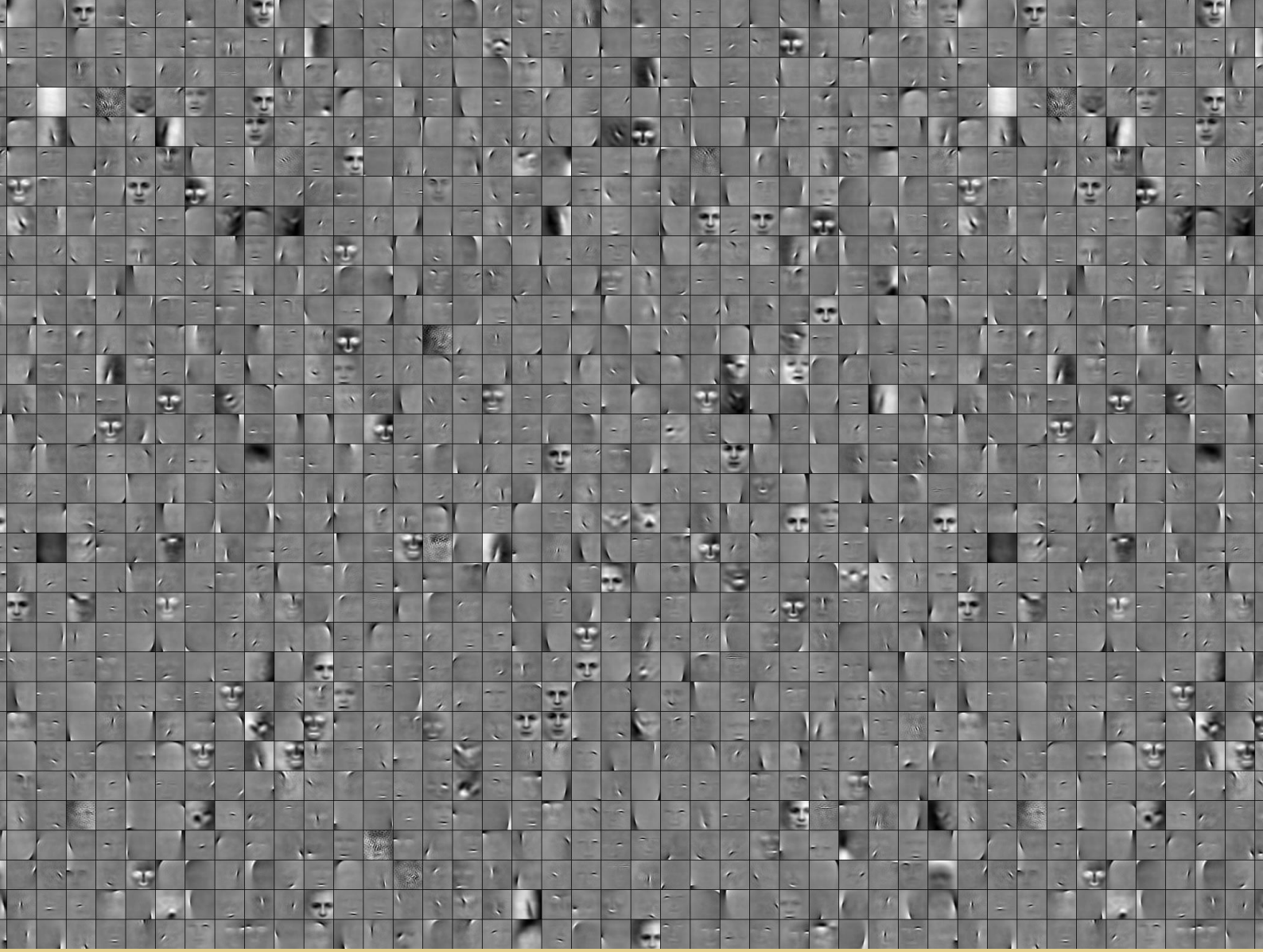


Pedro Miguel Neves Marques

Discriminative Sparse Representation for Expression Recognition

Master Thesis in
Electrical and Computer Engineering
September, 2014

· U C ·

Universidade de Coimbra

FCTUC FACULDADE DE CIÊNCIAS
E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Discriminative Sparse Representation for Expression Recognition

Author:
Pedro Marques

Supervisor:
Prof. Dr. Jorge Batista

Master Thesis in Electrical and Computer Engineering

President of the Jury:
Prof. Dr. Nuno Gonçalves
Jury:
Prof. Dr. João Ferreira
Prof. Dr. Jorge Batista

September 2014

In loving memory of my grandfather Manuel

Resumo

Esta tese tem como objetivo reconhecer emoções em imagens 2D. Iremos analisar as várias etapas que levam à resolução deste problema. Pretendemos tirar o máximo partido possível da informação disponível acerca das classes das amostras usadas nas fases de aprendizagem.

Vamos comparar vários métodos de processamento de imagem e mostrar que os métodos que usam histogramas como descritores são mais precisos no reconhecimento de emoções.

Uma vez que a popularidade dos métodos de representação esparsa em visão por computador, mais concretamente em reconhecimento facial, tem aumentado ultimamente, vamos testar um método esparso recente, que permite a construção de dicionários fazendo uso de informação acerca das classes usadas no treino. Vamos também compará-lo com outros métodos supervisionados de redução de dimensão.

Por fim vamos apresentar um novo método para reconhecimento de emoções, para o qual iremos obter taxas de classificação superiores a outros trabalhos nesta área. Este método fará uso de histogramas e uma combinação de redução de dimensão supervisionada e representação esparsa. Vamos testar o nosso algoritmo com várias bases de dados, contendo expressões, alinhadas ou não, de vários indivíduos.

Keywords

Reconhecimento de Emoções, Histogramas Descritores, Preservação da Relação entre Dados, Redução de Dimensão Supervisionada, Dicionários de Descritores, Representação Esparsa

Abstract

This thesis is focused on recognising emotions of different subjects through facial expressions in 2D images. We will go through the multiple stages of this problem where we aim to take maximum advantage of supervised algorithms and labelled information.

We will compare different pixel processing techniques and show that the histogram based ones, like HOG and LBP, have the best performance for this particular problem.

Sparse representation has definitely been proved to be a very good way to solve computer vision problems in facial understanding over the last couple of years. Therefore we will make use of a new label consistent singular value decomposition algorithm to learn a discriminative dictionary and compare its performance with several supervised dimensionality reduction techniques.

Finally we will obtain state-of-the-art classification accuracies for the problem of recognising facial expressions with our histogram supervised manifold preserving sparse representation technique. We will test different methods across multiple databases containing images of various subjects performing various expressions, aligned or non-aligned.

Keywords

Emotion Recognition, Histogram Descriptors, Manifold Preserving, Supervised Dimensionality Reduction, Orthogonal Matching Pursuit, Sparse Representation

Acknowledgements

Right off the bat I would like to thank Pedro Martins, João Faro and João Filipe for welcoming and helping me at the lab. It was good to have such a great atmosphere during all that work.

To my supervisor Prof. Jorge Batista for counselling and pushing me but most of all for his constant availability and will. His work ethic and candour makes it hard to imagine how would it be like without him.

To my family and friends for the endless care and support. It is a joy to be with you.

And on a special note I would like to thank my parents. Thanking seems to be so irrelevant when one just gives you so much comfort, love and confidence. They set me no barriers, were always present and support me endlessly. I know I will carry this debt for the rest of my life, the only thing that eases it is having a sense that I can make them proud. Mãe e Pai, não tenho a intençâo de vos fazer perceber o quanto valorizo o vosso esforço e dedicação, apenas espero que um dia se apercebam.

Contents

List of Tables

Acronyms

BU-3DFE 3D Facial Emotion Dataset [1]

CK+ Extended Cohn-Kanade Database [2]

DLPP Discriminant Locality Preserving Projections

DR Dimensionality Reduction

HDSMPSR Histogram Descriptors' Supervised Manifold Preserving Sparse Representation, our proposed method

HOG Histograms of Oriented Gradients

k-NN k-Nearest Neighbours

KDEF Karolinska Directed Emotional Faces [3]

KNDA Kernel Neighbourhood Discriminant Analysis

LBP Local Binary Patterns

LCK-SVD Label Consistent K-SVD

LDA Linear Discriminant Analysis

OMP Orthogonal Motion Pursuit

PCA Dimensionality Reduction

PPT Pixel Processing Technique

SR Sparse Representation

SVD Singular Value Decomposition

SVM Support Vector Machine

Chapter 1

Introduction

Feelings are expressed through a variety of manners but mainly we make use of our faces to display reactions accordingly to events. As most of the times we do it subconsciously, facial expressions are key for others to evaluate our mood. But what is truly interesting about them is that they are identical across different people having different gender, race or ages. After personal identification the information contained in a face through the emotion it portrays is of substantial importance. Accurately characterising it through computer vision could lead to varieties of applications. From surveillance security systems to human-computer interaction, all could benefit from information about the state of spirit of the target or the user respectively. Also in the area of psychiatry, emotion recognition could be useful to evaluate a subject's response to various stimulus.

Throughout this document we will often mention facial recognition and expression recognition. These terms are not to be confused as the first is about distinguishing subjects and the later about distinguishing emotions. Even though the development of computer vision over the years has been immense in the field of identity recognition, and that we are now comfortable to say that it is possible for an automated system to recognise faces associating them to an individual, facial understanding is still not developed at that level. Expression recognition algorithms are far from being as robust as identity recognition ones, since they still output mediocre classification results under uncontrolled environments. For each case discriminant features are different but much more subtle on emotion labelling. That is because instead of extracting distances between mouth, nose, eyes, ears and face's contours, we are mainly looking for textures.

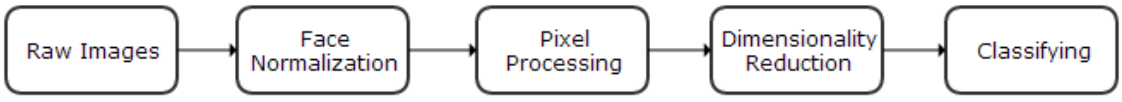


Figure 1.1: Different stages before classifying expressions.

(a) (b) (c) (d) (e) (f) (g) (h)

Figure 1.2: Examples of the emotions present in the CK+ database. (a)Neutral (b)Anger (c)Contempt (d)Disgust (e)Fear (f)Happiness (g)Sadness (h)Surprise

Needless to say that classifying expressions poses a bigger of a problem. As we will be dealing with natural images which are highly dimensional non-linear data, feature extraction and manifold learning are essential for good discrimination between different expressions across different individuals. After all the essential reduced information is at our disposal classification is what follows. At this stage focus has been given to sparse coding for this type of problems, not only because it emulates the nature of humans' primary visual cortex, but also because it can extract similar components of different expressions at a low computational cost.

For what follows we will attempt to look for the best combinations on each of the stages' techniques displayed in Figure 1.1. We will test them on several datasets containing images of frontal faces with the respective labelled emotions.

As there is a need to discuss methods and accuracy rates within the scientific community on emotion classification it is mandatory the use of an internationally recognised dataset. Only then we are able to really evaluate our proposed method. The database we will be using for algorithm development is the Extended Cohn-Kanade (CK+) database [2] which is the most widely used for individual facial expression detecting. This dataset contains images of subjects performing Paul Ekman's six basic emotions plus Contempt, from Neutral pose to the expression's appex. Examples can be seen in Figure 1.2. We will then test our algorithm on other datasets as well. And propose our best solution.

1.1 Related Work

In order to recognise expressions regardless of the environment much work has been recently developed on pixel processing techniques. Local binary patterns were first used for analysing textures [4, 5]. In 2006 Ahonen et al.[6] first implemented local binary patterns for face description. Later [7] combined it with spatial pyramid matching in order to deal with different poses and lighting conditions, and since then plenty of work [8] has been developed. Gabor filters are also very popular for feature extraction. Kekre et al.[9] used them for face recognition in mobile phones, while [10] for disguise proof recognition. Zhang et al.[11] even combined it with LBP in order to sharpen descriptors. In 2005 Dalal et al.[12] first described the concept of

histogram of oriented gradients for human detection in images. Since then more works have been developed in that area [13, 14]. Although they become extensively used for face detection [15, 16] and face recognition [17, 18], they are less common for expression recognition.

There has also been recent work developed on dimensionality reduction [19, 20] where emphasis is given to non-linear locally embedding dimension reduction. But we will mainly focus on the works [21, 22].

Besides the endless study on identity recognition there has been recent work developed on the subject of expression recognition. Thai et al.[23] chose to processed certain facial features separately to later use an artificial neural network, while Shan et al. [24] developed a boosting mechanism to automatically extract the most discriminant LBP features, and classify them using different SVMs. Ptucha et al.[25] used sparse representation and combined it with a regional based statistical inference model obtaining good results when classifying expressions. Zhang et al. used LBPs and Gabor wavelets and also obtained better results using sparse representation than neural networks and SVMs. Some other works opt to classify action units instead, such as [26, 27], to classify more frequent and general expressions. Mahoor et al.[28] used sparse representation to classify action units. There has also been algorithm development on expression classification using image sequences or video [29, 24, 30, 31], however these methods are quite different to those using raw images.

1.2 Overview

This thesis is divided into three main parts where we will go through all the techniques implemented for each stage of the problem of facial recognition, followed by the experimental results and conclusion. In this chapter we briefly presented information about the subject matter. In the next chapter, Chapter 2, we discuss four pixel processing algorithms that can better extract features from a raw face in a two dimensional image. In Chapter 3 four dimensionality reduction methods will be analysed and for each of them, advantages and handicaps are mentioned. As for Chapter 4 we approach three classification techniques and provide an insight on how they work. We will then discuss the experimental results in Chapter 5 and see how various combinations between the techniques mentioned perform. We will then take our conclusions in Chapter 6.

Chapter 2

Pixel Processing

As stated before the goal of this work is to recognise expressions in raw images. What it means is that our images are nothing but photos of multiple subjects performing up to eight expressions. For this reason they cannot be used for training and testing as is. After detecting the face on each of the photos, independently of the emotion, they are affine mapped to a reference and trimmed down only to contain the subject's face itself. The point of this is for each image to have the same size and contain roughly the same information per region. Meaning that the average of all processed images of a dataset be a slightly blurred face. Final images can then be compared and used for algorithm testing after being rearranged into vectors, i.e. if an image's size has h pixels by l, its vector will have a length of $h \times l$.

As Figure 1.2 suggests photos were taken under different environments with different luminosities and contrasts. This was purposely done as robust algorithms are intended. Same class expressions under different conditions increase the difference between them and complicate class clustering. For that reason and because natural images contain a vast amount of fuzzy information it is necessary to use pixel processing techniques. Despite having many of these techniques developed and available not all suit our interests the better.

The most simple pixel processing technique is image downscaling. There are several ways to do it, the majority of them using some kind of pixel colour value interpolation. This means that important details might be lost, such as wrinkles or small face patterns in this case. And even though sometimes downscaling alone obtains satisfactory results this is to be discouraged as better class discriminatory pixel processing techniques are available. However image down scaling is still useful for high resolution downsizing or subtle downscaling. If there is no considerable loss of information, downscaling is good for computational cost decrease and can really be interpreted as a first stage dimensionality reduction.

The following pixel processing techniques do collect discriminatory and key information present into images increasing between class scattering and consequently

algorithm accuracy. In the next four sections we will explain four techniques we consider to perform very well on facial understanding.

2.1 Sobel Filter

The Sobel filter [32] is a very popular operator in computer vision. Easy to compute and extremely useful for feature detection this filter allows for edge enhancement in images. It does so by exposing the biggest and most abrupt colour changes in an image using its gradient. This kind of points are best for reliable recognition, as they are fairly insensitive to noise and illumination.

To demonstrate how it works lets consider a grayscale image as an intensity function. This function varies discreetly between a maximum value that being white and a minimum correspondent to black. For each pixel a 3×3 neighbourhood pixel regions are defined around itself. In order to estimate the gradient inside this region the entries of G_x and G_y in Eq.(2.1) are multiplied with the intensity of the pixels in the 3×3 region. Pixel masks G_x and G_y are used as they respond maximally to horizontal and vertical edges respectively.

The results in Figure 2.1 are then achieved by simply convolving the Sobel filter Eq.(2.1) with the original image and calculating the results' magnitude 2.1 b) or the phase 2.1 c). As we can see in the Figure 2.1 the original is significantly altered only to display its edges, the best possible expression's feature. Both the edge's magnitude and phase can be used as image descriptors. In Figure 2.1 we can also evaluate the main advantage of using gradients, as no evidence of light variations influence the processed image. Although they do contain a fair amount of redundant information that can easily be dealt with afterwards. As we will see.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{2.1}$$

(a) Original Image (b) Edge Magnitude (c) Edge Phase

Figure 2.1: Example of Sobel filter usage.

2.2 Local Binary Patterns

Local Binary Patterns (LBP) have been used extensively in facial recognition. They are excellent texture descriptors as they are invariant to luminosity and pose changes. Which suits the needs for an accurate facial descriptor as Ahonen et al.[6] proved, obtaining very good results on identity recognition. Since then the use of LBP for facial understanding has been immense. It has a low computational cost and is easily understandable.

Firstly we define the radius R measured in pixels from a centre pixel and a number N of equidistant sample points placed over the respective circumference. Each of these points represents a pixel. Then the grayscale value of each sample pixel is compared to the centre one. If higher or equal the respective pixel assumes the value 1, and 0 otherwise. This process is pictured in Figure 2.2 and can be interpreted as the following equation:

$$LBP(x_c, y_c) = \sum_{p=0}^{N-1} 2^p H[I(x_p, y_p) - I(x_c, y_c)] \tag{2.2}$$

where $H[b]$ is an heaviside step function. Take note that sample points might not be placed in the centre of a pixel, in that case a bilinear interpolation is used. After defining a standard direction around the sample pixels and the starting bit the resulting binary values will originate a N bit binary word whose decimal value is added into an histogram of range 2^N. This process is repeated to all pixels except for those that are less than R pixels away from the image's margin. The notation for this is $\text{LBP}_{N,R}$.

2.2.1 Popular Variations

So basically $\text{LBP}_{N,R}$ maps patterns to a value between 1 and 2^N and counts the amount of times they occur in an image. But only having such an histogram however is still very limiting for facial understanding.

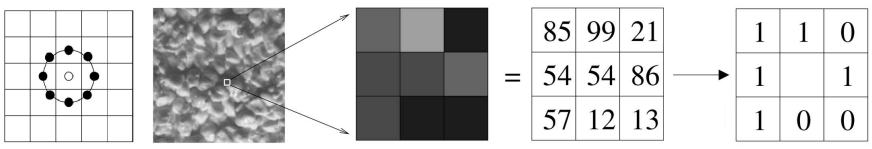


Figure 2.2: Illustration of the $\text{LBP}_{8,1}$ algorithm.

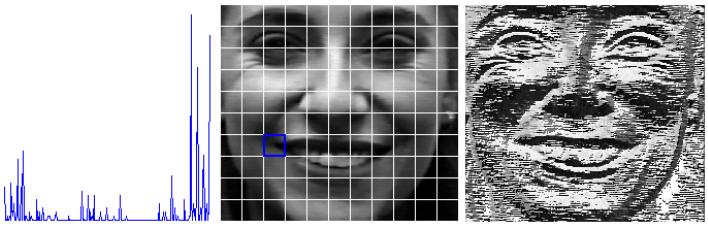


Figure 2.3: LBP histogram of a section, original image and pixel-wise LBP.

Image Decomposition

Ahonen et al. in their work [6] chose to decompose the images of the faces in sub-regions. This procedure is mandatory as not doing so "hides" the discriminant features of each face because the key binary patterns in key regions, as the eyes, mouth, nose, are added up to each other originating very similar different class histograms. For that reason LBP requires a prior image decomposition, pictured in Figure 2.3, sub-region histograms are then orderly concatenated into a single vector, that acts as the new image descriptor.

Images can be decomposed in any number of sections which do not even have to be equally sized. Different sections can also weight accordingly to its importance [6]. Having resulting LBP histograms multiplied by a weighting coefficient, for instance 0 for corners and 3 for mouth and eyes, can surely increase clustering accuracy.

Pixel-wise LBP

Another popular way of using LBP is to create a new image with each pixel having the decimal value of its correspondent in the original image according to the algorithm described above. This technique is known as pixel-wise LBP and is pictured left in Figure 2.3. As it is possible to see it eliminates all the original luminosity variations.

Uniform Patterns

A binary pattern is said to be uniform if it only contains up to two bit-wise transitions. Ojala et al.[5] shown that uniform patterns most frequently describe micro features such as edges and corners thence making better feature detectors. It is suggested for that reason that non uniform patterns be all assigned to the same bin in the LBP histogram. This is denoted as LBP^{u2}.

2.3 Gabor Filter

Gabor filters are very popular in computer vision because they mimic the visual cortex of mammals on image analysis. Its principle is straightforward but very

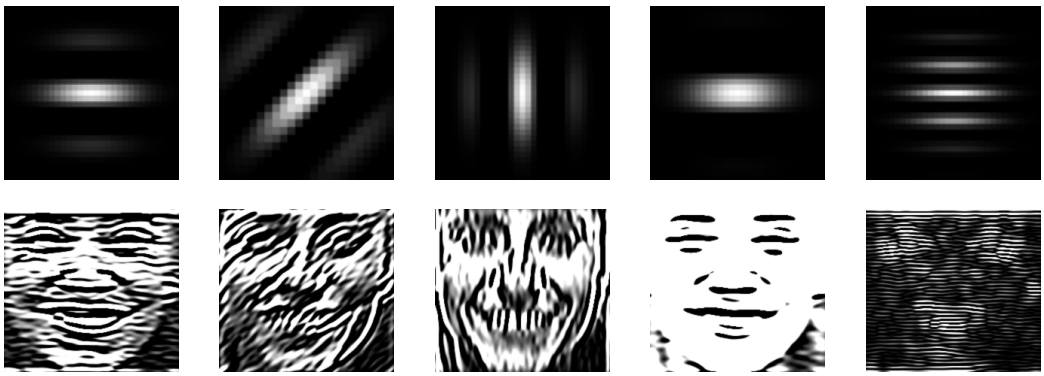
effective. It boils down to convolutions between the original image and a number of filters. Their impulse response is obtained by modulating a sinusoid wave with a Gaussian function, take a look at Eq.(2.3) and Figure 2.5.

$$\begin{gathered} x' = x\cos\theta + y\sin\theta \quad , \quad y' = -x\sin\theta + y\cos\theta \\ g(x, y; \lambda, \theta, \phi, \gamma, \sigma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right)\cos\left(2\pi\frac{x'}{\lambda} + \phi\right) \end{gathered} \tag{2.3}$$

For feature extraction a single filter generally proves to be ineffective, regardless of how tuned each of the five parameters might be. Instead a typical descriptor contains responses between the original image and a set of a few dozens of filters. There are five variable parameters in a Gabor filter, λ and ϕ are the frequency and offset of the sinusoidal component, θ the orientation of the filter, γ the ratio between the major and minor elliptical radius of the Gaussian function and σ^2 is the variance of the latter. As varying ϕ and γ has no advantage on finding Gabor patterns in images they can be removed from Eq.(2.3). The banks of filters are then generated by alternatively fixing two of the other three variables and vary the remainder. Still this method implies an high computational cost either to build the descriptors or to operate with them as they are very big.

In Figure 2.4 we can see how an image responds to few of the Gabor filters varying λ and *theta*. The binary like propriety of the responses is good for distinguishing relevant features amongst distinct classes and provides margin for data reduction later on.

Overall Gabor is a very solid method for feature extraction across all fields of computer vision. Its only drawback is the large descriptors it generates. Remember that each of the images in Figure 2.4 has the same size as the original and that a bank of filters can have between twenty to thirty different tuned filters which causes a Gabor descriptor to be that bigger than the original image. It is computationally



(a) $\lambda = 10\ \theta = 90°$ (b) $\lambda = 10\ \theta = 45°$ (c) $\lambda = 10\ \theta = 0°$ (d) $\lambda = 15\ \theta = 90°$ (e) $\lambda = 5\ \theta = 90°$

Figure 2.4: Some Gabor filters on a happy expression, fixing $\sigma = 5$, $\gamma = 1$ and $\phi = 0$

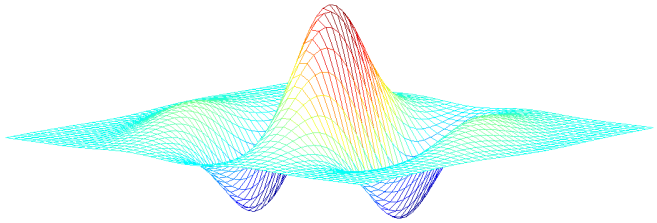


Figure 2.5: Example of Gabor filter.

costly and inefficient to deal with such large vectors which makes Gabor filters very dependent on dimensionality reduction techniques only to have conditions of operating later on.

2.4 Histograms of Oriented Gradients

Similar to LBP, Histograms of Oriented Gradients also output as a descriptor several histograms. But as its name suggests in this case the histograms contain information on the orientation of the gradients of the image rather than the patterns themselves. Truth is that gradients are as a power full of a tool.

In 2005 Dalal and Triggs [12] presented a method inspired in the already popular Scale Invariant Feature Transform and came up with extremely accurate descriptors for the problem of human detection in raw images. Although being a different reality we will put Histograms of Oriented Gradients (HOG) to the test and see how it performs on emotion recognition. We will not do this by chance. HOG does have proven results across several fields of computer vision. The HOG algorithm can be divided in four parts: horizontal and vertical gradient computation, orientation binning, block decomposition and block normalisation.

Gradient Computation

The actual foundation of HOG algorithm is the image's gradients. Their magnitude and phase will be used to calculate the histograms. The gradients are calculated using 1-D centred masks $G_x = [-1, 0, 1]$ and $G_y = [-1, 0, 1]^T$. Take note that other masks can be used, like the Sobel Filter in Section 2.1 for instance. But in [12] is shown that the simple $[-1, 0, 1]$ mask performs the best.

After gradient computation two matrices are outputted with the same size as the original image, respectively x and y image gradient's components. We then use Eq.(2.4) to calculate the magnitude and orientation of each pixel's gradient.

$$|G| = \sqrt{{G_x}^2 + {G_y}^2} \quad , \quad \theta = \arctan\left(\frac{G_x}{G_y}\right) \tag{2.4}$$

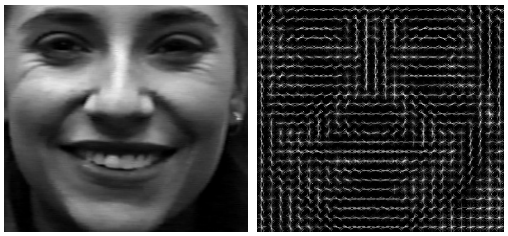


Figure 2.6: Illustration of HOG.

Orientation Binning

After having the orientations of each pixel's gradient we can now compose an actual histogram of oriented gradients. Like in the LBP in Section 2.2, HOG also requires an image decomposition into sections of $p \times p$ pixels named cells. Each cell will be described by one individual histogram.

An histogram can have as many orientation bins as desired between 0° and 180° or 360°. Each pixel of a cell will cast a weighted vote into the histogram. The votes are simply the pixel's gradient magnitude and they are interpolated bilinearly between the neighbouring bins.

Block Decomposition

Due to images' different illumination conditions and background to foreground contrast in human detection, [12] proposes the use of $c \times c$ cell overlapping blocks. Which means concatenating $c \times c$ cell's histograms and that each of them is used multiple times in the final descriptor. Although redundant, it significantly increases accuracy because each block is individually normalised. Rectangular blocks are called R-HOG but they can also be circular shaped, C-HOG.

In case of the most common R-HOG, $c \times b$ matrices where b is the number of orientation bins chosen will form the final HOG descriptor. The matrices will have integer approximates of h/p lines and l/p columns where h and l is the height and length of the original image.

Block Normalisation

So as mentioned above descriptors are normalised block wise. A number of normalisation schemes were tested in [12] and the best scheme was difficult to elect as all of them outputed good results. Besides clearly outperforming the results achieved with non normalised descriptors this step also makes it unnecessary for prepossessing raw images on colour and gamma normalisation. We will be using grayscale images but for RGB images it is usual to pick the colour channel with biggest overall contrast.

Chapter 3

Dimensionality Reduction

The majority of pixel processing techniques do enhance features present in images, but they do not necessarily discard any information. In fact the majority of descriptors resulting from different pixel processing techniques are way bigger than the original image. And not only are they too big for training and classifying as they are also extremely inaccurate descriptors per se. This is because they contain a considerable amount of noisy information likely to mislead expressions' classification. But even presumably important information like facial features do not necessarily mean distinction between classes, actually they can also cause classification errors particularly on emotion recognition since certain type of features are common to all facial expressions or just not characteristic of each of them. On top of that, the amount of redundant information present in descriptors is equally important. To face this and other issues it is very important to reduce data dimensionality.

Machine learning and data mining often deal with real world information and real world information often has high dimensionality. That is why dimensionality reduction is a serious concern in science and has been subject of relevant study over the years. Let us take a look at our case. We are dealing with raw images of size $h \times l$ meaning that every image can be considered a point in an Euclidean space of $h \times l$ dimensions. The number of dimensions can grow to hundreds of thousands after processing the images with techniques such as the ones mentioned in the previous Chapter. The descriptors vectors are then so big that it is impractical to use them for any kind of operation as it would take long and imply an elevated computational cost.

The challenge here is reducing the number of dimensions and still keep the intrinsic relationship between data samples. The premise of dimensionality reduction, demonstrated in Eq.(3.1), is to map information from a D-dimensional to a much smaller d-dimensional space.

$$x \in \Re^D \rightarrow y = \Psi(x) \in \Re^d \quad , \quad D \gg d \tag{3.1}$$

Throughout this Chapter we will consider each dimension to be a feature of the data, in this case images of faces. Reducing data dimensionality really can be seen as second stage feature extraction, after the pixel processing techniques. The main goal of performing mapping Ψ on the data is to eliminate all the possible redundant or correlated dimensions or features. Take in mind that dimensionality reduction is performed automatically on every descriptor, and not only we want to reduce their individual size but we still want to faithfully preserve the differences between different class descriptors and similarities within same class descriptors.

Dimensionality reduction (DR) techniques have been developed over the years and can be roughly divided into four categories: linear and non-linear, supervised and non-supervised. Linear techniques reduce data dimensionality through the product $y = Mx$ where $M_{d \times D}$ is the mapping matrix. While non-linear generally involve positive definite kernel operations, meaning that each descriptor has a different mapping matrix. Supervised DR techniques use before hand information, generally the class of each descriptor, in order to better cluster reduced data. As for non-supervised DR they don't. Each DR has its pros and cons and their performance varies with the type of data.

Throughout the next sections we will explain the techniques used in this work. They will give us an insight on how DR really works. The four categories mentioned before will be addressed. The first two sections are about basic classic techniques while the last two are more recent developed DR algorithms.

3.1 Principal Component Analysis

Principal component analysis (PCA) is by far the most popular non-supervised DR technique. It is very easy to compute and yet very efficient. It has been very useful not only for DR but also for signal processing, meteorological science, spectral decomposition, etc. It performs a linear mapping from D to d dimensions by orthogonally projecting points in the D-dimensional space to the d principal components of the data [33]. The principal components are vectors of the new linear subspace

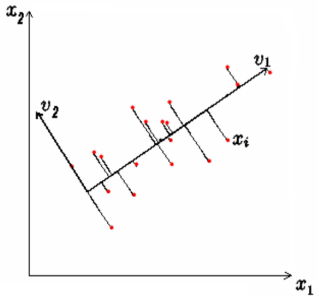


Figure 3.1: First and second principal components of a set of two dimensional data.

that better preserve the variability of the original data. See Figure 3.1 and imagine we wanted to reduce the dimension to 1-D, the points in red would be projected onto v_1 that has the direction of maximum variance of X. Performing DR with PCA will ensure that the reduced d dimensions correspond to the d principal components were the data is the most spread out, causing clusters to be more noticeable. A lower dimension d carries higher amount of features lost but also a decrease of possibly correlated ones and a bigger signal to noise ratio.

To explain how PCA works, picture a set of D-dimensional zero-mean data with n samples, $X \in \Re^{D\times n}$, that we wish to reduce to a d-dimensional subset, $\Re^{d\times n}$. Imagine as well vector v to represent a generic principal component, which can be calculated as a linear combination of X through the following:

$$v = \sum_{j=1}^{n} m_j x_j \qquad j = 1, \ldots, D \tag{3.2}$$

where $m_1 \ldots m_D$ are coefficients acting as mapping weights for the generic principal component and $x_1 \ldots x_D$ are the vectors of the original data. Lets consider now $S \in \Re^{D\times D}$ to be the covariance matrix of X and suppose we want to calculate the very principal component v_1, i.e. the one which assumes maximum variance:

$$var(v_1) = \max_{m^T m=1} var(m^T X) = \max_{m^T m=1} m^T S m$$

The condition $m^T m = 1$ is set for the weights not to be arbitrarily large. To solve this optimisation problem let us use the Lagrange multiplier α_1:

$$L(m, \alpha_1) = m^T S m - \alpha_1 (m^T m - 1)$$

Differentiating with respect to m gives:

$$Sm = \alpha_1 m$$

where S is a square matrix, m an eigenvector and α_1 its largest eigenvalue. The same analogy can be made for the remaining principal components through:

$$cov(X)M = \alpha M \qquad M^T M = I \qquad M \in \Re^{D\times D} \tag{3.3}$$

where M are the eigenvectors. To obtain the desired DR from the D to the d-dimensional space we must obtain the d largest eigenvalues in α and the corresponding d mapping vectors. We will name the final mapping weights M_* since they are a collection of d vectors from M. We then perform the following product:

$$Y = M_*^T X \qquad Y \in \Re^{d\times n} \quad M_* \in \Re^{D\times d} \quad X \in \Re^{D\times n} \tag{3.4}$$

where Y is the reduced data corresponding to the d principal components.

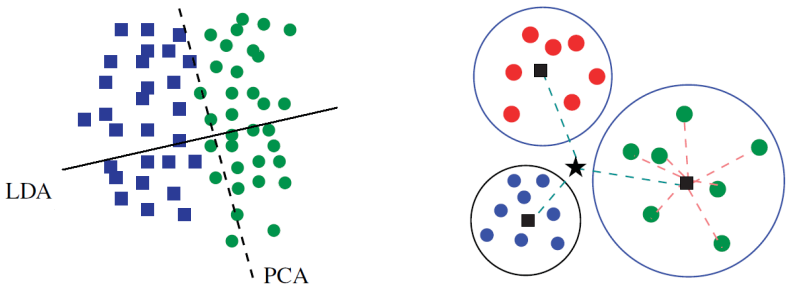
As we saw PCA is an orthogonal transformation that aims to establish new dimensions according to the data's variance. All there is to do is calculating the covariance of the centred data and use its largest eigenvalues' corresponding eigenvectors to map the data onto the new reduced dimension. Although being a good method to reduce data from very high dimensions, when the desired if the desired reduced space gets too small, PCA alone might not be able to handle the job.

3.2 Linear Discriminant Analysis

As we saw, PCA makes no distinction between samples when performing DR. This has its pros and cons. On one hand it is very fast, straightforward and efficient for very high dimensions. But on the chance of existing samples from different classes, we might be better off incorporating this information into the DR algorithm, see Figure 3.2(a). Linear Discriminant Analysis (LDA) does just that, which makes it a supervised DR technique. As the name suggests, LDA is also linear, so just like PCA it outputs a mapping matrix. Being discriminant enables LDA to weight more features that better approximate samples from identical class and features which most cluster samples from different classes. This allows the mapping matrix to form better clusters in the reduced data.

Just like in PCA or another DR technique, as we will see ahead, reduction of dimensionality generally assumes the form of an optimisation problem turned into an eigenproblem. In the case of LDA, one wants to maximise the ratio of the data's scatter between different classes over the scatter within classes themselves [34]. The solution to the eigenproblem obtained when performing this maximisation will define the axes of the reduced dimensionality.

To show how LDA works, lets again consider a set of centred data X having c



(a) (b)

Figure 3.2: (a) Example of PCA and LDA reduced 1-D axis when the scatter across classes is bigger than the one across the whole set of data. (b) The black squares are the classes' mean and the star the overall mean.

classes, n_k samples per class and a total of n samples. First we have to define a measure for the between-class and the within-class scatter. We will name these S_b and S_w accordingly:

$$S_b = \sum_{k=1}^{c} (\mu_k - \mu)(\mu_k - \mu)^T \tag{3.5}$$

$$S_w = \sum_{k=1}^{c} \frac{n_k}{n} \sum_{x_i \in X_k} (x_i - \mu)(x_i - \mu)^T \tag{3.6}$$

where vector μ is the overall data average and μ_k the average of class k (Figure 3.2 (b)). X_k is the subset of data vectors x_i belonging to class k. In Eq.(3.5) and Eq.(3.6) we are basically using the covariance of the classes' means and the covariance of the individual classes correspondingly. As we want to maximise the ratio of the first over the latter, the mapping matrix would be the one that solved the following optimisation problem:

$$M = \max_{M^T M = I} \frac{M^T S_b M}{M^T S_w M} \tag{3.7}$$

which translates into the eigenproblem:

$$S_w^{-1} S_b M = \alpha M \tag{3.8}$$

The major weakness of LDA is the number of reduced dimensions being limited to the number of classes. Due to the rank of S_b, α has $c - 1$ meaningful non-zero eigenvalues, which forces LDA to reduce data to $c - 1$ dimensions or less, due to the number of eigenvectors on M_*. However in our case that will mean no major objection as we will be dealing with up to eight expressions, seven features suffice. One other handicap of LDA is its inaccuracy when reducing dimensions from a number greater than the number of samples. This makes LDA and other methods very reliable on a first stage reduction with PCA.

3.3 Discriminant Locality Preserving Projections

It seems that reducing from potentially hundreds of thousands of dimensions to $c - 1$, c being the number of classes, would be too much loss of information. However to set c expressions apart $c - 1$ are in fact more than enough, if those are the features that are the most capable of distinguishing expressions. Both PCA and LDA can do this, but they have one flaw regarding the intrinsic relation between high dimensional expressions.

High dimensional datasets, regardless the type of data, are likely to create man-

ifolds. A manifold is a particular relationship between high dimensional points. It really can be seen as an high dimensional surface where points lay. In case of a flat or no manifold, PCA and LDA will not have their performances affected. But as the manifold gets more curved, these techniques are less able to preserve that same curve on the reduced dimension. To understand why, take a look at Figure 3.3 and imagine the several colours to be different classes. PCA and LDA would naturally focus on retaining pairwise Euclidean distances, hence interpreting the Swiss roll to be a regular cloud of points and ignoring large manifold distances over Euclidean distances [35], causing classes' mixture. To solve this problem several DR techniques have been developed. These techniques attempt to preserve curvilinear distances measured over the manifold. Discriminant Locality Preserving Projections (DLPP) [21] does just that.

DLPP is a supervised version of another DR technique named Locality Preserving Projections, a graph based projection DR [34]. Both construct a weighted adjacency graph aimed to preserve affinities between points. They do so by putting penalties on mapping neighbouring points far from each other. This assures that points that are very close in the original dimension, keep an identical relation once they have their number of dimensions reduced. One of the advantages of these techniques is being linear, so the mapping itself will not be different from PCA or LDA. Lets now see how DLPP works.

Much like any other supervised technique, DLPP also tries to maximise the ratio of the between class scatter over the within class scatter. But it does it in a slightly different way. Take note that the following definitions assume that data is organised by classes, according to $X = [X^1, ..., X^c]$ where $X^k = [x_1^k, ..., x_n^k]$ is the collection of vectors of class k. Although it is not mandatory, it facilitates comprehension. Lets define S_b and S_w, taking into account, the for now unknown linear mapping $y = M^T x$:

$$S_b = \sum_{i,j=1}^{c} (m_i - m_j)^2 B_{ij} \tag{3.9}$$

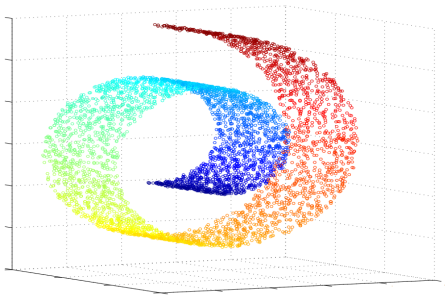


Figure 3.3: Swiss Roll Dataset.

$$S_w = \sum_{i,j=1}^{n} (y_i - y_j)^2 W_{ij} \tag{3.10}$$

where c is the number of classes, y_i a mapped vector, and m_k the average mapped vector of class k:

$$m_k = \frac{1}{n_k} \sum_{i=1}^{n_k} y_i^k$$

where n_k is the number of samples of class k. B_{ij} and W_{ij} are the weighted adjacency matrices. The scatters represented in Eq.(3.9) and Eq.(3.10) have two main differences from those of LDA. First they are measured on the reduced dimension and secondly they consist on the sum of differences between points or classes' average themselves.

The next thing to do is to find suitable weighting matrices B and W. The goal is just to penalise large distances in the original dataset, by making them less trustworthy on the reduced dimension, hence having a smaller weighting coefficient. Using a Gaussian kernel:

$$B_{ij} = \exp\left(\frac{\|f_i - f_j\|^2}{t}\right) \qquad i, j \in [1, ..., c] \tag{3.11}$$

$$W = \begin{bmatrix} W^1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & W^c \end{bmatrix} \quad W_{ij}^k = \exp\left(\frac{\|x_i^k - x_j^k\|^2}{t}\right) \qquad i, j \in [1, ..., n_k] \tag{3.12}$$

where f_i, like m_i, is class i's mean vector, but this time on the original dimension. As for t, it is a variable parameter, that should be adjusted experimentally.

By now we can see that, like in LDA, S_b works with classes' average, however S_w only considers samples within the same class. That is because distances between samples from different classes do not matter for the sake of tightening clusters together on the reduced dimension, thence their respective weights being zero. To solve for S_b divided by S_w, the following is demonstrated in [21]:

$$\frac{S_b}{S_w} = \frac{M^T F H F M}{M^T X L X M} \tag{3.13}$$

where $F = [f^1, ..., f^c]$, $H = E - B$ and $L = D - W$, according to:

$$E_{ii} = \sum_j B_{ij} \qquad D_{ii} = \sum_j W_{ij} \tag{3.14}$$

Lastly, we seek to maximise S_b over S_w once again, which gives:

$$M = \max_{M^T M=I} \frac{M^T FHFM}{M^T XLXM} \tag{3.15}$$

This can be solved with an eigenproblem similar to that of LDA. DLPP as well, only allows a reduction to a number of dimensions inferior to c, which is the rank of B. Take note that as we are performing a maximisation, the largest eigenvalues are taken, but one can invert the fraction and minimise it. In that case the lowest eigenvalues would be the same as the previous.

So we saw that isolating samples into c classes' sets, together with penalising cross manifold distances, allows for a supervised, manifold preserving reduction technique. The fact of originating a linear mapping allied with the single tuning parameter t, makes DLPP a very simple, robust and easy to use DR.

3.4 Kernel Neighbourhood Discriminant Analysis

Up until now we have been looking at linear DR techniques. Although they are simple to use, they are not capable of dealing with situations were data is subject to complex non-linear variations. An example of non-linear variations, in our case, are for instance pose and illumination changes. But also the changes within identical expressions, which in emotion recognition can be a great deal, since is not common that different people smile or do any kind of expression exactly alike. Another important non-linear variations are different genders, ages and races. Since we are bundling it all together, non-linear relationships within similar classes compromise good discriminant performance on linear DR.

To deal with this, the data is mapped from a D dimension to an even higher dimensionality, called the feature space. In a feature space, all non-linear components of the set of data are attenuated, only then to be reduced to the d dimensions desired. However finding an appropriate mapping as working with such high dimensions can be infeasible, so kernels are used instead. This is called the kernel trick and is defined as:

$$K_{ij} = k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle \tag{3.16}$$

where k is the chosen kernel, and $\Phi(X) = [\Phi(x_1), ..., \Phi(x_n)]$ is the feature space of the data. What it does is allow working with the high dimensional data using inner products, taken to be equivalent to a kernel of the original data. Kernel DR methods have been used extensively due to their great potential. In this section follows a brief explanation on how non-linear mapping works.

Kernel Neighbourhood Discriminant Analysis (KNDA), very recently proposed

by Zhao [22], is basically a kernel extension of DLPP. It tries to take advantage of the kernel trick to overcome non-linear relationships of the data, while at the same time maximising S_b over S_w. The scatter measure will be similar to that of DLPP:

$$S_b = \sum_{i,j=1}^{n} (V^T\Phi(x_i) - V^T\Phi(x_j))^2 B_{ij} \tag{3.17}$$

$$S_w = \sum_{i,j=1}^{n} (V^T\Phi(x_i) - V^T\Phi(x_j))^2 W_{ij} \tag{3.18}$$

The difference to DLPP is that here V is mapping data straight from the feature space and S_b is not working with classes' average, for that reason the only distinction to S_w are the affinity weights themselves. They are calculated according to:

$$B_{ij} = \begin{cases} 1 & \text{if } \|\Phi(x_i) - \Phi(x_j)\| \leq \varepsilon \text{ and } \text{class}(\Phi(x_i)) \neq \text{class}(\Phi(x_j)) \\ 0 & \text{otherwise} \end{cases} \tag{3.19}$$

$$W_{ij} = \begin{cases} 1 & \text{if } \text{class}(\Phi(x_i)) = \text{class}(\Phi(x_j)) \\ 0 & \text{otherwise} \end{cases} \tag{3.20}$$

Notice that both B and W are binary. Since we are dealing with points located on the feature space, we can take for granted a straightforward distribution scatter of data and there is no need for a more complex weighting procedure. For the between class affinity graph we will want to consider pairs of points that are located close from each other, but at the same time are from distinct classes. These are points that we wish to be the furthest possible. So B_{ij} will be positive if points i and j are closer than a threshold ε, that should be chosen to be to be just big enough to include a minimum number of points inside the $\Phi(x_i)$ centred sphere. The within class affinity graph is similar to that of DLPP, simply with binary entries.

Since $\|\Phi(x_i) - \Phi(x_j)\|$ in Eq.(3.19) is measured in the feature space, there is need to express it in function of the chosen kernel k, see Eq.(3.17). The norm is then computed according to:

$$\begin{aligned} &\|\Phi(x_i) - \Phi(x_j)\| = \\ &= \sqrt{(\Phi(x_i) - \Phi(x_j))^T(\Phi(x_i) - \Phi(x_j))} \\ &= \sqrt{\langle\Phi(x_i), \Phi(x_i)\rangle - 2\langle\Phi(x_i), \Phi(x_j)\rangle + \langle\Phi(x_j), \Phi(x_j)\rangle} \\ &= \sqrt{K_{ii} - 2K_{ij} + K_{jj}} \end{aligned} \tag{3.21}$$

After constructing both affinity graphs we proceed to maximise S_b over S_w. In [22] a demonstration can be found on Eq.(3.17) and Eq.(3.18) which shows that:

$$\max \frac{S_b}{S_w} = \max \frac{M^T K H K M}{M^T K L K M} \qquad H = E - B \qquad L = D - W \tag{3.22}$$

where E and D can be found in Eq.(3.14) and M yields the mapping matrix from the original D dimensions. The selection of the mapping eigenvectors $M_* \in \Re^{D \times d}$ is done according to the largest eigenvalues, exactly like the two previous methods.

As this time since we are working with kernels, the mapping is non-linear. So X is mapped into the reduced dimension according to $Y = M_*^T K$. This requires test samples to be in the feature space before they can be reduced:

$$y = M_*^T \Phi(X^{train})^T \Phi(X^{test}) \tag{3.23}$$

Which is equivalent to calculate the kernel between each of the train and test sample vectors according to $= k(x_i^{train}, x_j^{test}) = \Phi(x_i^{train})^T \Phi(t_j^{test})$.

Kernel methods and specifically KNDA require caution when choosing the parameters. There are plenty of kernels to use, and for each of them several tests with different tunings should be made. Due to threshold ε in the between class affinity matrix, different kernel set ups cause M to change dramatically if ε is maintained. But once one feels comfortable using kernels, the best can be taken out of them.

To sum up in this Chapter we covered the fundamental ideas behind all types of DR. We saw that by not being a discriminant technique PCA is best for reducing data from very high dimensions to a still considerable number of features. But from then on, supervised DR methods are the best for leveraging from labels, which allow them to do an efficient reduction to a very low number of dimensions. We presented a basic supervised technique broadly used in DR, but also a manifold preserving and a non-linear method that deal with more complex data. We also leaned the power of eigenproblems for reducing dimensions and explained what is and how to deal with a feature space.

Chapter 4

Classification

In this last theoretical chapter we will discuss several methods of classifying data. The classification stage evaluates the performance of the various algorithms mentioned before. But once data is clustered and ready for classification, different techniques yield different results. Classification is not solely a way to classify test samples and output accuracy, there is yet a margin to increase it.

The most popular classification technique, the nearest neighbour algorithm, is as plain as it gets. It simply classifies tests samples according to the class of the k nearest neighbours (k-NN) in the Euclidean Space. Evidently there are a number of variants to this algorithm, which are not part of our study, designed to boost classification performance.

Classifying expressions might pose challenges to a simple k-NN algorithm. Ultimately, depending on the type of data certain classification techniques might fail to stand out. Expressions for example, can be naturally deceiving even to humans, and that reflects onto the euclidean space, where it is likely to exist an overlapping of different expressions. More complex techniques can, on the other hand, deal with the problem just fine by not making use of the geometrical distances between samples, since it is normal for test samples to be slightly dislocated from their respective class cloud, specially when using supervised DR techniques. We will attest to that later on.

In this Chapter our focus will be mainly on sparse coding classification techniques. Sparse coding has recently generated interest in pattern recognition and computer vision. It can be used for both representation and classification, it minimises computational overhead and deals well with occlusions in the case of image classification. Advantages such as these make sparse representation (SR) very robust and attractive for the scientific community. In a SR system a test image is represented by a vector of coefficients $x \in \Re^n$, called the sparse code, that most faithfully reconstruct the image itself when multiplied by an a priori learnt overcomplete dictionary $D \in \Re^{d \times n}$, where d is the dimension of its vectors and n the

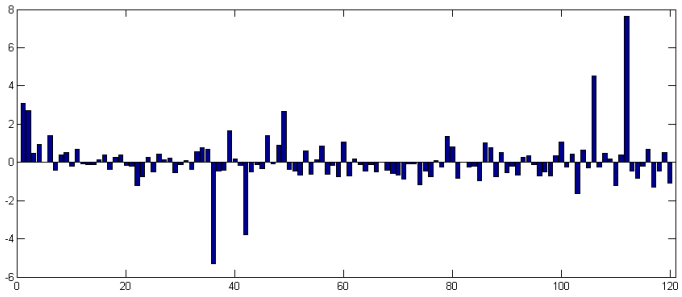


Figure 4.1: Example of a sparse code x, each atom weights a dictionary element.

number of elements. There are several ways to construct a dictionary, but fundamentally it contains a combination of training samples whom test samples can relate to. For a dictionary to be over-complete it has to have $n > d$. So an SR problem basically translates into:

$$\|y - Dx\|_2 < \varepsilon \qquad \min \|x\|_0 \tag{4.1}$$

where y is a test vector, ε the maximum admitted reconstruction error and the l_0 norm $\|x\|_0$ the sparsity constrain, i.e. the number of non-zero elements in the set of coefficients. We are then left with x, see Figure 4.1, that not only can be used to reconstruct the input signal but also to assign it to a certain class if that is the case. We can either assign the input to the class of the dictionary element having the biggest sparse coefficient as to the set having the biggest sum of coefficients.

In this chapter we will take a look at two SR algorithms but first we will give a brief insight on another very popular classification technique.

4.1 Support Vector Machine

This first method [36] has a totally different paradigm from k-NN or SR, it is however a very powerful, efficient and robust classification tool. Because of that, it is without a doubt one of the most used classification methods in machine learning, becoming a standard for evaluating other methods.

The idea behind Support Vector Machine (SVM) is simple to understand. In order to give a brief insight on how it works picture a set of data containing samples from two distinct classes. The goal of SVM is to construct an hyperplane which divides the two classes. In fact this principle is not very different to LDA in the sense that it tries to find a one dimensional projection that best separates distinct classes. Needless to say that SVM requires training data along with its labels.

Assuming that data's classes are linearly separable, SVM would trace an hyperplane passing between the two classes. An important notion in SVM are margins

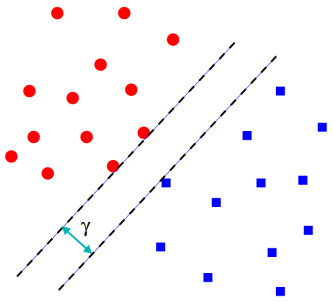


Figure 4.2: Principle of SVM.

[34]. Not only the hyperplane would divide both classes, as it would do it ensuring that the same plane would be the furthest away possible from the closest different class training samples, as in Figure 4.2.

For the two class linearly separable set of data, as in the figure above, lets take the hyperplane to be $\omega^T x + b = 0$, in which case the function $f(x)$ would classify depending on its sign. Lets consider the assigned value $y = -1$ for one class and $y = 1$ for the other. That being said one wants to find the limiting hyperplanes $\omega^T x + b \geq 1$ to include one whole class and $\omega^T x + b \leq 1$ to include the other, or just $y(\omega^T x + b) \geq 1$. The concept of margin mentioned above comes form maximising the distance between these two limiting hyperplanes. Geometrically one can understand that the distance between the two hyperplanes is $\gamma = 2/\|\omega\|$. So what SVM really does is the following:

$$\min_{\omega,b} \|\omega\| \qquad y_i(\omega^T x_i + b) \geq 1 \quad \forall x_i \tag{4.2}$$

Truth is, that most often classes are not linearly separable. In that case SVM is able to use the kernel trick, just like KNDA. Using this methodology allows SVM to linearly separate really complex data in the feature space. So a new input would be mapped to the feature space and depending on which side of the hyperplane it would fall onto, the class would be assigned.

Another variant is the multiclass SVM. It is common for problems in machine learning to have more than just a pair of classes. Juts in emotion recognition we can have up to eight. Multiclass SVM has two ways to deal with it. The winner takes all principle or a one versus one strategy. In the first, the test point is assigned to the class whose cloud calibrated classification function scores the highest, meaning that the point is the deepest into that class. As for the one versus one, the sample is tested between all possible pairs of classes, collecting votes along the way. The class that had the most votes is the assigned to sample.

In the end, what makes SVM so remarkable, is its capacity of adapting to all types of classification demands. This is because SVM is able to deal with very

complex non-linear data, having multiple classes, and managing to do it efficiently with a scarce number of training samples.

4.2 Orthogonal Matching Pursuit

Orthogonal matching pursuit (OMP) is a step wise SR algorithm that attempts to find in a dictionary $D \in \Re^{d \times n}$, a linear combination of vectors or elements, that have maximal projection onto an input vector $y \in \Re^d$. Where n is the number of dictionary elements and d their dimension. It does so in a greedy fashion which means that in each iteration the algorithm searches for the dictionary elements that have the biggest inner product with respect to the residual vector. Take note that the dictionary elements have to be normalised. OMP is fairly easy to understand and yet a very efficient. Lets see how it works.

In the very first iteration of the OMP, $k = 1$, let the residual vector $r_0 = y$, consider as well the sparse code zeroed vector $x \in \Re^n$. For the first step of the algorithm the goal is to find and store the index of the dictionary atom that satisfies:

$$i_k = \arg\max_i \langle r_{k-1}, d_i \rangle \tag{4.3}$$

where $d_i \in \Re^d$ is the i^{th} element of D. Then we want to retrieve the sparse code x that better portraits the input vector, using an l_2 norm:

$$x = \arg\min_x \|y - \sum_{j=1}^{k} d_{i_j} x_{i_j}\|_2^2 \tag{4.4}$$

x_{i_j} is a sparse code coefficient, member of $x = [x_1, ..., x_d]$, that weights d_{i_j}, while the remaining ones are zero. After calculating x we are ready to update the residual vector:

$$r_k = y - \sum_{j=1}^{k} d_{i_j} x_{i_j} \tag{4.5}$$

Then we test for the stopping condition, if that is not the case the iteration is incremented and the process repeated from Eq.(4.3). Take note that the number of non-zero elements of x is always equal or less than the number of iterations k.

Regardless of how effective OMP is, it still requires a decent dictionary and stopping condition. For both there are a number of options with similar principles. A dictionary should be relatively compact and comprehensive while the stopping condition must allow the algorithm to converge successfully and still keep x as sparse as possible.

4.2.1 Stopping Conditions and Variants of OMP

OMP itself is a derivative of an algorithm called Matching Pursuit. The difference is that the latter skips the l_2 norm optimisation stage right to subtracting to the residual signal the inner product resulting from the first step. Only to stop when the residual becomes lower than a predefined threshold.

The stopping condition of OMP is generally based on the l_0 norm of the sparse code x, according to $\|x\|_0 < T$, where T is an upper bound on the number of non-zero elements. But one could also want the number of iterations not to exceed a certain amount, or the residual value to be smaller than a specific threshold just like in Matching Pursuit. It is also an option to incorporate each of those together.

If there is any downsize to OMP it has to be the fact of not being a convex optimisation, which increases the odds for the algorithm not to converge in some practical problems. That is caused precisely by the l_0 norm which can be difficult to solve. For this reason two other variations of the OMP arose, which substitute the sparsity constrain factor by an l_1 norm.

The Basis Pursuit compromises to satisfy $\min \|x\|_1$, while the Lasso algorithm also known as Basis Pursuit De-noising reformulates the stopping condition to $\min \|x\|_1 < \lambda$. Substituting the l_0 by an l_1 norm, attributes less importance to the sparsity of the code and more to the value of its entries. Overall we might end up with solely non-zero coefficients, a large amount having residual values. In the end, these last two algorithms, do not necessarily lead to a better performance, they take more time and are more complex [37].

4.2.2 Choosing a Dictionary

For a pursuit algorithm to find a sparse code that most faithfully represent an input vector, a dictionary has to be carefully chosen. Every SR pursuit method takes the dictionary for granted, assuming that it will converge no matter what. No need to mention that having different classes' inputs aggravates the problem, as it is expected from a dictionary the ability to handle any possible input regardless the class. As a good dictionary choice can be decisive on the end performance of any sparse algorithm, one may opt to learn the most efficient dictionary possible for a certain k sample input Y. Here we are going to present a method [38] that is broadly used to do it.

First we have to select an initial dictionary for improvement. Throughout out this next algorithm the number of dictionary atoms will not change, only their content. Therefore it shouldn't be too small, to avoid lack of representativeness, neither too big, due to high computational cost. Generally it is okay to randomly select some training samples, provided that different classes should always be equally present in

the initial dictionary, if that is the case. Lets consider an initial dictionary $D \in \Re^{d\times n}$ for what we want to represent $Y \in \Re^{d\times k}$, where d is the size of the signals, n the number of dictionary atoms, and k the number of inputs. For starters we have to compute the sparse code $X \in \Re^{n\times k}$ with a pursuit algorithm like OMP.

$$X = \begin{bmatrix} x_{1,1} & \cdots & x_{1,k} \\ x_{2,1} & \cdots & x_{2,k} \\ \vdots & \ddots & \vdots \\ x_{n,1} & \cdots & x_{n,k} \end{bmatrix} \tag{4.6}$$

The columns of X are the sparse codes for each of the input signals. As for the rows they represent the contribution of the dictionary elements on each of the inputs. The next thing to do is to fix X and individually update each of the atoms $d_i \in D$. We do so by first calculating:

$$E^i = Y - \sum_{\substack{j=1 \\ j\neq i}}^{n} d_j x^j \tag{4.7}$$

which is the difference between the input vectors and the reconstruction contribution of every dictionary element but d_i. In Eq.(4.7) x^j is the j^{th} row of X. The next step is to remove from E^i the columns whose d_i had no initial contribution at all through the corresponding sparse code. In other words we eliminate from E^i every column e_h if $x_{i,h} = 0$, $1 \leq h \leq k$. The last thing to do is to perform a singular value decomposition (SVD) on $E^i = U\Delta V^T$ and update $d_k = u_1$, which is the first vector of the unitary matrix U. Finally we are able to proceed to the next dictionary element, incrementing i, recalculating Eq.(4.7) and again perform the SVD.

Once all the dictionary is updated, we are able to recalculate the new sparse codes X generated and repeat the whole process as many times as desired. As it takes quite a while and the dictionary updates start to be irrelevant at some point, it is okay to do no more than ten iterations. The algorithm just presented is called K-SVD and can be translated into the following problem:

$$< D, X >= \arg\min_{D,X} \|Y - DX\|_2^2 \qquad \forall i, \|x_i\|_0 < T \tag{4.8}$$

where $\|x_i\|_0 < T$ is the stopping condition of the pursuit algorithm used. Here we are assuming that the stopping condition is the sparsity constrain, and that x_i a column of X in Eq.(4.6), cannot surpass T number of non-zero elements.

In the next section we will see how to improve it for the purpose of boosting classification.

4.3 Label Consistent K-SVD

The performance of sparse codes as classifiers is highly dependent on the dictionary's quality, that has to be robust even when dealing with inputs from different classes. As we saw in the previous section, the K-SVD algorithm for dictionary learning makes no distinction between dictionary elements. This can be compared to the paradigm of dimensionality reduction, since we have information on the sample classes we might just use it to improve sparse codes discrimination between classes.

Jiang et al.[39] proposed a simple method for learning discriminative dictionaries. Like any other dictionary learning method, it uses the training set to find the dictionary that fits best that same type of data. The learning stage is similar to the one we just presented, the difference is that dictionary D and input Y contain more than just the samples themselves. The idea is to make use of their labels to comprehend better the relation between and within classes. In expressions that can be of great advantage since often there is a pattern of confusion between certain expressions.

In fact in [39] two methods are proposed, they are however very similar, as we will see. To explain how they work we are going to use the same notation as in Eq.(4.8). Lets assume the stopping condition to be $\|x_i\|_0 < T$ in both:

$$< D, A, X >= \underset{D,A,X}{\arg\min} \|Y - DX\|_2^2 + \alpha\|Q - AX\|_2^2 \tag{4.9}$$

$$< D, A, W, X >= \underset{D,A,W,X}{\arg\min} \|Y - DX\|_2^2 + \alpha\|Q - AX\|_2^2 + \beta\|H - WX\|_2^2 \tag{4.10}$$

We will name the first method Label Consistent K-SVD 1, shortened LCK-SVD1, and the second LCK-SVD2. To understand why they are label consistent, one needs to take a look at the terms used in each of them. The first term $\|Y - DX\|_2^2$ is the reconstruction error and is no different from what we have already seen, it uses no label's information whatsoever.

The second term, just like the first, is common to both algorithms, and it is called the discriminative sparse code error. $Q \in \Re^{n\times k}$, n being the number of dictionary elements and k the number of input signals, is a binary matrix whose entry q_{ij} is positive, if dictionary element d_i and input signal y_j share the same label. Take note that we are considering dictionary $D \in \Re^{d\times n}$ and input matrix $Y \in \Re^{d\times k}$, just like in the previous section, d being the smaples dimension.

The third term on LCK-SVD2 is what distinguishes it from LCK-SVD1. In $\|H - WX\|_2^2$, $H \in \Re^{c\times k}$, c being the number of classes, is also a binary matrix.

There is one column per input signal, and in each column the single positive entry dictates the class of the respective input signal. It is hereby called the classification error.

As for α and β they are coefficients which weight the discriminative and classification error's contribution. $A \in \Re^{n \times n}$ and $W \in \Re^{c \times n}$ are linear transformation matrices. A forces the sparse codes to be as discriminant as possible according to Q. While W is simply a predictive classifier. Both matrices in addiction to the dictionary must be initialised before being used in Eq.(4.9) and Eq.(4.10).

In [39] at first, one dictionary is learnt for each class, using the regular K-SVD algorithm presented in Section 4.2.2. Depending on the desired total number of dictionary elements, both LCK-SVDs randomly pick a number of training samples from each class to form the very initial class dictionary. The c dictionaries are independently learnt according to a predefined number of iterations, and then concatenated to form the overall initial dictionary D. Once D is obtained, a pursuit algorithm is used to compute a sparse code X that best suits the training input signals. With X we are now able to initialise the parameters A and W through the minimisations:

$$A = \arg\min_{A} \|Q - AX\|^2 + \lambda_1 \|A\|_2^2 \quad , \quad W = \arg\min_{W} \|H - WX\|^2 + \lambda_2 \|W\|_2^2 \tag{4.11}$$

which yield the following solutions using a Tikhonov regularisation [40]:

$$A = (XX^T + \lambda_1 I)^{-1} XQ^T \quad , \quad W = (XX^T + \lambda_2 I)^{-1} XH^T \tag{4.12}$$

Once we have the initial D, A and W we can finally proceed to learn the optimal dictionary according to Eq.(4.9) and Eq.(4.10). As they are similar one can explore LCK-SVD2 alone for simplicity, but all calculations can be used interchangeably. The sparse code X will leverage the supervised information through the following minimisation:

$$< D, A, W, X >= \arg\min_{D,A,W,X} \left\| \begin{pmatrix} Y \\ \sqrt{\alpha}Q \\ \sqrt{\beta}H \end{pmatrix} - \begin{pmatrix} D \\ \sqrt{\alpha}A \\ \sqrt{\beta}W \end{pmatrix} X \right\|_2^2 \quad \forall i, \|x_i\|_0 < T \tag{4.13}$$

We saw before that the goal of a sparse code is to minimise the difference between the input signals and dictionary elements. Since original input signals were concatenated along with the discriminative matrix Q and class matrix H, what the solution above presents is a simultaneous optimisation for all parameters, since X alone must meet all demands. An effort will then be made for the dictionary not

only to have its elements even the original signals, but also to incentive its atoms to match input classes while at the same time alerting them on mistakes.

The optimal dictionary can be learnt using the regular K-SVD previously presented, by simplifying Eq.(4.13) into:

$$< \hat{D}, X >= \arg\min_{\hat{D},X} \|\hat{Y} - \hat{D}X\|_2^2 \qquad \forall i, \|x_i\|_0 < T \tag{4.14}$$

where $\hat{Y}$ and $\hat{D}$ are the new input signals and dictionary. This way no distinction is made between matrices D, A, and W, and the sparse code will converge to better understand the intrinsic relationship of the data.

In [39] an l_2 column wise normalisation is made on $\hat{D}$ before using the K-SVD, so once we are done with the learning, we cannot just decompose $\hat{D}$ back into D, A and W, and the following computations have to be made:

$$D' = \left[\frac{d_1}{\|d_1\|_2}, ..., \frac{d_n}{\|d_n\|_2}\right] \qquad W' = \left[\frac{w_1}{\|d_1\|_2}, ..., \frac{w_n}{\|d_n\|_2}\right] \tag{4.15}$$

We are lastly ready to classify test samples. No A was calculated since just D' and W' are necessary at this point. D' is the optimal dictionary learnt and will be used to compute the sparse codes, according to:

$$x_i = \arg\min_{x_i} \|y_i - D'x_i\|_2^2 \qquad \forall i, \|x_i\|_0 < T \tag{4.16}$$

Finally W' will weight the codes and assign them to classes, then to predict which class does y_i in fact belong to, according to the highest scorer:

$$class = \arg\max_j (W'x_i) \tag{4.17}$$

This concludes our take on LCK-SVD2. Take note that the same applies to LCK-SVD1, the only difference is that no classification error term is used, so in Eq.(4.17) W' is absent.

Chapter 5

Experimental Results

Throughout this last chapter we are going to do a series of tests with the goal of achieving maximal accuracy on recognising emotions. In order to avoid over-fitting we are going to put three different datasets to the test, which we will describe next. As there are many varying parameters, it is mandatory to break them down and start off by excluding the methods obtaining the least satisfying results. First we will mainly focus our attention to the clustering capabilities of each of the pixel processing techniques (PPTs) with each of the dimensionality reduction (DR). We will then put sparse representation (SR) to the test and see how it performs against other methods. In the end we will propose a final method that most likely will achieve good results across different data bases, presenting itself to be a robust algorithm.

During testing no virtual datasets were used, neither individual samples were at any time removed from the datasets. Datasets were used as available online with original labelling. The results on the following tables are percentage accuracies, varying from 0 to 100%. Each result is averaged from a number of runs. The number of runs is inversely proportional to the size of the dataset. In between runs the parameters are absolutely identical, what varies is solely the training and testing expressions, that are selected randomly each run. The purpose of that is to stabilise the results, which are as likely to vary as small the dataset is. However this will not eliminate accuracy oscillations in between tests. All the tests were performed using a 90% training ratio.

Appendix A contains important practical information that can be used to replicate the results obtained. Very briefly some details are given about the algorithms used, such as parameters and source codes. Also some insight is given on tuning the sparse methods.

5.1 Datasets

In order to compare our method to other works published in the area of expression recognition it was mandatory to use the Extended Cohn-Kanade Database [2] as it is by far the most used for that purpose. Besides CK+ another two datasets were added for algorithm testing, the Karolinska Directed Emotional Faces (KDEF) [3] and the 3D Facial Emotion (BU-3DFE) dataset [1]. All three of them feature frontal facing subjects performing a variety of basic emotions, as well as different genders, races, ages and lighting conditions, and were specifically conceived for the matter of emotion recognition or facial understanding, either for machine learning or psychological studying. Also all of them were publicly available, with no permission request necessary from the developers.

Each dataset has its own characteristics and a purpose. The CK+ dataset will test the performance of the algorithm on a small set of data, where the faces contours, nose and eyes are aligned and the emotions are staged. The BU-3DFE dataset is a big dataset that we will use to the test non aligned, non staged 3D faces which are reconstructed to 2D. The KDEF is a medium sized dataset to test partially aligned, non staged expressions.

Besides expressions, most of the datasets contain more information than just the frontal facing pictures. Such as faces' landmarks, expression evolution from neutral to apex, various poses, action unit labelling, 3D facial models, etc. We are going to focus exclusively on the apex frontal facing labelled emotions on each of the datasets. Here follows an overview on each of them.

5.1.1 CK+

The Extended Cohn-Kanade Database, is an extension of a previous Cohn-Kanade database. In this extended version an emotion was added, and the others were relabelled by experts, as to classify the expression itself rather than the expression the subject was asked to do. Despite having a total of 123 subjects in the dataset, the expressions performed by each one vary. The total number of expressions performed and accepted as legitimate is 327, between 118 subjects. Also in this extended version Lucey et al.[2] propose a benchmark testing protocol, as to compare performance of algorithms developed around the world.

This dataset contains photos of the evolution of the various emotions performed by different people. From neutral to the emotion's apex, the number of photos can vary between expressions, however we are only interested in the last one, the totally developed emotion. In this work we decided to use as well, a neutral expression per individual. So in total the CK+ dataset will have 17 Contempt (Co) expressions, 25

Fear (Fe), 29 Sad (Sa), 45 Angry (An), 59 Disgust (Di), 69 Happy (Ha), 83 Surprised (Su) and 123 Neutral (Ne). This already counts one sad expression mislabelled as contempt.

Centring Faces

As mentioned before, the raw images of CK+ are simply photos of subjects performing expressions. Meaning that the location of faces in the images, or even their size is different from photo to photo. Since identical facial information per image pixel is mandatory, it is necessary to centre all faces together. Fortunately the dataset features facial landmarks, which allow for face tracking and mapping. A random face is chosen to be the reference and all the other faces are mapped onto itself using affine mapping. Affine mapping may distort the mapped image, so care need to be taken when choosing the mapping landmarks. For example, mouth and eyebrows are very likely to change position and shape between expressions, for that reason their landmarks were not used to map the faces together, as doing so may attenuate the expression concerned.

A good way to see if all faces are in fact aligned is to take a look at their average. As we can see in Figure 5.1 (a), this average resembles a face, which indicates that all faces are roughly in same position. More information can be taken out of the average expression, like the blurred contours, that indicate a bigger variation of that particular region.

After each face is mapped the excesses and background are cropped out, and the final face images, as in Figure 1.2, are obtained. The images are then saved and ready to be subject to pixel processing techniques. Finally they are stacked all together, being ready for dimensionality reduction.

On the tests performed using this dataset, a total of 144 expressions were picked out of the 327. The first 18 expressions per class on the dataset were chosen according to the original labelling. The accuracies tabled result from an average of 100 runs. In each run a set of 16 testing descriptors, 2 samples per class, were randomly selected from the 144, the rest being used for training.

5.1.2 BU-3DFE

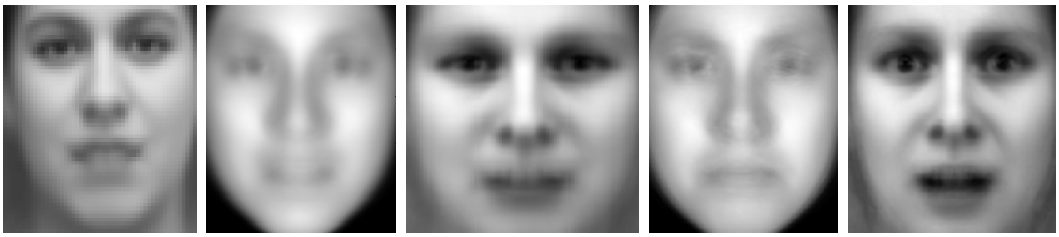
Expressions in 2D images are subject to pose variations and skin movements that may affect recognition rates. This type of problems, as well as lighting conditions can only be sorted out with 3 dimensional capturing of expressions. The BU-3DFE dataset [1] was created in 2006, for the reason of further developing the expression recognition testing into 3 dimensional images, as till then no major 3D image testing datasets were publicly available. This dataset is the largest used in this study, as it

contains 100 subjects performing 7 expressions, each one with four different levels of intensity, excluding the neutral expression. Just like CK+, it also features several races, ages and both genders, specifically 56 women and 44 men.

For each expression there is a 3D face model and respective feature points as well as a two angle facial view image and a 2D coloured image of frontal facial texture, the one we are going to make use. It is only of our interest the most intense expression, out of the four, which corresponds to the apex. Unlike for the CK+ we are doing no mapping of faces. As all faces are located in the same region of the image, we are simply going to crop them equally only to include the 2D texture face and graylevel them, to look like the sample in Figure 5.2. However, as centred the faces may look like this is not the case, as there is a slight offset between each of them, hence their average being considerably more blurred than in the CK+ dataset, see Figure 5.1.

On the tests performed on this dataset, we also used a 90% training ratio, with a total of 490 random training expressions along with 110 testing expressions. To stabilise results, the accuracies are averaged between 20 runs. Less runs are required compared to the CK+ dataset as the training and testing sets are much larger.

In Table B.2, Appendix B, the confusion matrix of the BU-3DFE displays a bad accuracy rate in particular for the sad expression, which shows that it often comes across as other expressions. In Figure 5.1 d), the average sad expression for this dataset can be seen. The sad expression is barely recognisable due to the blurred texture of the face. This happens not only because the faces were not mapped onto each other but also due to not staging expressions. Meaning that each individual interprets every emotion in his own different way. Although for humans it can be obvious to interpret any expression of sadness, for pattern recognition it is much more challenging, since every sad face does not necessarily exhibit an evidencing of the lower lip, for instance. Despite the confusion matrix in Table B.2 revealing an overall mediocre accuracy rate, it is not intended to compare these results to other datasets. As each one explores a different matter, the goal is to see whether or not our method is able to increase these accuracies.



(a) (b) (c) (d) (e)

Figure 5.1: Average apex expression of (a) CK+ (b) BU-3DFE (c) KDEF (b) Sad (BU-3DFE) (e) Fear (KDEF)

(a) (b) (c) (d) (e) (f) (g)

Figure 5.2: Sample expressions of BU-3DFE on the top and KDEF on the bottom row. (a)Neutral (b)Anger (c)Disgust (d)Fear (e)Happiness (f)Sadness (g)Surprise

5.1.3 KDEF

Both CK+ and BU-3DFE were conceived for the field of emotion recognition in computer vision. The Karolinska Directed Emotional Faces (KDEF) [3] on the other hand was developed for psychological and medical research purposes. KDEF has a total of 70 subjects, 35 men and 35 women, performing 7 expressions each throughout two separate sessions. Each expression was photographed at its apex from $-90, -45, 0, +45, +90$ degrees. All photos are coloured and had the eyes and mouth positioned in fixed image coordinates.

As in the BU-3DFE, to process this dataset to look like in Figure 5.1, all there is to do is to crop the $0°$ frontal facing expression and graylevel it. The pictures used in this work are the ones of the second session for no particular reason. As there has been a previous centring of expressions, the average KDEF looks quite more sharp than the BU-3DFE's. And despite having more accurate confusion matrix in Table B.3, the fear expression, for instance, is confused with surprise close to one fifth of the times. This is evident in Figure 5.1, and presents itself to be the typical SR coefficient mining problem that arises from expression similarity, that a robust algorithm has to deal with.

All the tests using the KDEF are an average of 40 runs. In each run a random set of 441 training faces are picked among a total of 490, regardless of identity. The rest corresponding to 10% are left for testing.

5.2 PPTs and DR Methods Accuracy

Now that we have analysed all datasets and explained how each of those are going to be tested, lets move on to the problem of recognising expressions. Obviously it is our desire to come up with a very robust method having the best possible clas-

sification accuracy, across all datasets. Throughout this work we described several techniques, largely used in face recognition or understanding. Processing an image, reducing its dimensionality and classify its descriptor, can be done in multiple ways. So all there is to do is to start off by excluding methods with less satisfactory results.

In this second section we are going to analyse PPTs and DR, then choose the best ones in order to move on. We are going to do this by simply understanding the clustering power of each possible pair of PPTs and DR methods combined. For each dataset there is a table in Appendix C with the classification accuracy achieved by a k-NN classifier. The PPTs used were, Downscalling, Phase and Magnitude Sobel Filters, LBP, pixel-wise LBP, Gabor filters, HOG and an additional row was added for the raw image with no PPT. As for DR, PCA with no further DR, LDA, DLPP, KNDA and LPP, the standard non-supervised version of DLPP, were used. PCA was used as a primary DR before other DR methods, reducing data to the corresponding 99% of eigenvalues, except for Gabor which yield better results with 90%, due to their size.

After taking a look at the Tables C.1, C.2 and C.3, several conclusions can be made. The first is that the results of various pairs of PPT and DR are relatively consistent across all three datasets, which suggests convergence on the best solution. Secondly, DR seems to have a bigger leverage than PPTs on the results, not surprising since the information is much more condensed on the later. In third place, and most importantly, both LDA and DLPP, as LBP and HOG, clearly stand out as the most accurate methods. This validates the power of histogram descriptors as well as supervised DR. Histogram descriptors allow for slight variations on heads' pose, size and location of features, while the others tend to span similar features across non identical dimensions. Although for supervised DR the advantage being obvious, KNDA failed to be any good at classifying expressions. This might be for several reasons, such as tuning and over complex assumption of manifold. In Figure 5.3 it is possible to see that KNDA clusters expressions rather well, but faces challenges when mapping out of training samples. KNDA also tends to oscillate considerably in between datasets and PPTs, which suggests dependency on parameter adjusting.

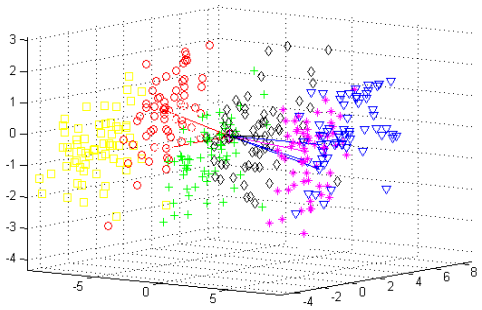


Figure 5.3: KNDA clustering and expression k-NN classification.

5.3 Classification Techniques

We have seen that LBP and HOG turn out to be the best PPTs for our problem. As for DR, LDA and DLPP clearly achieve the best results. For now, we are only going to focus our attention on this two pairs of techniques. In this section we are going test the several classification techniques. Lets take a look and analyse the following table.

	CK+				BU-3DFE				KDEF			
	LDA		DLPP		LDA		DLPP		LDA		DLPP	
	LBP	HOG	LBP	HOG	LBP	HOG	LBP	HOG	LBP	HOG	LBP	HOG
k-NN	81.7	86.8	87.6	88.1	70.3	66.2	70.1	69.3	82.3	77.9	82.2	82.5
SVM	81.3	87.6	86.9	89.2	70.3	65.4	69.5	69.7	82.5	76.6	82.6	82.4
OMP	87.0	89.2	86.9	89.1	68.8	70.0	69.5	69.2	79.5	83.2	79.7	83.0
LCK-SVD1	64.5	52.8	82.3	48.6	68.9	65.5	68.9	67.8	79.0	58.5	80.4	77.3
LCK-SVD2	84.3	74.2	86.9	77.9	68.9	68.4	69.0	67.9	78.4	80.2	80.4	77.3

Table 5.1: Accuracies obtained by the several classification techniques.

This table shows the results achieved by several classification methods, including the three SR methods, on each dataset, for both LDA and DLPP, and LBP and HOG. Each column had exactly the same set of training and testing sets, but they varied from column to column.

Here the results are less conclusive and no classification technique seems to stand out. They actually appear to oscillate in between datasets, PPTs and DR methods. The discriminative SR methods, LCK-SVD1 and LCK-SVD2, also have rather disappointing results, apparently not making much use of labelled dictionary atoms. This is mainly due to the number of features of the reduced descriptors. After reducing dimensionality with supervised DR methods such as LDA and DLPP, descriptors end up having a maximum of $c - 1$ dimensions, c being the number of classes. This complicates the task of dictionary learning, since the features in case are already very condensed. Being projected onto principal components using supervised information provides little margin for LCK-SVD to leverage more of that same information. The solution is not to use supervised DR before LCK-SVD, and use the descriptors as is, right after the PCA. Having more and unbiased features will loosen sparse coefficients providing them with more reconstructive options. Making effective use of discriminative sparse code and classification error terms.

5.3.1 Concatenating LBP and HOG descriptors

We have seen in Table 5.1 that LBP and HOG's compared performance variates between datasets. In other words, some datasets may have better classification accuracies with LBP while others with HOG. If that happens with different datasets, it sure happens expression wise. Take a look at Table 5.2, where we have two

	Co	Fe	Sa	An	Di	Ha	Su	Ne
Co	92.5	0	4.5	1.5	0	0	0	1.5
Fe	0	98.0	0	0	0	1.5	0	0.5
Sa	5.0	5.5	68.5	9.5	0	0	0.5	11.0
An	1.0	0.5	2.0	89.0	0.5	0	1.0	6.0
Di	0.5	0	0	0.5	98.0	0	0.5	0.5
Ha	0	1.0	0	0	0	99.0	0	0
Su	0	0	0	0.5	1.0	0.5	97.0	1.0
Ne	7.5	1.5	4.5	23.0	2.0	0.5	10.5	50.5

	Co	Fe	Sa	An	Di	Ha	Su	Ne
Co	84.0	1.0	1.5	0	0.5	1.5	0	11.5
Fe	0	78.5	9.0	0.5	0.5	3.0	0	8.5
Sa	0	4.0	63.5	12.5	1.0	0	0	19.0
An	0	0	8.5	67.5	0.5	0	0	23.5
Di	1.5	0	0.5	0.5	82.0	4.0	0	11.5
Ha	0	0.5	0	0.5	5.5	93.5	0	0
Su	0.5	0	0	0.5	0	0	94.0	5.0
Ne	0	0	5.5	15.0	0.5	0	0	79.0

Table 5.2: Confusion matrix of HOG and LBP on the CK+ dataset.

confusion matrices for LBP and HOG, on CK+ dataset, using LDA and a k-NN classifier. HOG's struggle with the neutral expression is evident, while LBP has close to 30% leap on that same expression. On the other hand HOG has an average more 20% accuracy on Fear and Anger expressions. This also happens in other datasets, possibly with different expressions.

By simply concatenating HOG and LBP descriptors, more consistent and better classification accuracies can be obtained. This is key for a robust algorithm, so from now on we are going to use the concatenation of both descriptors to face the problem of emotion recognition. Take note that when concatenating, the descriptors were normalised to the same range of values. In this case LBP was normalised to match the range of values of HOG. This must be done individually for every dataset.

5.4 Choosing Final Method

In this last section we are finally going to decide which combination of techniques yield the most robust method. To do that we are once again testing all classifying methods, this time with concatenated descriptors as well as without supervised DR before LCK-SVD. We will take some final notes based on Table 5.3 and finally test our method on the CK+ benchmark testing protocol [2] in the next section, comparing it with other works.

In Table 5.3, we can take a look at the accuracies obtained for the final descriptors. For each dataset, we rerun the experiments without the expression that most confusion caused according to Appendix B. Take note that despite the Sad expression having the lowest accuracy rate in Table B.1, that is mainly due to the presence of the Neutral expression, which causes most errors. For that reason in the case of the CK+, the Neutral expression will be removed.

Much like the previous table, k-NN and SVM have similar classification accuracies, while OMP seems to perform better the most of the times. Also OMP is consistent regardless of the DR, while k-NN and SVM have much better accuracies with DLPP, on bigger datasets. This might indicate that big datasets generate more complex manifolds, which OMP deals with quite well. DLPP is definitely a more

	CK+				BU-3DFE				KDEF			
	All		No Neutral		All		No Sad		All		No Fear	
	LDA	DLPP	LDA	DLPP	LDA	DLPP	LDA	DLPP	LDA	DLPP	LDA	DLPP
k-NN	87.0	87.1	94.9	95.6	66.0	69.2	72.4	79.7	74.8	84.4	84.3	89.9
SVM	87.5	87.5	95.1	96.2	65.6	69.3	72.3	80.0	74.4	84.5	83.4	89.9
OMP	89.4	89.6	95.7	97.1	69.1	70.2	78.5	79.8	81.8	82.5	88.7	90.2
LCK-SVD1	90.1		97.0		71.0		76.4		83.0		90.9	
LCK-SVD2	90.1		96.9		70.6		76.6		83.1		90.8	

Table 5.3: Final accuracies obtained by classification techniques.

robust DR technique as it is better in every occasion.

Sparse representation classification techniques are, as expected, the ones who benefit the most from the concatenation of descriptors. Both LCK-SVD have similar results, and the classification error minimisation in LCK-SVD2 seems to make no difference at all. Also the little advantage that LCK-SVD has over OMP seems to vanish when the expression that causes the most confusion is taken out from the dataset. This can signify that in case of not existing patterns of expression mislabelling, LCK-SVD has no advantage over other methods. This is not what happens with expressions, as naturally there are certain expressions which are tricky to distinguish. However the payoff for the higher computational cost and laboured tuning of LCK-SVD appears to be little.

5.5 Benchmarking Performance

For all the reasons stated in the previous section, OMP in conjugation with DLPP preceded by a PCA, using a concatenation of HOG and LBP descriptors is our proposed method. We shall name it Histogram Descriptors' Supervised Manifold Preserving Sparse Representation, abbreviated HDSMPSR. To show its power we are going to test it side by side with other recently proposed methods, on the CK+ dataset. The CK+ benchmark testing protocol [2] stipulates a leave-one-out cross-validation policy. Which yields 118 different testing sets, one per subject, and 327 expressions to classify.

	HDSMPSR	Ptucha et al.[25]	Jain et al.[41]	Lucey et al.[2]	Chew et al.[42]
Accuracy	96.3	91.4	84.1	83.3	74.4

Table 5.4: Expression classification accuracy of several methods according to the CK+ benchmark testing protocol.

Ptucha et al.[25] also used SR, employing a regional based statistical inference model. Jain et al.[41] and Lucey et al.[2] used active appearance models with linear SVM classifiers. Chew et al.[42] used normalised pixels from constrained local models and a linear SVM as well. Our method, HDSMPSR, with histogram descriptors followed by DLPP and OMP, got 315, out of 327 expressions, correctly classified.

Clearly outperforming any of the other methods. For the record, the use of SVM instead of OMP scored 92.6%, while LCK-SVD1 and LCK-SVD2 scored 83.8% and 84.1% respectively. One reason for LCK-SVD may be the need for retuning, as the paradigm of classification changed radically.

In the following table there is also a comparison between our method and those obtained by human observer results on the 490 apex expressions of the KDEF dataset. Still our HDSMPSR method, presents to be a far more reliable classifier.

	HDSMPSR	Goeleven et al.[43]
Accuracy	82.5	74.6

Table 5.5: HDSMPSR compared to human observer on KDEF.

Our method is clearly very good on the task of classifying expressions. It also gives positive indications on his robustness, since it behaves good on every dataset. Also, it is computationally efficient and easy to use, since it does not require any parameter adjusting.

Chapter 6

Conclusions

In this work, the basic tools for computer expression recognition are covered. We started of with an explanation on the most popular pixel processing techniques used in emotion recognition. We moved on, to present four pixel processing techniques of the various kinds. Two already established, and other two very recent developed. We ended up with the classification techniques, briefly explaining the concept behind SVM, perhaps the most used classification technique in computer vision, then to focus in sparse representation. Here a recently developed supervised manner for dictionary construction was presented.

A new way of classifying facial expressions was proposed. Our novel HDSMPSR method, obtained state of the art classification accuracy for emotion recognition in natural images when compared with methods developed over the last years, or even human classification. We believe that HDSMPSR can be successfully applied to other problems in machine learning as well.

Histogram descriptors, such as LBP and HOG are very powerful tools for learning and matching patterns and textures, particularly in faces, having the advantage of being forgiving with subtle pose changes, one of the big problems in facial understanding. A supervised DR method that preserves manifolds, turns out to be a very good way of reducing data to very low dimensions. Truth is, that DLPP together with OMP are more robust than LCK-SVD methods, since no tuning is required in between problems. Orthogonal matching pursuit has proven to be very sharp for classifying emotions, consistently yielding better performances than SVM.

6.1 Future Work

In our perspective the efforts on computer basic emotion classification, should be focused on natural 2D single images, as much of the information is or can be easily extracted that way. Also the use of 2D imagery reduces the need for complex applications and widens the opportunities to use them.

We do believe, that there is still margin for further improvements. Emotion detection in natural environment needs to successfully deal with radical head poses and occlusions, even when face tracking mechanisms are taken for granted. Non-staged and non-centred expressions also seem to pose a boundary on classification results.

For that reason a new and more exigent protocol would be important for expression classification. It is necessary at this point to focus efforts on dealing with non frontal poses as well as genuine expressions. The CK+ benchmark testing protocol seems to be falling behind on the demands of real world tasks, since in datasets like BU-3DFE and KDEF the accuracy decreases quite significantly.

Nevertheless, emotion recognition seems to be evolving fast and a huge leap has been given just in the last three years. It is of great importance to learn how apex expressions are connected with identity, race, gender and age. The goal is to enable a complete automated facial recognition from an image or a single frame of video, regardless of quality, head pose and occlusions.

Appendix A

Technical details

Here follows some briefing on the codes, specifications and tunings of the algorithms used on our proposed method. The technical details are common to all datasets used and were not altered during tests.

PPTs

The tuning of the various pixel processing techniques was not really part of our study, so no tests were performed whether to see which parameters yield best accuracies. Mostly default and/or recommend settings were used.

For HOG a toolbox available from [44] was used, with the cell size set to 5. As for LBP an algorithm provided by [6] was used on sets of 21×21 pixels. The images were first re-sized to 168×168 and then divided in 8×8 regions.

DR

A good dimensionality reduction toolbox was available on [35]. The codes of DLPP and KNDA were written according to the original papers [21, 22]. As stated before 99% of the highest eigenvalues resulting from PCA were used, while the supervised DR reduced to $c - 1$ dimensions, c being the number of classes. For DLPP we used $t = 1$ and normalised norms before using them for B and W. As for KNDA a linear kernel was used.

Classification, SR and Tuning LCK-SVD

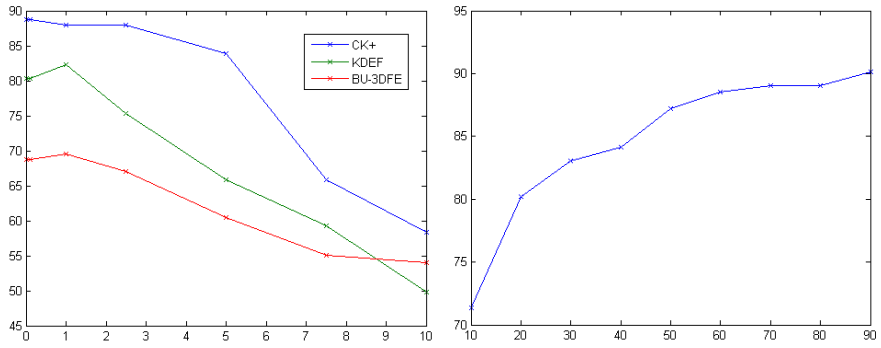
Recent Matlab editions have toolboxes for k-NN, SVM and OMP. But in this work we used code from [45] and [46]. The number of neighbours used for k-NN were 10, and for SVM a linear kernel with a cost coefficient of 0.001 was used. The dictionary used in OMP was the mean vectors of the different classes. So no dictionary was leaned for OMP. The sparsity constrain for OMP was $c - 1$ dimensions, c being the number of classes.

The code for the LCK-SVD was available in the original paper [39]. But there was still need to tune the parameters which is one of the downsides of LCK-SVD is the tuning. With about 6 parameters to tune, this process can turn out to be

a bit laborious, specially when it is desired to use a single configuration in several datasets. Fortunately in our case some parameters are fairly easy to tune, as all datasets seem to benefit equally from certain configurations.

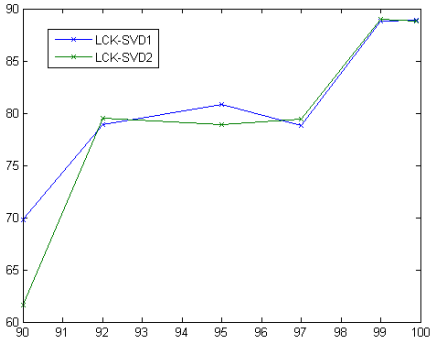
Increasing dictionary size betters LCK-SVD performance, but only until a certain point, from which on accuracy is maintained but computational cost augments. Increasing sparsity constrain, the number of non-zero sparse coefficients, also increases accuracy, for that reason it was set equal to the dictionary size, 15 times the number of classes. The number of iterations makes little difference on the accuracy, since in this case they converge rather fast, but mean a great deal on computational time and cost. For that reason they were set to a minimum possible of 2 initialising iterations and 5 for training.

The coefficients α and β for the reconstruction and classification errors, really depend on the type of data, and require some experiments, in this case both α and β were set to 1, despite being very flexible. Take note, that LCK-SVD1 can have very different accuracy rates from LCK-SVD2. In some cases it can even be half. But in the end if planning to use the classification error do not bother taking LCK-SVD1 as an indicator, since having one less term leads to quite different configuration. Figure A.1 shows a couple of experiments, proving some of our points.



(a)

(b)



(c)

Figure A.1: Accuracies on the y axis for the following varying parameters: (a) α on LCK-SVD1 (b) Sparsity constrain on CK+ (c) Percentage of PCA's eigenvalues for CK+

Appendix B

Datasets' confusion matrices

The confusion matrices of each dataset result from a k-NN classification of 26×20 downscaled photos, with an initial PCA DR to 99% of the eigenvalues followed by an LDA to $c-1$ dimensions, c being the number of classes. In this as in the following appendices the terms of testing used for each dataset were as described in Section 5.1.

	Co	Fe	Sa	An	Di	Ha	Su	Ne
Co	93	0	4.5	0	0	0	0	2.5
Fe	3	87.5	0	0	0.5	1.5	0	7.5
Sa	10.5	6	57	5	0	0	0	21.5
An	1.5	0	5	76	0	0	0	17.5
Di	0	0	0	0	90	4	0	6
Ha	0	2.5	0	0	0.5	97	0	0
Su	0	0	2.5	0	1	0	88	8.5
Ne	0	0	7	12	5	0	0.5	75.5

Table B.1: Confusion matrix of the CK+ dataset, by the end of 100 runs.

	Fe	Sa	An	Di	Ha	Su	Ne
Fe	54.3	9.4	5.8	12	6.3	3.1	9.1
Sa	10.7	47.2	19.4	4.4	1.7	1.8	14.8
An	4.9	19	51.1	10.6	1.7	0	12.7
Di	8.9	6.8	10.5	61.4	4.3	3.7	4.4
Ha	9	0	0.1	5.2	81.6	3.2	0.9
Su	3.7	2.5	1.1	4.2	0.5	85.7	2.3
Ne	6.1	13.4	6.9	3	0.7	1.7	68.2

Table B.2: Confusion matrix of the BU-3DFE dataset, by the end of 20 runs.

	Fe	Sa	An	Di	Ha	Su	Ne
Fe	58.6	12	1	0.1	2.6	19.1	6.6
Sa	10.3	73.1	2.6	4	0	0.7	9.3
An	4.7	8	72.3	10.7	0.3	0	4
Di	1.4	6.1	13.6	75.2	2.7	0.1	0.9
Ha	3.7	1.7	0	0	91.3	2.1	1.2
Su	18.1	0.4	0.2	0	0	80.1	1.2
Ne	2.3	7.4	0.9	0	0	0	89.4

Table B.3: Confusion matrix of the KDEF dataset, by the end of 40 runs.

Appendix C

PPTs and DR methods accuracy

For the three datasets we used a simple k-NN classifier, with k set to 10. So what we are accounting for is really the clustering capability of each pair of techniques. Except for PCA, the dimensions were reduced $c-1$ dimensions, c being the number of classes. The angry expression was left out of the BU-3DFE for results' proximity.

	PCA	LDA	LPP	DLPP	KNDA
Raw	34.1	82.5	20.5	81.8	12.4
Downscaled	31.4	84.2	20.9	82.5	12.1
Phase	46.0	74.1	32.8	75.3	29.7
Magnitude	14.2	61.1	19.2	59.3	29.2
LBP	51.3	88.3	30.9	86.9	26.4
pwLBP	38.3	75.1	19.9	72.9	18.8
Gabor	21.4	70.6	19.3	69.8	14.2
HOG	67.8	86.4	52.0	87.1	44.4

Table C.1: DRs and PPTs on the CK+ dataset.

	PCA	LDA	LPP	DLPP	KNDA
Raw	36.4	64.9	25.5	66.7	29.8
Downscaled	42.0	66.8	29.3	69.1	25.3
Phase	45.3	56.2	31.3	59.0	26.2
Magnitude	32.8	57.9	35.3	59.0	32.7
LBP	49.7	77.2	26.2	77.6	30.1
pwLBP	45.6	44.9	31.7	57.6	31.6
Gabor	27.0	60.9	22.3	59.6	27.3
HOG	65.8	75.2	41.6	77.4	34.7

Table C.2: DRs and PPTs on the BU-3DFE dataset.

	PCA	LDA	LPP	DLPP	KNDA
Raw	45.1	76.4	27.8	77.2	29.2
Downscaled	46.9	78.0	30.4	77.0	18.5
Phase	44.5	63.0	31.8	65.2	25.6
Magnitude	35.7	66.9	47.6	68.4	28.9
LBP	58.3	85.3	49.4	82.2	37.3
pwLBP	58.2	37.2	52.3	66.8	34.5
Gabor	19.5	71.5	13.0	69.4	12.4
HOG	63.9	77.1	51.8	83.1	38.4

Table C.3: DRs and PPTs on the KDEF dataset.

Bibliography

[1] L. Yin, X. Wei, Y. Sun, J. Wang, and M. Rosato, "A 3d facial expression database for facial behavior research," in *Automatic Face and Gesture Recognition, 2006. FGR 2006. 7th International Conference on*, pp. 211–216, April 2006.

[2] P. Lucey, J. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews, "The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, pp. 94–101, June 2010.

[3] O. Lundqvist, Flykt, "The karolinska directed emotional faces," tech. rep., Department of Clinical Neuroscience, Psychology section, Karolinska Institutet, 1998.

[4] T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative study of texture measures with classification based on featured distributions," *Pattern Recognition*, vol. 29, no. 1, pp. 51 – 59, 1996.

[5] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, pp. 971–987, Jul 2002.

[6] T. Ahonen, A. Hadid, and M. Pietikainen, "Face description with local binary patterns: Application to face recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, pp. 2037–2041, Dec 2006.

[7] D. Maturana, D. Mery, and A. Soto, "Face recognition with local binary patterns, spatial pyramid histograms and naive bayes nearest neighbor classification," in *Proceedings of the 2009 International Conference of the Chilean Computer Science Society*, SCCC '09, (Washington, DC, USA), pp. 125–132, IEEE Computer Society, 2009.

[8] D. Huang, C. Shan, M. Ardabilian, Y. Wang, and L. Chen, "Local binary patterns and its application to facial image analysis: A survey," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 41, pp. 765–781, Nov 2011.

[9] H. B. Kekre and V. A. Bharadi, "Article: Face recognition using gabor filter based feature vector for mobile phones," *IJAIS Proceedings on International Conference and workshop on Advanced Computing 2013*, vol. ICWAC, pp. 6–11, June 2013. Published by Foundation of Computer Science, New York, USA.

[10] M. Sharif, S. Mohsin, M. J. Jamal, M. Y. Javed, and M. Raza, "Face recognition for disguised variations using gabor feature extraction.," *Australian Journal of Basic & Applied Sciences*, vol. 5, no. 6, 2011.

[11] W. Zhang, S. Shan, W. Gao, X. Chen, and H. Zhang, "Local gabor binary pattern histogram sequence (lgbphs): a novel non-statistical model for face representation and recognition," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 1, pp. 786–791 Vol. 1, Oct 2005.

[12] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 886–893 vol. 1, June 2005.

[13] Q. Zhu, M.-C. Yeh, K.-T. Cheng, and S. Avidan, "Fast human detection using a cascade of histograms of oriented gradients," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 2, pp. 1491–1498, 2006.

[14] T. Kobayashi, A. Hidaka, and T. Kurita, "Selection of histograms of oriented gradients features for pedestrian detection," in *Neural Information Processing* (M. Ishikawa, K. Doya, H. Miyamoto, and T. Yamakawa, eds.), vol. 4985 of *Lecture Notes in Computer Science*, pp. 598–607, Springer Berlin Heidelberg, 2008.

[15] H.-X. Jia and Y.-J. Zhang, "Fast human detection by boosting histograms of oriented gradients," in *Image and Graphics, 2007. ICIG 2007. Fourth International Conference on*, pp. 683–688, Aug 2007.

[16] X. Zhu and D. Ramanan, "Face detection, pose estimation, and landmark localization in the wild," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 2879–2886, June 2012.

[17] O. Déniz, G. Bueno, J. Salido, and F. D. la Torre, "Face recognition using histograms of oriented gradients," *Pattern Recognition Letters*, vol. 32, no. 12, pp. 1598 – 1603, 2011.

[18] A. Albiol, D. Monzo, A. Martin, J. Sastre, and A. Albiol, "Face recognition using hog–ebgm," *Pattern Recognition Letters*, vol. 29, no. 10, pp. 1537 – 1543, 2008.

[19] Y. Pan, S. S. Ge, and A. A. Mamun, "Weighted locally linear embedding for dimension reduction," *Pattern Recognition*, vol. 42, no. 5, pp. 798 – 811, 2009.

[20] F. L. Rongbing Huang, Changming Su and M. Du, "Kernel discriminant locality preserving projections for human face recognition," *Journal of Information and Computational Science*, no. 7, pp. 925 – 931, 2010.

[21] W. Yu, X. Teng, and C. Liu, "Face recognition using discriminant locality preserving projections," *Image and Vision Computing*, vol. 24, no. 3, pp. 239 – 248, 2006.

[22] X. Zhao, "A kernel based neighborhood discriminant submanifold learning for pattern classification," *Journal of Applied Mathematics*, 2014.

[23] T. S. H. Le Hoang Thai, Nguyen Do Thai Nguyen, "A facial expression classification system integrating canny, principal component analysis and artificial neural network," *CoRR*, vol. abs/1111.4052, 2011.

[24] C. Shan and R. Braspenning, "Recognizing facial expressions automatically from video," in *Handbook of Ambient Intelligence and Smart Environments* (H. Nakashima, H. Aghajan, and J. Augusto, eds.), pp. 479–509, Springer US, 2010.

[25] R. Ptucha and A. Savakis, "Manifold based sparse representation for facial understanding in natural images," *Image and Vision Computing*, vol. 31, no. 5, pp. 365 – 378, 2013.

[26] Y.-L. Tian, T. Kanade, and J. Cohn, "Recognizing action units for facial expression analysis," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 23, pp. 97–115, Feb 2001.

[27] M. Bartlett, G. Littlewort, M. Frank, C. Lainscsek, I. Fasel, and J. Movellan, "Recognizing facial expression: machine learning and application to spontaneous behavior," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2, pp. 568–573 vol. 2, June 2005.

[28] M. Mahoor, M. Zhou, K. L. Veon, S. Mavadati, and J. Cohn, "Facial action unit recognition with sparse representation," in *Automatic Face Gesture Recognition and Workshops (FG 2011), 2011 IEEE International Conference on*, pp. 336–342, March 2011.

[29] I. Cohen, N. Sebe, A. Garg, L. S. Chen, and T. S. Huang, "Facial expression recognition from video sequences: temporal and static modeling," *Computer*

Vision and Image Understanding, vol. 91, no. 1–2, pp. 160 – 187, 2003. Special Issue on Face Recognition.

[30] P. Michel and R. E. Kaliouby, "Real time facial expression recognition in video using support vector machines," 2003.

[31] G. Fanelli, A. Yao, P.-L. Noel, J. Gall, and L. Van Gool, "Hough forest-based facial expression recognition from video sequences," in *Trends and Topics in Computer Vision* (K. Kutulakos, ed.), vol. 6553 of *Lecture Notes in Computer Science*, pp. 195–206, Springer Berlin Heidelberg, 2012.

[32] I. Sobel and G. Feldman, "A 3x3 isotropic gradient operator for image processing." Never published but presented at a talk at the Stanford Artificial Project, 1968.

[33] H. Hotelling, "Analysis of a complex of statistical variables into principal components," *J. Educ. Psych.*, vol. 24, 1933.

[34] E. Kokiopoulou, J. Chen, and Y. Saad, "Trace optimization and eigenproblems in dimension reduction methods," *Numerical Linear Algebra with Applications*, vol. 18, no. 3, pp. 565–602, 2011.

[35] L. Van der Maaten, E. Postma, and H. Van den Herik, "Dimensionality reduction: A comparative review," *Technical Report TiCC TR 2009-005*, 2009.

[36] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, pp. 273–297, Sept. 1995.

[37] P. Breen, "Algorithms for sparse approximation," tech. rep., School of Mathematics, University of Edinburgh, 2009.

[38] M. Aharon, M. Elad, and A. Bruckstein, "K-svd: Design of dictionaries for sparse representation," in *Proceedings of Spars'05*, pp. 9–12, 2005.

[39] Z. Jiang, Z. Lin, and L. S. Davis, "Learning a discriminative dictionary for sparse coding via label consistent k-svd," 2011.

[40] G. H. Golub, P. C. Hansen, and D. P. O'Leary, "Tikhonov regularization and total least squares," 1999.

[41] S. Jain, C. Hu, and J. Aggarwal, "Facial expression recognition with temporal modeling of shapes," in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pp. 1642–1649, Nov 2011.

[42] S. Chew, P. Lucey, S. Lucey, J. Saragih, J. Cohn, and S. Sridharan, "Person-independent facial expression detection using constrained local models," in *Automatic Face Gesture Recognition and Workshops (FG 2011), 2011 IEEE International Conference on*, pp. 915–920, March 2011.

[43] E. Goeleven, R. De Raedt, L. Leyman, and B. Verschuere, "The karolinska directed emotional faces: A validation study," *Cognition & Emotion*, vol. 22, no. 6, pp. 1094–1118, 2008.

[44] A. Vedaldi and B. Fulkerson, "VLFeat: An open and portable library of computer vision algorithms." `http://www.vlfeat.org/`, 2008.

[45] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[46] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *J. Mach. Learn. Res.*, vol. 11, pp. 19–60, Mar. 2010.