the support vectors may actually grow with the number of training instances, and in the worst case it will grow linearly. Note also that the complexity for the classification phase of the MDS approach is not listed, because all the query instances are embedded at the same time as the training instances. From this table we see that from the computational efficiency point of view using the dissimilarity space embedding to classify bags of features is preferable.

5.5.3 Theoretical Soundness

It is worth noting that using an SVM classifier on in the dissimilarity space is preferable from the theoretical point of view to using an SVM directly on the bags of features via the matching or the hybrid kernel. We have already mentioned in Section 5.3.2 that the matching kernel is non-Mercer, and consequently neither is the hybrid kernel. While the matching kernel has been shown to satisfy the probabilistic Mercer condition [6], meaning that the convex optimization has a high probability of converging to the global minimum, the convergence is not guaranteed. On the other hand, if one were to use an SVM in a Euclidean embedding space, then there would be a number of standard Mercer kernels from which to choose (e. g. polynomial, RBF, etc.). For such kernels the convex optimization is guaranteed to converge to the global minimum.

CHAPTER 6

COMBINING BAGS OF FEATURES WITH FEATURE VECTORS

Despite the advantages of bags of features, feature vectors are still useful, especially in applications where at least a rough segmentation of the object of interest is available. Due to the fundamental difference in how the two representations are computed, we expect that they would provide different kinds of information. For example, while the bags of features describe texture, they differ from global texture descriptors because the support over which texture is computed varies. Furthermore, local features do not capture shape information, so there is even more reason to expect that shape descriptors would capture a different kind of information about the object.

We expect classifiers that use bags of features will commit errors that differ from those of classifiers based on feature vectors [45]. Thus a classification system that made use of both representations would likely perform better than a classifier based on either type alone. In this chapter we first explore approaches to image classification that use feature vectors. We then discuss three methods that attempt to combine both representations into a single classification system, and compare their performance.

6.1 Feature Vector Representation of Images

In this work, we define a *feature vector* simply as a fixed size collection of ordered scalar values representing an image. The most important aspect of this definition is that components of a feature vector are directly comparable from image to image.

Feature vectors provide a convenient and compact representation that is compatible with many standard classification techniques.

This convenience may come at a cost, however. In particular, if an image measurement represents a property that is not easily measurable in every image, then the value of a particular component may be meaningless in some images. For example, a scalar value representing the eccentricity of a plankton is not at all robust to major occlusions of the organism. In particular, because our images are captured automatically by a camera and often suffer major occlusions, shape features may be meaningless for a considerable portion of our images. This problem is especially evident in the VPR image set. Nevertheless, on average there is still valuable information contained in these features.

6.1.1 Segmentation.

Most of the features we compute require a figure-ground segmentation of the object of interest. Fortunately, in our data sets an image often does contain a single object, although sometimes several organisms or particles are present in a single image. We have also found that a simple global bimodal segmentation is usually effective for separating the organism from the background, which tends to be significantly brighter than the object in the FlowCAM images, and darker than the object in the VPR images [4, 45]. We use expectation-maximization (EM) to fit two Gaussians to the histogram of gray values for a given image [14]. The Bayesian decision boundary between the two Gaussians defines the cut point between foreground and background. Subsequently, morphological hole filling [71] is used to capture the stray pixels inside the object that have intensity similar to that of the background. Figure 6.1 shows a sample image (Figure 6.1(a)) and the corresponding segmentation (Figure 6.1(b)).

6.1.2 Feature Vectors for the FlowCAM Data Set

We computed the following 225 features for the FlowCAM data set:

(a) Image (b) Segmentation

Figure 6.1. An example of image segmentation.

- *Simple Shape*: The 8 features in this category include, Area, Perimeter, Compactness, Eigenratio, Eccentricity, Standard deviation of area across connected components, Convexity, and Rectangularity.

- *Moment Invariants*: These 7 features are the logs of first seven moment invariants as proposed by Hu. The moments are computed over the binary image, so they are shape descriptors.

- *Granulometric Features*: To compute the granulometric features [48] a series of morphological erosions and dilations with structure elements of different size are preformed. Then the differences in area between the object before and after the erosion or dilation is recorded. We used four window sizes: 3×3, 5×5, 7×7, and 9×9, and two types of structure elements: the square and the diamond. This resulted in 16 granulometric features.

- *Moments of Intensity Histogram*: The 5 features here include mean, standard deviation, skewness, and kurtosis of the histogram of gray scale values and the entropy of the normalized histogram.

- *Local Binary Patterns* are gray-scale and rotation invariant texture operators [53]. Local binary patterns are computed from $P > 1$ image pixels, where each sampled image point p is calculated at a radius $R(R > 0)$ according to the formula

$(-R\sin(2\pi p/P), R\cos(2\pi p/P))$. Each pixel in the image then lies at the center of a circle of gray value samples. The intensity values of the samples are thresholded by the value at the center, and are encoded into a binary number. These codes are accumulated into a histogram, which is normalized by the object's size, and is treated as a feature vector. The segmentation is used to mask contributions to the histogram to only those pixels interior to the cell. In these experiments, we calculated local binary patterns for $R = \{1, 2, 3\}$ and $P = 8 * R$. This resulted in 54 features.

- *Co-occurrence Features*: These 140 texture measures have been derived from co-occurrence matrices. The entry (i, j) of the co-occurrence matrix $P_{d_x d_y}(i, j)$ is the number of occurrences of the pair of gray levels i and j at horizontal step d_x and vertical step d_y pixels. In our work, we calculate the energy, inertia, entropy, and homogeneity of the co-occurrence matrices [28].

6.1.3 Feature Vectors for the VPR Data Set

For the VPR data set we have used a total of 179 features.

- *Simple Shape features*: area, perimeter, and compactness (perimeter squared over area). These 3 features were computed from the segmentation.
- *Intensity Mean and Standard Deviation.*
- *Local Binary Patterns.* Same 54 features as for the FlowCAM data set.
- *Shape Index Histograms* are histograms computed using the isophote and the flowline curvatures of the intensity surface [60]. The total number of elements in this descriptor is 120.

We emphasize that the LBP and shape index histograms are logical components of a feature vector. Note that while the bags of features also capture texture measurements of the image, they are computed at specific interest points which may be

different for each image. LBPs and shape index histograms, on the other hand, are computed in a consistent manner across images so that every element of the resulting feature vector has the same meaning in each image. That is, the LBPs and shape index histograms are used to measure the same properties of each organism.

6.2 Stacking

Ensemble methods are learning algorithms that have been shown to improve performance by combining the outputs of multiple component classifiers. Ensemble methods for classification have been shown to have better accuracy than the component classifiers if the component classifiers are accurate and diverse [15]. An accurate classifier is one that outperforms random guessing, and diverse classifiers are those that commit independent errors. Even if the errors are not entirely independent using an ensemble can lead to an improvement in accuracy. One of the main areas of ensemble research is how to induce independence between the component classifiers. Independence of errors can be achieved by manipulating the training set, manipulating the feature set, or injecting randomness in the learning algorithm.

It is reasonable to expect that in general the bags of features and the feature vectors are likely to capture different types of information contained in images. If that is true, then classifiers using the two types of features respectively are likely to commit independent errors. To illustrate this point consider the confusion matrices showing the performance for an SVM classifier that uses feature vectors and the Maximum likelihood classifier that uses bags of features on VPR image set (Tables 6.1 and 6.2). By simply comparing the matrices we see that the two classifiers commit different errors that are likely to be independent.

In Section 6.5 we show the accuracy of methods that use both representations on the FlowCAM and the VPR data sets. These methods achieve an improvement on the VPR data set compared to using each representation alone, supporting our

Table 6.1. Confusion matrix for SVM with feature vectors on VPR data set.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	**21**	5	3	1	1	2	0	1	2	96	0	0	0	1
2	4	**33**	0	0	6	3	0	1	3	30	0	5	1	0
3	3	2	**22**	0	0	0	11	1	1	54	3	0	1	2
4	3	2	0	**10**	8	1	0	0	1	2	0	7	0	0
5	1	4	0	2	**94**	2	0	0	0	5	0	10	13	0
6	1	2	1	0	11	**4**	1	0	0	39	1	1	7	0
7	0	3	11	0	0	1	**29**	0	3	70	21	0	1	3
8	3	3	2	0	0	0	0	**83**	2	1	1	1	0	1
9	5	0	1	2	2	0	3	0	**89**	24	3	1	0	3
10	12	11	10	0	1	2	21	3	6	**339**	16	0	4	8
11	0	0	3	0	0	0	16	1	2	18	**67**	0	0	1
12	1	13	1	5	11	0	1	6	3	5	0	**155**	1	0
13	3	0	0	0	19	5	0	0	0	23	0	1	**30**	0
14	4	2	2	0	0	0	2	1	17	33	5	1	1	**10**

expectation of error independence. However, on the FlowCAM data set combining the representations produced no significant improvement, implying that in that particular case the errors were not independent. The reason for that is the peculiarity of the data set, which consists of very small images. As a result local features essentially capture the same information as the global texture descriptors (Section 6.5).

Many ensemble techniques use a fixed strategy for combining the outputs of their component classifiers, such as simple majority vote or a weighted vote. We call this strategy fixed, because it is decided upon ahead of time. On the other hand, an ensemble method called *stacked generalization* or *stacking* [81] uses another classifier to generalize over the space of outputs of the component classifiers (Figure 6.2). In this case the component classifiers are called the base-level classifiers, and the "meta-classifier" is called the top-level classifier, forming a two-level hierarchy. Thus, instead of using a pre-determined function to combine the outputs of the base-level classifiers, this function is learned by the top-level classifier.

Boosting [24] is another widely used ensemble technique in which each subsequent classifier is trained on the instances that are likely to be misclassified by the previous

Table 6.2. Confusion matrix for the Maximum likelihood classifier for bags of features on VPR data set

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	**18**	0	0	0	17	0	0	0	0	41	0	56	1	0
2	2	**13**	0	0	8	0	1	0	1	8	0	50	1	1
3	1	0	**47**	2	4	0	7	0	7	24	0	7	0	0
4	0	0	0	**0**	15	1	0	0	0	2	0	16	0	0
5	0	0	0	0	**87**	0	0	0	0	2	0	39	2	0
6	1	0	1	0	23	**6**	0	0	0	12	1	20	4	0
7	3	0	10	0	6	0	**74**	0	4	27	2	10	3	2
8	0	0	0	0	0	0	0	**89**	0	0	0	8	0	0
9	4	0	0	0	6	1	3	0	**57**	29	3	24	0	6
10	7	1	5	1	29	2	14	0	3	**274**	0	79	16	0
11	1	0	1	0	0	0	3	0	1	12	**88**	1	0	1
12	0	0	0	0	7	0	0	21	0	1	0	**173**	0	0
13	0	0	0	0	42	0	0	0	0	3	0	28	**8**	0
14	1	1	1	0	3	0	0	0	8	21	1	28	0	**14**

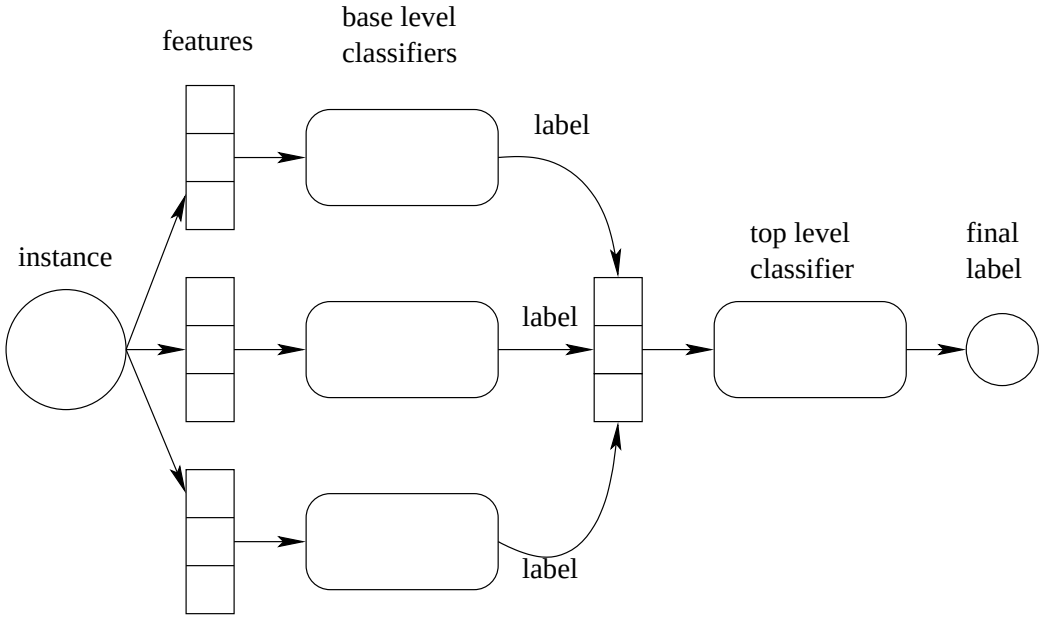


Figure 6.2. Stacking.

classifiers. In other words, boosting manipulates the training sets of the component classifiers to induce the independence of errors. In our case, however, we expect that the fact that the component classifiers use different feature types is sufficient to achieve the independence of errors. This implies that manipulating the training sets, which adds computational cost during the training phase, is necessary, and stacking is preferable to boosting.

We discuss here two main variations of stacking. In the first, the input to the top-level classifier is a concatenation of class labels produced by each of the base-level classifiers. In the second variation, each base-level classifier outputs a posterior distribution over class labels, rather than a single label. Distributions from the base-level classifiers are concatenated and used as input to the top-level classifier. Stacking with probability distributions, in essence, trains on an estimate of classification confidence from the base-level classifier. Any classifier can be used at the base level if we only require a single category label, but stacking with probability distributions restricts us to classifiers that output distributions over class labels. The choice of the top-level classifier is not restricted in any way.

In our case we use stacking to combine bags of features and feature vectors, and thus we only have two base-level classifiers (Figure 6.3) [45]. We use the Maximum likelihood classifier for the bags of features (Chapter 4), which naturally outputs a posterior distribution. For the feature vectors we use an SVM. Unfortunately most implementations of SVMs do not by default output a posterior distribution. To get around this problem we have modified the SVM implementation in the WEKA toolkit [80] to output the relative confidences for the predicted class labels. While these are not true posterior probabilities, they nevertheless indicate the SVM's confidence in its predictions. We then use another SVM as the top-level classifier.

The performance of stacking on FlowCAM and VPR data sets is shown in Figures 6.4 and 6.5 respectively. In our experiments we trained the Maximum likelihood

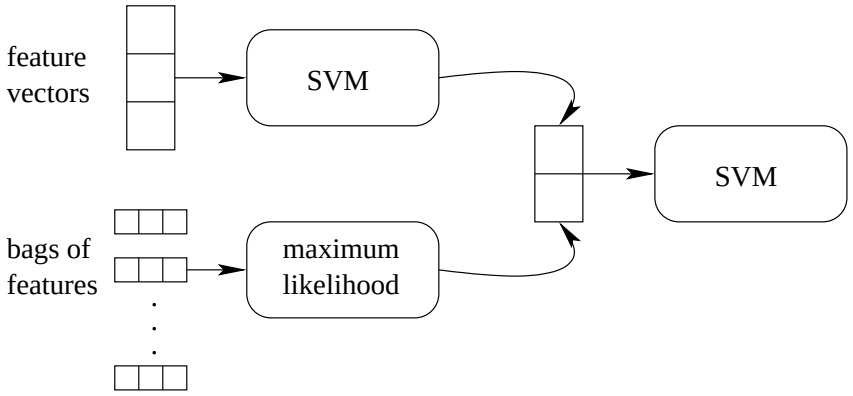


Figure 6.3. Using Stacking to Combine Bags of Features and Feature Vectors.

classifier on the bags of SIFT features whose dimensionality was reduced to 2, 4, 8, and 16 dimensions via PCA, just as in Section 4.3. Each of the resulting Maximum likelihood classifiers was combined with an SVM using the feature vectors via stacking, and the corresponding accuracies are shown in the figures.

For the FlowCAM data set using stacking actually resulted in a performance worse on average than that of using the feature vectors alone. The accuracy achieved by stacking was 71.63 $\pm$ 4.04% (Table 6.3), while the accuracy of the SVM using the feature vectors was 72.87 $\pm$ 3.42% [4]. The difference is not statistically significant ($p < 0.821$), so we can conclude that in this case stacking produced no improvement.

On the other hand, for the VPR data set stacking yielded a significant improvement in accuracy compared to the base-level classifiers. The accuracy of stacking was 67.34 $\pm$ 3.22% [45] compared to the best accuracy using the bags of features alone (56.60 $\pm$ 2.80% using an SVM with the hybrid kernel [5]), and the accuracy using the feature vectors (52.1 $\pm$ 3% [45]). The improvement over the SVM with the hybrid kernel is statistically significant ($p < 2.57 \times 10^{-7}$).

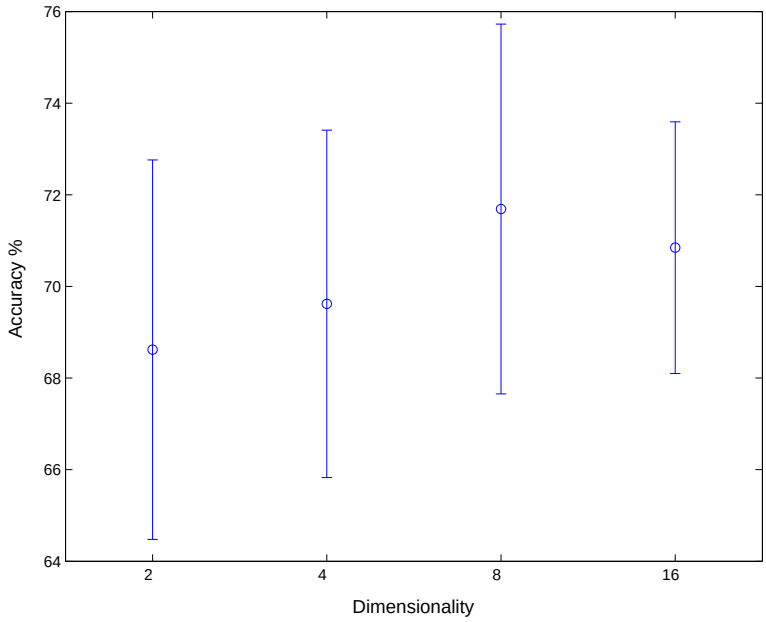


Figure 6.4. Stacking results on FlowCAM data set.

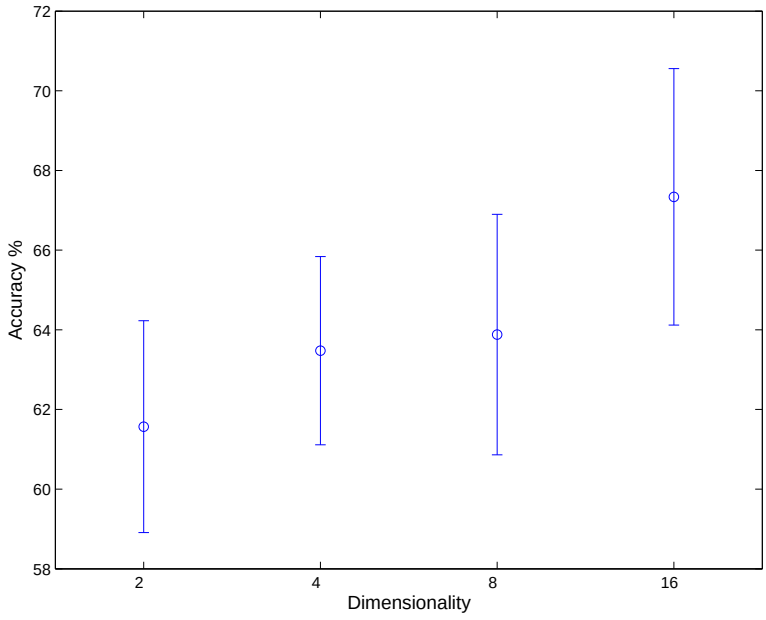


Figure 6.5. Stacking results on VPR data set

6.3 Combination of Kernels

Blaschko et al. [5] propose a different method for combining the feature vectors and the bags of features. They present a support vector machine classifier that operates on both representations simultaneously. They utilize the fact that the set of positive definite kernels is closed under addition and multiplication. In other words the sum of two positive definite kernels is a positive definite kernel, and so is the product.

Blaschko et al. [5] have experimented with several different combinations of kernels for bags of features (Section 5.3) with a Gaussian RBF kernel operating on the feature vectors. They have achieved the best result using a polynomial combination of kernels:

$$k_{poly}(x_1, x_2) = (k_1(x_1, x_2) + (1 - \alpha)) \cdot (k_2(x_1, x_2) + \alpha)\,, \tag{6.1}$$

where k_1 is a kernel for bags of features, k_2 is a kernel for feature vectors, and α is a parameter that controls the relative contribution of each kernel.

Recall that when we combined the two representations via stacking, the base-level classifiers that used the bags of features and the feature vectors respectively were trained separately. Each classifier was trained to achieve the best possible accuracy on its own, which does not necessarily yield the best accuracy for the combination. On the other hand, Blaschko et al. combine both representations in a single SVM via a combination of kernels. Thus one classifier is trained using both representations simultaneously attempting to achieve the best overall accuracy directly.

The results for this approach are show in Figures 6.6 and 6.7 for the FlowCAM and the VPR image sets respectively. The accuracy is plotted as a function of the bandwidth of the RBF kernel for the feature vector, and the bandwidth of the minor kernel of the matching kernel for the bags of features. Notice that the accuracy on the VPR data set appears to be more stable with respect to the parameters than that for the FlowCAM data set.

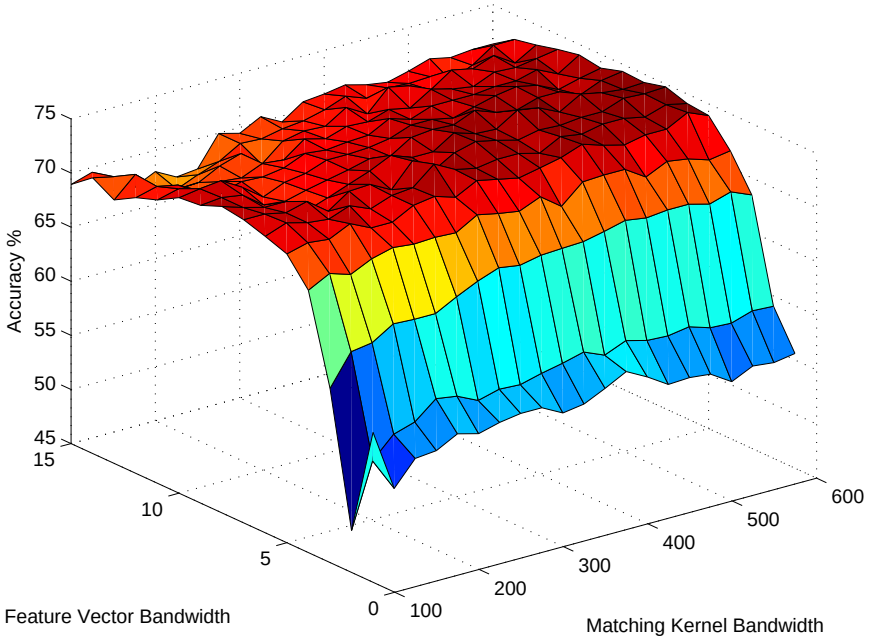


Figure 6.6. Classification accuracies for a polynomial combination of two kernels on FlowCAM data set.

The highest average accuracy on the FlowCAM images was $74.84 \pm 2.68\%$. This is not significantly better than the accuracy for an SVM on the feature vectors alone, which was 72.87 ± 3.42 ($p < 0.0611$), which leads us to the conclusion that combining the two representations is not worth the additional computation in the case of the FlowCAM data set.

On the VPR images the best accuracy for the polynomial combination of kernels was $71.90 \pm 3.2\%$. This is a significant improvement compared to the best result achieved by an SVM on bags of features ($56.6 \pm 2.80\%$ for the hybrid kernel), and the accuracy of an SVM on feature vectors ($52.1 \pm 3\%$). The two-sample t-test comparing the accuracy for the combination and the SVM on bags of features alone yielded $p < 10^{-9}$. The combination of kernels has also resulted in a statistically significant improvement over stacking ($p < 5.3 \times 10^{-4}$).

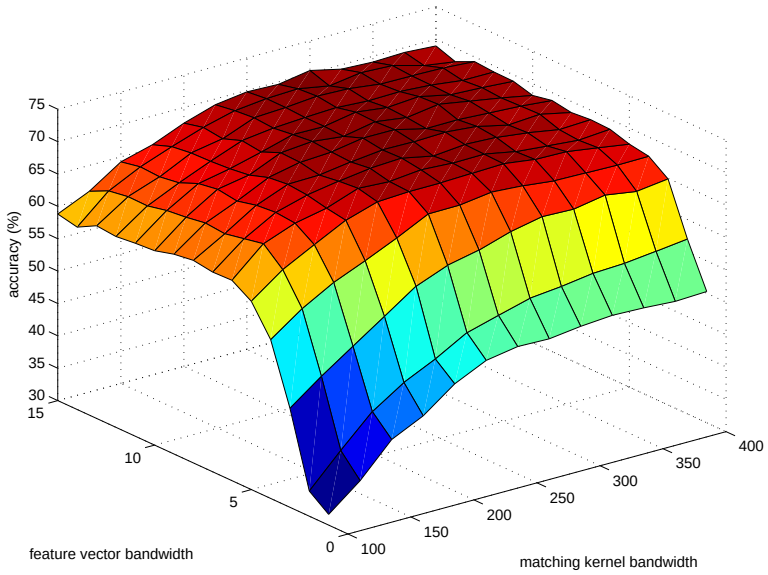


Figure 6.7. Classification accuracies for a polynomial combination of two kernels on VPR data set [5].

6.4 Dissimilarity Space

Embedding bags of features into a space (Section 5.4) also provides an opportunity to combine them with feature vectors in a very natural fashion. Once bags of features are represented as vectors, they can be simply concatenated with other features to form a higher-dimensional feature space. This approach has the same advantage as the combination of kernels, in that the training occurs using both representations simultaneously. However using an embedding also has the advantage of speed resulting from operating on vectors rather than bags of features.

In our experiments we have concatenated the vector representations of the bags of features produced by mapping them onto a dissimilarity space with the feature vectors, and trained an SVM classifier on the resulting space. The results for the FlowCAM and the VPR image sets are shown in Figures 6.8 and 6.9. The accuracy is plotted as a function of the dimensionality of the dissimilarity space (the number of prototypes). The best accuracy on the FlowCAM images was $73.19 \pm 3.05\%$. On

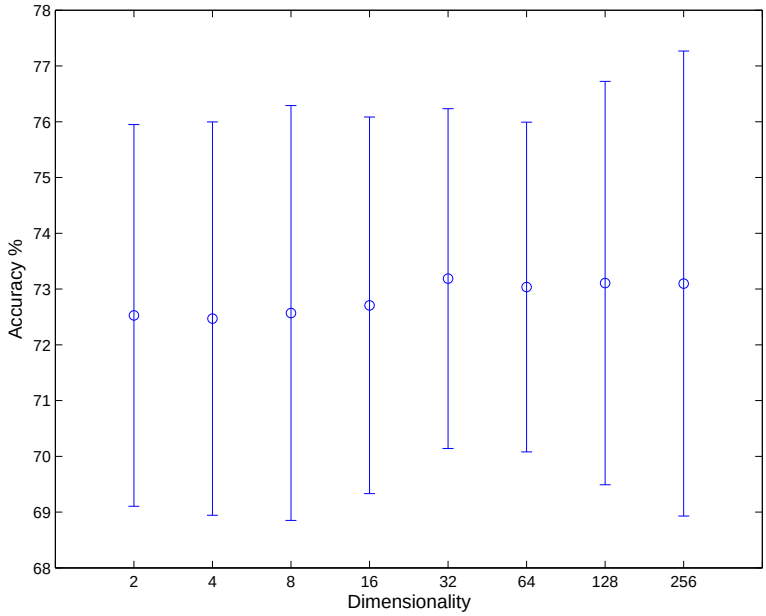


Figure 6.8. Accuracy of an SVM using the dissimilarity space representation concatenated with the feature vector on FlowCAM data set.

the VPR images the best accuracy was $68.25 \pm 3.44\%$. Notice that the accuracy on the FlowCAM data set does not change with the number of prototypes, suggesting that the dissimilarity space representation adds no information to the feature vectors. However on the VPR data set there is a clear trend of the accuracy increasing with the number of prototypes, indicating that using both the feature vectors and the bags of features together is beneficial.

6.5 Summary of the Results.

In this section we analyze the relative advantages of methods for combining bags of features with feature vectors discussed above. All the results for both our data sets are summarized in Tables 6.3 and 6.4. The rows of the tables are separated into three groups. The top group are the accuracies for the classifiers operating on bags of features. The middle group consisting of one row contains the accuracy for the

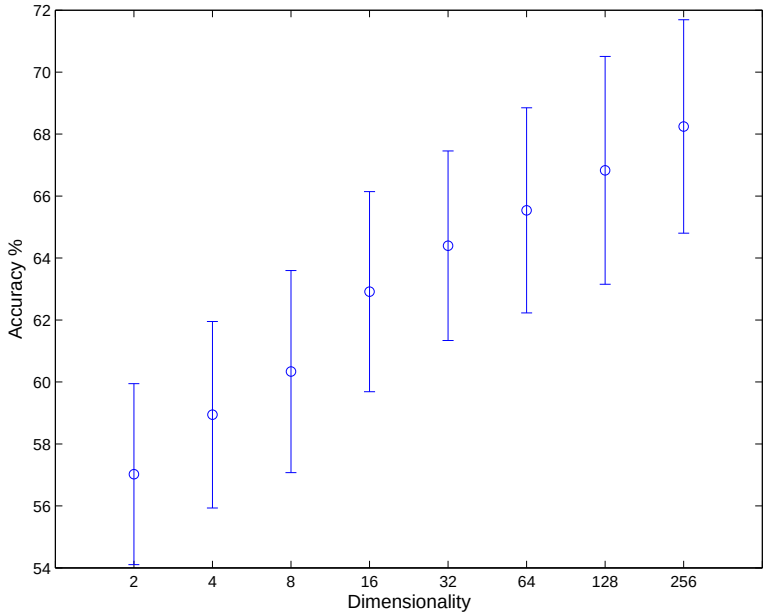


Figure 6.9. Accuracy of an SVM using the dissimilarity space representation concatenated with the feature vector on VPR data set.

SVM classifier on feature vectors. The bottom group contains the accuracies for the methods that combine the bags of features and the feature vectors.

For the FlowCAM data set, the difference between different combination methods is not statistically significant, according to two-sample t-tests. For example the t-test comparing stacking and the combination of kernels, whose average accuracies differ the most, resulted in $p < 0.548$. More importantly, for this data set set the increase in accuracy resulting from combining the two representations compared to using the feature vectors alone is not statistically significant. This suggests that combining the two representations for images that produce very few local features, such as the FlowCAM images, may not be worth the additional computational cost.

Furthermore, these results seem to indicate that we are observing a "ceiling effect" on the FlowCAM data set, meaning that we have achieved the best possible accuracy for these representations. Further improvement probably requires either designing domain-specific features or changing the labeling scheme. For example, categories

Table 6.3. Results Summary for FlowCAM Data Set

Method	Accuracy %
Maximum Likelihood	59.49 ± 6.36
KNN	62.45 ± 4.27
SVM with Matching Kernel	58.26 ± 2.13
MDS and SVM	63.61 ± 5.48
Dissimilarity Space and SVM	64.29 ± 4.40
SVM on Feature Vectors	72.87 ± 3.42
Stacking	71.63 ± 4.04
Combination of Kernels	74.84 ± 2.68
Diss. Space and Feature Vectors	73.19 ± 3.05

of organisms that are related taxonomically and are similar morphologically can be collapsed into larger categories.

For the VPR data set, the difference in accuracy between stacking and concatenating the dissimilarity space representation with the feature vectors was not statistically significant ($p < 0.5858$). However, the difference in performance between the concatenation and the combination of kernels was statistically significant ($p < 0.0243$). While the combination of kernels resulted in a higher accuracy, it is much less efficient than using the dissimilarity space. Recall that the time complexity of training an SVM directly on the bags of features is $O(n^2)$, while the complexity of mapping the instances onto a dissimilarity space is $O(n)$, where n is the size of the training set. Given that the absolute difference in the average accuracy of the two methods on the VPR data set is less than 4%, using the dissimilarity space for its efficiency may be a reasonable trade-off.

Note also that for the VPR data set using both the bags of features and the feature vectors results in a significant increase in accuracy. The reason that combining the two representations worked better for the VPR images than for the FlowCAM images is that the latter are generally much smaller. The FlowCAM captures photos of very small organisms at relatively low magnification. As a result many of the organisms

Table 6.4. Results Summary for VPR Data Set

Method	Accuracy %
Maximum Likelihood	54.84 ± 3.59
KNN	49.84 ± 3.85
SVM with Hybrid Kernel	56.60 ± 2.80
MDS and SVM	63.67 ± 3.20
Dissimilarity Space and SVM	62.77 ± 3.67
SVM on Feature Vectors	52.10 ± 3.00
Stacking	67.34 ± 3.22
Combination of Kernels	71.90 ± 3.20
Diss. Space and Feature Vectors	68.25 ± 3.44

have a diameter of less than 50 pixels in the image, and produce very few, or in some cases even no SIFT features [1]. This means that the information captured by the SIFT descriptors is not very different from that captured by the global texture descriptors, such as the local binary patterns or the shape index. Therefore adding the bags of features to the feature vectors does not introduce a sufficient amount of new information for a significant increase in classification accuracy.

The VPR images and the organisms they depict, on the other hand, are generally larger. Consequently they often produce hundreds of SIFT features. In this case the local SIFT descriptors preserve some of the information discarded by the histogram-based global texture descriptors, such as the local binary patterns and the shape index. Thus combining the bags of SIFT features and the feature vectors in the case of the VPR image set has produced a significant improvement in accuracy.

[1]Approximately 5% of the images in the FlowCAM data set produce no SIFT features. If the Maximum likelihood classifier is used, then an instance that contains no local features is assigned a random label, meaning that it is usually classified incorrectly. When we compute the Hausdorff distance between two sets of local features we define it to be 0 if both sets are empty, and ∞ if one set is empty and the other is not. In the FlowCAM data set most of the instances that produce no SIFT features belong to the same category, and are classified correctly by the classifiers using the pairwise distances.

CHAPTER 7

CONCLUSIONS

In this dissertation we have discussed two different image representations: bags of local features and feature vectors. We have also presented several image classification methods that use the bags of features, and several approaches that combine the two representations. These methods have been tested on two sets of real images of phyto- and zoo-plankton. This work is part of an on-going research project to develop software tools for marine biologists to automate identification of plankton images collected *in situ*.

Specifically, the contributions of this work are as follows:

- We have pointed out that computing point correspondences between images is not necessarily the best paradigm for object class recognition.

- We have proposed the Maximum likelihood classifier for bags of features, which models the distribution of features in a category of images using non-parametric density estimation.

- We have proposed to use the Hausdorff distance between bags of features for pairwise comparison.

- We have proposed to embed a bags of features into a dissimilarity space using its Hausdorff distance to a set of prototype instances and then train a classifier in that space. We have shown that the prototypes can be selected randomly, resulting in a very efficient classification method. The time complexity of the

training phase that is linear in the size of the training set, and the time complexity of the test phase is constant in the size of the training set.

- We have analyzed the relative advantages of different methods for classifying images represented with bags of local features. We have concluded that an SVM classifier on the dissimilarity space is the most efficient approach from the computational point of view, whose accuracy on our data sets is as high or higher than that of the other methods.

- We have proposed three methods for combining the bags of features and the feature vectors into one classification system: stacking, concatenating an embedding of bags of features with feature vectors, and using a combination of SVM kernels, one computed on an embedding of the bags of features and the other on the feature vectors.

- We have analyzed the performance of different approaches to combining bags of features and feature vectors, and we have concluded that the difference in accuracy among them on our data sets is generally not statistically significant.

- We have also concluded that on the VPR data set combining the two representations results in a statistically significant increase in accuracy over using either representation alone. However, the improvement was not statistically significant on the FlowCAM images. We believe that the reason for this is the fact that the FlowCAM images are generally very small and produce very few local features. As a result the bag of features representation differs little from a global texture descriptor. Thus, adding the bags of features to the feature vector introduces little additional information.

- We have concluded that using a combination of SVM kernels, one computed on an embedding of the bags of features and the other on the feature vectors

is the most computationally efficient classification method, whose accuracy on our images is as high as or better than that of the other methods.

As we have mentioned earlier this work is part of on-going project to automate labeling of plankton images. However the methods presented in this dissertation are generic and could be easily applied to other domains. In order to improve our results on the plankton images it may be necessary to devise domain specific features.

Another future research direction can be to explore unsupervised and semi-supervised classification algorithms. The FlowCAM and the VPR are capable of producing thousands of images in very short time, and manually labeling enough training images is a very time-consuming task for the marine biologists. Unsupervised or semi-supervised methods could significantly reduce this work load.

BIBLIOGRAPHY

[1] D. Ballard. Generalizing the Hough transform to detect arbitrary pattern. *Pattern Recognition*, 13(2), 1981.

[2] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (24):509–522, 2002.

[3] M. C. Benfield, C. S.Davis, and S. Gallager. Estimating the in-situ orientation of calanus finmarchicus on Georges Bank using the Video Plankton Recorder. *Plankton Biology and Ecology*, 47:30–33, 2000.

[4] M. Blaschko, G. Holness, M. Mattar, D. Lisin, P. Utgoff, A. Hanson, H. Schultz, E. Riseman, M. Sieracki, W. Balch, and B. Tupper. Automatic in situ identification of plankton. In *Proceedings of IEEE Workshop on Applications of Computer Vision*, 2005.

[5] M. B. Blaschko, D. A. Lisin, M. A. Mattar, M. C. Benfield, and E. G. Learned-Miller. Using kernels to integrate feature vectors and bags of features for object classification. *Submitted to International Journal of Computer Vision*, 2006.

[6] S. Boughorbel, J.-P. Tarel, and F. Fleuret. Non-mercer kernels for svm object recognition. In *Proceedings of British Machine Vision Conference (BMVC'04)*, pages 137 – 146, London, England, 2004. http://www-rocq.inria.fr/ tarel/bmvc04.html.

[7] M. Brown and D. G. Lowe. Recognising panoramas. In *International Conference on Computer Vision*, 2003.

[8] C. J. C. Burges. Geometry and invariance in kernel based methods. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, Cambridge, MA, 1998.

[9] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[10] M. Burl and P. Perona. Recognition of planar object classes. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1996.

[11] T. F. Cox and M. A. A. Cox. *Multidimensional Scaling*. Chapman & Hall, London, 1994.

[12] P. Culverhouse, R. Williams, B. Reguera, V. Herry, and S. Gonzalez-Gil. Expert and Machine Discrimination of Marine Flora: a comparison of recognition accuaracy of field-collected phytoplankton. In *IEEE International Conference on Visual Information Engineering*, 2003.

[13] P. F. Culverhouse, R. Williams, B. Reguera, V. Herry, and S. Gonzalez-Gil. Do experts make mistakes? a comparison of human and machine identification of dinoflagellates. *Marine Ecology Progress Series*, 247:17–25.

[14] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B*, 39(1):1–38, 1977.

[15] T. Dietterich. Ensemble methods in machine learning. In *1st Intl. Workshop on Multiple Classifier Systems*, 2000.

[16] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[17] I. L. Dryden and K. V. Mardia. General shape distributions in a plane. *Advances in Applied Probability*, (23):259–276, 1991.

[18] H. du Buf and M. M. Bayer. *Automatic Diatom Identification*. World Scientific Publishing Company, 2002.

[19] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, Inc, second edition, 2001.

[20] J. Eichhorn and O. Chapelle. Object categorization with SVM: kernels for local features. Technical report, Max Planck Institute for Biological Cybernetics, Tübingen, Germany, 2004.

[21] A. Faragó, T. Linder, and G. Lugosi. Fast nearest-neighbor search in dissimilarity spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):957–962, 1993.

[22] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[23] W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9), 1991.

[24] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.

[25] D. Hall, B. Leibe, and B. Schiele. Saliency of interest points under scale changes. In *Proc. British Machine Vision Conf.*, 2002.

[26] F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel. *Robust Statistics: The Approach Based on Influence Functions*. John Wiley & Sons, Inc, 1986.

[27] A. Hanson, M. Marengoni, H. Schultz, F. Stolle, E. Riseman, and C. Jaynes. Ascender II: a framework for reconstructino of scenes from aerial images. In *Workshop Ascona 2001: Automatic Extraction of Man-Made Objects from Aerial and Space Images*, 2001.

[28] R. Haralick. Statistical and structural approaches to texture. *Proceedings of the IEEE*, (4):786–804, 1979.

[29] G. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings Alvey Vision Conference*, 1988.

[30] T. P. Hettmansperger and J. W. McKean. *Robust Nonparametric Statistical Methods*. John Wiley & Sons, Inc, 1998.

[31] G. R. Hjaltason and H. Samet. Properties of embedding methods for similarity searching in metric spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5), 2003.

[32] D. Huttenlocher, D. Klanderman, and A. Rucklige. Comparing images using the Hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–863, September 1993.

[33] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, 1998.

[34] T. Jebara and R. Kondor. Bhattacharyya and expected likelihood kernels. In *Conference on Learning Theory*, 2003.

[35] T. Jebara, R. Kondor, and A. Howard. Probability product kernels. *Journal of Machine Learning Research*, 5:819–844, 2004.

[36] H. Jeffries, M.S.Berman, A. Poularikas, C. Katsinis, I. Melas, K. Sherman, and L. Bivins. Automated sizing, counting and identification of zooplankton by pattern recognition. *Marine Biology*, 78:29–334, 1984.

[37] T. Kadir and M. Brady. Saliency, scale and image description. *International Journal of Computer Vision*, 2001.

[38] T. Kadir and M. Brady. Scale saliency : A novel approach to salient feature and scale selection. In *International Conference Visual Information Engineering*, pages 25–28, 2003.

[39] T. Kadir, A. Zisserman, and M. Brady. An affine invariant salient region detector. In *European Conference on Computer Vision*, pages 228 – 241, 2004.

[40] Y. Ke and R. Sukthankar. PCA-SIFT: A more distinctive representation for local image descriptors. In *Proc. CVPR*, 2004.

[41] J. J. Koenderink. The structure of images. *Biological Cybernetics*, 50:363–396, 1984.

[42] U. Kreßel. Pairwise classification and support vector machines. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, Cambridge, MA, 1999.

[43] T. Lindeberg. Feature detection with automatic scale selection. *Intl. Journal of Computer Vision*, 30(2), 1998.

[44] D. Lisin, E. Riseman, and A. Hanson. Extracting salient image features for reliable matching using outlier detection techniques. In *Proceedings International Conference on Vision Systems*, 2003.

[45] D. A. Lisin, M. A. Mattar, M. B. Blaschko, M. C. Benfield, and E. G. Learned-Miller. Combining local and global image features for object class recogntion.

In *IEEE Workshop on Learning in Computer Vision and Pattern Recognition*, 2005.

[46] D. G. Lowe. Object recognition from local scale-invariant features. In *Proc. International Conference on Computer Vision*, 1999.

[47] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2004.

[48] G. Matheron. *Random sets and intergral geometry*. John Wiley and Sons, New York, 1975.

[49] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *Proc. International Conference on Computer Vision*, pages 525–531, 2001.

[50] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *Submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2004.

[51] H. P. Moravec. Towards automatic visual obstacle avoidance. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, page 584, 1977.

[52] P. Moreels and P. Perona. Common-frame model for object recognition. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*. MIT Press, Cambridge, MA, 2005.

[53] T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE PAMI*, 24(7):971–987, 2002.

[54] E. Pekalska and R. Duin. Dissimilarity representations allow for building good classifiers. *Pattern Recognition Letters*, 23(8):943–956, 2002.

[55] E. Pekalska, P. Paclik, and R. Duin. A generalized kernel approach to dissimilarity-based classification. *Journal of Machine Learning Research, Special Issue on Kernel Methods*, 2(2):175–211, 2002.

[56] J. Piater. *Visual Feature Learning*. PhD thesis, Department of Computer Science, UMASS Amherst, 2001.

[57] J. Piater and R. Grupen. Learning appearance features to support robotic manipulation. In *Cognitive Vision Workshop*, 2002.

[58] A. Pope. *Learning to Recognize Objects in Images: Acquiring and Uising Probabilistic Models of Appearance*. PhD thesis, Computer Science Department, University of British Columbia, 1995.

[59] A. Pope and D. Lowe. Learning probabilistic appearance models for object recognition. In S. Nayar and T. Poggio, editors, *Early Visual Learning*, pages 67–97. Oxford Universe Press, 1996.

[60] S. Ravela. *On Multi-Scale Differential Features and their Representations for Image Retrieval and Recognition*. PhD thesis, University of Massachusetts Amherst, 2002.

[61] S. Ravela and A. Hanson. On multi-scale differential features for face recognition. In *Vision Interface*, Ottawa, 2001.

[62] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

[63] J. T. Robinson. The k-d-b-tree: A search structure for large multidimensional receptive field histograms. *Transactions of the Associateion for Computing Machinery*, 1981.

[64] C. Schmid, R. Mohr, and C. Bauckhage. Evaluation of interest point detectors. *International Journal of Comptuter Vision*, 37(2), June 2000.

[65] B. Schölkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. MIT Press, 2002.

[66] D. W. Scott and S. R. Sain. Multi-dimensional density estimation. *Data Mining and Computational Statistics*, 23, 2004.

[67] N. Sebe, Q. Tian, E. Loupias, M. Lew, and T. Huang. Evaluation of salient point techniques. *Image and Vision Computing*, 21(13-14):1087–1095, December 2003.

[68] M. Shapiro and M. Blaschko. Stability of Hausdorff-based distance measures. In *Proc. of IASTED Visualization, Imaging, and Image Processing*, Marbella, Spain, 2004.

[69] A. Shokoufandeh, I. Marsic, and S. J. Dickinson. View-based object recognition using saliency maps. Technical report, Deparment of Computer Science, Rutgers University, 1998.

[70] C. Sieracki, M. Sieracki, and C. Yentsch. An imaging-in-flow system for automated analysis of marine microplankton. *Mar. Ecol. Progr. Ser.*, 168:285–296, 1998.

[71] P. Soille. *Morphological Image Analysis: Principles and Applications*. Springer-Verlag, 1999.

[72] G. Tian, D. Gledhill, D. Taylor, and D. Clarke. Interest points based fast 3d surface reconstruction. In *Computer Graphics and Imaging*, 2004.

[73] M. Turk and A. Pentland. Face recognition using eigenfaces. In *Proceedings IEEE Converence on Vision and Pattern Recognition*, 1991.

[74] S. Ullman. *High-Level Vision: Object Recognition and Visual Cognition*. MIT Press, 2000.

[75] J. Vleugels and R. Veltkamp. Efficient image retrieval through vantage objects. *Pattern Recognition*, 35(1):69–80, 2002.

[76] V.Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[77] K. N. Walker, T. F. Cootes, and C. J. Taylor. Locating salient object features. In *Proc. of BMVC*, 1998.

[78] C. Wallraven, B. Caputo, and A. B. A. Graf. Recognition with local features: the kernel recipe. In *International Conference on Computer Vision*, 2003.

[79] A. P. Witkin. Scale-space filtering. In *Proc. International Joint Conference on Artificial Intelligence*, 1983.

[80] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann, 2000.

[81] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

[82] Z. Zhang, R. Deriche, O. Faugeras, and Q.-T. Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artificial Intelligence*, 78(1), 1995.