Local Features for Image Classification

Diplomarbeit im Fach Informatik

Lehrstuhl für Informatik VI
Rheinisch-Westfälische Technische Hochschule Aachen
Prof. Dr.-Ing. H. Ney

vorgelegt von:

Tobias Gabriel Benedikt Kölsch
Matrikelnummer 215203

Gutachter:

Prof. Dr.-Ing. H. Ney
Prof. Dr. E. Vidal

Betreuer:

Dipl. Inform. D. Keysers

Hiermit versichere ich, dass ich die vorliegende Diplomarbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe. Alle Textauszüge und Grafiken, die sinngemäß oder wörtlich aus veröffentlichten Schriften entnommen wurden, sind durch Referenzen gekennzeichnet.

Aachen, November 2003

Tobias Gabriel Benedikt Kölsch

Acknowledgements

This work has been supervised by the Lehrstuhl für Informatik VI of the RWTH-Aachen and by the Instituto Tecnológico de Informática of the Universidad Politécnica de Valencia. It would not have been possible without the support and help of Prof. Dr.-Ing. Hermann Ney and Prof. Dr. Enrique Vidal who gave me the possibility to complete my diploma thesis at their respective institutes.

Also, I want to thank Dipl.-Inform. Daniel Keysers for the help and support he gave me in the course of this work.

I would like to thank Dr. Roberto Paredes for the helpful discussions and incitements. And also I would like to thank Juan-Carlos Pérez for inviting me to Valencia in such an unproblematic manner.

I thank everybody at the ITI who helped me make the stay in Valencia not only an academic, but also a personal experience. Especially, I would like to mention Javier Cano, Alejandro Héctor Toselli, José Ramón Navarro Cerdón, José García, Felix García, Elsa Cubel, Antonio Lagarda, and David Vilar.

I thank everybody at the I6 who helped in the course of this work. I am especially grateful to the members of the image recognition group who are Thomas Deselaers, Christian Gollan, Philippe Dreuw, and Ilja Bezrukov.

Also I thank Ansgar Prüne, Christian Haselbach, and Peter Schneider-Kamp for proof reading and commenting on my presentations.

Then I would like to thank Claudia Hernandez-Inojosa for bearing my absence and spoiling me when I was present.

And, of course, all my thanks are with my mother Marinette, my father Ulrich, and my sister Valérie who I did not see much, but who always assisted me when in need for help.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This work is about appearance based object classification in digital images. Image object classification is used for face recognition, license plate recognition, medical image classification and similar tasks. The problem is to assign an image to one of several known classes based on its appearance. This task usually incorporates two steps: object localization and recognition. Here, we assume that the localization has already taken place, leaving us with images containing the objects of interest for recognition.

Looking at the possible approaches for object recognition, we find two extremes. The first one is that the entire image is considered as a whole for recognition. This bears the problem that we run into difficulties in the presence of non-linear transformations but has the advantage that global dependencies are taken into account.

The other extreme is that only image extracts are compared, discarding all information about their global position. By this, extracts that are similar can come from incomparable positions in their respective images. However, this approach can deal well with non-linear deformations and with partial occlusion.

Additionally, there are all kinds of combinations of the two extremes. There are approaches using local characteristics and then evaluating their relative positions [Brown & Lowe 02] or using hidden Markov models that use local features instead of single pixel values to compute the emission probability [Gollan 03].

However, all these methods have in common that they compare entire images. This restriction significantly reduces the capability of generalization. Imagine a face classifier where the training images contain faces that are directly illuminated and others that are entirely shaded. If in a testing case we have to deal with an image that is illuminated from the left side, leaving the right one shaded, we would (provided no preprocessing that leads to invariance against this situation took place) have difficulties matching the testing image with either of the training images independently of the method we choose. The method we propose in this work differs in this aspect. Here, we do not look at every image of a training class independently, but regard all local features of one class at the same time. This gives us the flexibility to match the parts of the testing image with the parts of the training images that fit together best. This unconstrained search frees us from having to model variability explicitly to a certain degree, provided the respective variations of a specific image region are present in the training. The examined method does the matching between the image regions without global restrictions and solely based on local similarity.

This appearance based approach has proven successful on various image classification tasks including handwritten digits classification [Keysers & Paredes$^+$ 02], the classification of radiography [Paredes & Keysers$^+$ 02] and face classification [Paredes & Pérez$^+$ 01] where it outperformed all existing approaches. It also performed well in the face recognition contest described in [Messer & Kittler$^+$ 03].

Nevertheless, the presented method has a simple structure and disregards a lot of information. For example a direct voting scheme is chosen, so that only the best matching regions influence the decision process without regarding their goodness compared to the other matches. Moreover the Euclidean distance is chosen as distance measure. However, other distance measures offer more tolerance towards small transformations [Keysers 00].

In this work different probability frameworks for unconstrained local feature search are examined to consider larger amount of the available information. Furthermore, other distance measures that increase the invariance towards small local transformations like rotation, scaling, and noise are incorporated into the framework.

The main insights gained in this work are that recognition can be improved by using a kernel densities probability model and by using a distance measure that is invariant with respect to small local transformations, e.g. the tangent distance. Another interesting result is that the recognition is slightly improved by using an approximate nearest neighbor search.

The next section gives an overview of the state of the art in image object recognition. Section 1.2 exposes the motivations for this work. And in Section 1.3 the notation that is used in this work will be presented. In Chapter 2 an overview about general pattern recognition is given. This chapter is based on [Ney 00]. Chapter 3 is the main part, here the existing approaches are described in detail and variations are introduced and discussed. In Chapter 4 the databases used for evaluation are presented along with the best results from other groups. Our results on these databases are described in Chapter 5. In Chapter 6 the conclusions of the present work are drawn and a perspective to further investigation is given. The programs that have been written in the course of this thesis are described in appendix B. Appendix C presents all the programs and tools that were essential for the success of this work.

1.1 State of the Art

There are many publications on the subject of image object recognition. Some of them present holistic methods and others deal with local features. The approaches have been used for different kinds of tasks. There have been studies on object classification and recognition tasks, other groups have presented investigations about object detection or localization tasks, additionally there are publications about other tasks that use local and global features. First, the works on object recognition are presented.

[Fergus & Perona$^+$ 03] have used a framework that combines information about shape, appearance, relative scale and occlusion using probabilities. Local features of various sizes are extracted at image positions with a high local entropy, scaled to 11×11 sized windows and matched with reference features giving the probability for the appearance. Occlusion is modeled with a background model. From this matching the probability for the shape and the scale are estimated. The training is done using an EM-Algorithm. This method has been tested on different databases for the 2 class problem object absence versus object presence. However all presented results have been topped in [Deselaers 03] using simple texture features. This shows that it is important to find a representation that is appropriate to the problem. This can lead to good results more easily than complicated models.

[Deselaers 03] examines different features for image retrieval. Some of them are color histograms, Tamura features, local feature histograms, etc. The evaluation is done by classification tasks. As suggested earlier simple global classifiers did in some cases lead to better results than more complicated models.

In [Keysers 00] a measure that is invariant to small previously known transformations, the *tangent distance*, has been used for handwritten character recognition on the well known *US postal service* (USPS) corpus. This led to the best results of that time with 2.4% error. This method is also reported to have led to the result of 12.9% on the *Image Retrieval in Medical*

Applications (IRMA) corpus. Later a result of 8% has been achieved on IRMA using an *image distortion model* in addition to the tangents. This is presented in [Keysers & Dahmen$^+$ 03]. In the same publication another distance measure has been presented that does not measure the distance by directly comparing respective pixels. It is called *image distortion model.* Instead one pixel is compared with that pixel within a neighborhood that fits best. this leads to an error of 13.2% on IRMA. If this is combined with the tangent distance the error is reduced to 10.3%.

[Gollan 03] examines various non-linear transformation models, including an extended *pseudo two-dimensional hidden Markov model*, a real *two-dimensional hidden Markov model* and an extended *image distortion model* that takes into account the local context. Those methods have led to excellent results on all tested corpora, including 5.3% on IRMA, 1.9% on USPS with the extended pseudo two-dimensional hidden Markov model and 0.6% on MNIST with the extended image distortion model.

Global rotation, translation and scale invariant Mellin Fourier transforms are used together with Gaussian mixture densities in [Dahmen & Keysers$^+$ 00] for blood cell classification. This leads to a 15.3% error rate which is better than the human error of more than 20%.

In [Keysers & Paredes$^+$ 02] the local feature approach that is the base of this work has been used for handwritten digit recognition on the USPS corpus leading to the fairly good result of 3.0% error. It also has been shown that combining this method with the tangent vector approach presented above using majority vote results in an error of 2.0%.

[Keysers & Och$^+$ 02] presents a global, appearance based model for discriminative training of probabilities. It shows that the maximum entropy model outperforms Gaussian single densities for handwritten digit recognition on the USPS corpus with 14.2% versus 5.7%. Furthermore, the approach leads to results that are comparable to nearest neighbor 5.6%, but needing only about one fifth of the parameters.

In [Mohr & Picard$^+$ 97] the Bayesian decision is compared to direct voting for local features that have been extracted at salient locations. It shows that the probabilistic model is especially suited, when the single features are not that discriminative.

In [Deselaers & Keysers$^+$ 03] local features are used to identify possible object locations. The decision if an object is present or not is done using clustering or template matching. The tests have been done on a multi object database that has been generated out of the COIL corpus of common objects, by putting one, two or three COIL objects into a larger image. Further results are reported by permitting partial occlusion of the objects and by using heterogeneous backgrounds.

Another method for recognition and location of objects with occlusion and heterogeneous backgrounds is presented in [Reinhold & Paulus$^+$ 01]. The features are multi-resolutional Johnston wavelets that are extracted at evenly distributed positions of the image. The backgrounds are modeled as uniform distribution over the feature space and the objects features are modeled by mixture densities. The results of this approach are presented in Section 4.3 of this work.

[Wiskott & Fellous$^+$ 97] presents a higher level approach for face recognition using a single prototype. A graph of facial point positions, as pupils, tip of nose, corners of mouth, etc. is trained manually on a small number of faces. The facial positions are described by so called *jets.* Those are a number of values extracted at the wanted facial position of wavelet transforms of the image with different kernels (they differ in orientation and frequency). After the points have been found, the graphs are compared by a straight forward graph matching algorithm.

In [Lowe 99] local features that are invariant to translation, rotation, and scaling and partially invariant to illumination changes and projective transformations, are extracted at stable points in scale space. For testing the features are matched using nearest neighbor. The verification of the matches is done by finding a low residual least square problem for the global transformation parameters. According to the authors, this permits to find partially occluded objects in cluttered environments.

Face localization is the task presented in [Chen & Gu$^+$ 01]. They train representative local features from scale space by local non-negative matrix factorization to get an over-complete set of representatives. Then a small number of discriminative features is selected automatically. This method outperforms a Harr like classifier and is claimed to be comparable to other methods.

In [Brown & Lowe 02] invariant local features are used to localize matching positions in different images. This has been used to estimate transformations or to create $360°$ views by joining overlapping photographies. To do so features that are similar to those presented in [Lowe 99] are extracted at interest points in scale space. Then a nearest neighbor matching is performed between the two images. A transformation invariant outlier detection is used to identify false matches. Then a transformation is computed such that the interest points of the first image match those of the second.

Semi-local histogram features are presented in [Belongie & Malik$^+$ 00] for optical character recognition. They are extracted at edges within the image. In a second step the features of the query and the reference image are matched with the Hungarian algorithm and a least square transformation from one to the other is searched iteratively. A 0.63% error was achieved by this on the MNIST corpus.

[Kittler & Hatef$^+$ 98] examines which combination schemes are suitable for classification when joining the results of different methods. They compare the following combination schemes: min rule, product rule, majority voting, median rule, sum rule and max rule. The conclusion is drawn that the *sum rule* and the *median rule* are best suited for combination.

1.2 Motivation

This work has been motivated by the good results achieved with local features and direct voting in various image classification tasks, as those on the *Olivetti Research Laboratory* face classification database presented in [Paredes & Pérez$^+$ 01] and on a medical radiography classification corpus presented in [Paredes & Keysers$^+$ 02].

The goal was to evaluate the base method by systematic variations and extensions. Especially the simple probability model presented by R. Paredes et al. seemed to leave a lot of space for further investigation. Also the distance measure used, only implicitly leads to invariance towards transformations. Other distance measures are evaluated in the course of this work. The main points of interest are:

- Examine the effect of extracting features at multiple scales.
- Review alternate feature reduction techniques. Here especially the Fisher linear discriminant analysis and the discrete cosine transform are of interest.
- Inspect invariant distance measures, especially the tangent distance.
- Experiment with a kernel densities based approach instead of the direct voting.
- Examine log-linear models for non-equal weighting of the local feature probabilities.

All these single inspections are driven by the goal of better understanding why the local feature approach leads to such good results despite its apparent simplicity and identify and document the important aspects of the method. Finally, we hope to use these insights to improve the method. The effect of these changes is compared to the base method and other approaches by evaluating it on known data bases.

1.3 Notation

The symbols used in throughout work have the following meanings:

D	Dimension of vectors
K	Number of classes
X	Image
x	Local feature
$\mathfrak{X}_X$	Local features of image X
$\mathfrak{X}_k$	All local features of the training images of class k
$\mathfrak{X}$	Unification of all $\mathfrak{X}_k$
$\hat{x}$	Nearest neighbor of x
$\hat{x}_k$	Nearest neighbor of x in $\mathfrak{X}_k$
$\hat{x}_{k,l}$	l-th nearest neighbor of x in $\mathfrak{X}_k$

This notation has been used where possible.

Chapter 2

Pattern Recognition

In a typical pattern recognition task an input signal has to be assigned to one of K classes. For this general view it is not important whether we treat acoustical signals, EEG sequences or digital images. The process consists of three main steps,

1. preprocessing,
2. feature analysis and
3. application of a discriminant rule,

as demonstrated in Figure 2.1. In the preprocessing step basic transformations are performed on the signal. Examples are amplitude normalizations and de-noising.

The result of the feature analysis is a set of feature vectors that represent the input in a way that emphasizes the important aspects. For example in speech recognition doing the recognition on the plain acoustic signal does not lead to good results, as the information is only indirectly carried by pressure variations of the air. A more suitable representation for the input based on the frequency spectrum [Ney 01]. Some typical preprocessing steps in image recognition include Fourier analysis, gradient images, Laplace, and wavelet transforms [Jähne 02].

In a following step the *discriminant function* uses these feature vectors for classification. Such a function has the form:

$$\begin{aligned} g: \quad \mathbb{R}^D \times \{1,\dots,K\} &\mapsto \mathbb{R} \\ (X,k) &\mapsto g(X,k) \end{aligned} \tag{2.1}$$

Ideally g returns 0 or 1 for the wrong or right classes respectively but in most cases a discriminant function will only approximate this binary result. As we still would like to get a result even if the class cannot be determined with certainty, we introduce the decision rule r:

$$\begin{aligned} r: \quad \mathbb{R}^D &\mapsto \{1,\dots,K\} \\ X &\mapsto \underset{k}{\operatorname{argmax}}\{\, g(X,k) \,\} \end{aligned} \tag{2.2}$$

The discriminant function can be taken from a wide variation of methods. Examples are linear discriminant function that describe the class boundaries by a hyper plane in feature space. A further approach is that of artificial neural networks, these imitate the way neurons process data in nature.

Additionally, there are memory based techniques. These do not attempt to model the training data using parameters, but represent it directly. In a testing phase the test features are compared

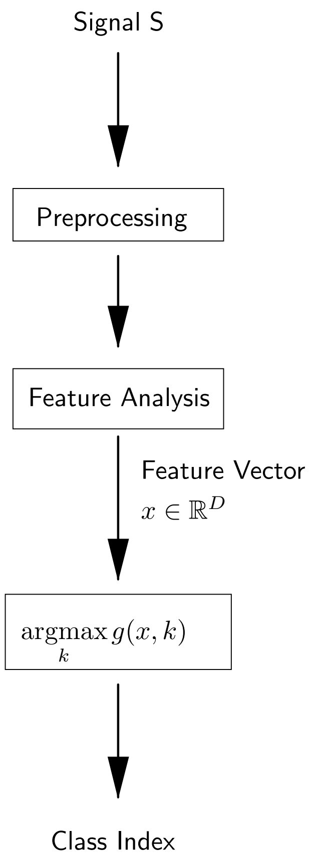


Figure 2.1: Structure of a recognition system.

to those seen in the training. The decision is now done using that comparison. A simple example for this is the nearest neighbor classifier that we will see in Section 2.1.

Another paradigm of pattern recognition is statistical pattern recognition. Here, the classes are modeled as probability distributions $p(k|X)$. This is then used as discriminant function

$$g(X,k) = p(k|X) \tag{2.3}$$

However this paradigm is orthogonal to the other presented methods so that they all can be interpreted in a probabilistic manner. This is presented in more detail in Section 2.3.

As this work deals with image recognition, we will often refer to images when presenting the pattern recognition techniques in this chapter. However, these techniques can generally be applied to all kinds of patterns.

2.1 Nearest Neighbor

The nearest neighbor classifier is a simple memory based method that, given the labeled training data $(X_1, k_1), \ldots, (X_N, k_N)$ and a distance measure d, decides the class affiliation of the testing data X using the following rule:

$$r(X) = k_n \text{ if } n = \operatorname*{argmax}_{n'}\{-\operatorname{d}(X, X_{n'})\} \tag{2.4}$$

This method's error rate has been proven not to be larger than twice the error rate of the Bayesian decision rule if the training set $\{(X_1, k_1), \ldots, (X_N, k_N)\}$ is infinitely large. A proof for this can be found in [Ney 00]. Due to its simplicity, the nearest neighbor classifier is well suited to calculate a baseline to which new methods can be compared.

This method can be extended to K-nearest neighbor classifier. Here the class that is most represented within the k-nearest neighbors of the test image is chosen.

A problem with these measures, as with all global methods, is that all images have to be of the same size. If this is not given they have to be scaled appropriately which might introduce unwanted effects for the recognition.

2.2 Distance Measures

Distance measures are an essential part of many classifiers. Usually the distance is the measure by which the dissimilarity between two patterns is expressed. To measure the distance between two images, these are usually interpreted as points in a high dimensional space. One measure to calculate the distance between them is the *Euclidean distance*. This is normally used to express distances in a vector space. It is also known as L_2-norm. Other simple distance measures that are from the same family are the city block distance or L_1-norm and the maximum or L_∞-norm. All these distances between two $I \times J$-dimensional images X and X' are given by

$$\mathrm{d}_l(X, X') = \left(\sum_{i=1}^{I} \sum_{j=1}^{J} |X_{i,j} - X'_{i,j}|^l \right)^{\frac{1}{l}} \tag{2.5}$$

Unfortunately these distance measures are quite sensitive to many linear and non-linear transformations. For example, a small change in brightness on one of the images leads to a large change in distance, as the distance of every single pixel pair has changed. This sensitivity can be reduced by applying some preprocessing steps to the images as for example a Fourier transformation.

A similar measure is the *Mahalanobis distance*. This is trained on some reference data $X_1, \ldots, X_M \in \mathbb{R}^{I \times J}$ with the covariance matrix Σ. For the images X and X' it is given by

$$d_m = (X - X')^T \Sigma^{-1} (X - X') \tag{2.6}$$

This can be interpreted as weighting the different pairs of components of the image inversely to their covariances. The idea behind this is that a large distance between components that usually have large distances between each other means less than a large distance between components that tend to be close to each other. This distance measure is typically used in statistical classifiers. This measure gives some transformation invariance in some cases by reducing the relevance of pixels that are typically subject to variations, this is especially true, if one covariances matrix is extracted for each class. This does usually not lead to invariance towards global transformations though.

Another distance measure that gives some invariance against small local transformations is the *image distortion model* as described in [Keysers 00]. In this model small local deformations are allowed, such that for $d_{IDM}(X, X')$ a pixel $X_{i,j}$ is not necessarily compared with $X'_{i,j}$ but with the best matching pixel of X' within some neighborhood $R_{i,j}$. The distance between the images is measured by

$$d_{IDM}(X, X') = \sum_{i=1}^{I} \sum_{j=1}^{J} \min_{(i',j') \in R_{i,j}} \{ \|X_{i,j} - X'_{i',j'}\|^2 + C_{(i,j),(i',j')} \} \tag{2.7}$$

where $C_{(i,j),(i',j')}$ is a cost function that associates a cost for the distortion.

But this again does not give invariance against global transformations as scaling or brightness changes. However a distance measure has been proposed that is invariant against small global transformations. The *tangent distance* as proposed in [Simard & Le Cun+ 93]. The idea here is to represent a set of previously known transformations as a subspace in the pattern space. This approach is described in more detail in Section 3.6.

All methods presented so far have the restriction that the images that are compared must have the same dimension. If the task does not fulfill this, this has to be achieved by scaling one or both images. Other methods have been proposed, as the non-linear deformation models investigated in [Gollan 03] that can, to a certain extent, deal with differently sized images and non-linear transformations.

Another way of comparing images of different sizes is by using local features for the comparison this is similar to the method described in [Paredes & Pérez+ 01]. This also gives invariance against global and local deformations as described in the introduction.

All these distance measures are not useful for classification by themselves, but are often part of the discriminant function g.

2.3 Statistical Classifiers

Another paradigm of pattern recognition is *statistical pattern recognition*. Here, the probability that a pattern x comes from a specific class is described by the *class conditional probability* $p(x|k)$. Together with the *a priori probability* $p(k)$ of a given class k we can calculate the *a posteriori probability* $p(k|x)$ using *Bayes' rule*:

$$p(k|X) = \frac{p(X|k) \cdot p(k)}{p(X)} = \frac{p(X|k) \cdot p(k)}{\sum_{k'=1}^{K} p(X|k) \cdot p(k)} \tag{2.8}$$

The a priori probability $p(k)$ is usually estimated from the relative frequency of k in the training data. In tasks where we do not want to favor any particular class we may set it to $\frac{1}{K}$. The classification decision is usually done using the *Bayes' decision rule*:

$$r(X) = \underset{k}{\operatorname{argmin}}\{p(k|X) \cdot L(k, k_X)\} \tag{2.9}$$

In this equation $L(k, k_X)$ is a cost function that describes the cost of choosing class k when k_X is the correct class. In many classification tasks it makes sense to define L as:

$$L(k, k') = 1 - \delta(k, k') \tag{2.10}$$

where δ is the Kronecker delta. If this is given, the decision rule is equivalent to:

$$\begin{aligned} r(X) &= \underset{k}{\operatorname{argmax}}\{p(k|X)\} \\ &= \underset{k}{\operatorname{argmax}}\,\{p(k,X) \cdot p(X)\} \\ &= \underset{k}{\operatorname{argmax}}\{p(k,X)\} \end{aligned} \tag{2.11}$$

This has been proven to be optimal with respect to the number of errors made, given that the probability distributions are known. A proof for this can be found in [Duda & Hart+ 01].

Unfortunately, the probabilities usually are not known and have to be modeled and estimated from some training data. This is one of the inherent difficulties of pattern recognition. There are several parameterized methods to represent densities. Those and the ways to estimate them will be presented in the following sections.

2.3.1 Single Densities

This representation is useful to model the distribution densities of a simple random variable. An example for this would be the measurement of a fixed distance. As the measuring tends not to be exact, successive measurements will not return the same value, but a value that is close to real one. The probability that a measurement k returns X if its variance is σ_k^2 and its mean value is μ_k is usually modeled using the well known *Gaussian density* or *Normal distribution*

$$p(X|k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(-\frac{1}{2}\frac{(X-\mu_k)^2}{\sigma_k^2}\right) \tag{2.12}$$

If the values whose probabilities we want to estimate have $D > 1$ dimensions, a covariance matrix Σ_k and a mean μ_k, the probability density is given by

$$p(X|k) = \frac{1}{\sqrt{2\pi|\Sigma_k|}} \exp\left(-\frac{1}{2}(X-\mu_k)^T \Sigma_k^{-1}(X-\mu_k)\right) \tag{2.13}$$

As the real values for μ_k and Σ_k are usually not known, they have to be estimated. This is typically done using the maximum likelihood criterion with some reference data. This signifies that the empirical mean $\hat{\mu}_k$ and the empirical covariance matrix $\hat{\Sigma}_k$ are taken as estimates for the real μ_k and Σ_k. This leads to good results, if the amount of reference data is sufficient and the assumption that it is a single density is correct.

Problems arise if the covariance matrix is not invertible due to insufficient data. One way to cope with this is by smoothing it or alternatively by using only the diagonal of $\hat{\Sigma}_k$.

2.3.2 Gaussian Mixture Densities

If the data is not generated by a single density, but from a combination of various densities, modeling it by a single density can lead to arbitrarily bad approximations. E.g. if the probability is the compound of two normally distributed probability densities, this usually cannot be modeled well by one single distribution. The approximation becomes worse if the distance between the centers of the distributions drift apart. In this case more complicated models are suited better.

One such model is the *Gaussian mixture densities*. It can approximate arbitrary density distributions and is especially useful when the density that is to be modeled is not normally distributed.

The goal is to model an unknown probability $p(X)$ by a family of probabilities $p(X|\lambda_i)$ that is parameterized by λ_i with $i = 1, \ldots, I$. The parameters can, for example, be the variances σ_i^2 and means μ_i and

$$p(X|\lambda_i) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{1}{2}\frac{(X-\mu_i)^2}{\sigma_i^2}\right)$$

is given by the Gaussian distribution as seen earlier. The mixture density is now given by the sum

$$p(X) = \sum_{i=1}^{I} c_i \cdot \hat{p}(X|\lambda_i) \qquad c_i \geq 0, \quad \sum_{i=1}^{I} c_i = 1 \tag{2.14}$$

A mixture density that is the sum of two distributions is depicted in Figure 2.2.

A problem with these distributions is that there usually is no closed form solution to train them from example data. So the training is done by iterative methods. One that is typically used is the K-means algorithm or one of its extensions. This family of algorithms is capable of approximating arbitrary distributions. The basic idea for these algorithms is that the problem of approximating a distribution is reduced to the problem of finding a clustering in the training data, and model each of these clusters by a single density.

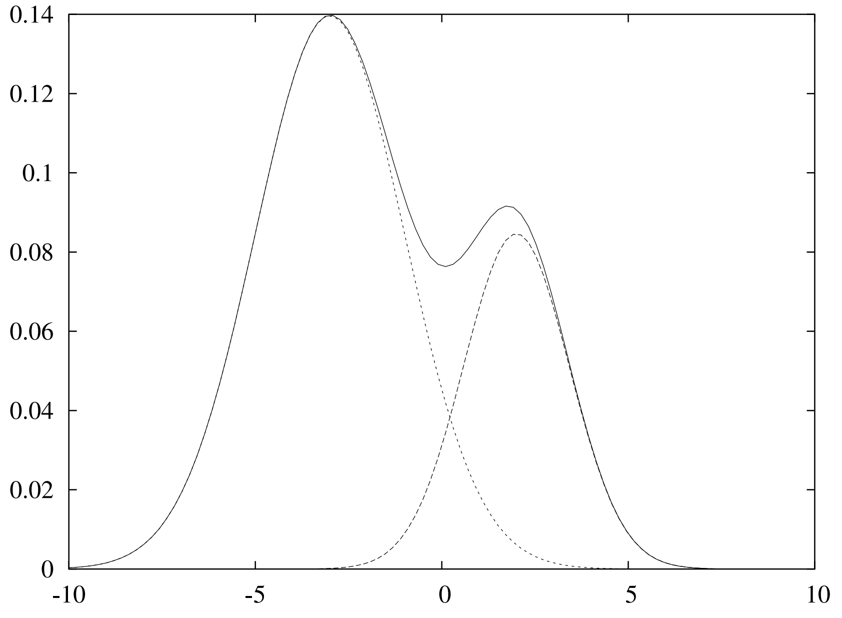


Figure 2.2: Two single Gaussian densities are composed to form a Gaussian mixture density.

K-Means Clustering

Clustering can be seen as quantization of data of arbitrary dimensionality. The goal is to minimize the squared error of the quantization on the training data. The K-means algorithm is an attempt to optimize this criterion. It is an algorithm of the expectation maximization family. The algorithm is supplied with a set of features $\mathfrak{X} = \{x_1, \ldots, x_N\}$ and some split-condition S that, for a given cluster, returns 1, -1 or 0 depending on whether the cluster has to be split into two sub-cluster, has to be merged with another cluster, or does not have to be changed.

Pseudo code for the algorithm can be seen in algorithm 1.

As the squared error always decreases when clusters are split, some kind of condition has to be given, such that not each data point gets one cluster. One such method, called ISODATA was presented in [Ball & Hall 65]. The idea here is that the user supplies the algorithm with parameters. The first one applies if clusters get too large or have an unusually large variance. Those get split perpendicular to the largest spread. Another parameter defines, when clusters are considered as being too small or too close to some other cluster. Those clusters then get merged.

Other algorithms also have outlier detection to prevent some outliers to deteriorate the entire clustering result. An extensive review on clustering methods is given in [Jain & Dubes 88]

The K-means algorithm is known to converge quite rapidly. An example in two dimensions with a bad initial clustering is given in Figure 2.3.

One notable variation of the K-means algorithm is the *fuzzy K-means* that is also described in [Ney 00]. In this algorithm not only the nearest neighbor is taken into account, but all other data points, weighted inversely to their distance using some probability distribution p, e.g. the Gaussian distribution. How it works can be seen in algorithm 2.

Algorithm 1: The K-means clustering algorithm.
Input: A set of features $\mathfrak{X} = \{x_1, x_2, \ldots, x_N\}$.
Output: The disjunct sets $\mathfrak{X}_1, \ldots, \mathfrak{X}_I$ that contain one cluster of $\mathfrak{X}$ each.
KMCluster($\mathfrak{X}, S$)

```
label Initialize:
    $\mathfrak{X}_i$ such that $\bigcup_{i=1}^{I} \mathfrak{X}_i = \mathfrak{X}$ and $\mathfrak{X}_i \bigcap \mathfrak{X}_{i'} = \{\,\}$ for all $i \neq i'$
label begin:
    while some $\mathfrak{X}_i$ changed
        Calculate the means $\mu_i$ of $\mathfrak{X}_i$ for $i = 1, \ldots, I$.
        $\mathfrak{X}_i := \{x_n : i = \mathrm{argmin}_{i'}\{\|x_n, \mu_{i'}\|\}\}$
    for i=1 to N
        if $S(\mathfrak{X}_i) = 1$
            Create $\mathfrak{X}_{I+1}$ and $\mathfrak{X}'_i$ such that $\mathfrak{X}_{I+1} \bigcup \mathfrak{X}'_i = \mathfrak{X}_i$ and $\mathfrak{X}_{I+1} \bigcap \mathfrak{X}'_i = \{\,\}$
            Set $\mathfrak{X}_i := \mathfrak{X}'_i$ and $I := I + 1$
            goto begin
        else if $S(\mathfrak{X}_n) = -1$
            Choose a $\mathfrak{X}_{i'}$ by some criterion
            Swap the Elements of $\mathfrak{X}_{i'}$ and $\mathfrak{X}_I$
            Set $\mathfrak{X}_i := \mathfrak{X}_I \bigcup \mathfrak{X}_i$
            Set $I = I - 1$
            goto begin
        else
            return $\{\mathfrak{X}_1, \ldots, \mathfrak{X}_N\}$
```

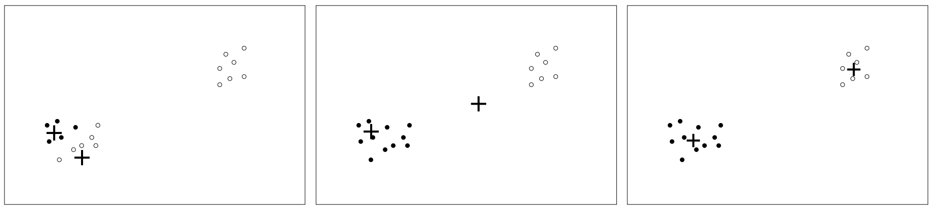


Figure 2.3: Convergence of K-means with bad initial condition.

2.3.3 Kernel Densities

The single density, that models a simple distribution, and the mixture densities for more complex distributions have been presented. The single density is an extreme case of the mixture densities to model a density trained by representative data. The other extreme is that every training example represents its own density. These densities are called *Parzen windows*, *Parzen densities* or *kernel densities*. The Gaussian density is typically used for this. However, the variance of a single point is 0. So we have to assume some other variance σ^2. One possibility is to multiply the variance of the data $\hat{\sigma}^2$ by some factor $\alpha \in \mathbb{R}$.

For some classification tasks, the class dependent variance is taken. However in image classification tasks this does not help, as can be seen in later chapters.

2.3.4 Log-Linear Classifiers

In [Jaynes 57] a new way of coding knowledge is presented. The underlying idea is to create a probability model that reflects the evidence that has been seen in the training but within these

```
Algorithm 2: The fuzzy K-means clustering algorithm.
Input: A set of features $\mathfrak{X} = \{x_1, \ldots, x_N\}$.
Output: A set of initial parameters $\{(c_1, \mu_1, \Sigma_1), \ldots, (c_I, \mu_I, \Sigma_I)\}$.
FKMCluster($\mathfrak{X}$)

while not converged
    Expectation Step:
        Calculate:
            $$p(i|x_n, \{c_i, \mu_i, \Sigma_i\}) := \frac{c_i \cdot p(x_n|i, \mu_i, \Sigma_i)}{\sum_{i'} c_{i'} \cdot p(x_n|i', \mu_{i'}, \Sigma_{i'})}$$
        Calculate:
            $$\gamma_i(x_n) := \frac{p(i|x_n, \{c_i, \mu_i, \Sigma_i\})}{\sum_{x_{n'}} p(i|x_{n'}, \{c_i, \mu_i, \Sigma_i\})}$$
    Maximization Step:
        Calculate new weights:
            $$c'_i := \frac{1}{N} \sum_n p(i|x_n, \{c_i, \mu_i, \Sigma_i\})$$
        Calculate new means:
            $$\mu'_i := \sum_n \gamma_i(x_n) \cdot x_n$$
        Calculate new Covariances:
            $$\Sigma'_i := \sum_n \gamma_i(x_n) \cdot [x_n - \mu'_i][x_n - \mu'_i]^T$$
    Update parameters:
            $$c_i = c'_i \qquad \mu_i = \mu'_i \qquad \Sigma_i = \Sigma'_i$$
```

constraints has the largest entropy possible. For this reason the method is called *maximum entropy modeling*. Another name used is *logistic regression*. This principle is widely used in computational linguistics [Ratnaparkhi 97, Berger & Della Pietra+ 96], but it also has been used for appearance based image object recognition [Keysers & Och+ 02]. However, here a short overview over the natural language processing usage of maximum entropy will be given, as the way maximum entropy is used here is closer to this than to the appearance based method presented in [Keysers & Och+ 02]. Thus, this part is mainly inspired by the two sources from computer linguistics presented earlier.

First we present a simple example from [Berger & Della Pietra+ 96]. In a translation system the task to find out the different French translations for the English word "`in`". The system is taught by reference translations. The first observation that is made is that "`in`" is always translated by one of the phrases: $\{\texttt{dans}, \texttt{en}, \texttt{á}, \texttt{au cours de}, \texttt{pendant}\}$. Another observation made, is that `dans` or `en` are used 30% of the time. As a result the probability model has the constraints

$$p(\texttt{dans}|\texttt{in}) + p(\texttt{en}|\texttt{in}) + p(\texttt{á}|\texttt{in}) + p(\texttt{au cours de}|\texttt{in}) + p(\texttt{pendant}|\texttt{in}) = 1$$

$$p(\texttt{dans}|\texttt{in}) + p(\texttt{en}|\texttt{in}) = 3/10$$

These constraints can be fulfilled by an infinite number of models. However following the principle of maximum entropy one would choose the following

$$\begin{aligned} p(\texttt{dans}|\texttt{in}) &= 3/20 \\ p(\texttt{en}|\texttt{in}) &= 3/20 \\ p(\texttt{á}|\texttt{in}) &= 7/30 \\ p(\texttt{au cours de}|\texttt{in}) &= 7/30 \\ p(\texttt{pendant}|\texttt{in}) &= 7/30 \end{aligned}$$

as it distributes the probability evenly between the possibilities to the degree permitted by the constraints. Until now it is intuitively clear what evenly means. This changes for example if the observation

$$p(\texttt{dans}|\texttt{in}) + p(\texttt{á}|\texttt{in}) = 1/2$$

is made. Now a measure for "evenly distributed" is needed. This measure is given by the entropy of the model. The entropy of a model $p(k|X)$ with respect to a set of examples $\mathfrak{X} = \{(X_1, k_1), \ldots, (X_N, k_N)\}$ is given by

$$H(p) = -\sum_{n=1}^{N} \sum_{k=1}^{K} p(k|X_n) \cdot \log p(k|X_n) \tag{2.15}$$

The entropy is maximized when the model is uniformly distributed.

The probability model used has the form

$$p(k|X) = c(X) \cdot \exp\left(\sum_{i=1}^{I} \lambda_i f_i(k, X) \right) \tag{2.16}$$

where $c(X)$ is a normalization term. The f_i are so called feature functions. In computer linguistics they are usually binary functions that return 1 if the input has a specific characteristic and 0 else. However, they are not restricted to being binary. Formally they are given by

$$\begin{aligned} f_i : \{1, \ldots, K\} \times \{1, \ldots, S\} &\mapsto \{0, 1\} \\ (k, X) &\mapsto f_i(k, X) \end{aligned}$$

An example for a feature function could be

$$f(k, X) = \begin{cases} 1 & \text{if } k = \texttt{en} \text{ and } X = \texttt{in} \\ 0 & \text{else} \end{cases}$$

It can be shown that finding the model with the maximum entropy is a convex problem with a unique global maximum. A general algorithms to find this maximum is known as *generalized iterative scaling*, which is described in [Ratnaparkhi 97]. It must be noted that the main problem in maximum entropy modeling is finding good feature functions f_i.

2.4 Local Feature Based Approach

In the local feature based approach that is the subject of this work, overlapping sub-windows, here called local features, are extracted from the training images at interest points. The extracts are then labeled with the name of the class of the image they are from and put together into a set. In testing, again local features are extracted at interest points. For each of these features the nearest neighbor in the training set is searched. Finally, all nearest neighbors vote for their class. The class that has the most votes is then chosen.

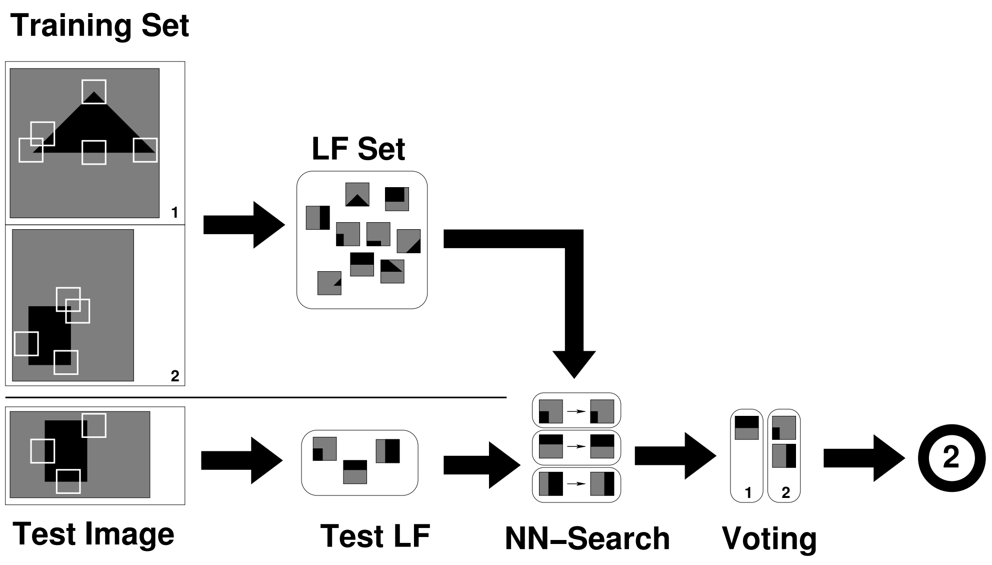


Figure 2.4: Overview of the local feature based approach.

A scheme of this is shown in Figure 2.4. On the top left of the figure are the training images and below is one test image. In the next step the local features are extracted from the training and the test images. Those of the training images are annotated with their class label and joined to a set. Then the nearest neighbor from the training feature set is searched for every test feature. In the following step every nearest neighbor votes for its class. Then that class is chosen, which receives most of the votes.

This is the method on which this work is based, it is described more thoroughly in the next chapter.

Chapter 3

Local Feature Based Classification

It has been suggested earlier that the local features approach presented in [Paredes & Pérez$^+$ 01] is effective for various classification tasks. Here the method is described in more detail. Then the investigations made in the course of this work are presented. The base method that is presented in the next section, is from now on called LF&DV.

3.1 Base Method

The process can be subdivided into the following basic steps

1. Feature extraction.
2. Feature set reduction.
3. Feature dimensionality reduction.
4. Nearest neighbor search.
5. Decision.

as depicted in Figure 3.1.

In most situations it makes sense first to do some preprocessing on the images. This consists of simple signal processing. Usually a brightness normalization is done to stretch the color histogram of images with low contrast. Also it is sometimes useful not to use the image only but its horizontal and vertical derivative as well. This gives further invariance towards brightness variations and contains the information whether a pixel is on an increasing or decreasing edge.

In the feature extraction step, all square sub images, from now on called *local features* (LF), of a fixed size are extracted. The size is usually chosen empirically. The right dimension is then found experimentally. The goal is to find the size that is most discriminative for the given task. That is the scale, at which a feature contains sufficient structure for recognition and still provide invariance towards global non-linear transformations. For face recognition it has been shown to be best to select about the size of an eye, whereas for a radiography classification task the best results have been obtained with extracts that were about two thirds the size of the original image.

Now a huge amount of data has been extracted. For images $X \in \mathbb{R}^{I \times J}$ and local features $x \in \mathbb{R}^{I' \times J'}$ that fulfill $I \times J \gg I' \times J'$ a little less than $(I \times J) \times (I' \times J')$ values have been extracted. Aside of memory and runtime concerns this leads to recognition problems, as many local features are from image regions that are not important for recognition. A local feature of the

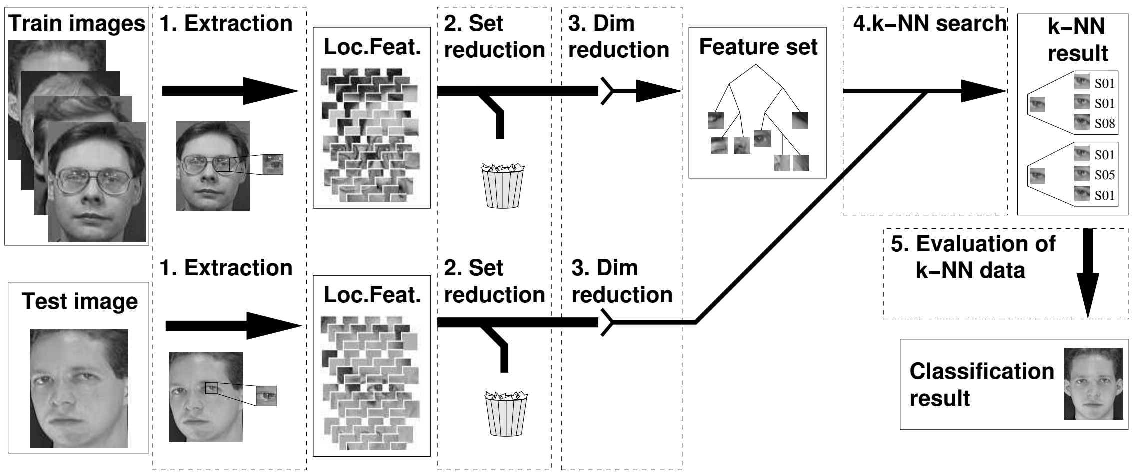


Figure 3.1: A schematic view of the local feature approach with direct voting. The Steps 1-3 are the feature extraction steps and in 4 and 5 the discriminant function is calculated. Preprocessing has been omitted for brevity.

background for example is not important for classification. For those reasons it makes sense to reduce the set of extracted features. The criterion by which this is typically done is local variance. Let X be an image and $\{x_1, \cdots, x_{N_X}\}$ its local features of dimension $D := I' \times J'$. Now a threshold t is chosen and the local feature x_n is taken into the feature set $\mathfrak{X}_X$ if the variance of their pixel values is greater or equal t. Formally this can be written as

$$x_n \in \mathfrak{X}_X \quad \text{iff} \quad \frac{1}{D}\sum_{d=1}^{D}(x_{n,d})^2 - \left(\frac{1}{D}\sum_{d=1}^{D}x_{n,d}\right)^2 \quad \geq \quad t \tag{3.1}$$

This however is only one possible method to reduce the feature set. Another will be presented later. The set of local features of image X that remain after the reduction is called $\mathfrak{X}_X$.

A further problem of recognition is the high dimensionality. The problem is that the feature space increases exponentially with the dimension. If the amount of training data is kept fixed the feature space quickly gets sparse if the dimension of the data increases. Under this condition distance based classifiers often produce unstable results. This problem is known as *curse of dimensionality* in the literature and described in detail by [Hastie & Tibshirani$^+$ 01].

This problem can be prevented by reducing the dimensionality of the feature space, such that the available training data can appropriately populate it. The question is now how to do this reduction. One possibility is to do a sub-sampling, e.g. discard every second component of the feature space. However, this is not recommended as the information that is in those discarded components is then lost. An alternative is to transform the data into another representation with less components without loosing that much information. One possibility is to do a *principal components analysis* on the data. This returns a linear transformation ψ of the same dimensionality, but where the components are sorted by importance, with regard to the squared error criterion. Now it can be chosen freely how many dimensions are to be kept. The decision is analogous to the decision what constitutes valuable information and what is noise. This transformation does lead to good results in LF&DV.

The steps presented so far are those that are executed as well on the training as on the testing data. At this point, the features from the training images are labeled and stored in a KD-tree search structure $\mathfrak{X} = \{(x_{1,1}, k_1), \dots, (x_{1,N_1}, k_1), \quad \cdots \quad, (x_{M,1}, k_M), \dots, (x_{M,N_M}, k_M)\}$. One such structure is described in [Arya & Mount$^+$ 98].

When testing, every local feature x_n of the testing image X is associated with its nearest neighbor $\hat{x}_{m,n'}$ of the training set according to the Euclidean distance.

For every local feature x_n with the nearest neighbor $x_{m,n'} \in \mathfrak{X}$ the a posteriori probability is now estimated by the formula

$$p(k|x_n) = \begin{cases} 1 & \text{if } k = k_{n'} \text{ and } n' = \text{argmin}_{n''}\{\text{d}(x_n, x_{n''})\} \\ 0 & \text{else} \end{cases} \tag{3.2}$$

The a posteriori probabilities of all local features x_n of X are combined using the sum rule

$$p(k|X) = \frac{1}{N_X} \sum_{n=1}^{N_X} p(k|x_n) \tag{3.3}$$

which is a good choice when combining noisy data [Kittler & Hatef$^+$ 98]. The decision is done as usually using the Bayesian decision rule:

$$r(X) = \underset{k}{\text{argmax}}\,\{p(k|X)\} \tag{3.4}$$

This strategy is known as *direct voting scheme* (DV).

The approach does not intend to be invariant towards local and global transformations. But implicitly the unconstrained search does lead to a potentially large amount of invariance, if the training set permits it, as mentioned in the introduction to this work. The invariance towards local and global transformations comes from the variance in the training data. So why does a method like nearest neighbor on entire images not benefit that much from this? Because the transformations on the training image are usually a superposition of various transformations. As a result even though the training corpus covers some variation of one class, the space of all transformations is too large to be reasonably populated. This is the reason why invariant distance measures, like the tangent distance that has been presented in [Simard & Le Cun$^+$ 93], are necessary.

Local features, on the other hand, are totally invariant towards translations, but also they have all kinds of invariances as local changes that are a result of a global change in the training image can be taken from all images at the same time, and so different local variations are simultaneously chosen out of various images approximating a different transformation that was not given in the training.

An example could come from a system that recognizes people from whole body photographs. Assume pictures of one person that have been taken form different distances. As a result the area she covers on these pictures varies from image to image. If the testing image is taken from a position slightly higher than the head of the person, such that the face is proportionally larger than the feet, the head can be compared to local features from a closer shot and the feet from a more distant one.

However, there are various alterations to the base method that can be tried, e.g. to get further invariances, to make a smoother decision and so on. Those variations are explored in this this work and will be presented in the following sections. The results of those variations will be presented in the following chapter.

3.2 Multi-Scale Feature Extraction

In the baseline method the window size is fixed and has to be chosen by hand. The result of this is that the single dimension is chosen that works best for the specified data. It is however probable that there is more than one relevant scale. To classify the radiography of a thorax for example a local feature of the shoulder might be a good indicator but the overall shape of the thorax probably will lead to a good vote as well.

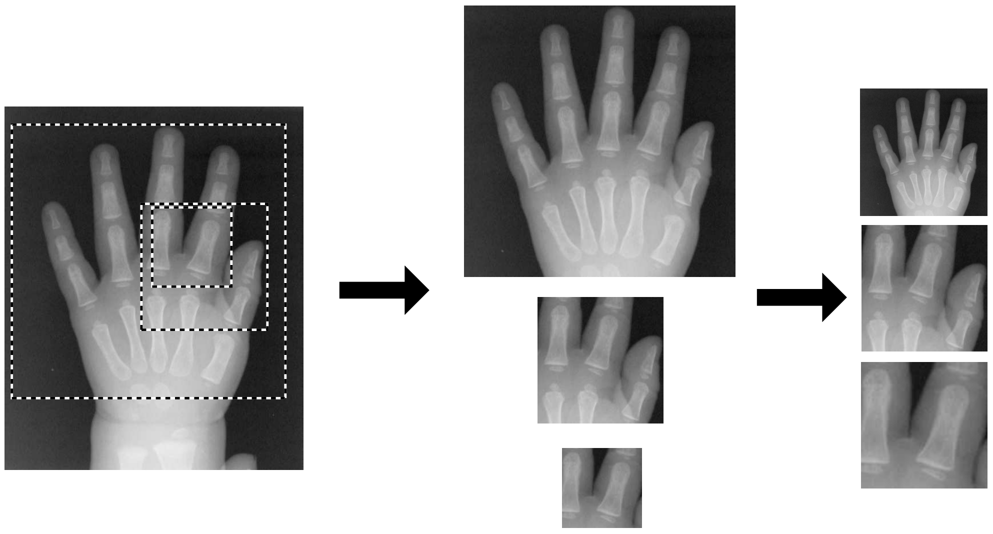


Figure 3.2: Extraction of multi-scale features.

To account for this in LF&DV features can be extracted at multiple scales. The inspiration for this comes from [Fergus & Perona$^+$ 03]. The method proposed in this paper measures the local entropy of differently sized windows of an image X and extracts those N local features with the highest entropy. The extracted features are then transformed to the same size by scaling.

The extension to LF&DV tested proceeds similarly. The local variance is measured at every pixel position for various window sizes. Those windows with a local variance above some threshold are extracted and scaled to a fixed size. This permits it to compare similarities in pictures with different resolutions. The window size is typically scaled to that size that worked best when only using one scale. The decision to compare features of different scales instead of only comparing features from the same scale is the same as the decision to do the unconstrained search of local features in the first place. It is done in the expectance that the context information that is contained in the local features will in most cases lead to a reasonable decision. This belief is plausible given the experiences made with local features.

The further proceeding after the extraction is the same as that described in the beginning of this chapter. That is, we do the reduction of the labeled features using the PCA, and save the features in the KD-tree.

The scaling is done using cubic splines. This has the advantage of being flexible and has proven successful for scaling in other contexts [Lehmann & Gönner$^+$ 99, Gollan 03].

Efficient Local Variance Computation. The LF&DV in general and especially the multi-scale feature extraction require it to compute a large number of local variances. If this is done naively for an image X of size $I \times J$ and local features of the size $D \times D$ the time consumption is proportional to $I \cdot J \cdot D^2$. This can be reduced to $I \cdot J$ with the following method. From X we compute $X^{(1)}, X^{(2)} \in \mathbb{R}^{I \times J}$ by

$$X^{(1)}_{i,j} = \sum_{i'=1}^{i} \sum_{j'=1}^{j} X_{i,j} \tag{3.5}$$

$$X^{(2)}_{i,j} = \sum_{i'=1}^{i} \sum_{j'=1}^{j} (X_{i,j})^2 \tag{3.6}$$

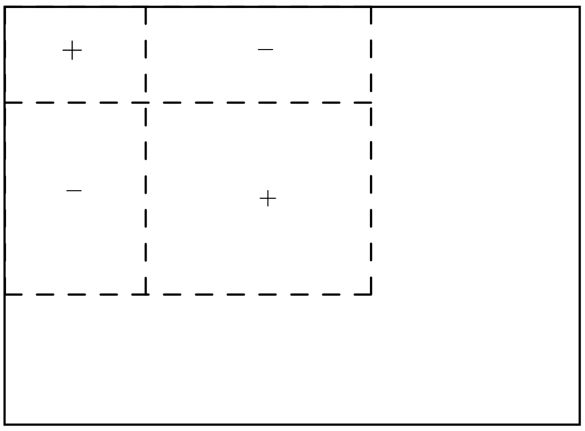


Figure 3.3: Addition chart for fast variance calculation.

$X_{i,j}^{(1)}$ now represents the sum of image values of X in the upper left corner of the image bounded by the pixel $X_{i,j}$ and $X_{i,j}^{(2)}$ is the equivalent for the sum of squared values of X. The variance $\sigma^2_{i,j,i',j'}$ of the sub window $X_{i,j,i',j'}$ is now calculated by

$$\sigma^2_{i,j,i',j'} = \frac{\left(X^{(2)}_{i',j'} - X^{(2)}_{i',j} - X^{(2)}_{i,j'} + X^{(2)}_{i,j}\right)}{(i'-i)\cdot(j'-j)} - \left(\frac{X^{(1)}_{i',j'} - X^{(1)}_{i',j} - X^{(1)}_{i,j'} + X^{(1)}_{i,j}}{(i'-i)\cdot(j'-j)}\right)^2 \quad (3.7)$$

In Figure 3.3 is demonstrated, which areas are added and which ones subtracted.

It can be seen that with this method, every local variance can be computed with 9 additions and 4 multiplications instead of $(i'-i)\cdot(j'-j)+1$ additions and $(i'-i)\cdot(j'-j)+2$ multiplications. The precalculation requires $2\cdot(I\cdot J-I-J+1)$ additions and $I\cdot J$ multiplications. On the *Olivetti research laboratory corpus* with an image size of 112×92 and a typical local feature size of 15×15 the naive method performs 2,325,540 additions and 2,335,830 multiplications per image. However, the method presented here only needs 205,744 additions and 51,464 multiplications per image. That is only 8.8% of the additions and 2.2% of the multiplications are needed for the same result.

3.3 Dataset Reduction

As has been said earlier, the local features are not extracted at all positions, but are restricted by a local variance threshold. If all local features are extracted, several problems emerge. One is that the amount of features is too large to be used easily. That is, the memory consumption is beyond the typical size of main memory. For example on the ORL corpus that is presented in Section 4.1 the extracted local features typically have a size of 15×15. Every image in this corpus is 112 pixel high and 92 pixel wide. That means that 7,469 local features are extracted per image. As the corpus contains 200 training images the training corpus would contain 1,493,800 local features. After the PCA, each local feature uses 320 bytes. This leads to a memory consumption of more than 450 MBs. This is without the search structure for the recognition task. The second and more relevant problem is that the search time increases as the amount of features increases. The third inconvenience of taking all local features is that local features are being included that are not discriminative. So for example sub-images from the background are not relevant if the task is face recognition. So it is tried to keep local features that are considered to be relevant. This is typically done by calculating the variance of the gray values of the local features and discarding all those with a variance that is below some threshold. However, other schemes for reduction can be applied, that lead to better discrimination results.

Feature Relevance Estimation

One possibility is to learn which features are suited for recognition. This can then be expressed by the probability distributions $p(g|x)$ and with $g \in \{0, 1\}$ where $g = 1$ can be seen as the class of discriminative features and $g = 0$ is that of non-discriminative features. After the distributions have been estimated, the training feature set $\mathfrak{X}$ is created by

$$\mathfrak{X} = \left\{ x_{m,n} \quad : \quad 1 = \underset{g \in \{0,1\}}{\operatorname{argmax}} \{p(g|x_{m,n})\} \right\} \tag{3.8}$$

The feature set $\mathfrak{X}_X$ of the text image X is also reduced using the same distributions

$$\mathfrak{X}_X = \left\{ x_n \quad : \quad 1 = \underset{g \in \{0,1\}}{\operatorname{argmax}} \{p(g|x_n)\} \right\} \tag{3.9}$$

After $\mathfrak{X}$ and $\mathfrak{X}_X$ have been extracted, we proceed as usually with LF&DV.

To estimate the probability distributions $p(g|x)$ for $g \in \{0, 1\}$ the following methodology is applied. For the labeled training images $(X_1, k_1), \dots, (X_M, k_M)$ a set $\mathfrak{X}' = \bigcup_{m=1}^{M} \mathfrak{X}'_{X_m}$ of all local features is extracted. $\mathfrak{X}'$ is split into the two disjunct sets

$$\mathfrak{H} = \{x_{m,n} \quad : \quad k_m = k_{m'} \wedge x_{m',n'} = \underset{x \in \mathfrak{X}' \setminus \mathfrak{X}'_{X_m}}{\operatorname{argmin}} \{\mathrm{d}(x, x_{m,n})\} \tag{3.10}$$

and

$$\overline{\mathfrak{H}} = \mathfrak{X}' \setminus \mathfrak{H} \tag{3.11}$$

These two sets are used to train the Gaussian mixture densities for $p(x|g)$ using the *fuzzy K-means* algorithm for Gaussian mixture densities. This algorithm was described in Section 2.3.2. The a-posteriori probabilities for $p(g|x)$ are now computed using Bayes' formula

$$p(g|x) = \frac{p(x|g) \cdot p(g)}{p(x)} \tag{3.12}$$

For the decision rule this is equivalent to

$$p(x|g = 1) \cdot \frac{|\mathfrak{H}|}{|\mathfrak{X}'|} \tag{3.13}$$

and

$$p(x|g = 0) \cdot \frac{|\overline{\mathfrak{H}}|}{|\mathfrak{X}'|} \tag{3.14}$$

This method can be seen as optimizing the feature extraction on the training set. The results achieved this way can be found in Section 5.2.

3.4 Dimensionality Reduction

As described earlier the amount of training data needed to represent a density sufficiently is in an exponential relation with the amount of dimensions of the feature representation. This was called curse of dimensionality, which has the result that distance based classifiers tend to produce unstable classifications [Hastie & Tibshirani$^+$ 01]. The local features usually have a window size from 15×15 to 19×19 pixel per feature which results in vectors of 225 to 361 dimensions. Such a feature space would need more then the given 100.000 to 1.000.000 local features that are normally extracted in training to be sufficiently populated. For this reason, the features should be

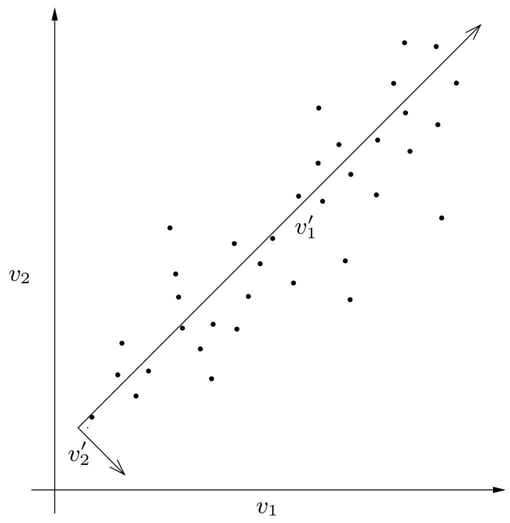


Figure 3.4: The data points are represented in the space that is spanned by v_1 and v_2 but there is a strong correlation of the two dimensions which is represented by v_1'. v_2' is orthogonal to v_1' and represents the least significant component.

represented in a more compact way. This is done by reducing the dimension. One way to reduce the dimension is by scaling down the local features. This is similar to discarding components in a regular manner. However, if components of a representation space are discarded we loose information which might be important for discrimination. To avoid this, it is usually attempted to represent the data with less dimensions in a way that most valuable information is preserved.

This transformation and reduction can be done either linearly or non-linearly. An overview of methods can be found in [Jain & Duin$^+$ 00]. Here only linear methods are examined.

3.4.1 Principal Component Analysis

One possible concern in the reduction of the dimensionality is to find the linear transformation $\psi : \mathbb{R}^{D' \times D}$ that, given the training data $\{x_1, \ldots, x_N\}$, minimizes the squared error of the representation:

$$\psi = \underset{\psi'}{\operatorname{argmin}} \left\{ \sum_{n=1}^{N} \|x_n - \psi'^T \cdot \psi' \cdot x_n\|^2 \right\} \tag{3.15}$$

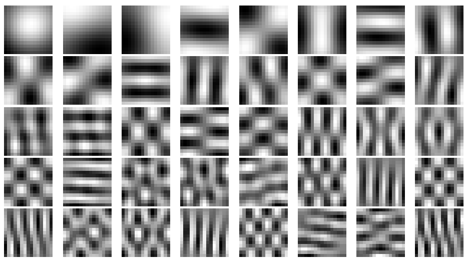
ψ can be calculated by computing the covariance matrix Σ of the training data. Its eigenvectors $v_1, \ldots, v_D$ are then calculated and the D' eigenvectors with the largest eigenvalues are chosen to span the new subspace. Figure 3.4 gives an example in a two dimensional space.

As we usually want $\|\psi \cdot x_n - \psi \cdot x_{n'}\| - \|x_n - x_{n'}\|$ to be as small as possible, the eigenvectors in ψ should be normalized to unit length.

This method has the advantage that it returns the eigenvectors sorted by importance with regard to the representation error such that it can be chosen freely what dimension the sub space should have.

3.4.2 Discrete Cosine Transform

One disadvantage of the principal component analysis is that it has to be calculated on the training data. This means an extra step that has to be performed. It would reduce the amount of


(a) PCA 40

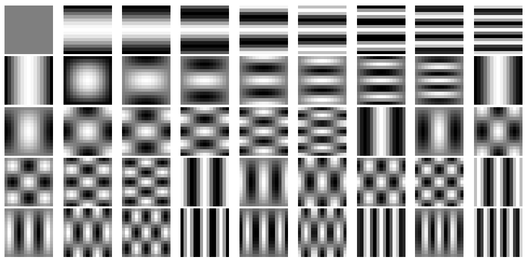

(b) DCT triangular 45

Figure 3.5: (a) The 40 first principal components, computed on the ORL face recognition corpus and (b) the 45 smallest frequencies using the DCT.

steps that have to be performed on training, to have a transformation that represents the essence of the data equally well, while being independent of the task. Taking a look at the principal components computed in real tasks, as can be seen in Figure 3.5 reveals that many components have a wave like structure. This fact and the experiences with the *discrete cosine transform* (DCT) for image compression (the well known JPEG-compression [Int 92] uses the cosine transform to get an efficient representation of the image data) suggests the cosine transform for the sub space calculation. Furthermore, it is known that the DCT decorrelates image data with the property that the correlation of the pixel values depends only on the relative position of the pixels.

The idea of the discrete cosine transform is to represent the image as amplitudes of waves with different periods. For an image $X \in \mathbb{R}^{I \times J}$ the cosine transformed image X' is calculated as described in [Press & Teukolsky$^+$ 02], by

$$X'_{i,j} = \sum_{i'=0}^{I-1} \sum_{j'=0}^{J-1} X_{i',j'} \cos \frac{\pi i(i' + .5)}{I} \cos \frac{\pi j(j' + .5)}{J} \tag{3.16}$$

However, this is just another representation of the data with the same amount of dimensions. The representation is split into different scale information. The positions $i'_{m,n}$ for larger m and n stand for higher frequency information, meaning higher detail. The dimensionality can now be reduced by discarding such high frequency information. This can be done in various ways that are all parameterized by some threshold t:

(a) Extraction of the square given by $n, m \leq t$.

(b) Extraction of the triangle given by $n + m \leq t$.

(c) Extraction of the quarter circle given by $\sqrt{n^2 + m^2} \leq t$.

The three methods are visualized in Figure 3.6. It can be seen that (a) favors high frequencies of the superposed waves compared to the simple ones. In (b) the higher frequency components are taken from the simple waves instead of the superposed ones and in (c) neither one is favored when extracting high frequencies.

3.4.3 Linear Discriminant Analysis

The two methods that have been presented so far reduce the dimension of the data without taking into account the class information. This can lead to unwanted transformations if the dimensions

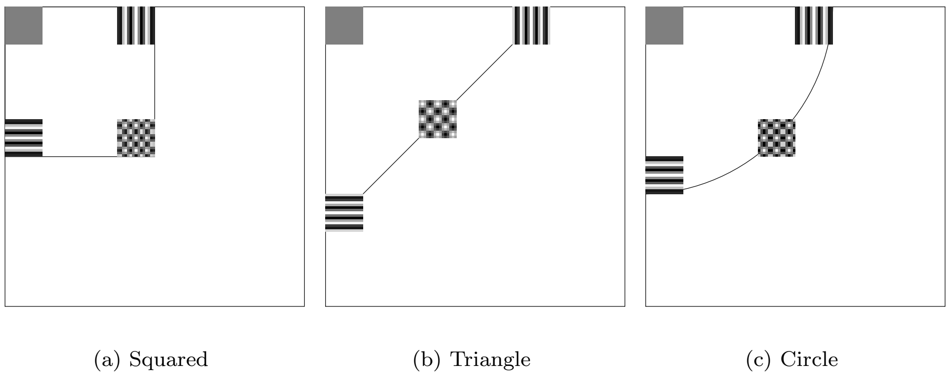


Figure 3.6: To reduce the dimensionality of the representation, only the low frequency components of the DCT are taken. These frequencies are within the separated region on the upper left of (a), (b) and (c).

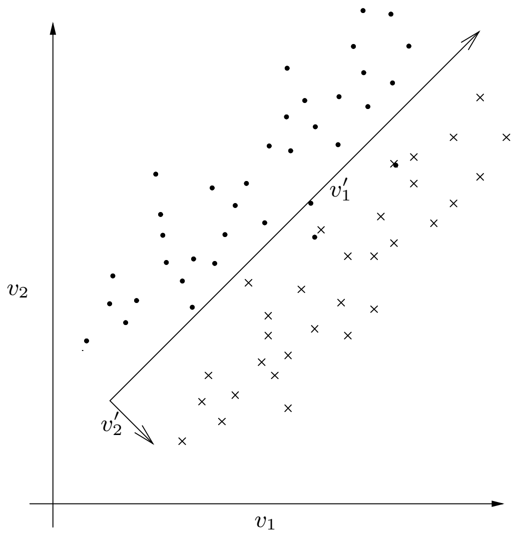


Figure 3.7: If the class means are close to one another and the class covariances are similar showing the strongest component perpendicular to the difference vector of one mean to the other, the PCA leads to bad results.

for the best representation are not those that permit the best discrimination. This situation is shown in Figure 3.7. However, for classification it is useful to use transformations that increase the distance between classes while reducing the within-class distances.

The *linear discriminant analysis* does this. It calculates a linear transformation that separates the class means as much as possible, while keeping the within-class distances constant. To calculate this transformation given the training data $\mathfrak{X} = \{(x_1, k_1), \ldots, (x_N, k_N)\}$, the overall mean has to be computed, with

$$\mu = \frac{1}{N} \sum_{n=1}^{N} x_n \tag{3.17}$$

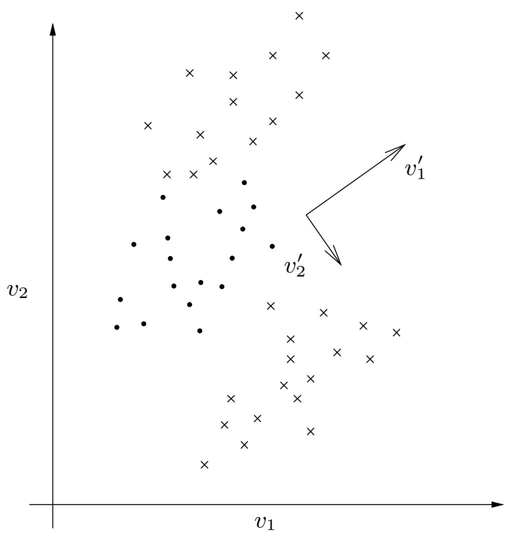


Figure 3.8: LDA is not suited to deal with clustered data.

and the class means with

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \delta(k, k_n) \cdot x_n \tag{3.18}$$

where N_k is given by

$$N_k = \sum_{n=1}^{N} \delta(k, k_n)$$

Then the *within-class scatter matrix* and the *between-class scatter matrix* are calculated by

$$S_W = \sum_{n=1}^{N} (x_n - \mu_{k_n})^2 \tag{3.19}$$

and

$$S_B = \sum_{k=1}^{K} N_k \cdot (\mu_{k_n} - \mu)(\mu_{k_n} - \mu)^T \tag{3.20}$$

The transformation is calculated by solving the generalized eigenvalue problem that, for $v \in \mathbb{R}^D$ and $V \in \mathbb{R}^{D \times D}$, is given by

$$S_B \cdot V = v \cdot S_W \cdot V \tag{3.21}$$

Where the i-th column of V contains the i-th eigenvector and the i-th component of v contains the i-th eigenvalue. The eigenvectors are now sorted according to their eigenvalues in descending order to the matrix $V' = \langle v_1, \ldots, v_D \rangle$. We now get the reduction matrix $\psi = \langle v_1, \ldots, v_{K-1} \rangle$ by discarding the $D - K + 1$ eigenvectors with the smallest eigenvalues.

It can be problematic to reduce a D dimensional space to $K - 1$ dimensions as it might not be sufficiently large to represent the important aspects of the classes. This is especially the case, if each class is made up of several cluster as is depicted in Figure 3.8. To circumvent this problem the classes can be split into pseudo classes. This can be done using a clustering algorithm, e.g. one of those presented in Section 2.3.

3.5 Probability Model

The probability model used up to now (see equation (3.2)) is binary, meaning that only the class that the nearest neighbor of the local feature belongs to gets the probability 1 and all other classes contain the probability 0. This might not be the best choice, especially if two classes have similar images. A solution for this problem can be to use more than one nearest neighbor and to weight the neighbors according to their distance from the test sample. A standard method to accomplish this is to use kernel densities.

3.5.1 Kernel Densities

The idea here is to approximate the densities of the data in feature space by smearing each sample point with a Gaussian density. The formula for this is

$$\phi(x, k) = \prod_{d=1}^{D} \frac{1}{\sqrt{2\pi v_k}} \exp\left(-\frac{(x_d)^2}{2v_k}\right) \tag{3.22}$$

as described in [Ney 00]. The parameter v_k is used to set the aperture of the densities. It is usually estimated from the class specific variance σ_k^2 by $v = \alpha\sigma_k^2$ with the empirical parameter $\alpha \in \mathbb{R}$. In practice, we used a pooled variance $\sigma_k^2 = \sigma^2$ as it led to better results. This is shown in Chapter 5. The probability $p(x|k)$ using the training data $x_{k,1}, \ldots, x_{k,N_k}$ is now estimated by

$$p(x|k) = \frac{1}{N_k} \sum_{n=1}^{N_k} \phi(x - x_{k,n}, k) \tag{3.23}$$

The a posteriori probability $p(k|x)$ for the local feature x is calculated using

$$p(k|x) = \frac{p(x|k)p(k)}{\sum_{k'=1}^{K} p(x|k')p(k')} \tag{3.24}$$

The a-priori probability $p(k)$ is usually assumed to be $\frac{1}{K}$ for all classes. In practice, this assumption even improves the results as classes that have many representatives have a higher probability to hold the nearest neighbor, such that the $p(k)$ is contained in the nearest neighbor search implicitly.

The probabilities $p(k|x)$ for the local features $x \in \mathfrak{X}_X$ of image X are then combined using the sum rule.

3.5.2 Direct Voting as Special Case of Kernel Densities

If we consider the case of a pooled covariance matrix (which leads to better results in local feature based recognition as will be seen in Chapter 5), it can be shown that direct voting is a special case of kernel densities. This will be proven in the following. For the sake of simplicity and readability the one dimensional case will be regarded here. The a-posteriori probability that a feature x belongs to the class k is, as said in the last section, given by

$$\begin{aligned} p(k|x) &= \frac{\sum\limits_{x' \in \mathfrak{X}_k} \frac{1}{\sqrt{2\pi\alpha\sigma^2}} \exp\left(-\frac{\mathrm{d}(x,x')^2}{2\alpha\sigma^2}\right)}{\sum\limits_{k'} \sum\limits_{x' \in \mathfrak{X}_{k'}} \frac{1}{\sqrt{2\pi\alpha\sigma^2}} \exp\left(-\frac{\mathrm{d}(x,x')^2}{2\alpha\sigma^2}\right)} \\ &= \frac{\sum\limits_{x' \in \mathfrak{X}_k} \exp\left(-\frac{\mathrm{d}(x,x')^2}{2\alpha\sigma^2}\right)}{\sum\limits_{x' \in \mathfrak{X}} \exp\left(-\frac{\mathrm{d}(x,x')^2}{2\alpha\sigma^2}\right)} \end{aligned} \tag{3.25}$$

As the variance σ^2 and the multiplier α are constants, the normalization term can be omitted. For all the distances $\mathrm{d}(x, x')$ between our local feature and the features $x' \in \mathfrak{X}$ we define

$$\mathrm{d}^2_{\min} = \min_{x' \in \mathfrak{X}} \{\mathrm{d}(x, x')^2\} \quad , \quad \gamma_{\min} = \frac{\mathrm{d}^2_{\min}}{2\alpha\sigma^2}$$

This value can be extracted from all exponents and we get

$$\begin{aligned} p(k|x) &= \frac{\exp\left(-\gamma_{\min}\right) \cdot \sum\limits_{x' \in \mathfrak{X}_k} \exp\left(-\left(\frac{\mathrm{d}(x,x')^2}{2\alpha\sigma^2} - \gamma_{\min}\right)\right)}{\exp\left(-\gamma_{\min}\right) \cdot \sum\limits_{x' \in \mathfrak{X}} \exp\left(-\left(\frac{\mathrm{d}(x,x')^2}{2\alpha\sigma^2} - \gamma_{\min}\right)\right)} \\ &= \frac{\sum\limits_{x' \in \mathfrak{X}_k} \exp\left(-\frac{\mathrm{d}(x,x')^2}{2\alpha\sigma^2} + \gamma_{\min}\right)}{\sum\limits_{x' \in \mathfrak{X}} \exp\left(-\frac{\mathrm{d}(x,x')^2}{2\alpha\sigma^2} + \gamma_{\min}\right)} \end{aligned} \tag{3.26}$$

which can be simplified again. There now exists at least one $x_{\min}$, such that

$$\frac{\mathrm{d}(x, x_{\min})^2}{2\alpha\sigma^2} - \gamma_{\min} = 0 \tag{3.27}$$

Assume first that there only exists one such $x_{\min}$ with its associated class $k_{\min}$. In this case $x_{\min} \in \mathfrak{X}_k$ or $x_{\min} \notin \mathfrak{X}_k$. This leads to the following equation

$$\begin{aligned} p(k|x) &= \frac{\delta(k, k_{\min}) \cdot \exp\left(-\frac{\mathrm{d}(x,x_{\min})^2}{2\alpha\sigma^2} + \gamma_{\min}\right) + \sum\limits_{x' \in \mathfrak{X}_k \setminus x_{\min}} \exp\left(-\frac{\mathrm{d}(x,x')^2}{2\alpha\sigma^2} + \gamma_{\min}\right)}{\exp\left(-\frac{\mathrm{d}(x,x_{\min})^2}{2\alpha\sigma^2} + \gamma_{\min}\right) + \sum\limits_{x' \in \mathfrak{X} \setminus x_{\min}} \exp\left(-\frac{\mathrm{d}(x,x')^2}{2\alpha\sigma^2} + \gamma_{\min}\right)} \\ &= \frac{\delta(k, k_{\min}) \cdot \exp\left(-\gamma_{\min} + \gamma_{\min}\right) + \sum\limits_{x' \in \mathfrak{X}_k \setminus x_{\min}} \exp\left(-\frac{\mathrm{d}(x,x')^2}{2\alpha\sigma^2} + \gamma_{\min}\right)}{\exp\left(-\gamma_{\min} + \gamma_{\min}\right) + \sum\limits_{x' \in \mathfrak{X} \setminus x_{\min}} \exp\left(-\frac{\mathrm{d}(x,x')^2}{2\alpha\sigma^2} + \gamma_{\min}\right)} \\ &= \frac{\delta(k, k_{\min}) + \sum\limits_{x' \in \mathfrak{X}_k \setminus x_{\min}} \exp\left(-\frac{\mathrm{d}(x,x')^2}{2\alpha\sigma^2} + \gamma_{\min}\right)}{1 + \sum\limits_{x' \in \mathfrak{X} \setminus x_{\min}} \exp\left(-\frac{\mathrm{d}(x,x')^2}{2\alpha\sigma^2} + \gamma_{\min}\right)} \end{aligned} \tag{3.28}$$

Now it can been seen that

$$\lim_{\alpha \to 0} \frac{\delta(k, k_{\min}) + \sum\limits_{x' \in \mathfrak{X}_k \setminus x_{\min}} \exp\left(\frac{-\mathrm{d}(x,x')^2 + \mathrm{d}^2_{\min}}{2\alpha\sigma^2}\right)}{1 + \sum\limits_{x' \in \mathfrak{X} \setminus x_{\min}} \exp\left(\frac{-\mathrm{d}(x,x')^2 + \mathrm{d}^2_{\min}}{2\alpha\sigma^2}\right)} = \delta(k, k_{\min}) \tag{3.29}$$

which in turn is equal to direct voting. Now assume there are more than one training features $x' \in \mathfrak{X}$ with $\mathrm{d}(x', x) = \mathrm{d}(x_{\min}, x)$. The set of those features shall be $\mathfrak{X}_{\min}$. The a-posteriori probability is now

$$p(k|x) = \frac{\|\mathfrak{X}_{\min} \cap \mathfrak{X}_k\|}{\|\mathfrak{X}_{\min}\|} \tag{3.30}$$

In direct voting this situation has to be caught and some rule for tie-brake has to be given.

As kernel densities are a generalization of direct voting, we have the security that it must be at least as good as direct voting in the case of $\alpha \to 0$.

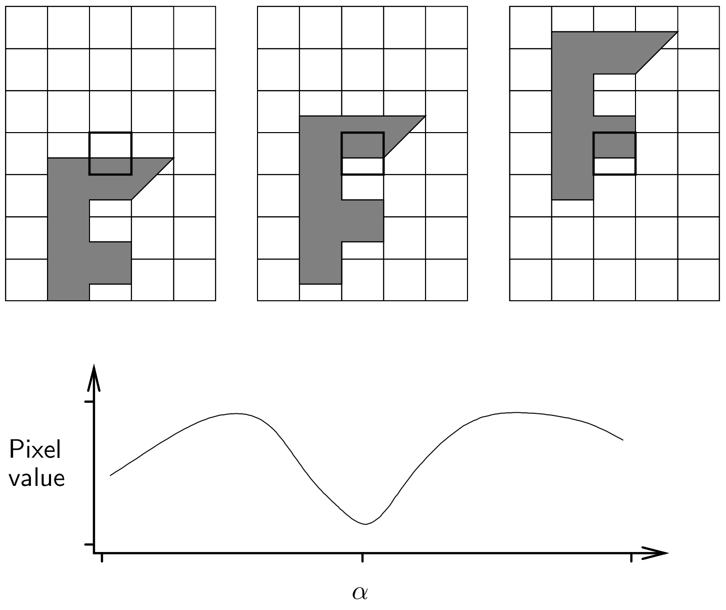


Figure 3.9: As the image is translated, the depicted component marks a path through a 1 dimensional feature sub space that cannot be described easily analytically.

3.6 Tangent Distance

In image classification one often has to deal with the situation that the objects are subject to small transformations, as rotations, scaling, shearing, etc. that do not affect the class membership. Appearance based methods like nearest neighbor using Euclidean or Mahalanobis distance run into problems with such transformations as for example the same object that is slightly shifted can result in large pixel distances.

In 1993 Simard et al. presented a new distance measure that is invariant to small transformations. The idea is to regard such a transformation $t(X, \alpha)$ that depends on the parameter α on the image $X \in \mathbb{R}^{I \times J}$ as a manifold in $I \times J$ dimensional space

$$\mathfrak{M}_X = \{t(X, \alpha) : \alpha \in \mathbb{R}^L\} \subset \mathbb{R}^{I \times J} \tag{3.31}$$

The squared distance between two images X and X' is measured as the minimal squared distance between the manifolds $\mathfrak{M}_X$ and $\mathfrak{M}'_X$ by

$$\mathrm{d}_{\mathrm{TD}}(X, X') = \min_{\alpha, \alpha' \in \mathbb{R}^L} \{\|t(X, \alpha) - t(X', \alpha')\|^2\} \tag{3.32}$$

Unfortunately the manifolds have no analytic expression in general. Furthermore, finding the minimal distances is a hard non-linear optimization problem with possibly various local minima. The effect of a translation on one pixel of an image is visualized in Figure 3.9. It also sometimes harms the recognition result to find the minimal distance on the entire manifolds. Consider the task of distinguishing a "9" and a "6" using the *manifold distance* where one transformation is a rotation. The approach of [Simard & Le Cun$^+$ 93] consists in approximating $\mathfrak{M}_X$ by the tangents to $\mathfrak{M}_X$ in the point X. Thus not $\mathfrak{M}_X$ is considered but

$$\overline{\mathfrak{M}}_X = \left\{ X + \left\langle \frac{\partial t(X, \alpha)}{\partial \alpha_1}, \cdots, \frac{\partial t(X, \alpha)}{\partial \alpha_L} \right\rangle \right\} \tag{3.33}$$

By this only small transformation to X will be close to it while large transformations on the image will return large distances. In addition the distance calculation consists now in finding minimal distance between two affine sub spaces which is easily computable.

The tangent distance of X and X' can now be calculated either using the tangents of only one or of both images. If the tangents of only one image are used we speak of the single sided (d_{SS}) and if those of both are taken of the double sided tangent distance (d_{DS}). The tangent $\frac{\partial t(X,\alpha)}{\partial \alpha_l}$ will be denoted as $\overline{t}_{X,l}$.

$$\mathrm{d}_{SS}(X, X') = \min_{\alpha \in \mathbb{R}^L} \left\{ \left\| X + \sum_{l=1}^{L} \alpha_l \cdot \overline{t}_{X,l} - X' \right\| \right\} \tag{3.34}$$

$$\mathrm{d}_{DS}(X, X') = \min_{\alpha, \alpha' \in \mathbb{R}^L} \left\{ \left\| X + \sum_{l=1}^{L} \alpha_l \cdot \overline{t}_{X,l} - X' + \sum_{l=1}^{L} \alpha'_l \cdot \overline{t}_{X',l} \right\| \right\} \tag{3.35}$$

It is clear that d_{SS} is not symmetrical, so one has to choose whether to take the training or the test images tangents.

In [Keysers & Paredes$^+$ 02] it is shown that combining LF&DV with the tangent vector approach from [Keysers & Dahmen$^+$ 00] leads to improvements on character recognition. However, the proposed combination is an outer one, meaning that the a posteriori probabilities of both methods are combined and that combination is used for the decision taking. The method used here can be seen as inner combination of both methods. This means that we use the tangent distance to find the nearest neighbor of the local features and then estimate the probability.

The tangent distance is calculated on the image directly. However, here the image is not used directly, but it is PCA-transformed before the distance calculation. This leads to the problem that the tangent distance does not make sense when applied to the PCA-transformed. A shift on the PCA-transformed results in some wave shift in the original image, which is not what we want. Furthermore, it is not clear how the PCA-transformed should be interpreted as image. Fortunately, the PCA is a linear transformation. This allows us to calculate the tangents on the features before we do the transformation. The tangents can then be transformed as if they were local features.

Now that the basic principles have been presented, the tangents that have been extracted will be presented. They are the same as those presented in [Keysers 00]. The first six tangents come from affine transformations on the images. If $(i, j)^T$ is a point in the image, the transformed position $(i', j')^T$ for an affine transformation can be described by

$$\begin{pmatrix} i' \\ j' \end{pmatrix} = \begin{pmatrix} 1 + \alpha_1 & \alpha_2 \\ \alpha_3 & 1 + \alpha_4 \end{pmatrix} \cdot \begin{pmatrix} i \\ j \end{pmatrix} + \begin{pmatrix} \alpha_5 \\ \alpha_6 \end{pmatrix} \tag{3.36}$$

The tangents to these transformations on an image x are denoted by $\overline{x}^1, \ldots, \overline{x}^6$. They are:

- horizontal translation:
  $\alpha_l = 0, l = 1, 2, 3, 4, 6 \qquad i' = i + \alpha_5 \qquad j' = j$

$$\overline{x}^1(i, j) = \lim_{\alpha_5 \to 0} \frac{x(i + \alpha_5, j) - x(i, j)}{\alpha_5} \tag{3.37}$$

- vertical translation:
  $\alpha_l = 0, l = 1, \ldots, 5 \qquad i' = i \qquad j' = j + \alpha_6$

$$\overline{x}^2(i, j) = \lim_{\alpha_6 \to 0} \frac{x(i, j + \alpha_6) - x(i, j)}{\alpha_6} \tag{3.38}$$

- rotation:
  $\alpha_l = 0, l = 1,4,5,6 \qquad \alpha_2 = -\alpha_3 \qquad i' = i + \alpha_2 j \qquad j' = j - \alpha_2 i$

$$\begin{aligned} \overline{x}^3(i,j) &= \lim_{\alpha_2 \to 0} \frac{x(i+\alpha_2 j, j - \alpha_2 i) - x(i,j)}{\alpha_2} \\ &= \lim_{\alpha_2 \to 0} \frac{x(i+\alpha_2 j, j-\alpha_2 i) - x(i, j-\alpha_2 i)}{\alpha_2} + \lim_{\alpha_2 \to 0} \frac{x(i, j-\alpha_2 i) - x(i,j)}{\alpha_2} \\ &= j\overline{x}^1(i,j) - i\overline{x}^2(i,j) \end{aligned} \tag{3.39}$$

- scaling:
  $\alpha_l = 0, l = 2,3,5,6 \qquad \alpha_1 = \alpha_4 \qquad i' = i + \alpha_1 i \qquad j' = j + \alpha_1 j$

$$\overline{x}^4(i,j) = i\overline{x}^1(i,j) + j\overline{x}^2(i,j) \tag{3.40}$$

- axis deformation:
  $\alpha_l = 0, l = 1,4,5,6 \qquad \alpha_2 = \alpha_3 \qquad i' = i + \alpha_3 j \qquad j' = j + \alpha_3 i$

$$\overline{x}^5(i,j) = j\overline{x}^1(i,j) + i\overline{x}^2(i,j) \tag{3.41}$$

- diagonal deformation:
  $\alpha_l = 0, l = 2,3,5,6 \qquad \alpha_1 = -\alpha_4 \qquad i' = i + \alpha_4 i \qquad j' = j - \alpha_4 j$

$$\overline{x}^6(i,j) = i\overline{x}^1(i,j) - j\overline{x}^2(i,j) \tag{3.42}$$

Note that equation 3.39 above does not exactly match the transformation named, as the rotation with an angle of ϕ would correctly be given by

$$\begin{pmatrix} 1+\alpha_1 & \alpha_2 \\ \alpha_3 & 1+\alpha_4 \end{pmatrix} = \begin{pmatrix} \cos\phi & sin\phi \\ -sin\phi & cos\phi \end{pmatrix} \tag{3.43}$$

However, this is not important as we do not look for an entire rotation, but only for the situation $\phi \to 0$. Under this condition both formulas give the same result. In any case this is not important, as the transformations 3.37 to 3.42 span all possible affine deformations. One could equally well use $\alpha_{l'} = 0, l' \neq l$ and then define $\overline{x}^l$ as

$$\overline{x}^l(i,j) = \lim_{\alpha_l \to 0} \frac{x(i \cdot (1+\alpha_1) + j\alpha_2 + \alpha_5, i\alpha_3 + j \cdot (1+\alpha_4) + \alpha_6) - x(i,j)}{\alpha_l} \tag{3.44}$$

The subspace spanned by the tangents would be the same. But in that case we could not give the tangents such catchy names.

Additionally to the given affine transformations the three further tangents $\overline{x}^7, \overline{x}^8 and \overline{x}^9$ are calculated. They are given as

- Line thickness:

$$\overline{x}^7 = (\overline{x}^1)^2 + (\overline{x}^2)^2 \tag{3.45}$$

- Additive brightness:
  $\alpha \in \mathbb{R}$

$$\overline{x}^8(i,j) = \lim_{\alpha \to 0} \frac{(\alpha + x(i,j)) - x(i,j)}{\alpha} \tag{3.46}$$

- Multiplicative brightness:
  $\alpha \in \mathbb{R}$

$$\overline{x}^9(i,j) = \lim_{\alpha \to 0} \frac{\alpha \cdot x(i,j) - x(i,j)}{\alpha} \tag{3.47}$$

Note that the tangents $\overline{x}^8$ and $\overline{x}^9$ are not approximations to a curve, but do represent the transformation exactly. Also note that if the tangent for the multiplicative brightness $\overline{x}^9$ transformation were to be used in the double sided tangent distance d_{DS}. This would always give the distance 0. This is because the origin always lies on $\overline{x}^9$.

3.7 Log-Linear Models

Until now all local features are assumed to have equal weight, or a weight that is computed from their distance to the reference feature. However, it would be nice to be able to learn their relevance from the training data. One way of representing weights in a probability measure, is using log-linear models. Those are frequently used in natural language processing. This framework assumes that some feature functions $f_1, \dots, f_S$ with $f_s : \mathbb{R}^{I \times J} \times \{1, \dots, K\} \mapsto \{0, 1\}$ are given. The goal is now to model a probability distribution that reflects the frequency with that the feature functions fire on the training data $\{(X_1, k_1), \dots, (X_N, k_N)\}$ without making additional assumptions. So for example if a feature function f_s does not fire for any (X_n, k_n) on the training data, we do not want it to be relevant for the probability calculation. Two patterns X and X' for that $f_{s'}(X, k) = f_{s'}(X', k)$ for all $s' \neq s$ and $k \in \{1, \dots, K\}$, and that satisfy $f_s(X, k) \neq f_s(X', k)$ for some k, should have the same a posteriori probabilities $p(k|X) = p(k|X')$ for all k.

Usually such probabilities are modeled with log-linear models. These have the form

$$p(k|X) = \frac{\exp\left(\sum_s \lambda_s f_s(X, k)\right)}{\sum_{k'} \exp\left(\sum_s \lambda_s f_s(X, k')\right)} \tag{3.48}$$

The goal is now to train the parameters λ_s such that the probabilities reflect the observations on the training data but are the least compromising otherwise. For this we calculate the values

$$F_s = \sum_n f_s(X_n, k_n) \tag{3.49}$$

The probability is now modeled such that

$$\sum_k \sum_n p(k|X_n) \cdot f_s(X_n, k) = F_s \tag{3.50}$$

is given and p is a probability distribution. This can be done with *maximum entropy* training. Here the goal is to fulfill (3.50) and otherwise maximize

$$-\sum_k \sum_n p(k|X_n) \cdot \log(p(k|X_n)) \tag{3.51}$$

An introduction to maximum entropy modeling can be found in [Ratnaparkhi 97].

A method called *generalized iterative scaling* exists to train the parameters λ_s of (3.48) given the feature functions. So the training of the model is solvable and the difficulty with maximum entropy modeling is to find suited feature functions. In [Keysers & Och$^+$ 02] the following feature functions $f'_1, \dots, f'_S$ have been proposed:

$$\begin{aligned} f'_s : \mathbb{R}^{I \times J} \times \{1, \dots, K\} &\mapsto \mathbb{R} \\ (X, k) &\mapsto X_{(s \mod I, \lfloor s/J \rfloor)} \end{aligned} \tag{3.52}$$

In this article KEYSERS et al. describe the relation between Gaussian single densities and this maximum entropy model. However, this approach has not been evaluated here as methods based on single prototypes are not useful in combination with local features. This is because the variance of the features within one class is too large to be represented by only one prototype. The method that is presented here is closer to the language processing way of using maximum entropy.

Assume the image X and its local features $x_1, \dots, x_{N_X}$. For every training feature from the training set $x_{n'} \in \mathfrak{X}$ and every class k', the feature function

$$f_{n,k'}(x, k) = \delta(k', k) \cdot \delta(n, \underset{n'}{\operatorname{argmin}}\{\mathrm{d}(x_{n'}, x)\}) \tag{3.53}$$

is defined such that there are in total $K \cdot |\mathfrak{X}|$ feature functions. The idea behind this is that it should be learned how good a feature is suited to predict a specific class. When looking at (3.48) and (3.53), it can be seen that by setting $\lambda_{n,k'}$ to

$$\lambda_{n,k'} = \delta(k_n, k') \tag{3.54}$$

if k_n is the class label of x'_n the log-linear classifier leads to the same classification decision as LF&DV with respect to the Bayesian decision rule. So log-linear models must be at least as good as LF&DV. At least on the training data, as in the testing features are considered that have not been seen in the training. The hope is now that the discriminative training with maximum entropy will further improve the results on the training and testing data.

Chapter 4

Databases

The evaluation of the methods that have been developed in the course of this diploma thesis has been done mainly on four classification tasks. These are the *Olivetti Research Laboratory* (ORL) corpus of face images, the *Image Retrieval in Medical Applications* (IRMA) corpus of radiography images, the *Erlangen* corpus of images of objects and the *Bloodcells* corpus of pathological red blood cells. Additionally a few tests have been done on the *US Postal Service* handwritten digits corpus.

These corpora will be presented in the following along with the results that have been achieved using other approaches.

4.1 Olivetti Research Laboratory Corpus (ORL)

This is a corpus for face recognition created at the Olivetti Research Laboratories. It consists of 400 images of 40 individuals such that there are 10 pictures of each person. All images are 92×112 pixel in size. The pictures contain the faces only. As a result the size changes of the projections from one person on the images usually do not exceed 15%. The facial expression of the persons vary as well as the illumination in the images. Additionally, the viewing direction varies slightly. Some examples are given in Figure 4.1. Some results that have been achieved on this corpus are shown in Table 4.1. As can be seen, LF&DV obtains a 0% error rate on this corpus. This is done by using the first 5 images of every person for training and the last 5 for evaluation. If only 3 images are used for training LF&DV obtains 2.6% error.

Table 4.1: Results of other approaches on the ORL corpus as reported in [Paredes & Pérez$^+$ 01].

Approach	**Error rate**
Volumetric Frequency Domain	7.5
Standard Hidden Markov Model	7.5
Probabilistic Neural Network	4.0
Convolutional Neural Network	4.0
Nearest Feature Line	3.0
Support Vector Machine	3.0
Embedded Hidden Markov Models	2.0
LF&DV	**0.0**

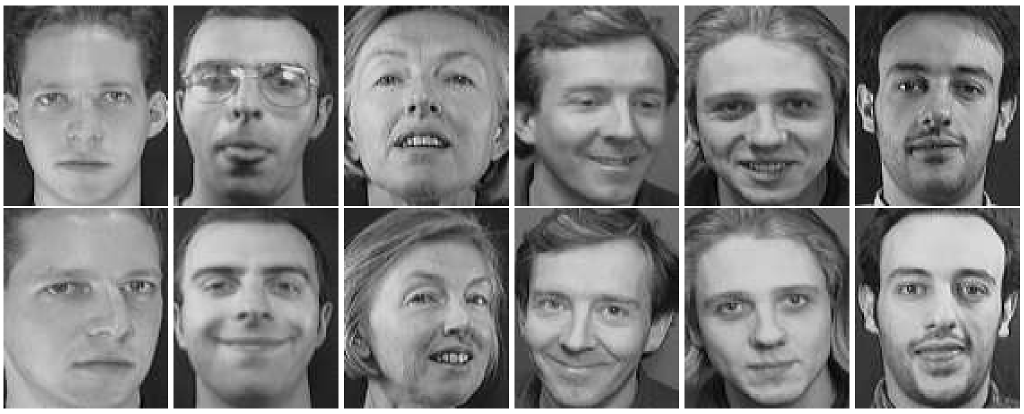


Figure 4.1: Examples of the ORL corpus.

4.2 Image Retrieval in Medical Applications (IRMA)

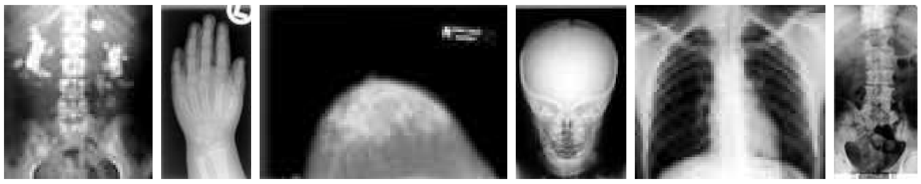
IRMA (Image Retrieval in Medical Applications) is a cooperative project of the Department of Diagnostic Radiology, the Department of Medical Informatics, Devision of Medical Image Processing and the Chair of Computer Science VI of the Aachen University of Technology (RWTH Aachen).

One result of this project is an image classification corpus with 1617 images that belong to 6 classes. These are abdomen, limb, mammography, cranium, thorax and vertebra. They are displayed in Figure 4.2(a). The images of the corpus are quite different in size and aspect ratio. The smallest image is 150×200 pixel in size and the largest is of 4000×4000 pixel. To make them comparable they have all been scaled such that the smaller side is 32 pixel long. Other approaches that require the images to have the same size make it necessary to scale all images to one fixed size, e.g. 32×32 pixel. This is not necessary with local features. Another preparation step that was applied to the corpus is a two bin histogram normalization to stretch the value range. This is especially important as the contrast of some images is especially poor and local feature extraction is done with a local variance threshold, such that no features would be extracted in those images otherwise. Furthermore, the boundary of classes is sometimes rather arbitrary as for example some images show half a pelvis and half a leg. An example image of every IRMA class is given in Figure 4.2(a) and the variance within the classes is demonstrated in Figure 4.2(b) by six examples of the class thorax.

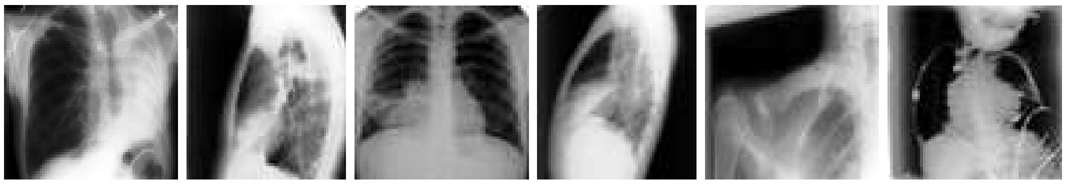
The best results that have been achieved so far on this corpus are presented in Table 4.2. Unfortunately IRMA is a corpus that has been created at the RWTH-Aachen and is not widely used, so that not many results of groups outside of Aachen have been published.

Table 4.2: Results of different approaches on IRMA.

Approach		**Error rate**
Cooccurrence Matrices	[Keysers & Dahmen+ 03]	29.0
Squared Images 1-NN	”	18.1
Squared Images, Kernel Densities	”	16.4
Thresholded Tangent Distance	”	11.1
LF&DV	[Paredes & Keysers+ 02]	10.6
Thresholded Image Distortion Model	[Keysers & Dahmen+ 03]	9.0
Distorted TD	”	8.0
Non Linear Distortion Models	[Gollan 03]	**5.3**



(a) Classes



(b) Thorax

Figure 4.2: (a) displays examples of classes of the IRMA corpus. They are abdomen, limb, mammography, cranium, thorax and vertebra. (b) shows different examples of the class thorax.

4.3 Erlangen

The corpus of the University of Erlangen-Nürnberg, Chair for Pattern Recognition, is an object recognition corpus with occlusion and changing backgrounds. The objects are two different cars, two different match boxes and one decorative box which also form the five classes. All images are 256×256 pixel in size with the object in their center. However, for evaluation they have been scaled down to 128×128 pixels to keep the local features approach manageable. The corpus is subdivided into two training corpora and six test corpora, where each training corpus is associated to three of the tests.

The training corpora consist of the mentioned objects rotated in steps of 10°, in the first one illuminated uniformly and the other time with two different illuminations, and hold 90 images plus two background images each. One background is simply black and the other one is the picture of a mouse pad.

As mentioned, there are three test corpora for each training corpus, each holding 170 images, which results in 34 images per class. For each task that is associated with first training corpus there is an analogous task for the second training corpus. The three conditions are

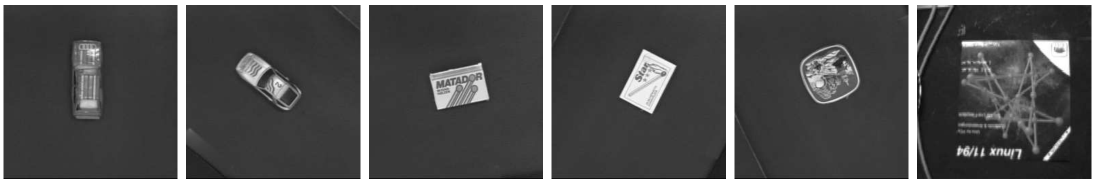
- a 25% occlusion of the object,
- a 50% occlusion of the object and
- changing backgrounds.

The objects are shown in Figure 4.3(a) and some examples of the test conditions are displayed in Figure 4.3(b).

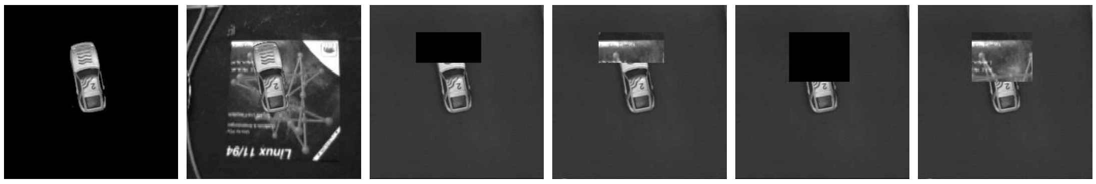
The results on this corpus, as published in [Reinhold & Paulus$^+$ 01], are shown in Table 4.3.

Table 4.3: Results of [Reinhold & Paulus$^+$ 01] on the Erlangen corpus.

Condition	**25% Occlusion**	**50% Occlusion**	**Changing background**
1 Illumination	0.0	2.3	0.0
2 Illuminations	0.0	4.8	0.0

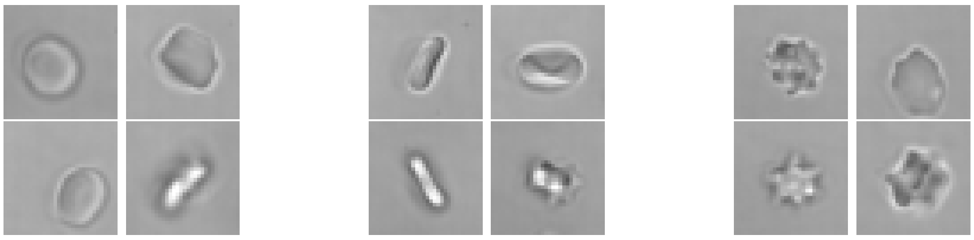


(a) Training



(b) Test

Figure 4.3: (a) The training images and the background of the Erlangen corpus and (b) examples of different test images.



(a) stomatocyte (b) echinocyte (c) discocyte

Figure 4.4: Examples of the three classes of the Bloodcells corpus.

4.4 Bloodcells

This is a corpus of red blood cells under the effect of different drugs. There are three classes of different size. The first class is represented by 3259, the second by 916 and the third by 887 images. Some example images are displayed in Figure 4.4. The pictures are 32×32 in size. Before classification, a two bin histogram normalization has been performed to augment the contrast of the images.

Some results on this corpus are displayed in Table 4.4. Even though this is just a three class problem the task is pretty difficult as can be seen in the human error rate of more than 20% and the best reported error rate so far of 15.3%.

4.5 US Postal Services (USPS)

This is the well known handwritten digit corpus of the US Postal Service. All images have a size of 16×16. The corpus is divided into 7291 training and 2007 testing images. Some example images are shown in Figure 4.5. Observe that some of the digits are cluttered or deformed heavily and that the line widths differ. The human error is about 2.5% which shows that it is a complicated task. Some of the results that have been published are presented in Table 4.5.

Table 4.4: Results on the Bloodcells corpus according to [Keysers & Dahmen+ 01].

Approach	**Error rate**
Human	$> 20\%$
Gaussian Mixture Densities	31.0
1-NN	21.4
Kernel Densities	19.6
Mixture Densities, RST-invariant	18.8
KD, Tangent Distance, Virtual Data	16.3
GMD, RST-invariant, LDA	**15.3**

Table 4.5: Some results on the USPS corpus.

Approach		**Error rate**
Human	[Simard & Le Cun+ 93]	2.5
Nearest Neighbor	[Keysers & Dahmen+ 00]	5.6
Relevance Vector Machine	[Tipping 00]	5.1
Neural Net (LeNet1)	[Simard & Le Cun+ 98]	4.2
Mixture Densities, LDA, Virtual Data	[Dahmen & Keysers+ 01]	3.4
Invariant Support Vectors	[Schölkopf & Simard+ 98]	3.0
LF&DV	[Keysers & Paredes+ 02]	3.0
KD, TD, Virtual Data	[Keysers & Paredes+ 02]	**2.4**

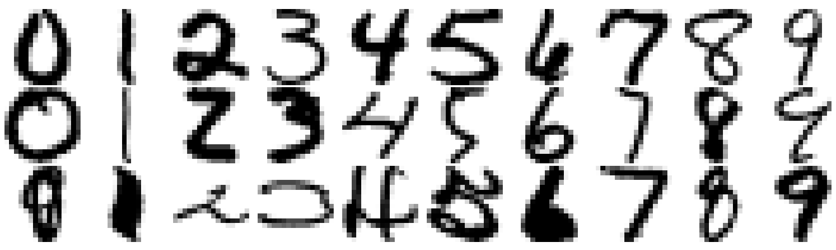


Figure 4.5: Examples of the USPS corpus.

Chapter 5

Experimental Results

Here the results of the methods that have been presented in Chapter 3 are presented and interpreted. The experiments have been carried out on the databases that have been presented in the previous chapter. However, the comparative results from other groups are presented again, along with the newly generated ones, where appropriate, and the differences are discussed.

5.1 Multi-Scale Feature Extraction

In Section 3.2 the motivation to extract local features at different scales has been discussed. Here we present the practical results of applying this technique to IRMA. The features are extracted with 13, 19, 25, 31 pixel width and then scaled to 19×19. Some examples of this are depicted in Figure 5.1. The variance threshold is 484 and no sub-sampling is done. The features are PCA-reduced to 40 dimensions. The results are shown in Table 5.1. It can be seen that the recognition result is increased as well for direct voting as for the kernel densities approach.

All scaling in this work has been done with spline-interpolation. However, scaling images down with splines can lead to artifacts. This can be seen in the two craniums on the lower left of Figure 5.1(b). So the results might be improved more by scaling down the features with another method.

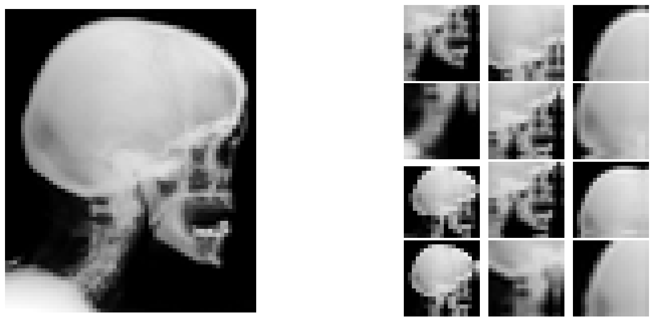

(a) Image (b) Features

Figure 5.1: Examples of features extracted at multiple scales and the originating image.

Table 5.1: Results with multi-scale feature extraction on IRMA.

Multi-scale	**Probability model**	**Error**
No	DV	10.3
Yes	DV	**10.0**
No	KD	9.7
Yes	KD	**9.4**

Table 5.2: Results of feature relevance estimation compared with variance threshold on ORL.

Approach	**Error rate**	**Features extracted**
Variance Threshold	2.5	157,756
FRE 50	4.6	133,074
FRE 150	**2.1**	**100,687**
FRE 500	**2.1**	124,538
FRE 1000	3.1	120,405
No reduction	2.9	917,280

5.2 Dataset Reduction

As presented in Section 3.3, usually the local variance threshold is used for set reduction. Here this method is compared with feature relevance estimation (FRE) that has been presented in the same section. The fuzzy K-means algorithm used has to be supplied with the number of densities that should be created. Tests have been made with 50, 150, 500 and 1000 densities. This has been tested on ORL using a local feature size of 15×15 and using the first three images of every person as training images and the last 7 for testing. The results can be seen in Table 5.2.

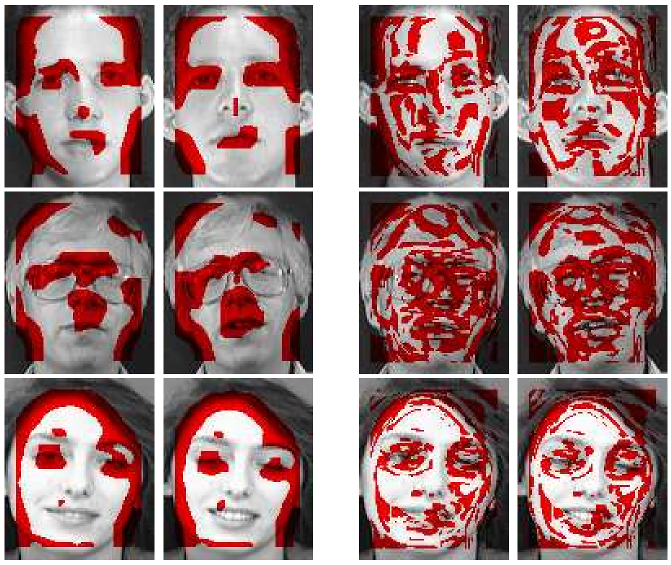
As can be seen, this method improves the recognition result slightly. What is remarkable though, is that it leads to the better results, while using significantly less features. Figure 5.2(b) shows some examples of where in the images local features have been extracted using FRE compared to the positions using the variance threshold that are displayed in Figure 5.2(a). It can be seen that FRE extracts features more regularly over the image. This could be an advantage. At the mouth and the nose of the woman, for example almost no local features have been extracted with the variance threshold. FRE did extract some features at those positions.

Additionally the effect of extracting all local features has been tested on IRMA the result can be seen in Table 5.3. The features have been extracted with a threshold of 400 and filtered with a horizontal and a vertical Sobel filter. The features have been PCA reduced to 40 dimensions. It is notable that even though only 0.4% more local features are extracted the result is one percentage point worse if no variance threshold has been applied. This shows that it definitely harms the result to include features with little local variance.

FRE leads to improvements. However, it must be noted that the approach is time consuming. This is partially because all local features have to be extracted, such that basic operations like reading and writing, already are costly tasks. Also the training of the mixture densities is computationally expensive. Finally, it makes evaluation techniques like leaving one out uncomfortable to execute as a new mixture densities has to be computed for every image that is classified. Since we

Table 5.3: The effect of taking all local features on the IRMA corpus.

Approach	**Error rate**	**Features extracted**
Variance threshold	**8.9**	780,653
No reduction	9.9	784,132



(a) Variance threshold (b) FRE

Figure 5.2: Positions at which local features have been extracted in the images. (a) Using the variance threshold the extraction is concentrated on few regions, whereas (b) using FRE the features are extracted more evenly over the entire image.

did not feel that the improvement does justify the effort, FRE is not considered in the following.

5.3 Dimensionality Reduction

The dimensionality has been reduced using principal component analysis, discrete cosine transform and linear discriminant analysis.

For the PCA, the 40 first principal components are chosen, so reducing the dimensionality from up to 361 dimensions down to 40. Some tests have been made regarding the dimensionality and the 40 dimensional sub-space has proven to be well suited for all tested corpora. Communication with the authors of [Paredes & Pérez$^+$ 01] revealed that they made similar experiences.

5.3.1 Discrete Cosine Transform

To reduce the dimensionality using DCT, only the largest wavelengths are chosen. As described in Section 3.6 they have been chosen by taking the components within a square, triangle or quarter circle in the upper left of the DCT-transform. One further design decision is whether the origin of the DCT is put in the center of the local feature or for example in its upper left corner. The results using the three wavelength extractions choices (circle, squared and triangular) as well as the two positionings of the origin of the DCT on IRMA are displayed in Table 5.4. As can be seen the DCT with its origin in the upper left corner of the local features lead to significantly better results than those with their origin in the center of the features. This is probably caused by the fact that the centered images cannot contain gradient information as they are symmetrical with respect to the origin. This can be seen in Figure 5.3(a). Those centered in the upper left corner of the features can be seen in Figure 5.3(b), it can be noted that they are neither symmetrical with respect to the horizontal axis nor to the vertical. It also must be noted that the best result is almost as good as the PCA with 40 dimensions which is 10.3%. This shows that the DCT is

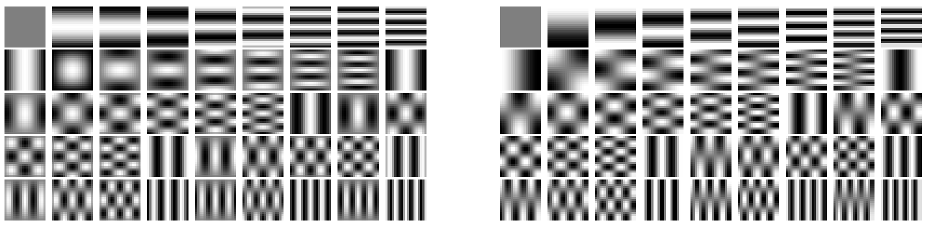

(a) Centered (b) Shifted to upper left

Figure 5.3: DCT (a) centered in local feature and (b) centered in upper left of feature.

Table 5.4: Results of different reductions using DCT on IRMA.

	Circle 33	**Circle 43**	**Square 36**	**Square 49**	**Triang 36**	**Triang 45**
Centered	24.5	23.1	24.6	23.1	23.0	21.8
Upper left	12.1	11.3	12.0	11.6	**10.9**	11.0

similarly well suited to reduce the representation dimensionality as the PCA.

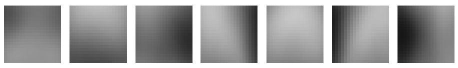
As a result the complexity of the method can be reduced by using the discrete cosine transform with small recognition performance losses only.

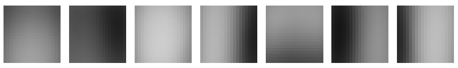
5.3.2 Linear Discriminant Analysis

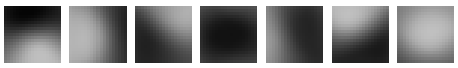
In the previous section the PCA has been compared to a method that does not have to be trained. The idea was to reduce the complexity of feature extraction. The question here is whether we can use the class information to increase class separability. To do so, the LDA, as described in Section 3.4.3, has been used. To keep it comparable with the results of the PCA, every class has been subdivided into 7 virtual classes. These were generated using K-means clustering as described in Section 2.3.2. The 42 means extracted on the IRMA corpus are displayed in Figure 5.4. Even though some means are quite close to each other, it can be seen that especially those of the thoraxes are easily distinguishable from those of other classes. Applying the LDA to the clustered vectors, we get the reducing transformation. The transformation vectors are given in Figure 5.5. These images reflect the salt and pepper patterns that can typically be observed with LDA transforms. However, some structures of the means can be found in a similar form in those vectors, especially those of the thorax class.

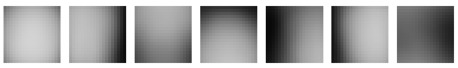
The tests of the LDA were done on IRMA with a feature size of 19×19 that were reduced to 41 dimensions as described above. The local variance threshold was set to 400, thus 780,653 features were extracted in each condition. But even though the images of the transformation vectors might suggest that the LDA might lead to good results, as some of them contain structures that resemble the class means of the virtual classes, the tests that are displayed in Table 5.5 demonstrate that the LDA is not suited for the local feature approach. This is similar to results reported in [Fergus & Perona$^+$ 03], where the authors compare different dimensionality reduction techniques for local features.

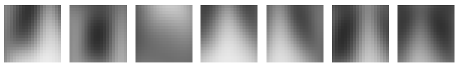
An explanation for this could be that the LDA expects the images of one class to be relatively similar. This however is not given in the local feature approach. After clustering, the average distances of the local features from their cluster mean is 1350. On the other hand the mean distance between the clusters is only 1306. It is also commented in [Duda & Hart$^+$ 01] that the LDA is problematic, if the clusters are not compact. This also might be a good indicator that generalizing approaches do not work well with local features.


(a) Abdomen


(b) Limb


(c) Mammography


(d) Cranium


(e) Thorax

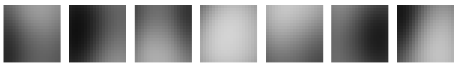

(f) Vertebra

Figure 5.4: The means of the clusters returned by the k-means algorithm.

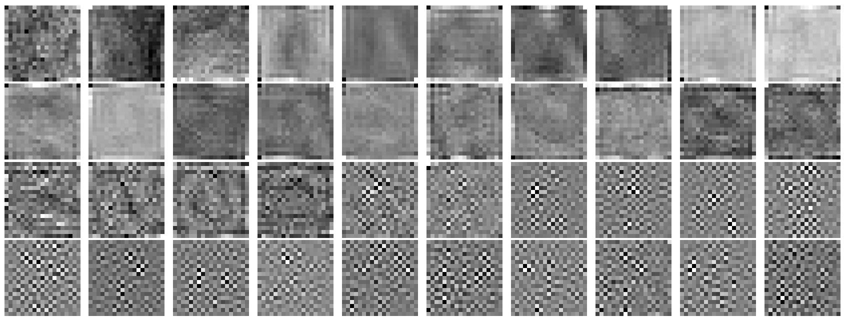


Figure 5.5: 40 first transformation vectors of the LDA.

5.4 Tangent Distance

In Section 3.6 we have proposed the tangent distance to give invariance towards small transformations. This has so far only been used for entire images. Here some results with the combination of tangent distance with local features and kernel densities are presented. The tangent distances are tested on the IRMA and the Bloodcells corpus. The parameters used on both corpora are displayed in Table 5.6. The tangents that lead to the good results on IRMA are horizontal and vertical translation, the axis and the diagonal deformation, the scaling, rotation and additive brightness changes. The tangents used for the Bloodcells corpus are the same as the tangents used for IRMA with the exception that the tangent for line thickness has been added. Adding this tangent has been motivated by the character of the images. Often the outlines of the blood cells contain borders of different widths, as is shown in Figure 5.6. These examples are all from the class stomatocyte.

Table 5.5: The results of the LDA compared to those gained with the PCA on IRMA.

Approach	**Error rate**
PCA	**10.4**
LDA	13.3

Table 5.6: Parameters used for tangent distance.

Corpus	**Normalization**	**Feature size**	**Threshold**	**Reduction**	α
IRMA	2 bin	19	400	PCA 40	0.008
Bloodcells	2 bin	15	1600	PCA 40	0.01

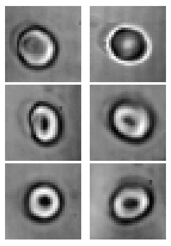
Table 5.7: The error rate of LF&DV could be reduced by measuring the nearest neighbor with the tangent distance instead of the Euclidean norm.

	IRMA	**Bloodcells**
LF&KD	10.3	17.2
LF&TD	7.4	**13.5**
Best other	[Gollan 03] **5.3**	[Dahmen & Hektor$^+$ 00] 15.3

These setups lead to improvements on IRMA and on the Bloodcells corpus as can be seen in Table 5.7. It must be noted, that the result on Bloodcells is the best reported so far, with the second best result of 15.3% that was reported in [Dahmen & Hektor$^+$ 00]. That method uses RST-invariant features and Gaussian mixture densities. This can be seen as a quite specialized design for blood cell classification. However, the rather general approach of local features, direct voting and tangent distances leads to a better result. On IRMA, [Gollan 03] presented the best error rate of 5.3%. This is the only publication of a better error rate than the one presented here. It has been achieved using pseudo 2D hidden Markov and distortion-models on Sobel transformed images with 3×3 local contexts. It is notable, that this approach only leads to very good result by using local contexts (even though the ones used there are a much smaller than those used in our approach).

If nearest neighbor is not searched for the PCA reduced of the local features, but on the horizontal and vertical Sobel transforms, the result for the Euclidean distance and direct voting increases as can be seen in Table 5.8. However the tangent distance does not lead to the good results it did on the gray values. A reason for this is that, by taking the Sobel transformed, the PCA reduction to 40 dimensions and the tangent distance the feature space is reduced that much, that the discriminative information is lost. If the tangent for multiplicative brightness is added to the configuration that leads to 7.8% error, the error jumps to 69.8%. This is because the distinction between steep and flat gradients is lost, such that images with little contrast get a small distance to images with big contrasts even though they do not carry suitable class dependent information.

Note that the chosen parameters were found without optimization on the specific data. For


(a) Thick lines

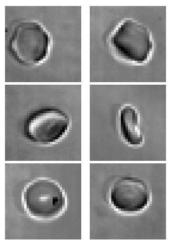

(b) Thin lines

Figure 5.6: A motivation to add the line thickness tangents for distance measurement.

Table 5.8: The effect of using Sobel transformed local features instead of the gray values on IRMA.

Distance measure	**Error rate**
Euclidean	8.9
Tangent distance 12sr	8.2
Tangent distance hv12sr	7.8

IRMA the window size presented in [Paredes & Keysers+ 02] has been taken without variation and for Bloodcells the sizes 13 and 15 have been tested without tangents and 15 pixel sized windows led to a slightly better result (17.6% for 13 and 17.2% for 15). On IRMA the only parameter that has been found empirically is the value of α for the kernel densities.

Also, in Section 3.6 an exterior combination of local features and tangent distance on the USPS corpus that has been presented in [Keysers & Paredes+ 02] was mentioned together with the idea to turn this into an inner combination. The authors report an error rate on USPS of 2.0% by combining LF&DV and TD with majority voting. The error rates of the methods without combination are 2.4% for the tangent distance with virtual data added to the training set. LF&DV leads to a 3.0% error (The LF&DV error rate could not be exactly reproduced, we just obtained a 3.4% error). Using the nearest neighbor with local features and direct voting using the tangent distance leads to an error rate of 2.6%. This is higher than that achieved using the tangent distance. However it is better than simply doing the LF&DV approach.

The results presented here is attained with the following configuration: The images are padded with a border of 7 pixel, the local features are 15×15 pixels in size, a low variance threshold of 100 was chosen, and no sub-sampling was performed. For the tangent distance all tangents are used but that for multiplicative brightness change.

5.5 Kernel Densities

As presented earlier, in the LF&DV approach the a priori probability $p(k|X)$ for a specific class k given an image X is directly approximated for each of its local features x by

$$p_{\text{NN}}(k|x_n) = \begin{cases} 1 & \text{if } \hat{x} \in \mathfrak{X}_k \\ 0 & \text{else} \end{cases}$$

The other possibility examined was the kernel densities, as presented in Section 3.5.1. However, we did not use the usual kernel densities as given in (3.24), but a variation known under the name of kernel densities with maximum approximation. It is given by:

$$p_{\text{KD}}(k|x) = \frac{\max_n \phi(x - x_{k,n}, k)}{\sum_{k'=1}^{K} \max_n \phi(x - x_{k',n}, k')}$$

with the kernel function that is given by

$$\phi(x, k) = \prod_{d=1}^{D} \frac{1}{\sqrt{2\pi\alpha\sigma_k^2}} \exp\left(-\frac{(x_d)^2}{2\alpha\sigma_k^2}\right)$$

This decision was mainly taken because of speed considerations. Using the local features, we have to deal with an amount of data that is by magnitudes larger than that of regular nearest neighbor approaches, such that an exhaustive search is expensive with respect to computing resources. The other reason why the usual kernel densities has not been used is that the local features within one class have a large variability. The consequence of this is that the kernel function returns values that are significantly larger than zero for the first few nearest neighbors only. In IRMA,

Table 5.9: The improvements using kernel densities.

	IRMA	**Erlangen 50% occ**	**Bloodcells**
NN&DV	10.3	1.2	17.7
KD	**9.7**	**0.6**	**17.2**

Table 5.10: Parameters used for kernel densities.

Corpus	**Normalization**	**Feature size**	**Threshold**	**Reduction**	α
IRMA	2 bin	19	400	PCA 40	0.03
Erlangen	None	13	400	PCA 40	0.01
Bloodcells	2 bin	15	2500	PCA 40	0.02

for example it does not make sense to estimate the probability of a local feature that shows an eye-hole from a local feature that shows a chin, even though they are both from the class cranium.

This approximation led to improvements of the recognition as can be seen in Table 5.9 for some corpora. The dependence of the result with respect to the parameter α can be seen in Figure 5.7. The graph shows the dependence of kernel densities of the weighting factor α, one time where the variance is pooled and the other where it is not. It can be seen that pooling the variance over the classes generally leads to better results, than using class dependent variances. Additionally, it can be seen that the result for the class dependent variance quickly deteriorates as α approaches 1. This is because then the distance looses weight as α increases. As a result the class with the highest variance is almost always chosen. The case with pooled variances on the other hand is much more insensitive towards this parameter. Also, it can be seen that the result converges to the result of LF&DV (that is 10.3%), when α gets sufficiently small. This behavior was explained in Section 3.5.2. The parameters for the results that are presented in Table 5.9 are shown in Table 5.10.

To estimate the impact of using the maximum approximation with the kernel densities a relaxed approximation with the 100 nearest neighbors from each class was tested. However, this does not improve the result with respect to direct voting. That is, the best result is achieved when α is set to 0. This is the result of direct voting with 10.3% error.

5.6 Local Features, Occlusion and Changing Backgrounds

When doing recognition with changing backgrounds and partial occlusion, local features are better suited than global approaches [Schmid 99, Mohr & Picard$^+$ 97]. Here, the Erlangen corpus with occlusion, brightness changes and changing backgrounds has been used for evaluation. The most important competition to our method is presented in [Reinhold & Paulus$^+$ 01]. There an approach based on local multi-resolution Gabor features and an object model is presented for the task. We will compare their results with ours, gained with the unconstrained local feature approach using direct voting and alternatively kernel densities. For these tests we used the following background model. It consists simply in an additional feature set $\mathfrak{X}_{bg}$ that has been extracted on the background images and represents the background class $K+1$. The decision rule is now given by

$$\underset{k\in\{1,\dots,K\}}{\operatorname{argmax}} \{p(k|X)\} \tag{5.1}$$

So the background class goes into the probability model, but is excluded from the decision rule. Practically this is realized by choosing the class with the highest probability, if it is not the background class and that with the second highest else. The results achieved for the three conditions

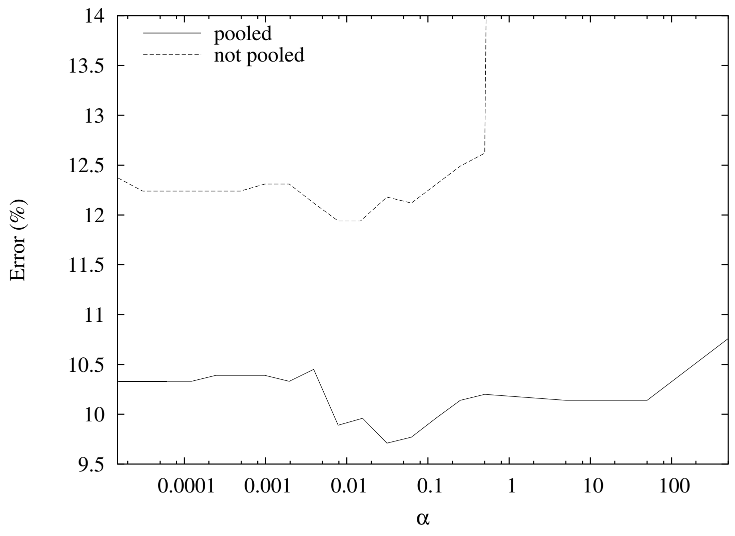


Figure 5.7: Dependence of α of kernel densities approximation with and without pooling of the variance on the IRMA corpus.

Table 5.11: LF&DV on Erlangen useful for partial occlusion and changing backgrounds.

Approach	**25% Occlusion**	**50% Occlusion**	**Changing Backgrounds**
[Reinhold & Paulus$^+$ 01]	0.0	4.8	0.0
LF&DV	0.0	1.2	0.6
LF&KD	0.0	**0.6**	0.0

with two illuminations are compared to those of [Reinhold & Paulus$^+$ 01] in Table 5.11.

It can be seen that, even though we have no object model, the results of local features with kernel densities is superior to the competitor. This shows that PCA-transformed local features are more appropriate to this task than stacked Gabor features.

5.7 Log-Linear Models

In this section we will present the results of using maximum entropy to estimate the quality of the local features. We have used the feature functions presented in Section 3.7.

As the log-linear models are computed iteratively, evaluation techniques like leaving one out are difficult to apply. For this reason a five fold cross-validation has been used. The tests have been done on IRMA with the parameters shown in table 5.12. The cross-validation corpora have been extracted by selecting every fifth image for evaluation and adding the others to the training corpora, starting at the first through fifth image.

One problem with this maximum entropy approach is that about 40% of the feature functions that fire in the test have not been seen in the training. As a result they do not affect the probability calculation in the test, which means that 40% of the local features are effectively ignored. As we want them to be included into the probability computation, we have to define a probability for unseen events. In this work it has been done as follows.

Let $(x_1, k_1), \ldots, (x_N, k_N)$ be some training data and let $f_{n,k}$ be the feature functions as defined in (3.53). Let $\mathfrak{F} = \{f_{n_1,k_1}, \ldots, f_{n_I,k_I}\}$ be the set of feature functions f_{n_i,k_i} for the nearest neighbor

Table 5.12: Parameters on IRMA to evaluate log-linear models.

Normalization	**Features size**	**Threshold**	**Reduction**
2 bin Sobel	19	400	PCA 40

Table 5.13: Results of Log-linear models on IRMA.

Approach	**1 of 5**	**2 of 5**	**3 of 5**	**4 of 5**	**5 of 5**	**All**
LF&DV	9.6	9.0	9.5	10.7	9.7	9.7
Max. Ent. (1)	9.0	11.1	12.1	10.0	10.8	10.6
Max. Ent. (2)	**8.3**	**8.6**	**9.0**	**7.1**	**9.6**	**8.5**

x_{n_i} and the class hypothesis k_i. We train the weights $\lambda_{n,k}$ with generalized iterative scaling. Then we calculate the mean $\hat{\lambda}^+$ of the feature weights $\lambda_{n',k'}$ with $\lambda_{n',k'} \geq 0$ as value for suited features and the mean $\hat{\lambda}^-$ of the features weights $\lambda_{n'',k''} < 0$ as value for unsuited features. The weights $\lambda_{n,k}$ of the feature functions $f_{n,k} \notin \mathfrak{F}$ that have not been seen in the training are set to

$$\lambda_{n,k} = \begin{cases} \hat{\lambda}^+ & \text{if} \quad k_n = k \\ \hat{\lambda}^- & \text{else} \end{cases} \tag{5.2}$$

Then the classification is done.

The generalized iterative scaling, as well as the evaluation are done with the program `Yasmet`. The results are shown in table 5.13. In the row of Max. Ent. (1) are the results when the features that have not been seen in the training are ignored and the row of Max. Ent. (2) holds the results that are achieved when using the proposed method.

It can be seen that the result is improved for every cross-validation corpus with the proposed method. This shows that it can be learned which local features are well suited for classification using maximum entropy modeling. It is interesting that we come to a similar conclusion in Section 5.2 using an appearance based approach. When only considering the feature functions that have been seen in the training the result is generally worse than by doing the direct voting. Only this one method for modeling the unseen feature functions was tested in this work. It might therefore be interesting to try other methods for this task.

5.8 Impact of Approximative Search

Execution speed is not as important in scientific research as, say in industrial applications. However, it may not be neglected entirely, especially in the case of local features, where we are sometimes dealing with over a million vectors with a dimensionality of 40. The first step to improve performance is by using fast search structures. However, at dimensionalities greater than 8 most structures do not perform significantly better than the brute force approach. This is described in [Arya & Mount$^+$ 98]. In this situation, approximative search strategies can lead to a significantly increased speed while limiting the maximum approximation error by some boundary.

The approximative search algorithm we use is the approximative KD-tree search as described in [Arya & Mount$^+$ 98]. The idea is to subdivide space into axis aligned cells $C_1, \ldots, C_L$. For the query feature x these cells are sorted by their distance to x and their content is searched in increasing order until the first feature x' is found in C_l that fulfills the condition

$$\mathrm{d}(C_l, x) > \frac{\mathrm{d}(x, x')}{1 + \varepsilon}$$

where $\mathrm{d}(C_l, x)$ is the distance between x and the closest side of C_l to the query feature. ε is a parameter that can be chosen freely. For the proposed algorithm, the query feature x, its real nearest neighbor $\hat{x}$, a returned approximative nearest neighbor x' and the parameter ε that is supplied by the user, the condition

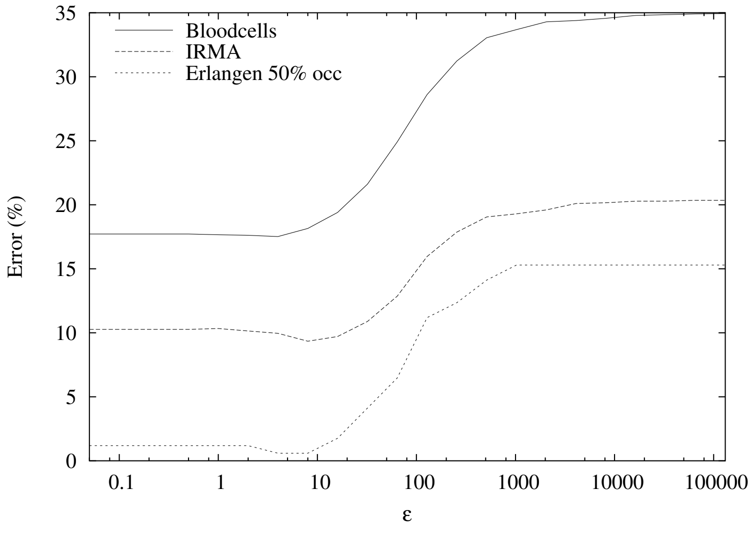
$$\mathrm{d}(x, x') \leq (1 + \varepsilon) \cdot \mathrm{d}(x, \hat{x}) \tag{5.3}$$

will always hold. The experimental results showed that the real error is usually significantly smaller than the theoretical bound in practical situations, though.

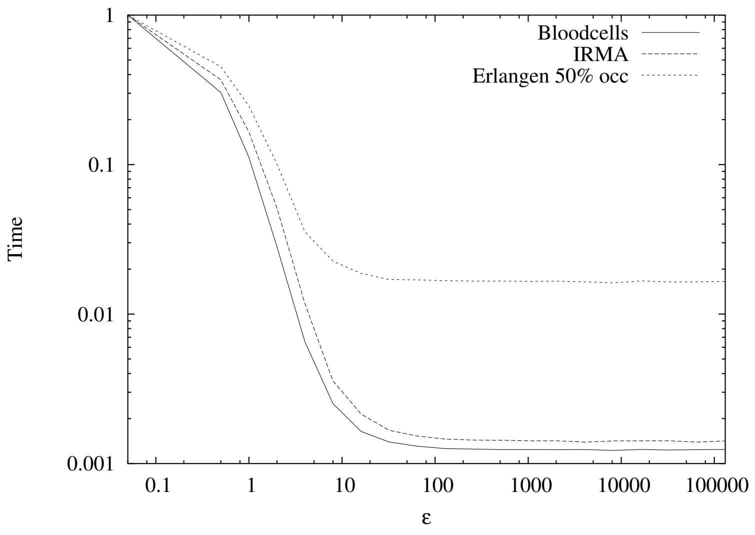
This strategy should cope quite well with the concept of local features with direct voting. As here one single result has little importance, it is the mass of, by itself unreliable, votes that leads to the good overall recognition results. Thus, we expect the result to get a little smeared out but to be roughly the same.

In the experiments, this expectation was not only confirmed, but the approximative search even turned out to reduce the recognition error on IRMA, Erlangen and Bloodcells as can be seen in Figure 5.8(a). At the same time, execution time decreases rapidly as the value of ε increases. This is depicted in Figure 5.8(b).

One explanation is that the approximate search is a probabilistic approximation to an L nearest neighbor search. Let $\mathfrak{X}_k = \{x_{k,1}, \ldots, x_{k,N_k}\}$ be the set of local features of class $k \in \{1, \ldots, K\}$, let x be the query feature and let $\{\hat{x}^x_{k,1}, \ldots, \hat{x}^x_{k,L}\} \subseteq \mathfrak{X}_k$ its L nearest neighbors from class k. If ε increases, the probability of finding $\hat{x}^x_{k,1}$ decreases and some other $\hat{x}^x_{k,l} \in \mathfrak{X}_{k,L}$ will be found. As this is done approximately, a class k' with more features similar to x will provide a good approximation to the real nearest neighbor $\hat{x}^x_{k',1}$ with a higher probability than any other class k'' with less features similar to the query. As a result the distance to the approximate nearest neighbors reflects the feature densities of the classes $k \in \{1, \ldots, K\}$ near x.



(a) Errors



(b) Computation Times

Figure 5.8: Error rates and relative execution times for Bloodcells, IRMA and Erlangen in dependence of ε.

Chapter 6

Conclusion and Outlook

In this work we have investigated a local feature based image classification approach. We have broken down the method into five distinct steps. These are:

- Feature extraction.
- Feature set reduction.
- Feature dimensionality reduction.
- Nearest neighbor search.
- Decision rule.

While working with the unconstrained search using local features we observed that this approach is well-suited for various image classification tasks, especially for recognition of partially occluded objects. Furthermore, its local nature makes the approach invariant with respect to translation and leads to some invariance with respect to some non-linear global transformations.

Extraction of features on different scales did lead to slightly improved results.

We have shown that it is possible to model which local features are most discriminative using Gaussian mixture densities. This leads to some improvements compared to the local variance threshold and at the same time extracts only two thirds the amount of local features.

Training weights for the local features using maximum entropy did lead to a significantly improved recognition. This and the conclusion drawn in the previous paragraph supports the assumption that it can be learned which local features are discriminative and which ones are not.

Another result of our research was that the use of a linear discriminant analysis instead of the principal component analysis does not improve the result. This is because the distance between the class means is small compared to the distances within the classes.

The use of the discrete cosine transform instead of the principal component analysis did not decrease the recognition result significantly, but it can be computed independently of the training data. As a result the entire process is simplified by one step.

Incorporating the tangent distance resulted in a further improvement of the recognition. On two corpora it leads to the best results published.

Also, kernel densities are suited to extend the probability model of the local feature approach as it increases the recognition result in most cases. Also, pooling the variances leads to significant improvements of the result compared to using class dependent variances.

Searching the nearest neighbor approximately improved the recognition result in all investigated cases if the approximation was not too rough. We assume that the approximation causes the probability to estimate the density of the training data that is close to the training point rather than being based directly on the nearest neighbors. Another benefit is the increase of recognition speed up to a factor of twenty.

Further interesting investigations are the combination of appearance based features with other, higher-level features as texture or shape characteristics. This could increase the recognition result, as larger scale information is added to the model.

Using a weighted dissimilarity measure to weight the components of the feature vectors might as well lead to improvements, especially after the dimension reduction is done. It also can be applied, when combining different kinds of features as described before.

However, we think that the most promising further exploration is the combination of the local feature approach with global models. We especially think that non-linear warping models could combine well with appearance based local features. Also, it would be interesting to explore global models that permit multi-object recognition in images.

Appendix A

Local Variance Histogram Features

In the course of this work we have almost exclusively worked with appearance based local features. However, one other type of features has been examined as well: *local variance histogram features* (LVHF). The basic idea is to build a histogram over the spatial distribution of variances relatively to some image points. The idea was inspired by [Belongie & Malik$^+$ 00], where similar features are used. These features are matched with the prototype features and adapted iteratively according to their position. This method leads to an error rate of 0.63% on the MNIST data set for handwritten digit recognition.

LVHFs are created as follows: for an image X, a factor $0 \leq t \leq 1$ and a local variance function

$$\mathrm{V} : \mathbb{R}^{I \times J} \times \mathbb{N} \times \mathbb{N} \mapsto \mathbb{R} \tag{A.1}$$

the local variance image X^{LV} is given by

$$X^{LV}_{i,j} = \begin{cases} 1 & \text{if} \quad \frac{|\{X_{i',j'} : \mathrm{V}(X,i',j') > \mathrm{V}(X,i,j)\}|}{I \cdot J} \leq t \\ 0 & \text{otherwise} \end{cases} \tag{A.2}$$

The image positions (i, j) where $X^{LV} = 1$ form the following set

$$\hat{X} = \{(i,j) : X^{LV}_{i,j} = 1\} \tag{A.3}$$

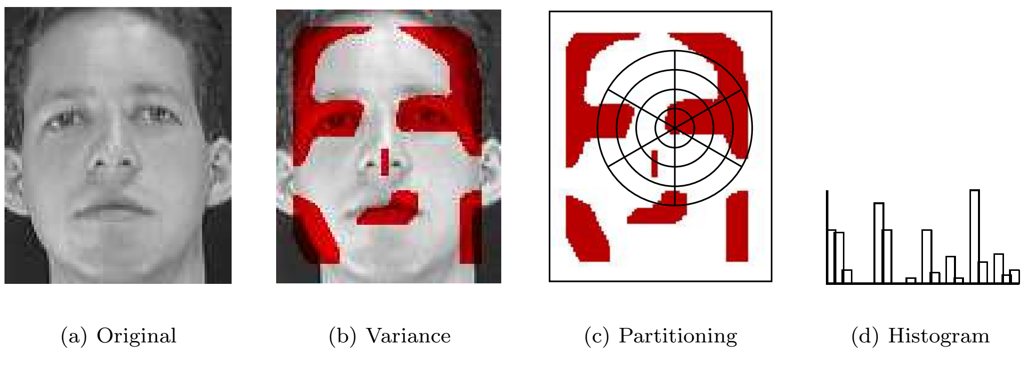


Figure A.1: Extraction process of the LVHFs: (a) original image, (b) places of high variance in the image, (c) local polar coordinate system with bins and (d) local variance histogram feature.

Table A.1: Results using LVHFs.

Features	**t**	**Subsampling**	**Error**
LVHF	0.3	1	7.5
LVHF	0.5	1	9.5
LVHF	0.125	0	12.0
LF			0.0

Then, for each image position $m = (i, j) \in \hat{X}$, the relative position to every other position $(i', j') \in \hat{X}$ is converted to polar coordinates. Let $\rho = (i' - i, j' - j)$ be one such relative coordinate for m and let $\rho^\circ = (\psi, r)$ be the polar representation of ρ. Then, the polar coordinate system is partitioned by taking rings of equal width and dividing these into the segments $S_1, \ldots, S_\infty$. The L first elements of this partitioning are the basis for the histogram $x^m = (x_1^m, \ldots, x_L^m)$. The entries of these histogram bins are determined by counting

$$x_l^m = |\{\rho'^\circ : \rho'^\circ \in S_l\}| \tag{A.4}$$

This process is depicted in Figure A.1. These histograms are then used like the local features.

This feature extraction method has been evaluated on ORL by using 50% of the images as training and 50% as test corpus. This contrasts to the 30-70 split of the corpus that is done instead throughout this work. The results are shown in table A.1. Even though the result stays beneath the 100% recognition of LF&DV, it can be seen that LVHFs are discriminative. As a result they could be used together with local features to add semi global shape information. This might be an interesting investigation for the future.

Appendix B

Software

To evaluate the methods described in this document, several programs have been written. The most important ones are introduced shortly in this chapter and their command line options are explained. In addition to those C++ programs, many small to medium size scripts were used for data transformation, program execution, result presentation and other tasks. Those were mainly written in bash, awk and the wonderful Python language.

drawBorder

The feature extraction cannot extract features that go beyond the border of the image. If this is needed the image can be padded with `drawBorder`. The program can deal only with PGM-images. It is called as follows

```
drawBorder [options] [<infile> [<outfile>]]
  options
    -c <borderWidth>  how large the borders should be.
    -s <color>        set border value to <color>.
    -q                Pssst, no output!
```

With `<infile>` and `<outfile>` the input and output filenames are given. If they are omitted "standard in" and "standard out" are used. With `-c` the width of the border is specified and with `-s` the gray value to be used for the padding is set.

tk_extract_local_features

This is the main program for the local feature extraction. It is given an image in the PGM-format and returns a file of local features. The program is executed as follows:

```
tk_extract_local_features <class> <file> [options]

  options:

     -D <directory>  Path to the image.
     -w <winRad>     The radius of the local feature.
     -v <varRad>     The radius of the area where the variance is
                     beeing calculated.
```

```
    -p <prune>      The amount of local features that are kept.
    -t <threshold>  The minimal standard deviation threshold that is
                    taken into account.
    -s <stepSize>   The step-size for the feature extraction.
    -g              The local features of the gradient image are
                    extracted.
    -n              Normalize the image before extracting local
                    features.
    -c              Get shape context instead of crop image.
    -o <ofilename>  Where the result should be written to.
    -i              Read image from stdin.
```

The `<class>` is the class that the image belongs to and the `<file>` is the name of the file that will be written in the second column, it is also the file that will be used as input. If the working directory is not the directory in which the file is located, use the `-D` command to give the location of the file. Every image should have a unique filename, as some other programs rely on this. Input will be taken from "standard in" if `-i` is given on the command line. The result is written to "standard out" unless a filename is given with the `-o` parameter.

The radius of the extracted window is given after `-w` and the window in which the variance is measured for extraction is given with `-v`. A radius of r results in a window of size $(2r+1) \times (2r+1)$ and a local feature vector with $(2r+1)^2$ components. The step size of the extraction is controlled with the parameter `-s`. If it is 1 all local features are taken into account for the variance threshold, if it is $s \in \mathbb{N}$ only the pixel positions $X_{s \cdot i, s \cdot j}$ are taken into account. `-t` is used to give a variance threshold to the program. All local features that have been returned after the sub-sampling and that have a variance of its gray values below this value will be discarded. If only some fraction of the local features that passed the variance threshold should be taken, this can be given as argument of the parameter `-p`. If its parameter is 1 all local features will be kept, if it is .5 half of the local features will be taken and if it is 0 none will be taken (this of course does not make a lot of sense unless you need some load on your computer). If `-g` is given, the local features are extracted on the gradient image. And if `-c` is given, not the local features, but local variance histograms are extracted as described in appendix A.

The format of the output file is as follows:

```
<class> <file> <x-pos> <y-pos> <value_1> <value_2>  . . .  <value_D>
<class> <file> <x-pos> <y-pos> <value_1> <value_2>  . . .  <value_D>
    .
    .
    .
<class> <file> <x-pos> <y-pos> <value_1> <value_2>  . . .  <value_D>
```

`<class>` is the name the class the image belongs to and `<file>` is the image's file name. `<x-pos>` and `<y-pos>` is the position at which the feature has been extracted and `<value_d>` are the values of the local feature. As the format has neither header nor terminating element it can easily be used with many Unix tools like `cut` and `awk`.

tdlf_create

This program is used to calculate some tangents on the local features. It is given a local feature file, as that described in the last section, on the command line or on "standard in" and returns a file with tangents.The program and its parameters are

```
tdlf_create [options] [infile]
```

```
  options
    tangents:
      -h:     Horizontal Shift.
      -v:     Vertical shift.
      -1:     Hyp1.
      -2:     Hyp2.
      -s:     Scale.
      -r:     Rotate.
      -t:     Line thickness.
      -a:     Additive brightness.
      -m:     Multiplicative brightness.
    -b <v>: How the borders will be calculated (m: mean, b: mean
            of border, f<num>: set bg to value <num>).
    -o <name>: Outfile name.
```

The parameters `-h`, `-v`, `-s`, `-r`, `-a` and `-m` are self explanatory. `-1` and `-2` are the axis and the diagonal deformation respectively and `-t` is the line thickness deformation. `-b` sets which color to assume for pixels outside of the feature. If it is given `f<num>` the number given by `<num>` is taken as background color. If `m` is given, the mean color of the local feature is taken and if `b` is given the mean of the border pixels is chosen for pixel outside the feature. The parameter `-o` gives the output file name. If it is not given, “standard out” is used.

Usually, when using this program, the local features are extracted 4 pixel larger than wanted, e.g. if we want 19×19 sized features we extract them with 23×23, calculate the tangent and crop off a border of 2 pixel later with the program `lf_crop` that is presented hereafter. If the important parts of the original image are close to the border, the image should be padded. This can be done with the program `drawBorder` for example, or with `convert` from the *ImageMagick* package.

The file format is

```
<rem_tans> <tan_name> - - <value_1> <value_2>  . . .  <value_D>
<rem_tans> <tan_name> - - <value_1> <value_2>  . . .  <value_D>
    .
    .
    .
<rem_tans> <tan_name> - - <value_1> <value_2>  . . .  <value_D>
```

`<rem_tans_t>` specifies the number of tangents left from the respective local feature. `<tan_name>` is a one character identifier for the tangent, the identifiers are identical to the command line parameters of `tdlf_create`. And the `value_i` are the components of the tangent.

lf_crop

This program is used to crop out the center of local features. This is necessary, as sometimes the local features have to be extracted with a frame to reduce effects at the borders during processing. This is for example important when filtering the local features or extracting the tangents. The program is called by

```
lf_crop [options] [infile]
  options
    -c <borderWidth>: How much should be cut of at the borders.
    -a <nrFeat>: How many features are per line (for example when using color
                 images) it is expected that every feature is in one piece
```

```
                        and is squared!
    -o <name>:        Name of output file. If this is ommitted or if '-' is given,
                      standard out is taken for output.
```

The argument of the parameter `-c` specifies the size of the border that should be cropped. If a `*.vec`-file contains more than one local feature per line (this can happen when using color images), the number of features per line has to be given to the program as argument to the `-a` parameter. The file that the results should be written to is given with the `-o` parameter. It is written to “standard out”, if the flag is omitted or if its argument is `'-'`. Otherwise the result is written to the file indicated by the argument. If its name ends in `.gz` the output is gzipped.

pca_reduce

This program is used to compute the principal components of the given training data. The program has to be used in two steps. In the first one the reduction matrix is computed and in the second one the data is reduced. The program and its parameters are

```
pca_reduce
  usage:
    pcareduce -c <outfile.mat> <reduceDim> <file1.vec> [<file2.vec> [ ... ] ]
    pcareduce -r <transform.mat> [<infile> [<outfile>]]
```

If the parameter `-c` is given, the transformation matrix is computed out of training data. The result is written into `<outfile.mat>`. `<reduceDim>` the dimension to which the local features should be reduced and the `<file.vec>` are local feature files as described in Section B.

The parameter `-r` indicates that it should be reduced. The parameter `<transform.mat>` is an indicator to the transformation matrix that has been calculated in the former step. Of course the matrix does not have to be from a PCA, but every matrix with the right format will be used to reduce. The format is

```
<H> <W>
<val_1_1>   . . .   <val_1_W>
    .                   .
    .                   .
    .                   .
<val_H_1>   . . .   <val_H_W>
```

create_tree

The next step is to create the search structure. This is done with `create_tree`. The program is called with

```
create_tree [-o <path>] trainfile.vec[.gz]
```

`trainfile.vec` is a concatenation of all training features, it can be gzipped. For example if the local feature files are gzipped and in the directory `lf`, this can be created by

```
$ zcat lf/*.vec.gz | gzip > train.vec.gz
```

If the results should not be written to the local directory the output directory can be given with the `-o` parameter. This creates KD-trees of the form `<class>.kdt` and a file that is called `fileToKdtree` and contains information about the training. It has the following form

```
vecDim: <dim> globalVariance: <variance> [other keyword, value pairs]
<C> <Path> <NLCF> <CVar> <NNVar> <I> <INr> <NLIF> . . . <I> <INr> <NLIF>
 .
 .
<C> <Path> <NLCF> <CVar> <NNVar> <I> <INr> <NLIF> . . . <I> <INr> <NLIF>
```

The meanings of the symbols are:

`<dim>`	:	Dimension of the features.
`<variance>`	:	Global variance.
`<C>`	:	Name of the class. All information of that row belongs to this class.
`<Path>`	:	Path to the KD-tree of the class.
`<NLCF>`	:	Number of the last feature belonging to the class.
`<CVar>`	:	The variance of gray values within the class.
`<NNVar>`	:	The variance of nearest neighbors within the class. This is typically not computed for the corpora, as it is quite time consuming. In this case a `0` is written.
`<I>`	:	Name of first image from which the local features are, from the actual
	:	class.
`<INr>`	:	The number of the image. This is needed for leaving one out.
`<NLIF>`	:	Number of the last feature from that image. This counter starts from 0 for every class.

This file and the KD-tree files are needed for the classification program.

lf_classify

This is the main program for evaluation. It loads the KD-trees that have been generated by the previous program and that are indicated by the `fileToKdtree` file. Then it classifies every `*.vec`-file that is either given on the command line or in a file. The parameters of `lf_classify` are:

```
lf_classify [options] [testfile1 [testfile2 [ ... ] ] ]

options:
        -E              Euclidean distance measure for nearest neighbour search.
        -T              Tangent distance measure for nearest neighbour search.
        -m <mul>        A multiplier that specifies how many nearest neighbours
                        should be pre selected with the Euclidean distance
                        extracted before doing the tangent distance.
        -t <thresh>     A threshold for thresholded distance (now only works for
                        tangent distance).
        -K              Kernel densities.
        -w <weight>     Gives the weight for the kernel densities.
        -P              Pool the variance of the classes.
        -V              Use the class variance.
        -D              Use direct voting.
        -p              Use product rule.
        -s              Use sum rule.
        -f <fileToKdtree> File that describes the training results.
        -o <outfile>    Name of the file where the results are written to.
        -k <num>        How many nearest neighbours are searched.
        -e <eps>        Error of the approximate nearest neighbour search.
        -F <fileOfTestFiles>    File where each line contains the information of
```

```
                      one test.
    -Y <yasmetFileN> Write output to <yasmetFileN> that is suitable for
                      Yasmet (or with '-' write to stdout).
    -S [diDI]         Set which kind of feature functions you want for Yasmet.
            d: Per local feature, class dependent search.
            i: Per local feature, class independent search.
            D: Global, class dependent search.
            I: Global, class independent search.
    -q                Quiet, give less output.
```

The parameters are of three categories.

1. Organizational: those are parameters that deal with the interaction with the operating system, as for example file locations, etc. `-f`, `-o`, `-F`, `-Y`, and all arguments after the last parameter are of this category.
2. Setup parameters: those deal with which kind of experiment will be done. `-E`, `-T`, `-t`, `-K`, `-D`, `-p`, `-s` and `-S` are of this type.
3. Setting parameters: those are used to set some parameters within the experiments. `-m`, `-w`, `-k` and `-e` are from this group.

I will describe the parameters in the order given here.

From the organizational parameters, `-f` is the most important. It specifies the path to the `fileOfTestFiles`-file. This file describes where the training data is stored, and how it is composed. The name of the file that the results are written to is given as argument to the `-o` parameter. The test-files are given on the command line or in a file by giving one line per test. This parameter has to be used when taking the tangent distance, as in that case the test-file name, as well as the tangent-file name as well as the used tangents have to be specified. But it is also useful, if there are more test files than the operating system supports as command line arguments (yes, this happens!). The parameter `-Y` specifies to produce output that can be used as input for the `Yasmet` program. This program implements a generalized iterative scaling algorithm, to calculate the maximum entropy probability distribution. Also it can run tests with this distribution. Different feature functions can be extracted. One is that described in Section 3.7.

The setup parameters `-E` and `-T` specify that Euclidean or tangent distance are to be used. If `-T` is given the parameter `-F` from the previous paragraph must be used to specify the testing conditions. The file given to `-F` must be of the following form.

```
<testfile_1> <tangentfile_1> <tangents>
<testfile_2> <tangentfile_2> <tangents>
     .
     .
     .
<testfile_N> <tangentfile_N> <tangents>
```

The `<tangents>`-part is a substring of `hv12srtam`. Each letter `<c>` stands for the tangent that is extracted by `tdlf_create` when using the command line parameter `-<c>`. The parameter `-t` must be given, if the thresholded distance should be used and its argument defines the threshold. The parameters `-D` and `-K` specify how the a-posteriori probability should be estimated, by direct voting or kernel densities. By using the parameters `-p` or `-s` one can decide, how the probabilities of the features are combined. The parameter `-S` needs an argument that specifies which feature functions is to be calculated, and if yasmet should train the feature probability or the overall probability.

The meanings of the setting parameters are as follows. The tangent distance is not calculated between all local features, but a set of similar features is searched using the Euclidean distance. The tangent distance is only calculated for those. The argument of the `-m`-parameter is a multiplier and the product of this and the argument of `-k` specifies the size of the set. `-k` is the number of nearest neighbors that are searched. If it is chosen large along with `-K` we get a better approximation to the kernel densities. The argument of `-w` will be the α-factor to the variance for the kernel densities. And finally the argument to `-e` specifies the ε parameter for the approximate tree search.

The local feature files as well as the tangent distance files can be given gzipped. However gzipped files must have a `.gz` suffix.

The output is of the following form:

```
#lf_classify <command line arguments used>
<testImageName_1> <class_0> <res> <class_1> <res>  . . .  <class_N> <res>
<testImageName_2> <class_0> <res> <class_1> <res>  . . .  <class_N> <res>
       .
       .
       .
# <testImageName_m>
       .
       .
       .
<testImageName_M> <class_0> <res> <class_1> <res>  . . .  <class_N> <res>
```

The first line contains the command line with all its arguments commented out by a `#`-symbol. Then the results for every test image follow, one per line. They are given by the image name, followed by the class-labels with their respective scores. The class with the highest score is the Winner. The scores are not probabilities, as they have not been normalized. This is not necessary, because they would have to be divided by the same number for all classes and so there would be no difference for the decision rule. The error rate in percent is returned by the `evalresult.py` script. This is presented in the following.

evalresult.py

This *Python* script is used to calculate the error rate from the result file of `lf_classify`. Its first argument specifies the corpus that is used. The possibilities are `blood`, `irma`, `orl` and `erlangen` for the respective corpora. The second, optional parameter is the name of the result file.

Example session

The following is an example of how the presented programs are used. The scenario is that the error rate should be estimated for the IRMA-corpus. However, the distances should not be calculated on the images but on their horizontal and vertical Sobel transforms. Additionally, we would like to use the tangent distance instead of the Euclidean norm.

First, we calculate the PCA-Matrix. This is estimated on the local features, so that we extract those.

```
$ cd IRMA

$ # first enlarged the images
$ for i in *.pgm; do drawBorder -c 2 $i > $i"_big"; done
$ # then extract local features with a border of 4 that is cropped later
$ for i in *.pgm_big
  do
    cat $i | tk_extract_local_features ${i%%.*} $i -w 11 -t 20 |
      lf_filter -f ~/filters/sobelH.fil -f ~/filters/sobelV.fil |
      lf_crop -c 2 -a 2
  done | gzip > ../train_pca.vec.gz
$ pca_reduce -c ../IRM_SOB_PCA_40.mat 40 ../train_pca.vec.gz
$ rm ../train_pca.vec.gz

$ # now the local features are extracted
$ mkdir ../IRM_SOB_PCA_40
$ mkdir ../IRM_SOB_PCA_40/lf
$ for i in *.pgm_big
  do
    cat $i | tk_extract_local_features ${i%%.*} $i -w 11 -t 20 |
      lf_filter -f ~/filters/sobelH.fil -f ~/filters/sobelV.fil |
      lf_crop -c 2 -a 2 |
      pca_reduce -r ../IRM_SOB_PCA_40.mat |
      gzip > ../IRM_SOB_PCA_40/lf/${i%.pgm}.vec.gz
  done

$ # and the tangents
$ for i in *.pgm
  do
    cat big_$i | tk_extract_local_features ${i%%.*} $i -w 11 -t 20 |
      tdlf_create -h -v -1 -2 -s -r -a -m |
      lf_filter -f ~/filters/sobelH.fil -f ~/filters/sobelV.fil |
      lf_crop -c 2 -a 2 |
      pca_reduce -r ../IRM_SOB_PCA_40.mat |
      gzip > ../IRM_SOB_PCA_40/lf/${i%.pgm}.tan.gz
  done

$ # the search structure is being created
$ cd ../IRM_SOB_PCA_40
$ zcat lf/*.vec.gz | gzip > train.vec.gz
$ create_tree train.vec
$ rm train.vec.gz

$ # decide which test files and which tangents are used
$ for i in lf/*.vec.gz
  do
    echo $i ${i%.vec.gz}.tan.gz hv12sra
  done > tst_tan_hv12sra

$ # and finally do the tests
$ lf_classify -T -m 250 -K -w .1 -P -f fileToKdtree -o result -F tst_tan_hv12sra

$ # now evaluate the result file
$ evalresult.py 'irma' result | less
```

By splitting all tasks in different applications, the process is flexible and can easily be extended.

Appendix C

Programs Used

In the course of this work I used a lot of free software. I would like to thank the authors of the following tools: For the development of the programs and to write this thesis I used XEmacs and Vim, for compiling I used tools from the gnu compiler collection, a lot of small programs were developed with python, swig, awk, bash, and octave. LaTeX was used for typesetting this work, GNU textutils were used for data processing, Yasmet was used for experiments with log-linear models.

Bibliography

[Arya & Mount$^+$ 98] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, A. Y. Wu. An Optimal Algorithm for Approximate Nearest Neighbor Searching. *Journal of the ACM*, Vol. 45, pp. 891–923, 1998.

[Ball & Hall 65] G. Ball, D. Hall. ISODATA, A Novel Method of Data Analysis and Classification. Technical report, Stanford University, Stanford, CA, USA, 1965.

[Belongie & Malik$^+$ 00] S. Belongie, J. Malik, J. Puzicha. Shape Context: A New Descriptor for Shape Matching and Object Recognition. Proc. *Advances in Neural Information Processing Systems*, pp. 831–837, Denver, CO, USA, 2000.

[Berger & Della Pietra$^+$ 96] A. L. Berger, S. A. Della Pietra, V. J. Della Pietra. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, Vol. 22, No. 1, pp. 39–71, 1996.

[Brown & Lowe 02] M. Brown, D. Lowe. Invariant Features from Interest Point Groups. Proc. *British Machine Vision Conference*, pp. 656–665, Cardiff, Wales, UK, Sept. 2002.

[Chen & Gu$^+$ 01] X. Chen, L. Gu, S. Z. Li, H.-J. Zhang. Learning Representative Local Features For Face Detection. Proc. *IEEE Computer Vision and Pattern Recognition*, Kauai Marriott, Hawaii, USA, Dec. 2001.

[Dahmen & Hektor$^+$ 00] J. Dahmen, J. Hektor, R. Perrey, H. Ney. Automatic Classification of Red Blood Cells using Gaussian Mixture Densities. Proc. *Bildverarbeitung für die Medizin*, pp. 331–335, Munich, Germany, March 2000.

[Dahmen & Keysers$^+$ 00] J. Dahmen, D. Keysers, M. O. Güld, H. Ney. Invariant Image Object Recognition using Mixture Densities. Proc. *15th International Conference on Pattern Recognition*, Vol. 2, pp. 614–617, Barcelona, Spain, Sept. 2000.

[Dahmen & Keysers$^+$ 01] J. Dahmen, D. Keysers, H. Ney, M. O. Güld. Statistical Image Object Recognition using Mixture Densities. *Mathematical Imaging and Vision*, Vol. 14, No. 3, pp. 285–296, May 2001.

[Deselaers & Keysers$^+$ 03] T. Deselaers, D. Keysers, R. Paredes, E. Vidal, H. Ney. Local Representations for Multi-Object Recognition. Proc. *25th DAGM Symposium*, Vol. 2781 of *Lecture Notes in Computer Science*, pp. 305–312, Magdeburg, Germany, Sept. 2003. Springer Verlag.

[Deselaers 03] T. Deselaers. Features for Image Retrieval. Diploma thesis, Lehrstuhl für Informatik VI, RWTH Aachen, Aachen, Germany, Dec. 2003.

[Duda & Hart$^+$ 01] R. O. Duda, P. E. Hart, D. G. Stork. *Pattern Classification.* John Wiley & Sons, Inc., New York, 2001.

[Fergus & Perona$^+$ 03] R. Fergus, P. Perona, A. Zisserman. Object Class Recognition by Unsupervised Scale-Invariant Learning. Proc. *IEEE Computer Vision and Pattern Recognition*, Vol. 2, pp. 264–275, Madison, Wisconsin, USA, June 2003.

[Gollan 03] C. Gollan. Nichtlineare Verformungsmodelle für die Bilderkennung. Diploma thesis, Lehrstuhl für Informatik VI, RWTH Aachen, Aachen, Germany, Sept. 2003.

[Hastie & Tibshirani$^+$ 01] T. Hastie, R. Tibshirani, J. Friedman. *The Elements of Statistical Learning.* Springer Verlag, Berlin, Germany, 2001.

[Int 92] The International Telegraph and Telephone Consultative Committee. *Information Technology - Digital Compression and Coding of Continuous-Tone Still Images - Requirements and Guidelines*, Sept. 1992.

[Jähne 02] B. Jähne. *Digitale Bildverarbeitung.* Springer Verlag, Berlin, Germany, 5th edition, 2002.

[Jain & Dubes 88] A. K. Jain, R. C. Dubes. *Algorithms for Clustering Data.* Prentice Hall, 1988.

[Jain & Duin$^+$ 00] A. K. Jain, R. P. W. Duin, J. Mao. Statistical Pattern Recognition: A Review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 1, Jan. 2000.

[Jaynes 57] E. T. Jaynes. Information Theory and Statistical Mechanics. *Physical Review*, Vol. 106, pp. 620–630, 1957.

[Keysers & Dahmen$^+$ 00] D. Keysers, J. Dahmen, T. Theiner, H. Ney. Experiments with an Extended Tangent Distance. Proc. *15th International Conference on Pattern Recognition*, Vol. 2, pp. 38–42, Barcelona, Spain, Sept. 2000.

[Keysers & Dahmen$^+$ 01] D. Keysers, J. Dahmen, H. Ney. Invariant Classification of Red Blood Cells. Proc. *Bildverarbeitung für die Medizin*, pp. 367–371, Lübeck, Germany, March 2001.

[Keysers & Dahmen$^+$ 03] D. Keysers, J. Dahmen, H. Ney, B. B. Wein, T. M. Lehmann. A Statistical Framework for Model-Based Image Retrieval in Medical Applications. *Journal of Electronic Imaging*, Vol. 12, No. 1, pp. 59–68, Jan. 2003.

[Keysers & Och$^+$ 02] D. Keysers, F. Och, H. Ney. Maximum Entropy and Gaussian Models for Image Object Recognition. Proc. *24th DAGM Symposium*, Vol. 2449 of *LNCS*, pp. 498–506, Zürich, Switzerland, 2002. Springer Verlag.

[Keysers & Paredes$^+$ 02] D. Keysers, R. Paredes, H. Ney, E. Vidal. Combination of Tangent Vectors and Local Representations for Handwritten Digit Recognition. Proc. *International Workshop on Statistical Pattern Recognition*, Vol. 2396 of *LNCS*, pp. 538–547, Windsor, Ontario, Canada, Aug. 2002. Springer Verlag.

[Keysers 00] D. Keysers. Approaches to Invariant Image Object Recognition. Diploma thesis, Lehrstuhl für Informatik VI, RWTH Aachen, Aachen, Germany, June 2000.

[Kittler & Hatef$^+$ 98] J. Kittler, M. Hatef, R. P. Duin, J. Matas. On Combining Classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 3, pp. 226–239, March 1998.

[Lehmann & Gönner$^+$ 99] T. M. Lehmann, C. Gönner, K. Spitzer. Survey: Interpolation Methods in Medical Image Processing. *IEEE Transactions on Medical Imaging*, Vol. 18, No. 11, pp. 1049–1075, Nov. 1999.

[Lowe 99] D. G. Lowe. Object Recognition from Local Scale-Invariant Features. Proc. *International Conference on Computer Vision*, pp. 1150–1157, Corfu, Sept. 1999.

[Messer & Kittler$^+$ 03] K. Messer, J. Kittler, M. Sadeghi, S. Marcel, C. Marcel, S. Bengio, F.Cardinaux, C. Sanderson, J. Czyz, L. Vandendorpe, S. Srisuk, M. Petrou, W. Kurutach, A. Kadyrov, R. Paredes, B. Kepenekci, F. B. Tek, G. B. Akar, F. Deravi, N. Mavity. Face Verification Competition on the XM2VTS Database. Proc. *4th International Conference on Audio and Video Based Biometric Person Authentication*, pp. 964–974, Guildford, UK, June 2003.

[Mohr & Picard$^+$ 97] R. Mohr, S. Picard, C. Schmid. Bayesian Decision Versus Voting for Image Retrieval. Proc. *7th International Conference on Computer Analysis of Images and Patterns*, pp. 376–383, Kiel, Germany, 1997.

[Ney 00] H. Ney. Mustererkennung und Neuronale Netze. Script to the lecture on Pattern Recognition and Neural Networks held at RWTH Aachen, Aachen, Germany, 2000.

[Ney 01] H. Ney. Speech Recognition. Script to the lecture on Speech Recognition held at RWTH Aachen, Aachen, Germany, 2001.

[Paredes & Keysers$^+$ 02] R. Paredes, D. Keysers, T. M. Lehmann, B. B. Wein, H. Ney, E. Vidal. Classification of Medical Images Using Local Representations. Proc. *Bildverarbeitung für die Medizin*, pp. 171–174, Leipzig, Germany, March 2002.

[Paredes & Pérez$^+$ 01] R. Paredes, J. C. Pérez, A. Juan, E. Vidal. Local Representations and a Direct Voting Scheme for Face Recognition. Proc. *Workshop on Pattern Recognition in Information Systems*, pp. 71–79, Setúbal, Portugal, July 2001.

[Press & Teukolsky$^+$ 02] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery. *Numerical Recipes in C++*. Cambridge University Press, 2002.

[Ratnaparkhi 97] A. Ratnaparkhi. A Simple Introduction to Maximum Entropy Models for Natural Language Processing. Technical Report 97-08, Institute for Research in Cognitive Science, University of Pennsylvania, Pennsylvania, USA, May 1997.

[Reinhold & Paulus$^+$ 01] M. Reinhold, D. Paulus, H. Niemann. Appearance-Based Statistical Object Recognition by Heterogeneous Background and Occlusions. Proc. *23rd DAGM Symposium*, pp. 50–58, Sept. 2001.

[Schmid 99] C. Schmid. A Structured Probabilistic Model for Recognition. Proc. *Conference on Computer Vision and Pattern Recognition*, Vol. 2, pp. 485–490, Fort Collins, Colorado, USA, June 1999.

[Schölkopf & Simard$^+$ 98] B. Schölkopf, P. Simard, A. Smola, V. Vapnik. Prior Knowledge in Support Vector Kernels. Proc. M. I. Jordan, M. J. Kearns, S. A. Solla, editors, *Advances in Neural Information Processing Systems*, Vol. 10, pp. 640–646. MIT Press, 1998.

[Simard & Le Cun$^+$ 93] P. Simard, Y. Le Cun, J. Denker. Efficient Pattern Recognition Using a New Transformation Distance. Proc. S. Hanson, J. Cowan, C. Giles, editors, *Advances in Neural Information Processing Systems*, Vol. 5, pp. 50–58, San Mateo, CA, USA, 1993.

[Simard & Le Cun$^+$ 98] P. Simard, Y. Le Cun, J. Denker, B. Victorri. Transformation Invariance in Pattern Recognition — Tangent Distance and Tangent Propagation. *Lecture Notes in Computer Science*, Vol. 1524, pp. 239–274, 1998.

[Tipping 00] M. E. Tipping. The Relevance Vector Machine. Proc. S. Solla, T. Leen, K. Müller, editors, *Advances in Neural Information Processing Systems 12*, pp. 332–388, Cambridge, MA, USA, 2000. MIT Press.

[Wiskott & Fellous$^+$ 97] L. Wiskott, J.-M. Fellous, N. Krüger, C. von der Malsburg. Face Recognition by Elastic Bunch Graph Matching. Proc. G. Sommer, K. Daniilidis, J. Pauli, editors, *7th Intern. Conf. on Computer Analysis of Images and Patterns*, Vol. 1296 of *LNCS*, pp. 456–463, Heidelberg, Germany, 1997. Springer Verlag.

Index