TAMPERE UNIVERSITY OF TECHNOLOGY

JYRKI NUMMINEN

PEOPLE IN PHOTOS

Master of Science Thesis

Examiner: Prof. Irek Defee
Examiner and topic approved in the Faculty of Computing and Electrical Engineering council meeting on 8th of November 2013

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Signaalinkäsittelyn ja tietoliikennetekniikan koulutusohjelma
NUMMINEN, JYRKI: Ihmisryhmät valokuvissa
Diplomityö, 49 sivua, 2 liitesivua
Marraskuu 2013
Pääaine: Signaalinkäsittely
Tarkastajat: Professori Irek Defee
Avainsanat: Kasvojentunnistus, Ilmeidenntunnistus, Koneoppiminen

Hyvälaatuisten digitaalikameroiden sekä kameralla varustettujen älypuhelimien yleistyminen on avannut uusia tarpeita ja mahdollisuuksia automaattiselle kuvien luokittelulle. Aihetta on tutkittu useista lähtökohdista, kuten sukulaisuussuhteiden automaattisesta tunnistuksesta tai kuvassa olevan ihmisryhmän sosiaalisen alakulttuurin löytämisestä.
Tämä diplomityö käsittelee valokuvista tunnistettavien ihmisryhmien automaattista luokittelua käyttäen erilaisia koneoppimisen menetelmiä. Diplomityö tehtiin Tampereen Teknillisen Yliopiston (TTY) signaalinkäsittelyn laitokselle vuonna 2013. Koska käytetyt menetelmät tarvitsevat myös tietoa kasvokohtaista ominaisuuksista etsin ja toteutin useita menetelmiä niin sukupuolen, iän kuin ilmeidenkin tunnistamiseksi. Rakensin esimerkkiohjelman näiden menetelmien pohjalta Android-käyttöjärjestälmälle ja lopuksi arvioin käytettyjen menetelmien käytännöllisyyttä.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Signal Processing and Communications Engineering
NUMMINEN, JYRKI: People in photos
Master of Science Thesis, 49 pages, 2 Appendix pages
November 2013
Major: Signal processing
Examiner: Prof. Irek Defee
Keywords: Face recognition, Expression recognition, Machine Learning

As digital cameras and camera equipped smart phones have become commonplace both the needs and the opportunities to automatically categorize photographs have increased. The subject has been researched with several goals in mind, for example how to find family relations in group photos or how to categorize a groups of people by their subculture.

This thesis was made for Tampere University of Technology (TUT) department of signal processing in 2013. In this thesis I review and analyze recent research into face and face properties recognition. Specifically methods of identifying individuals identity, age, gender and expression from known data sets for labeling purposes are considered. Additionally labeling methods of photo categorization based on groups of people in them are examined. Based on this analysis I then constructed an example implementation of these methods as a part of a camera application on Android mobile platform. Finally the suggested methods are evaluated in terms of practical usability.

PREFACE

This thesis was made for the Tampere University of Technology department of signal processing. Its goal was to review and test methods of automatic group photo classification based on face content.

I would like to thank prof. Irek Defee for guidance and the examination of this thesis.

Tampere, November 2013

Jyrki Numminen
Tel.: +358 40 717 2885

TABLE OF CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

AR-LBP	Asymmetric Region Local Binary Pattern
CLBPM	Completed Local Binary Pattern Magnitude
ELBP	Elliptical Local Binary Pattern
GV-LBP	Gabor Volume Local Binary Pattern
JNI	Java Native Interface
LBP	Local Binary Pattern
LDA	Linear Discriminant Analysis
PCA	Principal Component Analysis
SVD	Singular Value Decomposition
SVM	Support Vector Machine
$Sgn(x)$	Sign function
XOR	Exclusive Or
α_k	A Lagrange multiplier
$\{\omega_1, \dots, \omega_n\}$	Set of labels
$\bigotimes$	Convolution operator.
θ	Threshold value
$\boldsymbol{a}$	Augmented weights vector
$C(x)$	A classifier function
D	Set of features and labels for teaching an supervised system
E_k	Training error
x_k	A feature in a pattern
y_k	A label
$\boldsymbol{z}$	Augmented and mapped feature vector
Z_k	Normalizing constant
W	Area of pixels defining a window
$W_k(i)$	Weight or voting coefficient for a pattern-label pair
$\bar{X}$	Mean value of X

1. INTRODUCTION

The problem machine learning is trying to solve is, broadly speaking, categorization. For example a simple linear division over two variables can try to answer the question "is this wood sample from a pine or a fir tree" when the input is the texture of a sample and a density measurement. Tricky part is finding the best way to split the *feature space* for sample measurements in a way that the error rate is as low as possible. Additionally the method to accomplish this task should be adaptive to new input, therefore being a learning system.

These kind of learning systems can be split into two main categories, supervised and unsupervised systems. The difference between the two is data labeling. Unsupervised methods know only about the features of a sample, not its category. These kind of methods are often used in data mining as they try to extract labeling information from data. For example a question of "How many species of trees were these samples collected from" might be solved using an unsupervised learning algorithm. The line between supervised and unsupervised learning can be a blurry one though. In some cases we want to learn categorizations like "how these samples can be arranged into 5 different groups" or "We know these five samples are from pines and these other five are from firs. What would be a good classifier for these species of trees when there are samples from eight different species in the data set?"

In this thesis the interest is on recently developed, supervised learning algorithms which are used in and relating to face recognition. Commonly used mathematical tools in these algorithms are introduced in chapter 2. In face recognition the goal is to match a section of a photo (*a window*) against a previously learned model so the algorithm can return the areas containing faces. Different methods to achieve this goal are discussed in chapter 3. From these face matches a new set of features can be extracted through different methods. Using these extracted features it is often possible to recognize the identity and even emotional state of the target. These methods are discussed in chapters 4 and 5 respectively. Additionally it is possible to measure spatial relations from the found faces. This relational arrangement of people can be used to further categorize photographs. These methods are discussed in chapter 6. Finally in chapter 7 I discuss the implementation of these algorithms in Android-based software and consider the practical effectiveness of these methods.

2. MATHEMATICAL TOOLS USED IN FACE AND PATTERN RECOGNITION

In this chapter some of the commonly used tools relevant for this thesis are presented. They are used to either simplify larger datasets into more manageable forms or in extraction of meaningful data. Principal Component Analysis is used in “Eigenfaces” face identification method, local binary patterns are used in nearly every recognition method in this thesis and the Gabor filters are used in the Gabor Volume LBP-identity recognition method.

2.1 Principal Component Analysis, PCA

Principal Component Analysis [2], also known as discrete Karhunen-Loéve transform is an orthogonal linear transformation of a dataset, one of the fundamental tools used in any kind of high dimensional data analysis. The goal of PCA is to maximize the variance along one dimension while minimizing it on others. If the variance is small enough (below some arbitrary threshold) it may be considered irrelevant to the problem in hand and discarded altogether. In the figure 2.1 the two dimensional dataset on the left side is processed through PCA and the result can be seen on the right. The axis with the highest variance, the principal component, is now the “new axis 1”. As this is two dimensional example the “new axis 2” is automatically the least meaningful dimension when it comes to separating the samples. The process to calculate a PCA solution can be seen in algorithms 2.1 and 2.2.

Although PCA is straightforward to use, the user must take care of keeping the input set balanced with intended use in mind. The only parameter the analysis cares is the sum of squared sample distances per dimension. If the samples are not measuring the same type of variable in every dimension, they need to be somehow normalized to avoid situations similar to the dataset in the lower right corner of figure 2.2.

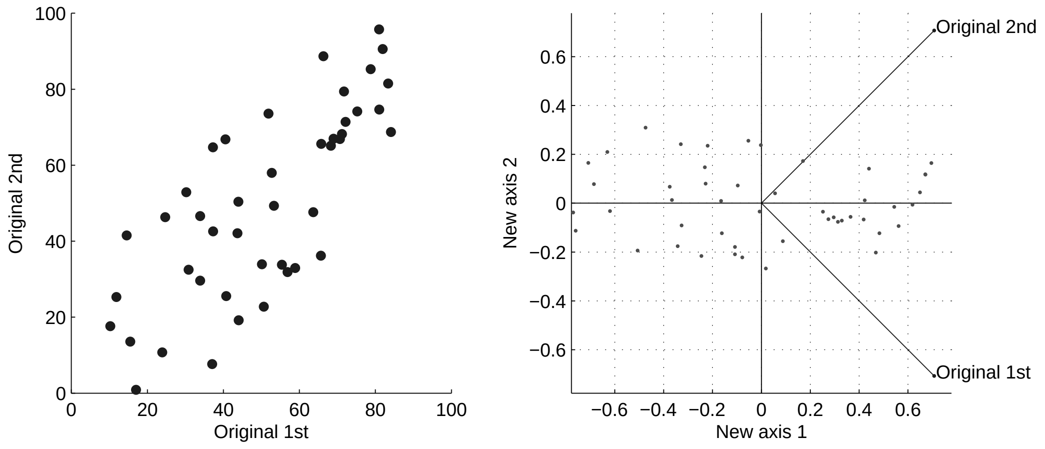


Figure 2.1. *Example result of PCA-analysis. Generated with script A.3*

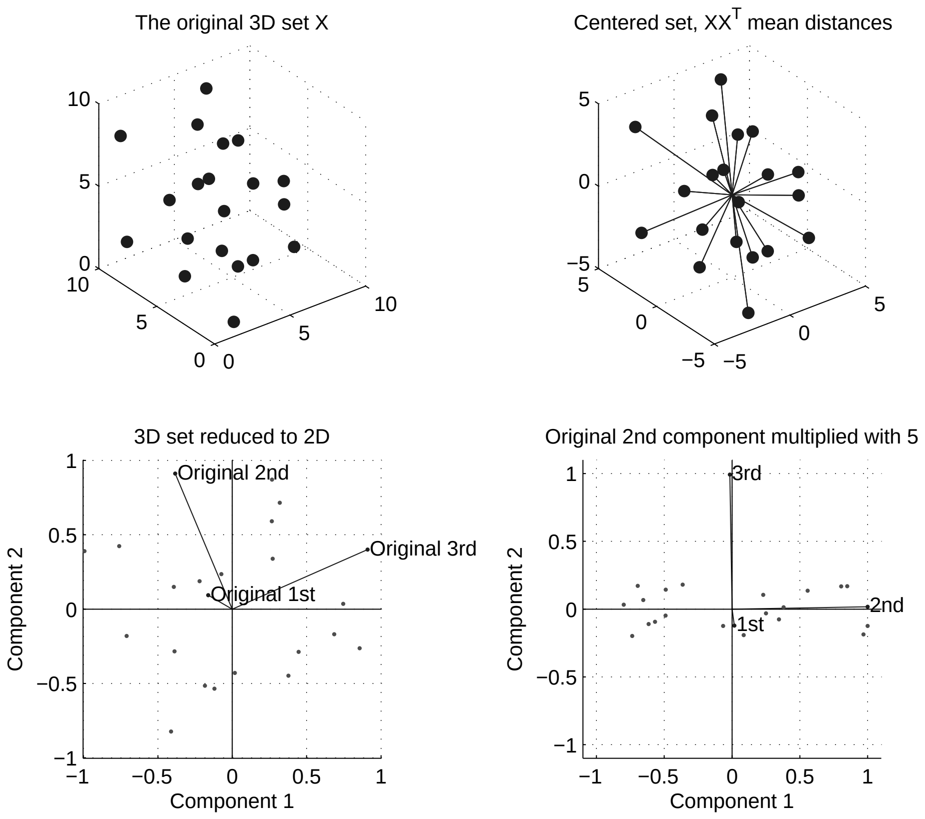


Figure 2.2. *Steps in PCA. Generated with script A.4.*

Algorithm 2.1 Definition of PCA-analysis [2]

```
1: begin initialize $X_k$ =Dataset to be analyzed,$(k = 1, \ldots, N), X_k \in IR^D$
2: Center the dataset so that $\sum_{k=1}^{N} X_k = 0$
3: Covariance matrix $C = \frac{1}{N} \sum_{k=1}^{N} X_k X_k^T$
4: $C$ can be factorized into form $\Psi \Delta \Psi^T$ where $\Psi = [\phi_1, \phi_2, \ldots, \phi_N]$ is a matrix
   with Eigenvectors of $C$ and $\Delta = [\Delta_1, \Delta_2, \ldots, \Delta_N]$ which is the diagonal matrix
   with Eigenvalues $\{\lambda_1 \lambda_2 \ldots \lambda_N\}$ on its diagonal. Eigenvectors should be sorted
   to descending order.
5: As first M Eigenvectors match M largest Eigenvalues,
   create matrix $\Psi^* = [\phi_1, \phi_2, \ldots, \phi_M]$, where $M \leq D$ is the wanted number of
   remaining dimensions.
6: return Original dataset projected into new PCA space by $\Psi^*$
7: end
```

Algorithm 2.2 Practical implementation of PCA-analysis

```
1: begin initialize $X_k$ =Dataset to be analyzed,$(k = 1, \ldots, N), X_k \in IR^D$
2: Subtract the mean $\bar{X}$ from each member of $X$.
3: Per demands in algorithm 2.1, line 4:
       $Y^T = X^T W, Y$ is transformation of X by W
           $= (W \Sigma V^T)^T W$, Singular value decomposition (SVD) of $X$
           $= V \Sigma^T W^T W = V \Sigma^T$
4: The colums of $Y^T$ now hold the score values of their corresponding eigenvectors.
5: To get $M$ first reduced dimensional representations of $X$ the projection for $W_M$
   is $Y = W_M^T X = \Sigma_M V^T$, where $\Sigma_M = I_{M \times m} \Sigma$, and $I_{M \times m}$ is identity matrix of
   suitable size.
   As square roots of eigenvalues in $\Sigma$ are relative to $X$:s covariance matrix $XX^T$
   the results are proportional to variance within $X$ because we are calculating
   with deviations from the mean $\bar{X}$.
6: return PCA projection $Y$
7: end
```

2.2 Local Binary Patterns, LBP

Local Binary Patterns are mostly non-parametric descriptors, originally used for texture analysis by Ojala, Pietikäinen and Harwood [3]. Two dimensional face picture is texture in itself, but by applying LBP it can be seen as local texture feature extractor, producing similar results to similar part of faces. LBP is simple, but surprisingly powerful feature extraction method detecting per pixel gradients. The whole operation is a small sliding window producing a binary image by comparing its center pixel value to neighboring pixels. If windows center pixel value is f_p :

$$LBP(p_c) = \sum_{k=0}^{7} \delta(f_{k+1} - f_p)2^k, \text{ per fig.2.3, } \delta(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{2.1}$$

In other words, the region values with numbering scheme from figure 2.3 are determined by *thresholding* area average pixel values against the middle pixel value of the scanning window. The resulting binary values form an LBP-code by concatenation in order, i.e. if areas 8 and 3 are the only areas with higher average value than the middle pixel, the resulting LBP binary code will be 10000100_b which can then be interpeted as an 8-bit integer like in eq. 2.1. Areas outside the image are considered to be removed from the area, both in value and area size when calculating averages later on.

Other variants based on the original have been created for specific areas of research. Naika, Das and Nair [4] have suggested an Asymmetric LBP specifically for face recognition tasks. In this configuration the LBP kernel is adjusted in size by parameters n and m, scaling the size in width and height. The values compared against the center pixel value are the averages of areas seen in figure 2.3. With $n, m = 1$ the AR-LBP is identical to the original LBP. The third variant of LBPs used in this thesis is the *Elliptical LBP*, or Elongated LBP [5]. The basic operating principle is the same than in original LBP but using a cell selection based on elliptical path around the center. The length of the code can be adjusted by selecting the number of divisions and size of the selection ellipse. An example of an ELBP kernel can be seen in figure 2.3.

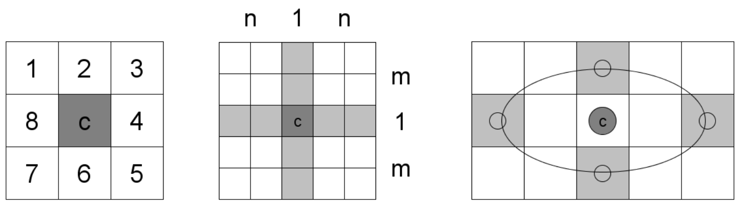


Figure 2.3. *The original LBP kernel, AR-LBP kernel and ELBP kernel. Indexing matches to k in eq. 2.1*

2.3 Gabor filters

Gabor filters are adjustable bandwidth filters originally defined by Dennis Gabor in 1946. [6] In image processing Gabor filters are extended to 2D space and used, for example, in edge detection, feature extraction and finding parallax shifts in stereoscopic photographs. A Gabor filter can be constructed by taking any complex plane harmonic function and modulating it by a Gaussian function to envelope it. They are specially interesting due to their similarity to real biological sensing processes and are also computationally pleasant as they can be calculated with Fourier transforms. If $\Im(\cdot)$ is a Fast Fourier Transform, $z = (x, y)$–coordinates and $\bigotimes$ is convolution operator, then

$$O_{\mu,\nu}(z) = I(z) \bigotimes \psi_{\mu,\nu}(z)$$

$$\Im\{O_{\mu,\nu}(z)\} = \Im\{I(z)\}\Im\{\psi_{\mu,\nu}(z)\}$$

$$O_{\mu,\nu}(z) = \Im^{-1}\{\Im\{I(z)\}\underbrace{\Im\{\psi_{\mu,\nu}(z)\}}_{\text{Pre-calculable}}\}$$

where $O_{\mu,\nu}$ is the filtered result of image I using filter $\psi_{\mu,\nu}$. A Gabor filter in this thesis (or a Gabor kernel) is defined like in [8].

$$\psi_{\mu,\nu}(z) = \left\|\frac{k^2_{\mu,\nu}}{\sigma^2}\right\| e^{\frac{k^2_{\mu,\nu}z^2}{2\sigma^2}}\left(e^{ik_{\mu,\nu}z} - e^{-\frac{\sigma^2}{2}}\right) \tag{2.2}$$

$$k_\nu = \frac{\pi/2}{f^\nu}, f = \sqrt{2}, k_{\mu,\nu} = k_\nu e^{\frac{i\pi\mu}{8}}, \sigma = 2\pi \tag{2.3}$$

This kernel can be used as a mother filter to generate several kernels for different scales μ and rotations ν. The real parts of filters generated using values $\mu = 0 \dots 7$ and $\nu = 0 \dots 4$ can be seen in figure 2.4.

An image can be Gabor-filtered spatially using the example implementation in script A.2. The result from this filtering can be seen in figure 2.5. For more complete practical implementation guide, see the research report on the subject by Ilonen, Kämäräinen and Kälviäinen [7]. As the guide recommends the filtering in the application is done in frequency domain as it is noticeably faster in face recognition. This is due to large enough face images resulting high feature dimensionality. Comparison between spatial and frequency domain efficiency can be found in [7, p. 23].

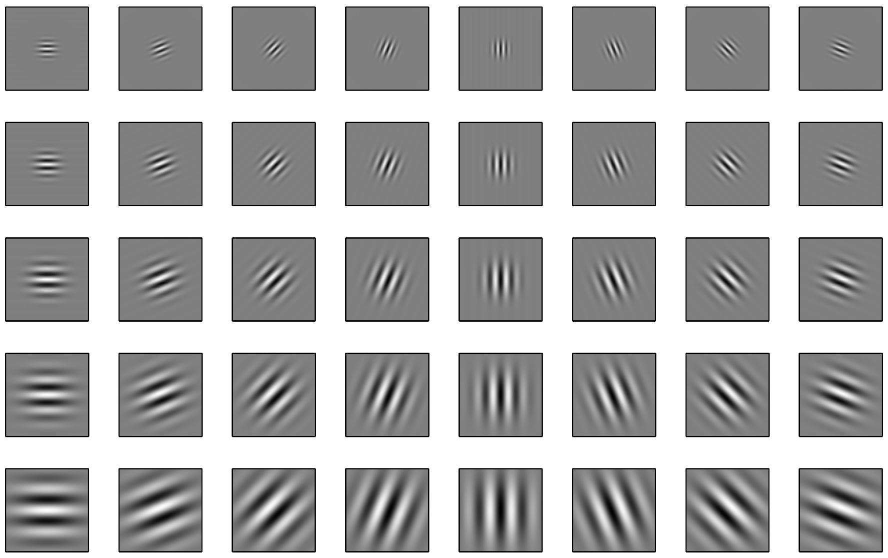


Figure 2.4. *Real parts of Gabor filter bank used later on in this thesis. Scale ν increases downwards, rotation μ increases from left to right. White indicates positive value, black negative. Generated using script A.1*

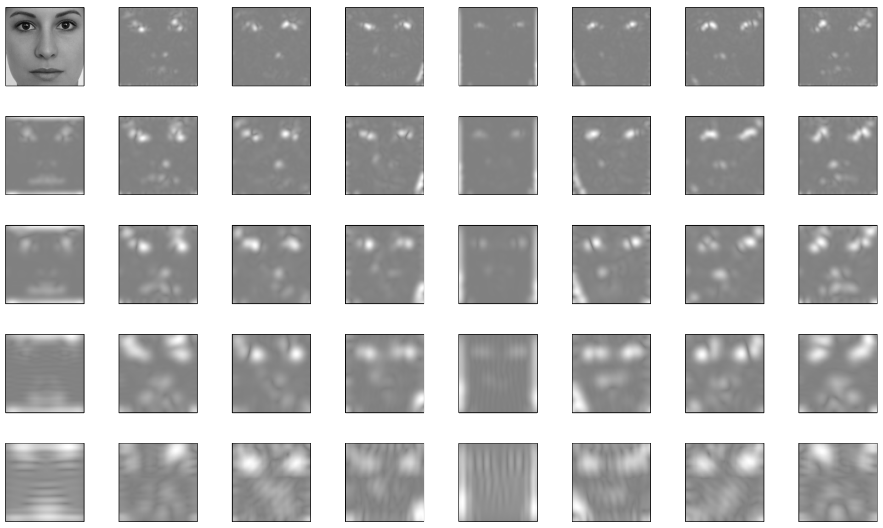


Figure 2.5. *Filtering results from script A.1 with first image replaced with the original face.*

3. METHODS OF MACHINE LEARNING

The goals of machine learning are achieved through learning classifiers. Classifier itself is a system which splits the feature space into segments corresponding to labels. When a set of features (*a pattern*) is fed into a classifier it computes a label for that pattern. If this label is externally found correct or incorrect the classifier then corrects itself accordingly, trying to minimize the error rate of the classifier. For example a classifier with input pattern {red light, clear way, engine running} might produce a label {accelerate}. The wanted result should most likely be {idle}, so after teaching this error to the classifier it would then adjust itself towards producing the correct label in the future.

Classifier system can usually be split into commonly found parts. Firstly the pattern to be recogniced must be sensed. This phase can be any kind of transducer, for example a microphone or like in our case, a camera. The resulting signal might then be segmented into parts depending on the goal of the classifier. For example a photo could be split into areas of interest and irrelevance based on skin color thresholding. Green areas are most likely not faces so they can be ignored.

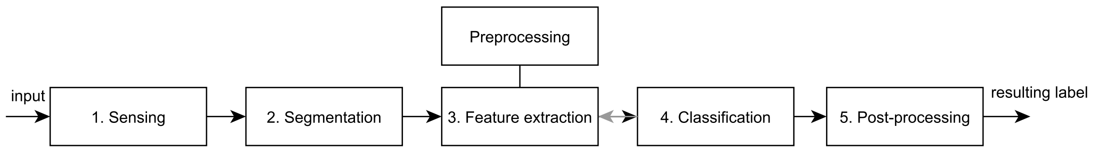


Figure 3.1. *Common stages of a pattern classifier*

The line between next two phases, feature extraction and classification, is often blurry. After the signal is segmented into *regions of interest* these areas are processed into recognizable features. This phase might include preprocessing like deformation of elongated pictures or sensing and reacting for occlusions covering parts of faces. In face recognition these sensed features are mostly spatial distances of facial features or statistically generated similarity measurements. Whatever the features are the feature extraction should be invariant to irrelevant transformations. In facial recognition the classifier system should find faces regardless of rotation or scale. In speech recognition the classifiers should label the same utterance with the same label regardless of signal amplitude and so on.

After the features are extracted and processed they can be then classified through different methods. The literal classification itself is usually the easiest part as it only evaluates the learned classification function for the pattern. Finally the classifier might do some post-processing, like discarding results with too low probability to be a good match or take advantage of contextual hints like not classifying two faces in the same photo with the same identity. In the example seen in figure 3.2 a classification system has been taught to separate the feature space into two parts. If a sample is measured to be on the left side of the dividing line it is deamed to belong in the class on that side of the line.

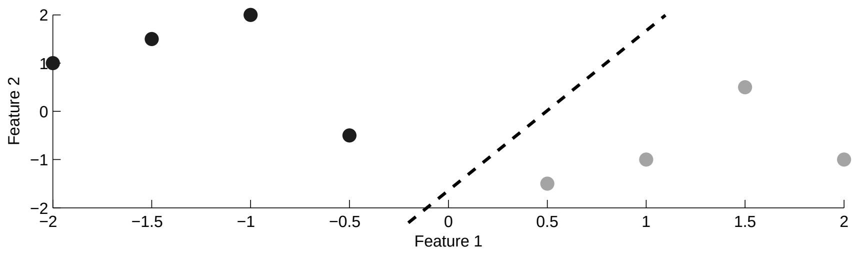


Figure 3.2. *A two feature linear classifier separating two categories (a dichotomizer)*

The simple linear classifier

The simplest general classifier possible is a two category dichotomizer like in figure 3.2. In order to split a feature space in two, we can define a weight vector $\boldsymbol{w}$, and a discriminant function $g(\cdot)$.

$$g(\boldsymbol{x}) = \boldsymbol{w}^t\boldsymbol{x} + w_0 = r\|\boldsymbol{w}\| \tag{3.1}$$

$$y = \begin{cases} \omega_1 & \text{if } g(\boldsymbol{x}) < 0 \\ \omega_2 & \text{if } g(\boldsymbol{x}) > 0 \end{cases} \tag{3.2}$$

In the simplest, linear and one dimensional case this vector will define a point splitting the feature space in two. In two dimensional case $\boldsymbol{w}$ will define a line and in general the $\boldsymbol{w}$ defines an *hyperplane.* The decision rule for this kind of classifier is the *sign-function* like in eq. 3.2. The equation can be also interpreted using $\boldsymbol{r}$, which is the distance of a sample from the closest point on the hyperplane.

In order to use this kind of categorization in multi-class cases, that is, cases where the answer can be from larger label set than two, the discriminant function $g(\cdot)$ can be generalized as

$$g(\boldsymbol{x}) = w_0 + \sum_{i=1}^{d} w_i x_i \tag{3.3}$$

where d is the number of dimensions in the feature space and w_i and x_i are components of weight vector $\boldsymbol{w}$ and input sample $\boldsymbol{x}$. To use this kind of *linear machine* [9, p. 218] to categorize a sample, a set of several discriminant functions need to be defined as :

$$g_i(\boldsymbol{x}) = \boldsymbol{w}_i^t \boldsymbol{x} + w_{i0}, i = 1, \dots, c \tag{3.4}$$

where c is the number of categories. Finding the correct category is done by comparing the pairwise categorizations and assigning $\boldsymbol{x}$ to ω_i when $g_i(\boldsymbol{x}) > g_j(\boldsymbol{x}), \forall i \neq j$.

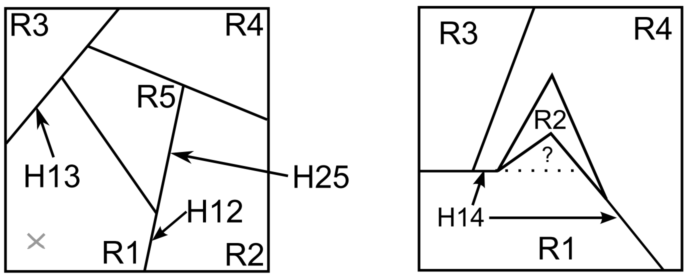


Figure 3.3. *Example of a possible (left figure) and impossible (right figure) two dimensional linear machine*

The H:s in figure 3.3 are the boundaries $g_i(\boldsymbol{x}) = g_j(\boldsymbol{x})$ between regions of R:s corresponding to labels ω_i. In many cases a linear machine like this is sufficient to solve a learning task, often resulting a decision tree for efficient repetition. This kind of classifier is however limited by the need to have only convex regions which can be separable but only singly connected. For example in right side of the figure 3.3 regions 1 and 4 have two separate connections which is impossible to express in one linear equation. The concave area of R2 leaves an unclassifiable area between R1 and R2. A way to overcome this limitation is discussed in section 3.2.

3.1 Boosting methods

Boosting methods are collection of pick-and-choose operations for selecting iteratively the most descriptive features from a set of all possible features, or the best categorizer from set of all possible categorizers and using them in parallel to produce better results than any of them could achieve alone. For the sake of brevity only the ADA-boost algorithm is discussed.

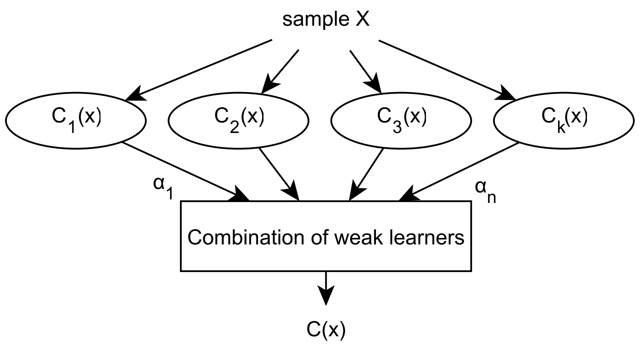


Figure 3.4. *The idea behind boosting methods*

Adaptive boosting, ADA-boost

Adaptive boosting (see [10]) is a technique to improve pattern recognition methods based on creating several cascading weak classifiers to form one strong classifier. In context of this thesis it is notable for being the basis for the methods used in Viola-Jones [11] face recognition.

In algorithm 3.1 describing the ADA-boost method of creating an cascade classifier, D is a teaching set of matching patterns and labels, x^i and y_i respectively. $W_k(i)$ is voting coefficient which is initialized to be equal between pattern-label pairs in D. Z_k is a normalizing constant, and $h_k(x^i)$ is the resulting label from classifier C_k with pattern x^i. As $W_k(i)$ is set to be equal in the beginning the first round of sample selection from D is random. The creation of a weak learner classifier on line 5 in algorithm 3.1 is done by first creating any kind of classifier (C_1) with classification error under $\frac{1}{2}$. In other words, any kind of classifier better than pure chance. In basic boosting the following classifiers are then created by picking new training sets from D by evenly choosing as many samples the earlier classifier fails and succeeds to classify correctly. This way there will be enough of an classifier overlap on areas harder to label. In other words, areas with apparent complexity will be given more attention. Compared to basic boosting, adaptive boosting adds per sample tracking of classification difficulty $W_k(i)$. The harder a sample is to classify, the more likely it is to be included into the next classifier to be created. As each classifier must have error rate smaller than $\frac{1}{2}$ the total classification error is guaranteed to decrease every iteration of the algorithm.

Algorithm 3.1 ADA-boost aided classifier creation algorithm [9]

```
1:  **begin initialize** $D = \{x^1, y_1, \dots, x^n, y_n\}, k_{max}, W_1(i) = \frac{1}{n}, i = 1, \dots, n$
2:  $k \leftarrow 0$
3:  **repeat**
4:      $k \leftarrow k + 1$
5:      train weak learner $C_k$ using $D$ sampled according to $W_k(i)$
6:      $E_k \leftarrow$ training error of $C_k$ measured on $D$ using $W_k(i)$
7:      $\alpha_k \leftarrow \frac{1}{2} ln[(1 - E_k)/E_k]$
8:      $W_{k+1}(i) \leftarrow \frac{W_k(i)}{Z_k} \times \begin{cases} e^{-\alpha_k} & \text{if } h_k(x^i) = y_i \text{ (correctly classified)} \\ e^{\alpha_k} & \text{if } h_k(x^i) \neq y_i \text{ (incorrectly classified)} \end{cases}$
9:  **until** $k = k_{max}$ **or** $E_k$ is small enough
10: **return** $C_k$ and $\alpha_k$ for $k = 1$ to $k_{max}$ (ensemble of classifiers with weights)
11: **end**
```

Algorithm 3.2 Creation of a weak learner-classifier [9]

```
1:  **begin initialize**
2:  $i \leftarrow 0, D_0 \in D$
3:  **repeat**
4:      $i \leftarrow i + 1$
5:      **while** selection of $s$ is possible **do**
6:          $r \leftarrow$ random value $N(0, 1)$
7:          **if** $r < \frac{1}{2}$ **then**
8:              find: $s \in D \wedge s \notin D_i$ so that $h_{i-1}(x^s) \neq y_s$
9:          **else**
10:             find: $s \in D \wedge s \notin D_i$ so that $h_{i-1}(x^s) = y_s$
11:         **end if**
12:         $D_i \leftarrow \{D_i, s\}$
13:         $s \notin D_{rest}$
14:     **end while**
15: **until**
16: **end**
```

The final classification (eq. 3.5) for any sample x is the weighted sum of all the classification results from simple classifiers C_k for that sample and correctness coefficients of those classifiers. In case of two class problem the simple and original way is to use labels $\{-1, 1\}$ and $Sgn(g(x))$ as the final classifier from eq. 3.5.

$$Sgn(x) = \begin{cases} -1 & x \leq 0 \\ 1 & x > 0 \end{cases} \qquad g(x) = \sum_{k=1}^{k_{max}} \alpha_k h_k(x) \tag{3.5}$$

3.2 Support Vector Machines

Support Vector Machines (SVM) are often used in machine learning problems as they give good results while being relatively simplistic. The original idea of optimal hyperplanes is from 1963 by Vladimir Vapnik, but the most commonly used soft-margin SVM [12] is from 1998.

The problem SVM solves when compared to linear machines is the ability to classify linearly non-separable cases. Firstly the dimensionality of feature space can be increased using nonlinear mapping $\Phi(\cdot)$ until the data becomes lineary separable. In this larger space different classes are then separated using any classifier. Commonly used example to visualize this kind of case is the XOR-classification problem. If we have samples $\boldsymbol{x} = \{(-1,1),(1,1),(-1,-1),(1,-1)\}$ and their corresponding labels $\boldsymbol{y} = \{\omega_1, \omega_2, \omega_2, \omega_1\}$ they cannot be separated using a linear classifier in their own space. In the XOR example the $\{x_1, x_2\}$ feature space is mapped into $\{\sqrt{2}x_1x_2, \sqrt{2}x_1\}$ space. In this space the optimal classifier is found to be $g(x_1, x_2) = x_1x_2 = 0$.

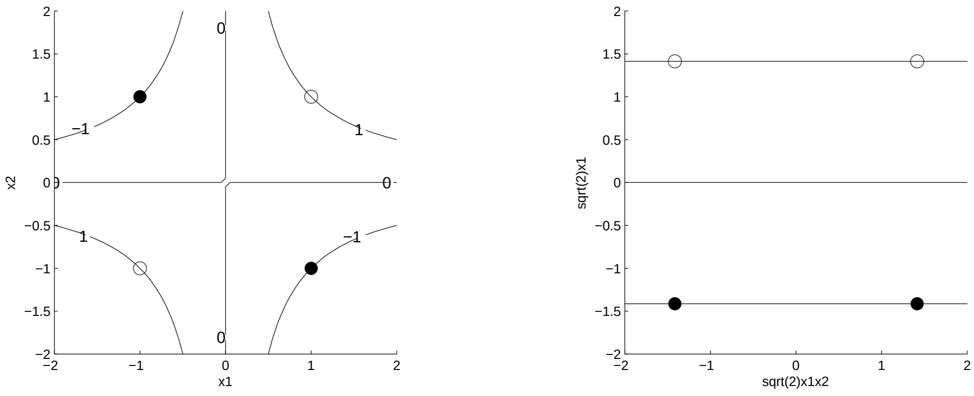


Figure 3.5. *SVM Solution for XOR classification [9]*

With SVM the underlying classification equation is similar to linear machines. If we have a mapping of patterns $\boldsymbol{x}_k$ to arbitrarely high dimension vector of $\boldsymbol{z}_k$ and labels $\boldsymbol{y}_k \in [\omega_1 = -1, \omega_2 = 1]$ then we can define $\boldsymbol{a}$ to be an augmented weights vector.

$$\boldsymbol{a} = \begin{bmatrix} w_0 \\ \boldsymbol{w} \end{bmatrix} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} 1 \\ \boldsymbol{x} \end{bmatrix} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix}, \quad \boldsymbol{z}_k = \Phi(\boldsymbol{x}_k) \tag{3.6}$$

By augmenting the sample vector $\boldsymbol{x}$ with a leading 1 the discriminant function is

$$g(\boldsymbol{x}) = \boldsymbol{a}^t\boldsymbol{z} = \sum_{i=0}^{d} \boldsymbol{w}_i\boldsymbol{z}_i$$

Unlike in linear machines, the goal of SVM is not to find a vector that minimizes any kind of error function, but to find *support vectors* which lie as close to class borders as possible. The best categorizer is then between these vectors. Like in equation 3.1 the distance from any sample to any hyperplane $\boldsymbol{w}$ is $r = \frac{|g(z_k)|}{||w||}$. Therefore the goal of finding the best separating hyperplane is done by maximising the function in eq. 3.7.

$$\frac{\boldsymbol{y}_k g(\boldsymbol{z}_k)}{||\boldsymbol{w}||} \geq b, k = 1, \ldots, n \quad [9, \text{chapter } 5, eq.106] \tag{3.7}$$

where b is a margin between two classes. Additionally by requiring that $||\boldsymbol{a}||b = 1$ the resulting hyperplanes will be unique. Because the labels are defined as ($\boldsymbol{y}_k \in [-1, 1]$) the margins can be described by

$$\boldsymbol{w} \cdot \boldsymbol{x} - b = -1 \quad \text{for } \omega_1$$

$$\boldsymbol{w} \cdot \boldsymbol{x} - b = 1 \quad \text{for } \omega_2$$

which then lead to demads for keeping the marginal clean of any samples.

$$\boldsymbol{w} \cdot \boldsymbol{x} - b \leq -1 \quad \text{for } \omega_1$$

$$\boldsymbol{w} \cdot \boldsymbol{x} - b \geq 1 \quad \text{for } \omega_2$$

$$\rightarrow y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i - b) \geq 1 \quad \text{for all } 1 \leq i \leq n \tag{3.8}$$

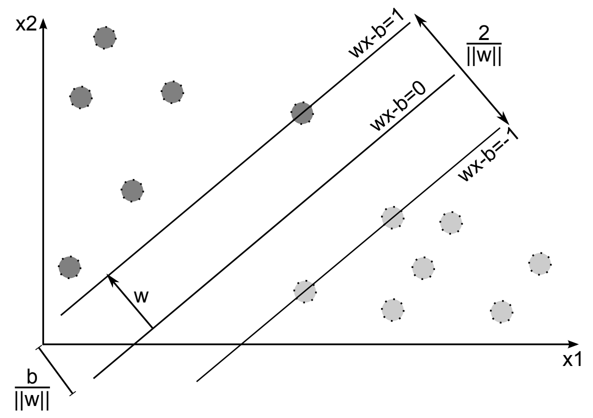


Figure 3.6. *SVM support vectors and classification hyperplane*

Training and use of SVMs

SVMs can be trained similarly to linear machines by using the currently worst pattern to update the weights, instead of any randomly selected pattern. This is however impractically slow excluding the most basic cases. More effective way to train a SVM is to use Lagrange undetermined multipliers ($\boldsymbol{\alpha}_k$). Like shown in [9, chapter 5, eq. 108] the minimization of $||a||$ can be done by minimizing the function with respect to $\boldsymbol{a}$. This can be reformulated into maximization of equation 3.10. Both of these are so called *quadratic programming* problems where linear requirements limit the optimization.

$$L(\boldsymbol{a}, \boldsymbol{\alpha}) = \frac{1}{2}||\boldsymbol{a}||^2 - \sum_{k=1}^{n} \alpha_k [z_k \boldsymbol{y}_k \boldsymbol{a}^t - 1] \tag{3.9}$$

$$L(\boldsymbol{\alpha}) = \sum_{k=1}^{n} \alpha_k - \frac{1}{2}\sum_{k,j}^{n} \alpha_k \alpha_j \boldsymbol{y}_k \boldsymbol{y}_j \boldsymbol{z}_j^t \boldsymbol{z}_k \quad \text{[9, eq. 109]} \tag{3.10}$$

When applying these with constraints of $\sum_{k=1}^{n} \boldsymbol{y}_k \boldsymbol{\alpha}_k = 0, \quad \boldsymbol{\alpha}_k \geq 0, \quad k = 1, \ldots, n$. The Karush-Kuhn-Tucker rule [21] states that the resulting classifier must be a linear combination of all the training samples with $\alpha_k = 0$ in most indices.

$$\boldsymbol{a} = \sum_{k=1}^{n} \alpha_k \boldsymbol{y}_k \boldsymbol{z}_k$$

Those indices which have $\alpha_k > 0$ are the support vector candidates as they satisfy the equation 3.8. Finding the values for these Lagrangian multipliers is processing power consuming task and methods to improve this search are being researched. Commonly used process to do this search is the sequential minimal optimization-method described by Platt [13]. After the suitable support vectors are found (if any are found) the final classifier can be calculated by finding the correct translation from origin to the resulting vector from equation 3.8. The notation * signifies the found values for this training set.

$$b^* = -\frac{1}{2}(\max_{i:y_i=-1} \boldsymbol{a}^{t*}\boldsymbol{x}_i + \min_{i:y_i=1} \boldsymbol{a}^{t*}\boldsymbol{x}_i) \tag{3.11}$$

To classify a new sample based on this training, only a sign-function is needed for the hyperplane-equation. Note that the inner product of the new sample and the training set samples needs to be calculated only if $\alpha_i > 0$.

$$\boldsymbol{w}^{t*}\boldsymbol{x} + b^* = (\sum_{i=1}^{n} \alpha_i \boldsymbol{y}_i \boldsymbol{x}_i)^t x + b^* = \sum_{i=1}^{n} \alpha_i \boldsymbol{y}_i \langle \boldsymbol{x}_i, \boldsymbol{x} \rangle + b^* \tag{3.12}$$

4. FACE RECOGNITION FROM PHOTOS

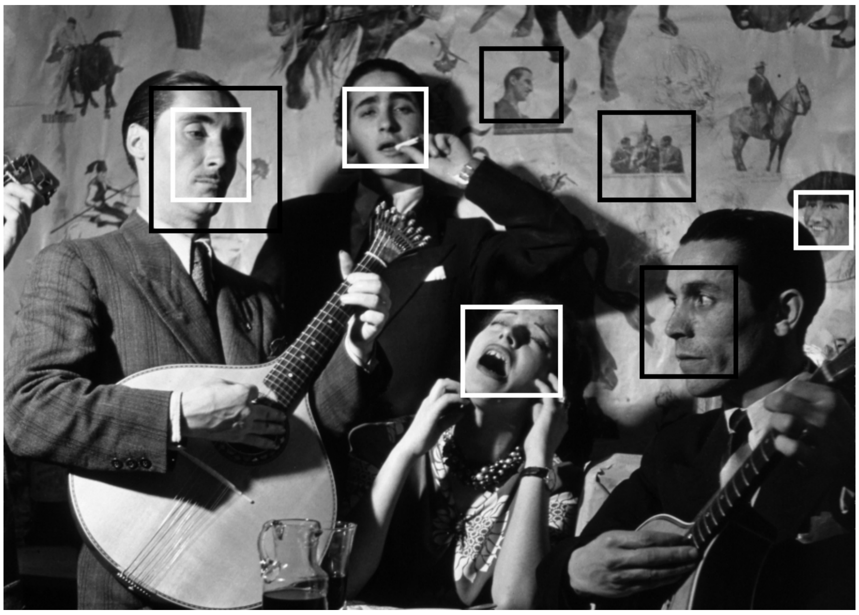


Figure 4.1. *Examples of positive and negative face matches [14]*

The underlying process of recognizing faces is similar to any machine learning problem. Using any method a set of features are extracted from a photo or video frame. These features are compared against a learned model where the categories are “these features represent a face”, or “these features do not represent face.” In the example photo 4.1 the white windows are the supposed face matches and black windows are failed matches. The areas in this photo were handpicked to demonstrate few key problems in face recognition as reasons for a face to be unmatchable are numerous. Even perfectly tuned system will miss obvious face matches from a human perspective due to practical limitations in artificial vision systems. As mentioned earlier in chapter 3, a face recognition system should be immune to scale and rotation of faces. In practice this is a hard task and often the ranges of these properties are limited. Several sensing problems can be solved by preprocessing but in some cases, for example, a face might simply have too low dynamic range for a classifier to find a match.

4.1 Pre- and post–processing

The process of face searching can be enhanced by limiting the search space only to areas with possible faces. One way to eliminate areas with no possible face areas is by thresholding the image with skin color value ranges. In his paper on using chrominance information characteristics Lin [15] suggests using the procedure described in algorithm 4.1. Thresholding limits were the result of manually picking face areas and clustering those samples in HSI-color space using standard Gaussian model. (see eq. 4.1)

$$\begin{aligned} &\text{x = sample in (H,S)-subspace} \\ \bar{u} &= \frac{1}{n}\sum_{i=1}^{n} x_i, \quad C = (x-\bar{u})(x-u)^T \\ G(x) &= \frac{1}{(2\pi)^{n/2}|C|^{1/2}} e^{-\frac{1}{2}(x-\bar{u})^T C^{-1}(x-\bar{u})} \end{aligned} \tag{4.1}$$

Algorithm 4.1 Skin color masking by [15]

```
1: begin initialize $I$=Input image
2: Apply gamma correction to $I$ so that $\approx 3\%$ of pixels can be quantizised as
   “reference black” and $\approx 5\%$ can be quantized as “reference white”
3: $R_{hs} = \begin{cases} H & \in (0,16) \cup (5.6, 2\pi) \\ S & \in (0.01, 0, 95) \end{cases}$
4: Pixel $x$ is interesting if $(x \in R_{hs}) \cap (G(x) > 0.75)$
5: return Masked image $I$
```

Another model which was finally used in this work for skin color thresholding was proposed by Kovac, Peer and Solina [16]. In their model the skin color is thresholded and processed by algorithm 4.2.

Algorithm 4.2 Skin color masking by [16]

```
1: begin initialize $I$=Input image
2: Threshold the image to interesting and irrelevant pixels by
   $$\text{Pixel is interesting if} \begin{cases} \{R > 95, G > 40, B > 20\} \\ \max\{R,G,B\} - \min\{R,G,B\} > 15 \\ |R-G| > 15 \wedge R > G \wedge R > B \end{cases}$$
3: return Masked image $I$
```

In practice this kind of preprocessing wasn't that useful as it increased both processing time and the chance of cutting off areas with faces. However the secondary use for this thresholding was to reduce the number of false face matches by counting the number of skin colored pixels in every potential face match and discarding matches with too low ratio of pixels per face area. Experimentally this ratio was set to 0.4 so only a rare correct match was removed falsely while improving the overall accuracy of the application.

Local normalization

Uneven lighting conditions is a common source for problems when the goal is to extract features for recognition. One way to tackle this problem is local normalization. In [17] Xie and Lam describe a method to effectively remove the local lighting variation while still retaining statistical properties of the image.

$$f_p(x,y) = \frac{f(x,y) - E(f(x,y))}{Var(f(x,y))}, (x,y) \in W \tag{4.2}$$

Here the W is the window under consideration, $E(f(x,y))$ is the local intensity mean value and $Var(f(x,y))$ is the local variance. In this step the low frequency components are seen as part of the mean, and the interesting high frequency components are part of the variance. The result is therefore free of globally varying effect of lighting levels but retains and normalizes the edges and other possible identifying features. The whole post processing chain can be seen in figure 4.2.

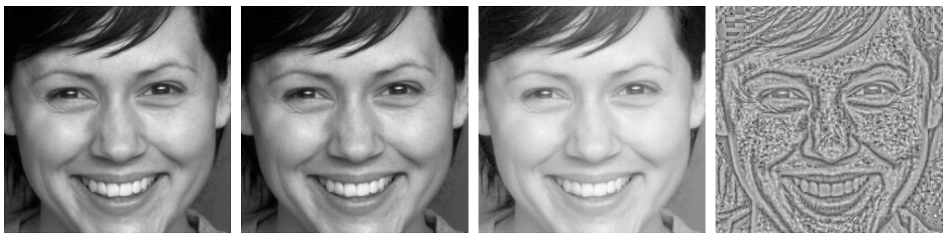


Figure 4.2. *Example of post processing steps using FACES [18] model 140. From left to right: Original in black and white - cropped, histogram equalization, gamma correction with* $\gamma = 0.5$*, local normalization using eq.4.2, window size* 7×7

This kind of processing was applied to recognized faces before classification steps in attempt to increase recognition rates. The results weren't positive, most likely due to both training sets and testing photos being high quality with only little to gain from this kind of processing.

4.2 Viola-Jones method

Viola-Jones is the current *de-facto* face recognition system and commonly used baseline to compare the effectiveness of a new system. The original definition [11] is from early 2000's and describes an efficient method of face matching while being computationally light when compared to earlier works. The main improvements to achieve this result were the replacement of image pyramids with *integral images* and using ADA-boost-like categorizer for fast negative match rejection.

Features

The common method of picture pyramids in image processing creates a set of down sampled copies of the original photo. By creating a set of different sized photos to match against the matching itself can be simplified to fixed size targets. Downside is the computational need for such operation. In Viola-Jones method this preprocessing phase was made redundant by selecting a feature set which can be easily calculated from integral images.

$$ii(x,y) = \sum_{x' \leq x, y' \leq y} i(x', y') \tag{4.3}$$

$$s(x,y) = s(x, y-1) + i(x,y)$$

$$ii(x,y) = ii(x-1, y) + s(x,y) \tag{4.4}$$

The two descriptions seen in equations 4.3 and 4.4 calculate the integral image from grayscale image i. As it can be seen this colour conversion and calculation can be done while loading an image in common, top down, left to right fashion. On the first glance this kind of image seems to have no real use. The usefulness becomes aparent however when it is used to calculate *Haar-like* features from the image.

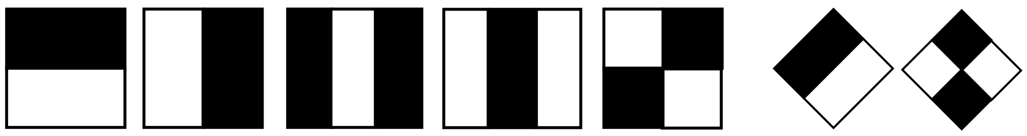


Figure 4.3. *Haar-like feature masks*

A Haar-like feature is a simple mask for calculating a sum of pixel values. In figure 4.3 are some very basic feature masks used in Viola-Jones. Black areas represent mask to be counted as addition and white areas as subtraction. The sum of these areas is a feature to be used to match face shapes. The two diagonal masks can be used just like any square based construction, but they require a second integral image which needs to be built diagonally.

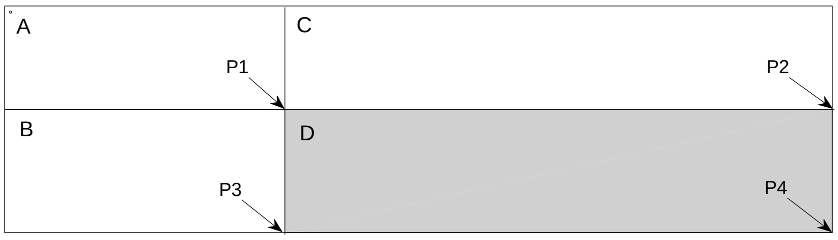


Figure 4.4. *Calculating area values from an integral image [11]*

If we want to know the sum of all pixel values in area A with upper left corner at $(0,0)$ all we need to do is read the value of the integral image at point $P1$. The area value of B is $ii(P3) - ii(P1)$, C is $ii(P2) - ii(P1)$ and finally D, which can be used to calculate any square in an integral image is $ii(P4) - ii(P3) - ii(P2) + ii(P1)$ as A is removed twice with B and C. With these kind of features the feature extraction is then a matter of few table lookups and sums per mask square.

Classifier training

Training a basic Viola-Jones classifier is done by first creating a positive set of face images in fixed window size and similar sized set of random, non face images. The original paper suggested using 24×24 pixel window as it performed reasonably while being computationally frugal. Inside a 24×24 window there can be more than 160000 ways to select a rectangular feature location, so the training process is aided by ADA-boost-like feature selection process. (See chapter 3.1) Even with boosting the learning phase is time consuming process as the candidate features need to be matched against every face and non face sample to calculate the error rate. With a reasonable amount of thousands of training samples the total number of possible features to be evaluated rises to hundreds of millions.

In the feature selection process the aim is to pick features which can categorize a sample with error rate as low as possible. As Haar-like features produce a single integer as feature value the evaluation of a feature is simply

$$h(x,f,p,\theta) = \begin{cases} 1 & pf(x) < p\theta \\ 0 & otherwise \end{cases} \tag{4.5}$$

If feature f is applied to window x the weak classifier h produces a binary classification depending on the value being either above or under a threshold value θ. The actual parameter to be learned is then the threshold value witch splits the value range of f best to separate faces and not faces. p is polarity function to choose the direction of inequality. As the goal is to create a cascade classifier the error rate of a single feature doesn't need to particularly low. In case of these features even the best classifier might have error rate as high as 0.3. The method to rank features by their classification capability is described in algorithm 4.3.

Algorithm 4.4 will produce a chain of classifiers with still relatively high error rates per layer. If the false positive rate of a single stage is f_i and the detection rate of a single stage d_i then the false positive rate and total detection rate of the entire cascade with K stages is

$$F = \prod_{i=1}^{K} f_i \qquad D = \prod_{i=1}^{K} d_i \tag{4.6}$$

Therefore to achieve a total detection rate of 0.9 in a 10 stage cascade a single stage classifier must have a detection rate of 0.99. Although this is a high rate to achieve the flip side is that the false positive rate of a single stage needs to be only 0.3. In a 10-stage cascade this would lead only to $0.3^{10} \approx 6 \times 10^{-6}$ rate of total false positive matches.

Face search

In order to scan faces from a photo a sliding window is moved across the image starting with desired size. From each window a set of features from the first stage of the cascade classifier is evaluated against the learned model. In the first stage are the strongest classifiers, so by evaluating only few features the obviously bad matches are discarded immediately. If the sample passes the threshold value of the first stage it is then passed to second stage and so on. Face match is found if the entire cascade agrees the sample represents a face.

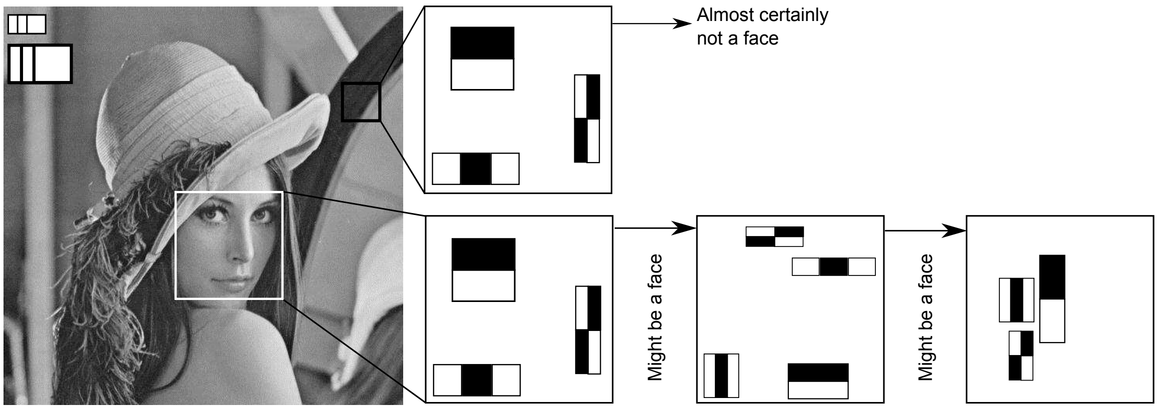


Figure 4.5. *Face search from an image using Viola-Jones method*

After the image has been scanned the sampling window can be scaled to search different sized faces by simply multiplying every feature coordinate with a desirable value. The original paper used geometric scaling of 1.25 and window step size of

1.5 pixels. Because testing face matches is done by sampling the integral image the scale of the sampling window doesn't affect the speed of testing at all. Compared to image pyramid methods this results at least 15 times faster processing.

4.3 Local binary pattern based method

As discussed in section 2.2, the local binary pattern-descriptor is a low impact method to extract texture information from an image. LBP require only a miniscule amount of processing power when compared to methods like image pyramids and yet give surprisingly good results in describing local image properties. Another benefit in image wide LBP calculation is the need to do only a single pass over the search area just like in integral images, but the resulting image doesen't need as much as 64 bits per pixel. The overall training is done similarly than in Viola-Jones, only using uniform LBP histogram features instead of Haar-like features.

In the application the user has the choice to use LBP detector instead of the Viola-Jones detector, but it does have a higher miss rate using the given default classifier data and might thus skip finding existing faces from group photos.

Algorithm 4.3 Selecting the best features by evaluating a feature against every training sample. (T total samples) [11]

```
1: begin initialize
2: $P = (x_1, y_1), \ldots, (x_n, y_n)$ where $x_n$ are the image samples and $y_n = [0, 1]$ for
   negative and positive training samples (faces and not faces) repectively
3: $w_{1,i} = \begin{cases} \frac{1}{2m} & y_i = 0 \\ \frac{1}{2l} & y_i = 1 \end{cases}$ where m,l are the number of positive,negative samples.
4: for $t = 1 \to T$ do
5:     Normalize weights, $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$
6:     Select the best weak calassifier $h_t$ with respect to the weighted error $\epsilon_t$
       $\epsilon_t = min_{f,p,\theta} \sum_i w_i |h(x_i, f, p, \theta) - y_i|$
       $h_t = h(x, f_t, p_t, \theta_t)$ where $f_t, p_t$ and $\theta_t$ are the minimizers of $\epsilon_t$.
7:     Update the weights: $w_{t+1,i} = w_{t_i} \beta_t^{1-e_i}$
       where $e_i = 0$ if $x_i$ is classified correctly, $e_i = 1$ otherwise.
       $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$
8: end for
9: return The final strong classifier:
   $C(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$
   where $\alpha_t = log \frac{1}{\beta_t}$
10: end
```

Algorithm 4.4 Training an ADA-boost-like cascade detector [11]

```
1: begin initialize
2: $f =$ maximal acceptable false positive rate
3: $d =$ minimum acceptable detection rate per layer.
4: $F_{target} =$ target total false positive rate
5: $P =$ set of face images, $N =$ set of not face images, $F_0 = 1.0, D_0 = 1.0, i = 0$
6: while $F_i > F_{target}$ do
7:     $i \leftarrow i + 1$
8:     $n_i = 0; F_i = F_{i-1}$
9:     while $F_i > f \times F_{i-1}$ do
10:        $n_i \leftarrow n_i + 1$
11:        Train a classifier with $n_i$ features using $P$ and $N$ with ADAboost.
12:        Evaluate current cascaded classifier on validation set to determine $F_i$ and
           $D_i$.
13:        Decrease threshold for the $i$th classifier until ht ecurrent cascaded classifier
           has a detection rate of at least $d \times D_{i-1}$ (this also affects $F_i$)
14:    end while
15:    $N \leftarrow \emptyset$
16:    If $F_i > F_{target}$ then evalueate the current cascaded detector on the set of
       non-face images and put any false detections into the set N
17: end while
18: return Cascade classifier chain
19: end
```

5. IDENTIFICATION OF FACES AND FACIAL PROPERTIES

After a face is found from a photo, a number of methods can be used to extract features for further categorization. The most common task is to try to identify the target in the picture from a database of previously collected feature sets. Other tasks considered in this thesis are the extraction of facial expression, gender and age range.

The main difference from algorithmic view is that finding a face is a two category problem – search window either contains or does not contain a face. When trying to match a pattern against multiple possibilities the task turns into a probabilistic multi class matching problem. This task can be divided again into several common steps.

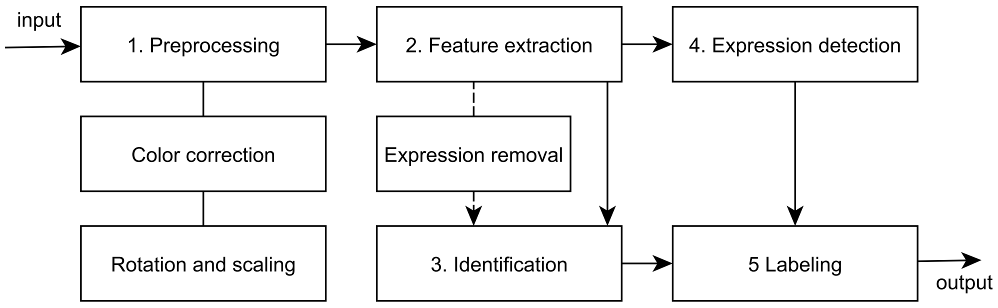


Figure 5.1. *Common stages of face identification and labeling*

In controlled conditions where it is possible to take an ideal photo system can be trained to attempt to remove any expression the target might have. By having only neutral samples to compare the recognition system can be more accurate, provided that the removal is successful. In this thesis the expressions are only detected and used to label faces as samples to be identified are picked from group photos.

5.1 Methods of face identification

Several methods have been developed during last few decades to gather biometric data by measurements made on images. One of the more obvious methods is the distance measurements between *salient* facial feature points like mouth edges and eye socket dimensions. These kind of measurements are relatively simple to extract in ideal conditions but won't necessarily survive many real life photography hindrances like occlusion or bad pose. In this thesis I used statistics based methods as the targets are from "real life" situations. Specifically I used the "Eigenfaces" method described in subsection 5.1.1 and the Gabor-volume local binary patterns method described in subsection 5.1.2.

5.1.1 Eigenfaces

The staple of face recognition and one of the earliest practical methods, "Eigenfaces" [19] uses PCA-analysis to produce an average image of all available sample photos and a set of most differentiating photo mixtures. In figure 5.5 is an example of "Eigenface" classifier trained with 25 different face photos. On the left is the mean photo and then four difference faces given by PCA-analysis.

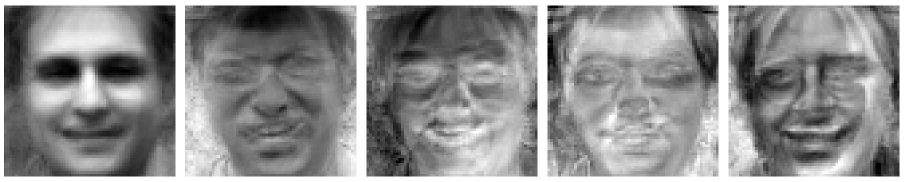


Figure 5.2. *Example of "Eigenfaces" components.*

Any of the original photos can be reconstructed, to a degree, by combining these photos. Accuracy and processing power requirements can be adjusted by choosing the number of mixture photos created and used. Every identity can be expressed, per picture, as a vector of coefficients for the difference photos. For example, start with the mean photo, add 10% of the intensity values of difference photo 1, -20% from the second and so on. The identity vector for that photo is then, for example $[0.1, -0.2, 0.0, 0.0]$. When matching a new identity against pre-calculated vectors the sample is projected into this models PCA-space and classified using any suitable distance based method, for example an SVM or a simple pairwise comparison. If the distance between the best match and the sample is small enough the face is considered to be an identity match.

"Eigenfaces" produces usable results but it does have several downsides. In order to get a good match the lighting, pose and expression need to be near ideal as the matching is against fixed pixel positions. Any movement will lead to a negative distance value shift in both old and new location.

5.1.2 Gabor volume local binary patterns, GV-LBP

Gabor Volume-LBP by Lei, Liao and Pietikäinen [20] uses images to create a 3D volumes using Gabor filters. This volume is then processed with specialized LBP operator to produce a classifiable pattern. As the pattern is result of an LBP-thresholding the result is more robust against lighting conditions than most methods.

In this method an identity vector for a face image is calculated by first processing it with a Gabor filter bank (see section 2.3) with orientation range (μ) from 0 to 7 and scale range (ν) from 0 to 4. To the resulting 3D volume a LBP is applied per every plane, i.e. a set of 2D LBPs are calculated by "viewing" and summing up the cube from orthogonal base directions ($j = 0 : XY; 1 : XT; 2 : YT$). This will produce a representation of a face f_j by taking sub image histograms from each of the slices.

$$H_j(l) = \sum_{x,y} I(f_j(x, y) = l), \quad l = 0, 1, \ldots, L_j - 1 \tag{5.1}$$

in which $I(\cdot) \in \{0, 1\}$ is the binary value of f_j. L_j is the last LBP code on that plane. Finally the resulting three histograms are concatenated into a singe feature vector $H = [H_1, H_2, H_3]$. To improve computational efficiency the same result can be achieved by scanning through the volume with a 3 dimensional LBP.

$$GVLBP(I_c) = \sum_{p=0}^{7} 2^P S(I_p - I_c) \tag{5.2}$$

$$S(I_p - I_c) = \begin{cases} 1 & I_p - I_c \geq 0 \\ 0 & I_p - I_c < 0 \end{cases} \tag{5.3}$$

I_c is the pixel under test, I_p is the pixels around I_c so that I_0 and I_4 are the neighboring pixels in orientation plane, I_2 and I_6 are the neighbors in scale plane, while the rest are neighbors in spatial plane. (See figure 5.3) On edge cases, for example when $\mu = 0$, the code for I_0 is set to 0.

The histogram is built by splitting the image into 64 non-overlapping areas and adding up values from all orientation and scale pictures. As there are 8 orientations and 5 scales the resulting single sub-image histogram is a sum of 40 histograms. Finally the histograms are multiplied by preset weight and normalized to emphasize the importance of eyes and mouth area in identifying people. After the full histogram has been acquired the best face match is found by calculating the best correlation from known samples.

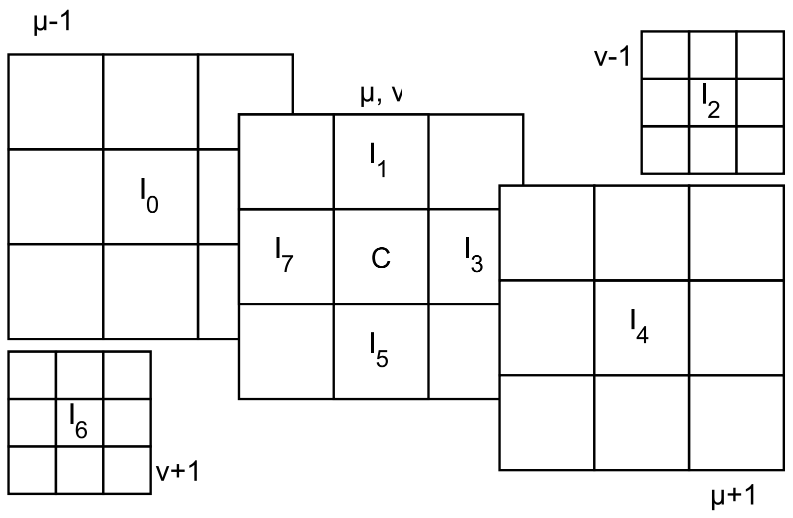


Figure 5.3. *Pixel neighbour selection for 3D LBP operation.*

5.2 Facial properties recognition

In order to classify photographs several methods were used to calculate further classifications for faces. Each face is classified by gender, age range and expression. Gender as a binary choice between man and a woman, age range is split into three categories and expression into six categories by convention used in FACES database [18]. These properties are used later on to match group similarities in group photos.

5.2.1 Gender classification

Gender classification was implemented using three methods which were considered after review of recent papers on the subject. Baseline choice was the well known and commonly used "Fisherfaces"-method which uses LDA [24] to separate classes and is suitable for gender classification as there is only two clearly separable classes created from a large dataset. LDA is similar to PCA, but it has added artificial class separation when projecting to model space.

The second method chosen was SVM based classifier trained on uniform LBP patterns [27]. The idea behind uniform LBP patterns is to use only the patterns which match gradients in the picture. This is done by selecting only LBP codes which have maximum of two binary transitions in them. Some details are lost, but these details are irrelevant in gender classification. By reducing the number of LBP codes to only 58 *cyclic* uniform patterns in an 8-bit code the histogram used to train and classify genders is only one fourth of the original length. Cyclic means here treating the binary pattern as if the first and last bits were neighboring each other. This way a pattern like in the center of figure 5.4 is clearly uniform. Any LBP code which doesn't match an uniform pattern is discarded.

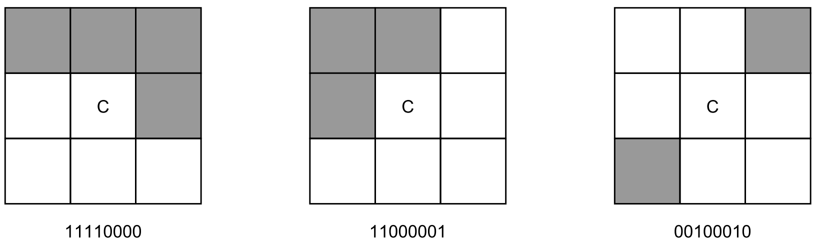


Figure 5.4. *From left, two uniform LBP patterns and a non-uniform LBP pattern*

The third method, which was found to be the best using FACES training data is by A. R. Ardakany and A. M. Joula [28]. In their method a photo is transformed into an edge map by convoluting it with simple edge filters $\{-1, 0, 1\}$ and its transpose into horizontal (d_y) and vertical (d_x) edge maps. These maps are then combined into full magnitude and direction maps by eq. 5.4 and 5.5. After quantizing the results by M for magnitudes and by N for directions these maps are combined into a single per pixel value of $(m-1) \times N + \theta + 1$. The final feature vector is a histogram concatenated from sub-images just like in LBP methods. The best values for these parameters were picked from the article as $M = 10, N = 8$ and 8×8 sub-images for the histogram building.

$$m = \sqrt{d_x^2 + d_y^2} \tag{5.4}$$

$$\theta = \tan^{-1}\left(\frac{d_y}{d_x}\right) \tag{5.5}$$

5.2.2 Expression classification

For expression classification I used method suggested by Naika, Das and Nair [4]. In their method a variable sized LBP operator is used to generate a histogram sequence which in turn is used to train an SVM. The LBP operator in question is the AR-LBP shown in section 2.2. In AR-LBP the operator works like in normal LBP but compares neighboring area averages instead of neighboring pixel values. The averaging area size of the operator was set to three pixels high times fifteen pixels wide as it was the smallest size still producing the best possible classification performance. (95.71% in the original paper)

5.2.3 Age classification

The age classification was implemented based on method by Ylioinas, Hadid and Pietikäinen [30]. Like other methods in this thesis this is a LBP-histogram based method using the elliptical LBP variant. ELBP is a variant of LBP where the kernel is defined by two semi-axes, division count and a possible phase addition.

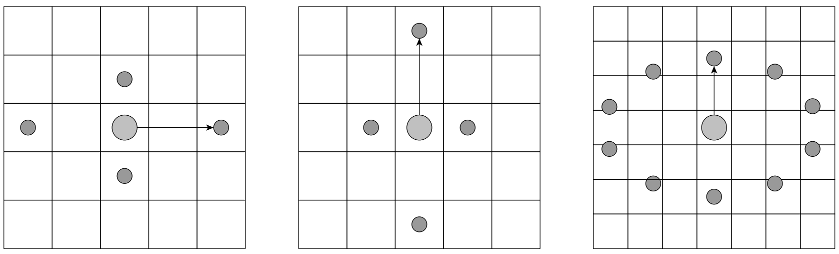


Figure 5.5. *ELBP kernels. From left to right (horizontal distance, vertical distance, phase, divisions): (2,1,0,4), (1,2,$\frac{\pi}{2}$,4) and the kernel used to classify ages (3,2,$\frac{\pi}{2}$,10). Arrow points to first pixel in LBP code, moving counter-clockwise.*

The generated histogram is a concatenation of two ELBP operations. First part is the normal LBP histogram and the second, known as completed LBP magnitude (eq. 5.6) is calculated from pixel magnitude differences. This operation defined in [29] is similar to normal LBP but instead of comparing pixel intensity values the comparison is between mean of local absolute differences (m_p) and the global mean (c) of these differences. This process is visualized in figure 5.6. After concatenating these two histograms the result is normalized before using it to teach a SVM.

$$CLBPM_{P,R} = \sum_{p=0}^{P-1} t(m_p, c)2^p \tag{5.6}$$

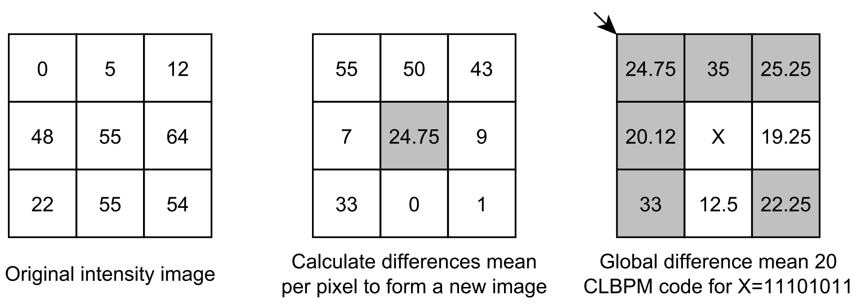


Figure 5.6. *The process to calculate CLBPM images*

6. SPATIAL AND SOCIAL ARRANGEMENTS OF PEOPLE IN PHOTOS

The final classification task in this thesis is the attempt to categorize photos by their spatial and socialy relevant content into general categories and to find photos closely resembling each others. To do this I chose two methods. For general situational classification I chose the method described in [31]. In this method described by Shu, Gallagher, H. Chen and T. Chen the information gathered from earlier recognition tasks is combined with spatial distances between found faces. The goal in this method is to find similarity values between every photo pair and then cluster the photos into groups representing different classes based on those distances. Like in the original paper the simplest method of nearest neighbor was used as it is readily available as part of OpenCV.

The calculation of these pairwise similarities is done by first forming a fully connected graph between the two photos under inspection. For photos I_1 and I_2 containing set of faces x_i and x_j the edge weights in the graph are calculated by

$$w_{i,j} = \alpha \, ||x_i - x_j|| + \sum_l \beta_l h_l(a_l(i), a_l(j)) \tag{6.1}$$

which simply compares the properties (a) of each face against each other and applies a weight function (h) to them for tuning purposes. In this case the used properties are gender and age. The weights α and β are adjustable by the user.

To account for photos taken from different distances the in–photo distances between the faces are normalized and mean corrected to match each other. The edge matrix $w_{i,j}$ is then processed with the *Hungarian Algorithm* [32, 33] in order to pick the smallest assignment to match every face in the photo with fewest faces. As this method favors photo pairs with large difference in face count a penalty is added as a final step to the selected weights. This penalty weight γ is also adjustable by the user. The final distance is then

$$d(I_1, I_2) \equiv w^*_{I_1,I_2} + \gamma \, ||I_1 - I_2||_{faces} \tag{6.2}$$

The w^* in eq. 6.2 are the edge values picked by the Hungarian algorithm. By calculating these distance values between all photographs the most similar photos can be found by simply sorting by this distance. By using these distances to train a K-nearest neighbors classifier a photograph can be then assigned to general categories like wedding photos or friend gatherings by calculating distances to training photos. This method doesn't however take account for the structure of the group.

For the task of finding best possible group photo matches I used a very recent method to describe groups by Y. Chiu, C. Li, C. Huan, P. Chung and T. Chen in their paper "Efficient graph based spatial face context representation and matching" [34]. In their method the faces from a group photo are arranged to an *Urquhart graph.* This graph is calculated by forming a Delaunay triangulation T_c by using the face locations as vertices and then removing the longest edge from each triangle. Spesificly the Urquhart graph is defined as

$$\{A_{ij}^U = 1 \mid ||v_i - v_j|| < \max\{||v_i - v_k||\,, ||v_j - v_k||\} \& \{v_i, v_j, v_k\} \in T_c\} \tag{6.3}$$

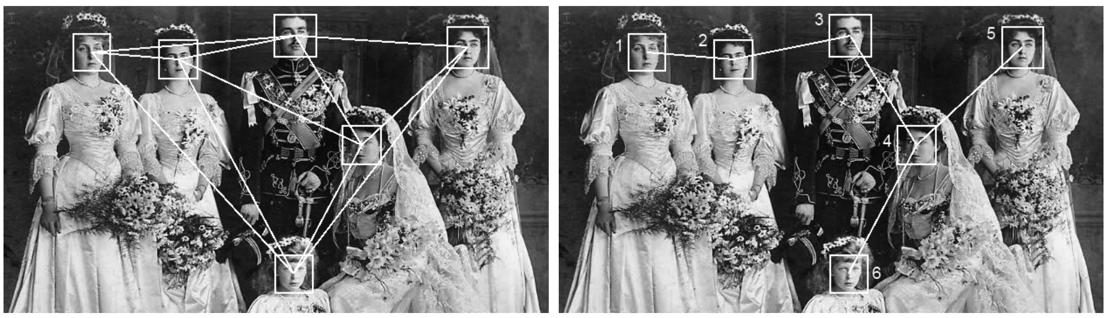


Figure 6.1. *Full Delaunay graph applied to face matches and matching Urquhart map. Photo taken from [35]*

After finding the Urquhart graph the edges of it can be used to form the adjacency matrix A for the graph. An undirected connection between face i and j is made by setting $A_{i,j} = A_{j,i} = 1$. Equation 6.4 is an example matching the photograph in fig. 6.1. This matrix thus defines the structure of the graph. When extracting this graph the selection of indexing can not be guaranteed to match the locations in the photos. To match the order of vertices a *permutation matrix* needs to be found.

A permutation matrix is a square matrix, matching the size of the adjacency matrix with column and row sums of 1. An example of an permutation matrix can be found in eq. 6.4. With P any suitably sized adjacency matrix can be rearanged to match another by matrix multiplication, $A' = PAP^T$.

$$A_{i,j} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \qquad P_{i,j} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{6.4}$$

The most straightforward method to find a best structural match is by Umeyama [36]. In his method two graphs are matched by eq. 6.6. The absolute values of eigenvectors of adjacency matrices are calculated and then multiplied in order to get a cost matrix for selecting each vertex combination.

$$A_1 = U_1 \Lambda_1 U_1^T, \quad A_2 = U_2 \Lambda_2 U_2^T \tag{6.5}$$

$$C = |U_1||U_2^T| \tag{6.6}$$

The optimal permutation matrix can be then found by applying the Hungarian algorithm to this cost matrix. The Hungarian algorithm solves the assignment problem by selecting one value from each row in a way that the sum of these selections is the maximum when only one assignment per column is allowed. By definition these selections define an permutation matrix.

When testing this method it gave rather unreliable results while having the additional downside of needing each of the graphs to have the same number of vertices. In practice the most realistic solution was to permutate every possible P with matrix sized $N^x \times N^x$ where N^x is the vertex count of largest graph of the two under comparison. While testing every possible permutation is usually moronic from algorithmic viewpoint, in group photos the number of vertices is relatively low ($N < 10$) so using a graph matching algorithm like PATH [37] wouldn't help.

To test each P candidate a similarity measurement function seen in eq. 6.7 was used. In it an adjacency matrix A^y is rearranged to match A^x using a $P_{i,j}$ candidate. $A_{i,j}$ is set to zero when filling the smaller graph to match the larger graph. Overall the best structural match is found by minimizing $F_0(P)$, which is the Frobenius norm of difference between the two graphs being compared. Trace is the sum of values on matrix diagonal.

$$\min_{P_{i,j} \in P} F_0(P) = \left|\left|A^x - A^{P(y)}\right|\right|_F^2 = \left|\left|A^x - P_{i,j} A^y P_{i,j}^T\right|\right|_F^2 \tag{6.7}$$

$$\text{where } ||E||_F^2 = \text{trace}(EE^T)$$

We also have labeling data ($C_{i,j}$) in form of gender, age and expression. With these a fitness matrix can be calculated by comparing them between each face from both photos. As expression is not a property on a scale like age a or single difference between genders I decided to define a distance table for pairwise expression distances. For example, it makes intuitive sense that the value for distance of a happy expression to a sad expression is greater than distance of a happy expression to a neutral expression. The distance mapping for expression can be seen in table 6.1.

Table 6.1. *Expression distance function values for different pairwise comparisons. Large=1, Medium=0.5, Small=0.1*

Distance	Happy	Sad	Angry	Disg.	Afraid	Neutral
Happy	0	L	L	L	L	S
Sad	L	0	M	M	M	M
Angry	L	M	0	M	M	L
Disgusted	L	M	M	0	M	L
Afraid	L	M	M	M	0	L
Neutral	S	M	L	L	L	0

Additionally an identity match can be used to mark two faces to be an exact match. Using this matrix a best possible labeling P can be found with eq. 6.8. By mixing these two metrics with adjustable variable $\alpha \in [0, 1]$ a good graph matching $P_{i,j}$ can be found with eq. 6.9.

$$\min_{P_{i,j} \in P} tr(C^T P) = \sum_{i}^{N^x} \sum_{i}^{N^x} C_{i,j} P_{i,j} \tag{6.8}$$

$$\min_{P_{i,j} \in P} (1 - \alpha) F_0(P) + \alpha \, \mathrm{trace}(C^T P) \tag{6.9}$$

Using this best P the vertices from the second photo (described by A^y) can be then rearranged to match the vertices in the first photo (A^x). To calculate the final distance value from these matching vertices the assumption is that the edges connecting similar vertices are orientated the same way. If an edge is defined as $e_{ij} = \{v_i^x, v_j^x\}$ where v_i and v_j are the normalized vertex coordinates from a photo then the difference in the edge orientations between these two photos can be expressed as

$$O(e_{ij}^x, e_{ij}^{x'}) = \left|\left| \frac{v_j^x - v_i^x}{||v_j^x - v_i^x||} - \frac{P^T v_j^y - P^T v_i^y}{||P^T v_j^y - P^T v_i^y||} \right|\right| \tag{6.10}$$

With this definition the value of O increases as the angle between matching edges in photos increases. If P matches an edge to a filler vertex ($e^x_{ij} = 0$ or $e^{x'}_{ij} = 0$) the distance value gets a value of 1. This is not too punishing but still adds large enough penalty to push results with different number of vertices between the photos down the matching list. The same value results from a real match when the angle between the edges is 90°. The final distance between two photos is the sum of edge similarities.

$$D(photo_1, photo_2) = \sum_{ij} O(e^x_{ij}, e^{x'}_{ij}) \tag{6.11}$$

By using this metric to compare photos against the example query photo in figure 6.2 the photos seen in figure 6.3 will have a small distance (are more similar) and the photos in figure 6.4 will have a large distance.

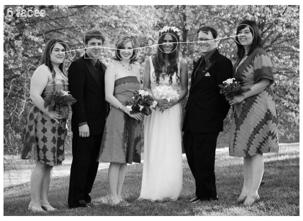


Figure 6.2. *Example query photo from [38]*

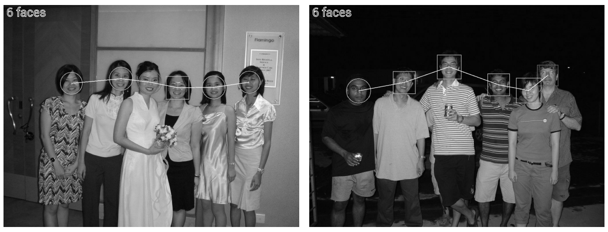


Figure 6.3. *Photos with short distance to the query photo. (Similar to the query photo) [38]*

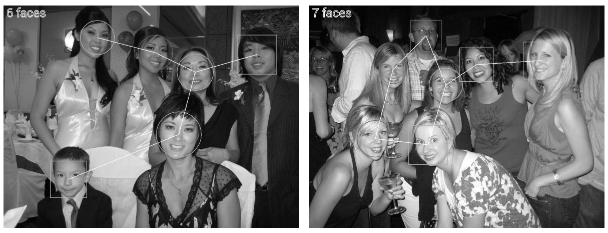


Figure 6.4. *Photos with large distance to the query photo. (Dissimilar to the query photo) [38]*

7. IMPLEMENTATION OF AN PHOTOGRAPH CLASSIFIER APPLICATION ON ANDROID

The main goal of this thesis was to build a mobile camera application capable to demonstrate automatic photo categorization with the aid of several face recognition and classification methods. The application was written to Android operating system using Java and C++-languages.

7.1 Software Platform

The Android operating system was selected as it is well established mobile operating system with open source based development tools and Java-based ecosystem providing C++ support through the Native Development Kit. Additionally OpenCV software library was chosen to speed up the development process as it is open sourced, well maintained and commonly used library in machine vision projects. The very first plan was to use Qt–Necessitas package of development tools, but it didn't support all needed features. Qt 5.2 would have been preferable as it is promised to have full Android support while running natively, but at the time it wasn't released yet. On table 7.1 are the details on used software.

Table 7.1. *Software used in this thesis project.*

Software	Version
Android OS	API-Level 11 (3.0 Honeycomb)
Android SDK Tools	21.1
Native Development Kit	rev. 8e
OpenCV	2.4.5

7.2 Implemented methods

Implemented methods were selected by comparing given results from articles while limiting the choice to methods relying on statistical measurements. This limit was due to uncertainty of photo quality and the fact that the methods were to be used to classify faces picked from group photos. As the photos were small selections from larger photo it was likely that they would be of small resolution and therefore unlikely to be less useful in methods relying on anthropometric methods.

The face identity and property feature calculations were implemented as a native library and called through JNI. Originally the goal was to implement some of these functions in Java, but the result was unbearably slow as the methods operate on per pixel level. This is either slow through JNI by calling matrix functions for every pixel or wasteful memory-wise by having to use matrix copies in memory.

The group distance calculations were implemented on Java as they weren't so dependent on raw processing speed throughput. For classification several readily available OpenCV methods were used. These can be seen in table 7.2.

Table 7.2. *Used classification methods in OpenCV*

Operation	Class in OpenCV
SVM	CvSVM (Java)
K-nearest neighbors	CvKNearest (Java)
Face detection (photos)	CascadeClassifier (Java)
Face detection (video)	DetectionBasedTracker (Java)
Eigenfaces and Fisherfaces	FaceRecognizer (C++)

7.3 Camera software

The software is based on OpenCV face search example as it demonstrates all necessary parts for basic camera operations and provides a basic foundation to implement user interaction and face recognition methods discussed in this thesis. By default the program tries to recognize and process any faces in real time video feed captured from the devices primary camera.

The user interface is minimal (see figure 7.1) as user input is required only to set up the wanted camera parameters and choose the wanted recognition methods. The main view when the application is started is the `CameraView` retrieved directly from OpenCV video capture method. When the application has found a face it will be highlighted with a green square by default. If the application is able to identity the target the name is added on top of the face square. When the user selects the option button the application will display the option menu used to select the used processing methods. From the menu the user can access all functions of the application. All group classification related tasks are found in "Photo Gallery". By selecting "Update matches" from photo gallery (fig. 7.3) the application will calculate the best matches for all pictures in the default android photo directory. (usually `\DCIM\Camera`) If the user wants to adjust the weights of the group classifiers (see eq. 6.1 and the

$C_{i,j}$–matrix in eq. 6.8) they can be adjusted by selecting “Set weights”. By selecting the “Similar photos”-button the application will show the best matching photos for the currently shown photo. In figure 7.5 the swipe operated result viewer is in mid change between two results.

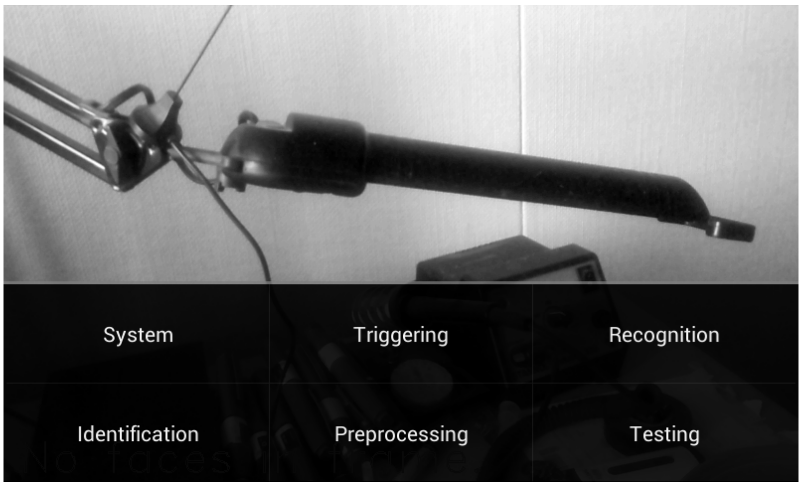


Figure 7.1. *The system menu of the application over live camera feed.*

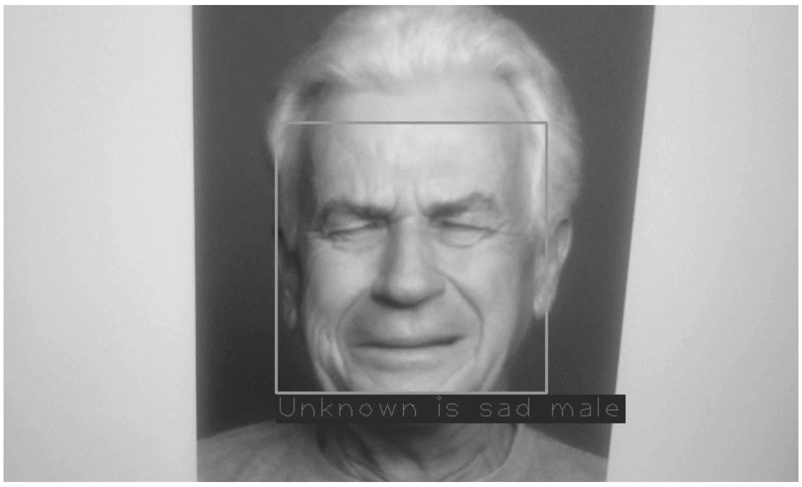


Figure 7.2. *Face recognized and categorized from live video feed.*

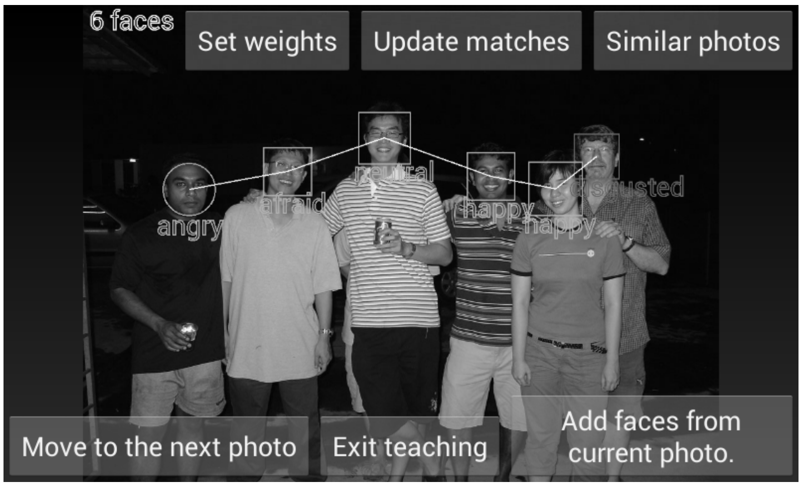


Figure 7.3. *The photo gallery.*

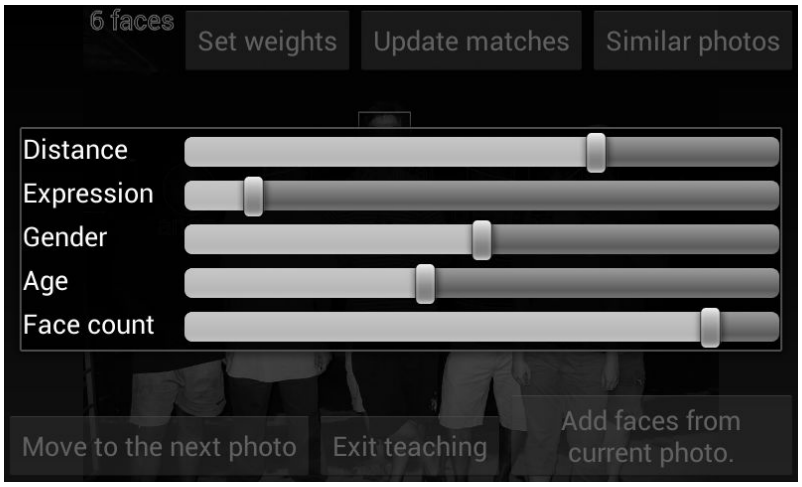


Figure 7.4. *The weights adjustment for group classifiers.*

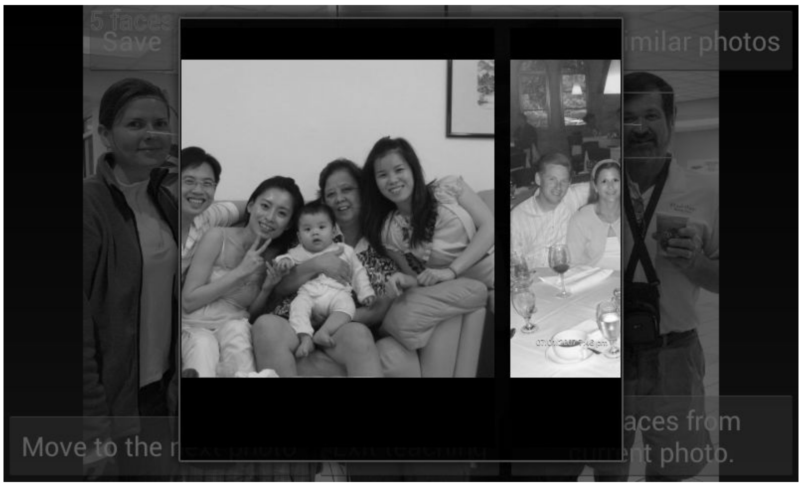


Figure 7.5. *The photo gallery match results viewer between two selections.*

8. RESULTS AND DISCUSSION

Testing different identification and labeling methods was done using pre-labeled datasets built just for such tasks. Most tests were done using the FACES-dataset. [18] FACES contains labeled photo sets from 171 individuals of different age and gender with range of expressions. Each expression is duplicated by having two sets of six photos per individual, resulting in 2052 unique photos. These photos were used to train gender, age and expression classifiers discussed earlier by selecting the first 1800 photos to train classifiers and using the remaining 252 photos to test the trained classifiers. In other words, the classifier was used only on test samples which were not used in training sets.

8.1 Identity classification methods

To test identification methods number of samples were selected per identity and the classifier was trained using a number of identities. These identities were selected randomly from the FACES data. The classifier was then tested against the remaining samples from the selected identities and a fixed number of other identities.

When testing "Eigenfaces"-method the results were reasonable, peaking at 70% recognition rate, but this was on ideal data matching single portrait photos. When testing with faces extracted from group photos the recognition rate was too low to be considered to be usable. This wasn't a real surprise as the training data was made up from several expressions and so resulting to large in-class differences which are problematic for "Eigenfaces". Overall the use of "Eigenfaces" was rejected as impractical.

When testing the Gabor LBP-method the number of selected sets and the number of samples used to train classifiers was varied as can be seen the figures 8.1 and 8.2. By adjusting the acceptable correlation match limit the relation of false positives and recognition rate could be adjusted. Overall the method works quite well and so it is used to match identity similarities in this thesis.

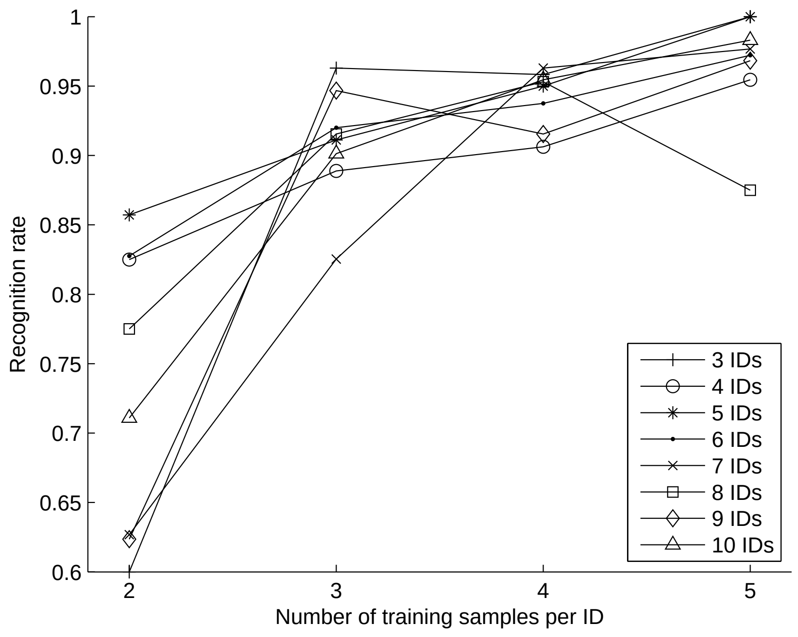


Figure 8.1. *Recognition test results for Gabor Volume LBP using FACES*

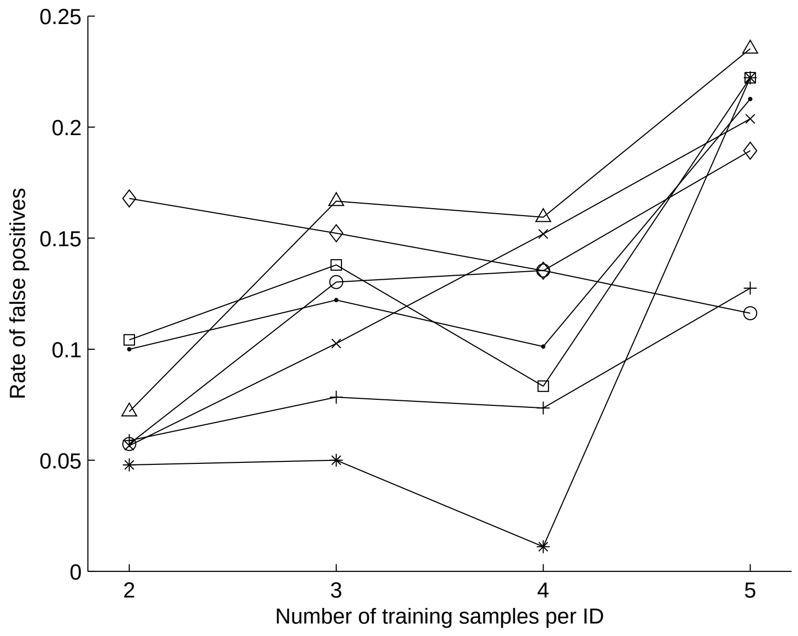


Figure 8.2. *False positive test results for Gabor Volume LBP using FACES*

8.2 Face property classification methods

Gender classification was tested with three different methods described earlier in section 5.2.1. The results of these classifications can be seen in tables 8.1 – 8.3. As it can be seen the edge-method was the best and it was selected to be used as a gender classifier. The "Fisherface"-method was tested using the implementation found in OpenCVs contrib–module. Two others were implemented based on the original papers and the resulting feature vectors were classified using OpenCVs multi class SVM without smoothing.

Table 8.1. *Gender confusion matrix for Fisherfaces method using FACES (N=253)*

Truth	**Male**	**Female**
Recognized as male	113	20
Recognized as female	41	79

Table 8.2. *Gender confusion matrix for LBP-histograms using FACES (N=253)*

Truth	**Male**	**Female**
Recognized as male	71	62
Recognized as female	6	114

Table 8.3. *Gender confusion matrix for Edge histograms using FACES (N=240)*

Truth	**Male**	**Female**
Recognized as male	126	6
Recognized as female	19	89

Table 8.4. *Used parameters and results for gender classifiers*

Method	SVM kernel	Kernel parameters	Classification acc.
Fisherfaces			75.9%
LBP histograms	Radial Basis Function	$C = 100, \gamma = 0.01$	73.1%
Edge histograms	Linear	$C = 100$	89.6%

The resulting accuracies are calculated from these measurements as number of correct classifications divided by number of all samples shown to the classifier. The tally of gender classifier results can be seen in table 8.4.

The test set division for expression classification was the same as in gender classification. Results were reasonable at 80% recognition rate although in real life situations the classification accuracy drops. The classifier also has some issues separating negative expressions from each other. This problem is mitigated by having these expressions mapped to shorter distances from each other when classifying expressions in group photos. This distance mapping can be seen in table 6.1 and an example test run results in table 8.5. The SVM was trained using the default grid search method [25, p.5] in OpenCV using the radial basis function kernel with parameters of $C = 100, \gamma = 0.01$.

Table 8.5. *Expression confusion matrix for AR-LBP histograms method. (N=240)*

Truth	**Happy**	**Sad**	**Angry**	**Disg.**	**Afraid**	**Neutral**
Recognized as happy	39	0	0	1	0	0
Recognized as sad	3	26	1	3	5	2
Recognized as angry	0	3	26	6	0	5
Recognized as disgusted	0	1	0	39	0	0
Recognized as afraid	2	5	0	1	31	1
Recognized as neutral	1	0	0	6	2	31

The testing of age classifier was done similarly to gender and expression classifiers. The final classification error rate hovered between 85-90%. For the test run seen in table 8.6 the classification error was 87.9% when using similar SVM parameters as the expression classifier.

Table 8.6. *Age confusion matrix for ELBP histogram method using FACES (N=240)*

Truth	**Young**	**Middle-aged**	**Old**
Recognized as young	96	0	0
Recognized as middle-aged	16	56	0
Recognized as old	9	4	59

8.3 Photo classification and group similarity methods

To test photo categorization I used the “Images of Groups Dataset” [38] which contains 5080 images split into three categories – wedding pictures, family gatherings and other group pictures. All 28 231 faces in these photos has been hand labeled by age and gender. To train the classifier equally sized random sets of photos were chosen from each class with equally sized test set. These sets do not overlap. A distance matrix between these photos like in figure 8.3 was then used as a training set. As these are distances, note that that $d(i, j) = d(j, i)$ and $d(i, i) = 0$. The best result was achieved when the classifier was trained using 115 samples per class and using 5 closest neighbors to vote. This resulted to 72.3% accuracy which is clearly above random chance. ($^1/_3$ in three classes case) Overall the classifier works reasonably well whenever using more than 30 samples per class to train.

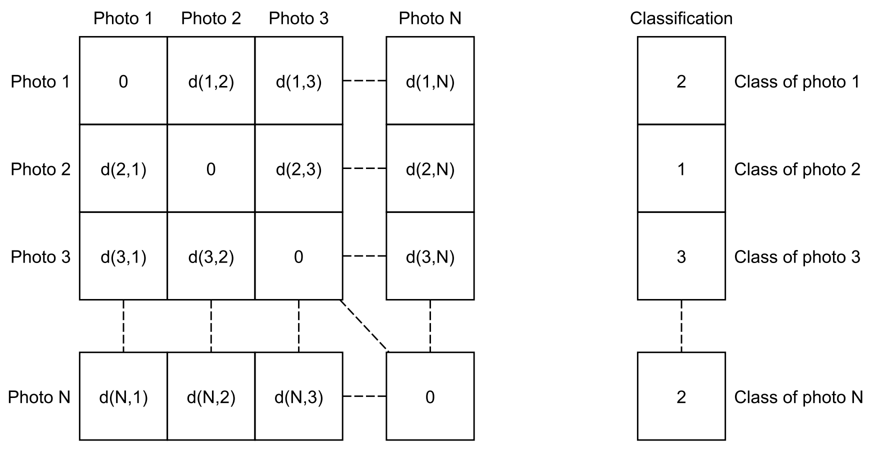


Figure 8.3. *Training data structure for the k-NN classifier.*

To test the photo similarity matching a group of images taken from [38] were processed using results from the face recognizer and trained property classifiers while ignoring the dataset labeling. The weights for different properties were set to the same value. The used example α–values are near extremes to avoid tied scorings. The resulting matches can be seen in tables 8.7 and 8.8. As it can be seen the matching process results to subjectively good matchings. When browsing through different queries it was also noticeable that the overall ordering of pictures was mostly reasonable. The most common type of picture out of its place were photos with small groups of people in a loop structure or in line with large height differences. As this kind of structure can be matched inside a larger one the distance score can be artificially small. It was however rare that this kind of score skewing led to best matches to be replaced with such photos.

Table 8.7. *Examples of best matches using different α-values. Wedding picture, faces in single line.*

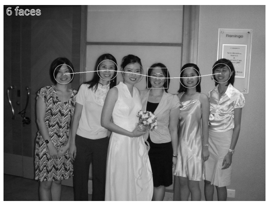
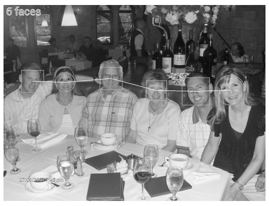
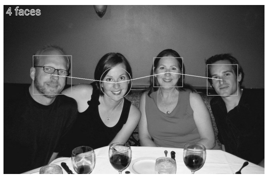
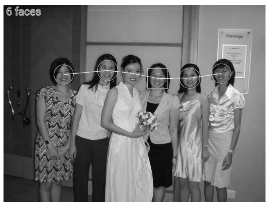
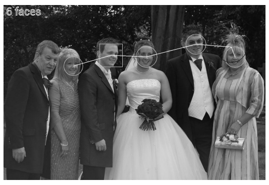
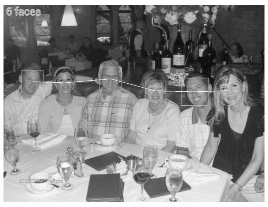
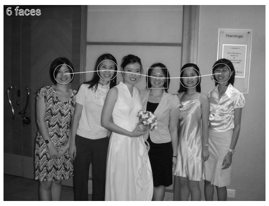
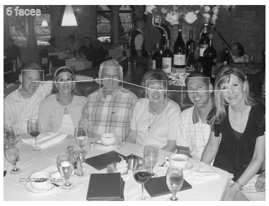
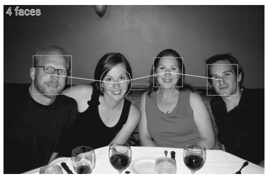
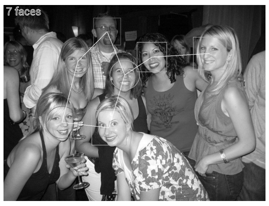
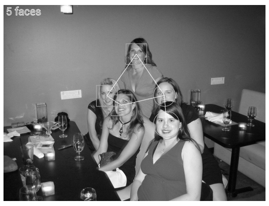
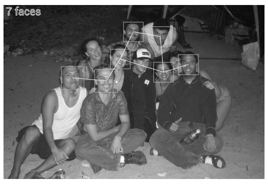
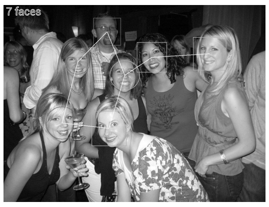
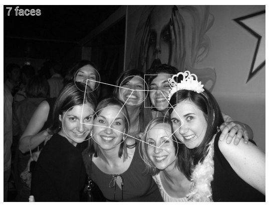
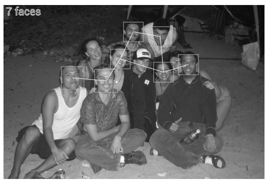
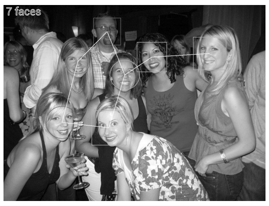
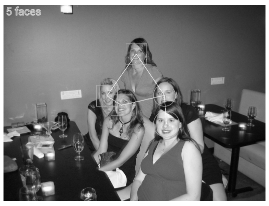
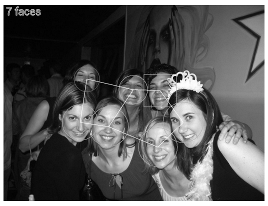
Parameters	Query	Best match	Second match
$\alpha = 0.05$ (Structure)			
$\alpha = 0.95$ (Label)			
$\alpha = 0.5$ (Mix)			

Table 8.8. *Examples of best matches using different α-values. Party picture, faces as a complex group.*

Parameters	Query	Best match	Second match
$\alpha = 0.05$ (Structure)			
$\alpha = 0.95$ (Label)			
$\alpha = 0.5$ (Mix)			

9. CONCLUSIONS

In this thesis the goal was to review methods of automatic group photo classification and implement an Android photo classification application using those methods. In order to implement these methods a further review was made to gender, age and expression classification from face photos. The methods of detecting and identifying faces from photos were also discussed.

Based on the review of all these methods an application was written by implementing several feature classifiers. The accuracy of these classifiers were measured by using labeled sample data from FACES [18] database and they were deemed to be practically accurate. Using these classifiers two group photo classifying methods were then implemented and tested using the "Images of groups dataset" [38]. These photo classifiers produced reasonably good results enough to be called successful methods to classify photographs by their face group content. The concept of a good match in this context is subjective but it is arguable that these methods delivered on what they promised as they could be used, for example, to automatically create photo albums by using handful of photos as training hints.

For future work a good approach would be to creatively mix the two methods used in this thesis. The result of photo similarity matching might be improved if the classifier included the labeling data in scoring after the graph matching. Although the group structure alone is good enough metric it ignores hints clear to the human sensibilities and leads to occasional bad scoring.

REFERENCES

[1] Available at: `www.opencv.org`

[2] Pearson, K.; "On Lines and Planes of Closest Fit to Systems of Points in Space", Philosophical Magazine, Vol. 2, No. 6. (1901), pp. 559–572

[3] Ojala, T.; Pietikäinen, M.; Harwood, D.; "Performance evaluation of texture measures with classification based on Kullback discrimination of distributions", Proceedings of the 12th IAPR International Conference on Pattern Recognition (ICPR 1994), vol. 1, pp. 582–585.

[4] Naika, C.L.S.; Das, P.K.; Nair, S.B.; "Asymmetric Region Local Binary Pattern Operator for Person-dependent Facial Expression Recognition", Communication and Applications (ICCCA), 2012 International Conference on Computing, DOI: 10.1109/ICCCA.2012.6179199

[5] Liao, S.; Chung, Albert C. S.; "Face Recognition by Using Elongated Local Binary Patterns with Average Maximum Distance Gradient Magnitude", Lecture Notes in Computer Science Volume 4844, 2007, pp 672–679. DOI: 10.1.1.75.8354

[6] Gabor, D.; "Theory of communication. Part 1: The analysis of information", Electrical Engineers – Part III: Radio and Communication Engineering, Journal of the Institution of, vol.93, no.26, pp.429–441, November 1946, DOI: 10.1049/ji-3-2.1946.0074

[7] Online, Referenced 6.11.2013, Available at: `http://www2.it.lut.fi/project/simplegabor/downloads/laitosrap100.pdf`

[8] Liu, C.; Wechsler, H.; "Gabor feature based classification using the enhanced fisher linear discriminant model for face recognition", Image Processing, IEEE Transactions on, vol.11, no.4, pp.467–476, Apr 2002, DOI: 10.1109/TIP.2002.999679

[9] Richard O. Duda, Peter E. Hart, David G. Stork, "Pattern classification", Wiley-Interscience; 2nd edition, November 9, 2000

[10] Yoav, F.; Schapire, Robert E.; "A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting", 1995, CiteSeerX:10.1.1.56.9855

[11] Viola, P.; Jones, M.; "Robust Real-time Object Detection", International Journal of Computer Vision, 2001, DOI: 10.1.1.110.4868

[12] Corinna, C.; and Vapnik, Vladimir N.; "Support-Vector Networks", Machine Learning, 20, 1995. Available http://link.springer.com/article/10.1007%2FBF00994018

[13] Online, Referenced 9.5.2013, Available at: http://research.microsoft.com/en-us/um/people/jplatt/smo-book.pdf

[14] Toni Frissell: Fado singer in Portuguese night club, Lisbon, 1946, http://www.flickr.com/photos/trialsanderrors/3108655995/

[15] Lin, Hai-bo; "A Kind of Human Face Region Detection and Recognition Method Based on Chrominance Information Characteristics", 2012 International Conference on Control Engineering and Communication Technology, pp. 469–472, DOI: 10.1109/ICCECT.2012.82

[16] Kovac, J.; Peer, P.; Solina, F.; "Human Skin Color Clustering for Face Detection", EUROCON 2003. Computer as a Tool. The IEEE Region 8 (Volume:2), pp. 144–148, DOI: 10.1109/EURCON.2003.1248169

[17] Liao, P.; Wang, Y.; Wang, M.; Ding, S.; Ma, H.; "An Effective Preprocessing Scheme for Face Recognition Based on Local Gabor Binary Pattern Histogram Sequence", Computer Science and Automation Engineering (CSAE), 2012 IEEE International Conference on (Volume 3), DOI: 10.1109/CSAE.2012.6273020

[18] Ebner, N.; Riediger, M.; Lindenberger, U.; (2010). "FACES—A database of facial expressions in young, middle-aged, and older women and men: Development and validation.", Behavior research Methods, 42, pp. 351–362. DOI:10.3758/BRM.42.1.351

[19] Turk, M.; Pentland, A.; "Face recognition using eigenfaces", Proc. IEEE Conference on Computer Vision and Pattern Recognition, 1991, pp. 586–591, DOI: 10.1109/CVPR.1991.139758

[20] Lei, Z.; Liao, S.; Pietikainen, M.; Li, S.Z.; "Face Recognition by Exploring Information Jointly in Space, Scale and Orientation", Image Processing, IEEE Transactions on , vol.20, no.1, pp.247–256, Jan. 2011, DOI: 10.1109/TIP.2010.2060207

[21] Online, Referenced 7.8.2013, Available at: http://www.encyclopediaofmath.org/index.php/Karush-Kuhn-Tucker_conditions

[22] Modified, originally from https://commons.wikimedia.org/wiki/File:Eigenfaces.png, Referenced 18.4.2013

[23] Online, Referenced 6.11.2013, Available at: `http://www.gnu.org/software/gsl/`

[24] Fisher, R. A.; "The Use of Multiple Measurements in Taxonomic Problems", Annals of Eugenics 7 (2): 179–188, 1936, DOI:10.1111/j.1469-1809.1936.tb02137.x. hdl:2440/15227.

[25] Online, Referenced 6.11.2013, Available at: `http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf`

[26] Han, B.; Luo, Y.; "Accurate Face Detection by Combining Multiple Classifiers using Locally Assembled Histograms of Oriented Gradients", Audio, Language and Image Processing (ICALIP), 2012 International Conference on, pp. 106–111, DOI: 10.1109/ICALIP.2012.6376595

[27] Shan, C.; "Learning local binary patterns for gender classification on real-world face images", Pattern Recognition Letters, Volume 33 Issue 4, March, 2012, pp. 431–437, DOI: 10.1016/j.patrec.2011.05.016

[28] Ardakany, A. R.; Joula, A. M.; "Gender Recognition Based on Edge Histogram", International Journal of Computer Theory and Engineering Vol. 4, No. 2, pp. 127–130, April 2012

[29] Guo, Z.; Zhang, L.; Zhang, D.; "A completed modeling of local binary pattern operator for texture classification", Image Processing, IEEE Transactions on (Volume:19 , Issue: 6), 19(6):1657–1663, 2010, DOI: 10.1109/TIP.2010.2044957

[30] Ylioinas, J.; Hadid, A.; Pietikäinen, M.; "Age Classification in Unconstrained Conditions Using LBP Variants", Pattern Recognition (ICPR), 2012 21st International Conference on, November 2012, pp. 1257–1260

[31] Shu, H.; Gallagher, A.; Chen, H.; Chen, T.; "Face-graph matching for classifying groups of people", International Conference on Image Processing (ICIP), September 2013.

[32] H.W. Kuhn, "The Hungarian method for the assignment problem," in Naval Research Logistics Quarterly, Volume 2, Issue 1–2, pages 83–97, March 1955, DOI: 10.1002/nav.3800020109

[33] Online, Referenced September 2013, Available at: `https://github.com/KevinStern/software-and-algorithms/blob/master/src/main/java/blogspot/software_and_algorithms/stern_library/optimization/HungarianAlgorithm.java`

[34] Chiu Y.; Li C.; Huang C.; Chung P.; Chen T.; “Efficient graph based spatial face context representation and matching”, IEEE ICASSP 2013, pp. 2001–2005, DOI: 10.1109/ICASSP.2013.6638004

[35] Online, Referenced October 2013, Available at: `http://www.theroyalforums.com/attachments/blogs/uploads/2013/05/ip5kcn.jpg` 10.9.2013

[36] Umeyama, S.; “An eigendecomposition approach to weighted graph matching problems”, Pattern Analysis and Machine Intelligence, IEEE Transactions on (Volume 10, Issue 5), pp. 695–703, September 1988, DOI: 10.1109/34.6778

[37] Zaslavskiy, M.; Bach, F.; Vert, J.-P.; “A Path Following Algorithm for the Graph Matching Problem”, Pattern Analysis and Machine Intelligence, IEEE Transactions on (Volume 31, Issue 12), p.2227–2242, October 2008, DOI: 10.1109/TPAMI.2008.245

[38] Gallagher, A.; Chen, T.; “Understanding Images of Groups of People”, Proc. CVPR 2009

A. APPENDICES

A.1 Matlab scripts used in figure generation

Script A.1 gaborgenerator.m

```
close all; clear all;
colourmap(grayscale);
for scale = 0:4
  for rot = 0:7
    [G,GIMAGE]=gaborfilter(I,0.05,scale,rot,0);
    subplot(5,8,n); n=n+1;

    t=127.5/max(max(abs(G)));
    image(uint8(t*real(G)+127.5));
    axis square;
    set(gca, 'XTickLabelMode', 'Manual');
    set(gca, 'YTickLabelMode', 'Manual');
    set(gca, 'XTick', []); set(gca, 'YTick', []);
  end
end
```

Script A.2 gaborfilter.m

```
[G,GIMAGE]=GABORFILTER(Image,Sigma,Freq,Rotation,Phase)
size=fix(1.5/Sigma);
for x=-size:size
  for y=-size:size
    theta = pi*Rotation/8;
    kv=(pi/2)/(sqrt(2)^Freq);
    k = kv*exp(1i*theta);
    G(size+x+1,size+y+1)=(k^2/(4*pi^2))* ...
    exp( -(abs(k)^2*(x*x+y*y)) / (8*pi^2))* ...
    (exp(1i* (real(k)*x + imag(k)*y))-exp(-2*pi^2+(1i*Phase)));
  end
end
GIMAGE=conv2(Image,double(G),'same');
```

Script A.3 pcademo.m

```
close all; clear all;
a = 10; b = 90; x = a + (b-a).*rand(50,1);
c = 20; d = 80; y = c + (d-c).*rand(50,1);
y=y+(x-50);
z=[x y];
subplot(1,2,1);
scatter(x,y,30,'filled');
hold on; axis([0 100 0 100]); axis square;
xlabel('Original␣1st'); ylabel('Original␣2nd');
subplot(1,2,2);
[pc,score,latent]=princomp(zscore(z));
biplot(pc(:,1:2), 'Scores', score(:,1:2),'VarLabels', ...
    {'Original␣1st' 'Original␣2nd'});
axis square; xlabel('New␣axis␣1'); ylabel('New␣axis␣2');
```

Script A.4 pcasteps.m

```
X = rand(20,3)*10; subplot(2,2,1);
scatter3(X(:,1),X(:,2),X(:,3),'filled'); axis square
title('The␣original␣3D␣set␣X');
subplot(2,2,2);
for a=1:3 X(:,a) = X(:,a) - mean(X(:,a)); end
    scatter3(X(:,1),X(:,2),X(:,3),'filled'); axis square
hold on
P = [0 0 0];
for p=1:size(X,1) P = [P;X(p,:);[0 0 0]]; end
line(P(:,1),P(:,2),P(:,3));
title('Centered␣set␣XX^T␣producing␣mean␣distances');
subplot(2,2,3);
[pc,score,latent]=princomp(X); %means calculated inside
biplot(pc(:,1:2), 'Scores', score(:,1:2),'VarLabels', ...
    {'Original␣1st' 'Original␣2nd' 'Original␣3rd'});
title('3D␣set␣reduced␣to␣2D'); axis square
subplot(2,2,4);
X(:,2) = X(:,2)*5;
[pc,score,latent]=princomp(X);
biplot(pc(:,1:2), 'Scores', score(:,1:2),'VarLabels', ...
    {'Original␣1st' 'Original␣2nd' 'Original␣3rd'});
title('Original␣2nd␣component␣multiplied␣with␣5'); axis square
```