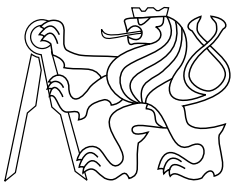
CENTER FOR
MACHINE PERCEPTION



CZECH TECHNICAL
UNIVERSITY IN PRAGUE

MASTER'S THESIS

ISSN 1213-2365

Structural classifier for gender recognition

Ondřej Fišar Bc.

ondra.fisar@gmail.com

CTU–CMP–2011–09

August 15, 2011

Available at
http://cmp.felk.cvut.cz/~fisarond/thesis.pdf

Thesis Advisor: Ing. Vojtěch Franc, Ph.D.

The authors were supported by EC projects FP7-ICT-247525
HUMAVIPS and PERG04-GA-2008-239455 SEMISOL.

Research Reports of CMP, Czech Technical University in Prague, No. 9, 2011

Published by

Center for Machine Perception, Department of Cybernetics
Faculty of Electrical Engineering, Czech Technical University
Technická 2, 166 27 Prague 6, Czech Republic
fax +420 2 2435 7385, phone +420 2 2435 7637, www: http://cmp.felk.cvut.cz

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra kybernetiky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Ondřej Fišar**

Studijní program: Otevřená informatika (magisterský)

Obor: Počítačové vidění a digitální obraz

Název tématu: **Rozpoznávání pohlaví z obrázku tváře**

Pokyny pro vypracování:

1. Navrhněte a implementujte klasifikátor, který bude odhadovat pohlaví člověka na základě obrázku jeho tváře. Výsledný klasifikátor musí být dostatečně robustní vůči změnám osvětlení, pozice, rotace a měřítka.

2. Dále navrhněte a implementujte algoritmus pro automatické učení parametrů klasifikátoru z databáze anotovaných obrázků.

3. Funkci klasifikátoru ověřte na reálných datech.

Seznam odborné literatury:

I. Tsochantaridis, T. Joachims, T. Hofmann and Y. Altun; Large Margin Methods for Structured and Interdependent Output Variables, Journal of Machine Learning Research (JMLR), 6(Sep):1453-1484, 2005

Vedoucí: Ing. Vojtěch Franc, Ph.D.

Platnost zadání: do konce letního semestru 2011/2012

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta elektrotechnická L.S.

prof. Ing. Vladimír Mařík, DrSc.
vedoucí katedry

prof. Ing. Boris Šimák, CSc.
děkan

V Praze dne 25. 3. 2011

Acknowledgment

I am heartily thankful to my advisor, Ing. Vojtěch Franc Ph.D., whose ideas led me during the whole work on my thesis and who corrected my English in the paper. Also I would like to thank Dr. Yea-Shuan Huang, who helped me with the thesis during my studies at Chung Hua University in Taiwan.

Lastly, I offer my regards and blessings to all of those who supported me in any respect during the completion of the project.

Declaration

I hereby declare that I have completed this thesis independently and that I have listed all the literature and publications used.
I have no objection to usage of this work in compliance with the act §60 Zákon č. 121/2000Sb. (copyright law), and with the rights connected with the copyright act including the changes in the act.

In Prague on August 15, 2011 .....

Abstract

This thesis deals with the problem of recognizing gender of a person from an image of his/her face. One of the major problems in the face classification comes from a large image variance caused by unknown pose of the recognized face. Existing approaches for gender recognition are based on feature classifiers trained from examples. The feature classifiers are themselves not invariant against pose transformations of the input face. There are two common strategies to make the classifier invariant. The first strategy is based on registering the input images prior to their classification. The second strategy is based on generating synthetic training examples by applying all pose transformations which do not change the class membership. In this thesis we propose a new approach based on using a structural classifier which treats the unknown face pose as an additional hidden parameter. The proposed structural classifier performs face registration and face classification simultaneously in one step. We formulate learning of the structural classifier from annotated examples as a convex optimization problem. We experimentally compare the proposed structural classifier against one baseline approach and one state-of-the-art approach on a large corpus of faces. The experiments show that the proposed structural classifier outperforms the state-of-the-art method achieving relative improvement of 10% in the classification accuracy. We also propose an improvement of the Bundle Method for Regularized Risk Minimization which is an optimization algorithm suitable for solving large instances of the learning problem. Preliminary results demonstrate that the improved BMRM algorithm can significantly reduced the number of iterations of the original method. The last result of this thesis is an open source library implementing the proposed classifier and its learning algorithm.

Resumé

Tato práce se zabývá problémem rozpoznávání pohlaví osoby z obrazu jejího obličeje. Jeden z hlavních problémů v rozpoznávání tváře tkví ve vysoké variancí obrázků způsobené neznámou polohou rozeznávané tváře. Existující řešení pro rozpoznávání pohlaví jsou založena na příznakových klasifikátorech učených z příkladů. Tyto příznakové klasifikátory nejsou samy od sebe invariantní vůči transformacím polohy zkoumané tváře. Obecně existují dva postupy rozšiřující klasifikátor o invarianci. První postup je založen na registrování vstupního obrázku ještě před samotnou klasifikací. Druhý postup je založen na generování syntetických trénovacích příkladů aplikováním všech transformací polohy obličeje (dále VE postup), které nemění příslušnost příkladů do třídy. V této práci navrhujeme nový přístup založený na použití strukturálního klasifikátoru, který řeší neznámou polohu tváře zavedením nového skrytého parametru. Navržený strukturální klasifikátor provádí registraci tváře a její klasifikaci zároveň v jednom kroku. Učení tohoto strukturálního klasifikátoru z anotovaných příkladů formulujeme jako konvexní optimalizační problém. Experimentálně jsme porovnali navržený strukturální klasifikátor se základním postupem a se zatím nejlepším známým VE postupem. V experimentech se ukázalo, že námi navržený klasifikátor překonává VE postup dosažením relativního zlepšení o 10% v přesnosti klasifikace. Navíc jsme také navrhli vylepšení optimalizačního algoritmu „Bundle Method for Regularized Risk Minimization“ (BMRM), který je vhodný pro učení klasifikátorů z velkého počtu příkladů. Předběžné výsledky prokázaly, že vylepšený BMRM algoritmus může znatelně snížit počet iterací vůči původní verzi. Posledním výsledkem této práce je open source knihovna realizující navržený klasifikátor a jeho učení.

Contents

Abbreviations

Used abbreviations are mainly from domain of computer science and machine learning.

BMRM	Boundle Method for regularized Risk Minimalization
GNU GPLv3	GNU General Public Licence, version no. 3
GUI	Graphical User Interface
HLBP	Histogram Of Local Binary Patterns
HOG	Histogram Of Gradients
LBP	Local Binary Patterns
LBP-pyramid	Local Binary Patterns with pyramidal improvement
MPI	Message Passing Interface
P-BMRM	Improved version of BMRM
SVM	Support Vector Machine

1 Introduction

The aim of this thesis is to design a classifier for categorization of face images into two classes. Expected input of the classifier is an unaligned face image (see Figure 1.1). The gaze angle is assumed to be roughly orthogonal to the camera plane. An output is a recognized class of the input image. In this thesis we are interested in classifying gender, however, the proposed methods are general and they can be used for visual classification of any kind of objects separable into two classes.

Figure 1.1 Example of input image

An image processing is becoming ordinary in our life. It is not surprising that there is a lot of possibilities where a gender recognition can be used. Not one science fiction writer predicts a future with cameras on streets following our steps. Nowadays when a mass media churns out a huge amount of an advertisement, these cameras can be a valuable information source for banners and billboards that will offer us a targeted advertisement in a real life on public places. It means we will get only a kind of an advertisement that is interesting for us. In the gender recognition case it means that a man get only an advertisement for men like expensive cars or Scotch whiskey and vice versa.

Figure 1.2 Humanoid robot NAO

Another application area for the gender recognition technology is in robotics. One of the hot topics in nowadays robotics is to develop robot with behavior similar to humans. This nearly impossible goal pushes robotics research forward. An example of such affords is the EU project HUMAVIPS run at the Center for Machine Perception. One of the major goals of the HUMAVIPS project is to allow humanoid robots to interact with people in a way natural for humans. To this end it is necessary for the robot to interact with a woman in different way than with a man. Moreover, in most cultures there exist social conventions which require different behavior in male and female company. In all these cases the gender recognition module becomes a necessary part of the robot's software. Figure 1.2 shows the humanoid robot NAO used in the HUMAVIPS project. The HUMAVIPS project provided the main motivation for this

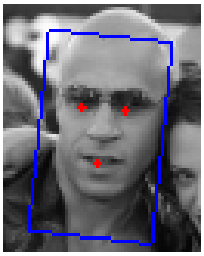


Figure 1.3 Example of the prediction provided by the proposed classifier (blue color stands for male).

thesis.

One of the major problems in the face classification comes from a large image variance caused by unknown pose of the recognized face. Existing approaches for gender recognition are based on feature classifiers trained from examples. However, the feature classifiers are themselves not invariant against pose transformations of the input face. The most typical approach to make the classifier invariant involves two stages. In the first stage, a set of salient feature points (like centers of eyes and mouth) are detected and then the input image is transformed into aligned form where each point is at predefined position. In the second stage, the feature based classifier is applied on the aligned input image. This approach has one principal disadvantage. An imprecise detector of the salient features can spoil the classifier soever good it is. Moreover, designing a reliable detector of salient facial features is itself a difficult problem.

In this thesis we propose a novel structured output classifier which tries to remove the deficiency of the two stage approach. The proposed structural classifier joins both stages together, that is, the classifier performs face alignment and the face classification simultaneously in one step (see Figure 1.3). This is the main feature distinguishing the proposed method from existing approaches all being based on two-class classifiers.

The contributions of this diploma thesis are as follows:

- We formulated a novel structural classifier for gender classification which models the unknown face pose as an additional hidden parameter. The classifier performs the face alignment and classification simultaneously in one step.
- We formulated learning of the structural classifier from annotated examples as a convex optimization problem. The learning problem can be seen as a variant of the Latent Support Vector Machines algorithm [11].
- We performed an extensive experimental comparison of the proposed structural classifier against one baseline approach and one state-of-the-art approach on a large corpus of faces. The obtained results show that the proposed structural classifier outperforms the state-of-the-art method achieving relative improvement of 10% in the classification accuracy.
- We propose an improvement of the Bundle Method for Regularized Risk Minimization [10] which is an optimization algorithm suitable for solving large instances of the learning problem. Preliminary results demonstrate that the improved BMRM algorithm can significantly reduced the number of iterations of the original method.
- We implemented an open source library which contains the proposed structural classifier and its learning algorithm.

The remaining text of the thesis is organized as follows. Existing approaches for gender recognition are outlined in chapter 2. The structural classifier for

gender recognition, formulation of the learning problem and the improved BMRM algorithm are proposed in chapter 3. Experimental evaluation of the proposed classifier and its comparison against existing approaches is presented in chapter 4. Chapter 5 describes the open source library. Finally, chapter 6 concludes the thesis.

2 Approaches of others

Most approaches designed for the gender recognition from facial images use feature based two-class classifiers. Popular classifiers for this purpose are Support Vector Machines [3, 5], Neural Networks [9], Linear Discriminant Analysis in combination with Principal Component Analysis [1], etc. There are various methods how to represent data in feature space. For example: Local Binary Patterns (LBP), Gabor filters, gray-scale values, features designed for the specific problem, histogram of LBP, etc. Parameters of a gender classifier are typically learned from examples of face images along with information about their true gender.

It is a common knowledge that feature base classifiers perform well provided the input face images are accurately aligned. This means that salient facial features like centers of eyes, nose or mouth should always be at the same positions. In typical application, however, images are obtained from responses of a face detector which returns face positions far from being accurately aligned. There are two main approaches to cope with the unaligned face images:

Approach 1: Image alignment + classification This approach involves two stages: i) image alignment and ii) classification. Firstly, salient points are detected and then the input image is transformed (typically applying affine transformation) into aligned form where each point is at predefined position. In the second stage feature based classifier is applied on the already aligned input image.

This approach requires sufficiently accurate and robust detection of salient facial features which is a non-trivial problem itself. Currently, not many open-source implementations of acceptable facial feature detector are currently available.

Approach 2: Method of virtual examples (VE method) Translation, rotation, change of scale and aspect-ration form a basic set of image transformations which do not change the class membership of the input images. The VE method allows to learn a classifier which is invariant against this set of transformation. The VE method is based on generating synthetic training examples by applying the set of transformations which do not change the class membership on a set of aligned images. The synthetic examples contain all the variance originating from unaligned images. Having the synthetic training examples the standard two-class classifier is trained.

A big advantage of the virtual examples method is that it can be used to learn classifiers invariant against arbitrary finite set of transformations. A notable disadvantage are huge memory and computational requirements caused by the size of the synthetic training set. The number of the synthetic examples is proportional to the number of transformations which is in real application large. This limitation makes the application of many

learning algorithms prohibitive. An efficient algorithm for training linear SVM classifier from virtual examples with application to gender recognition was proposed in [7].

In this thesis we propose a different approach based on a structured output classification instead of the common two-class classifier. Unlike the approach 1 our structural classifier caries out image alignment and the classification simultaneously in one step. Unlike the approach 2 our classifier treats the unknown transformation of the input faces as an additional hidden parameter to be estimated during the classification.

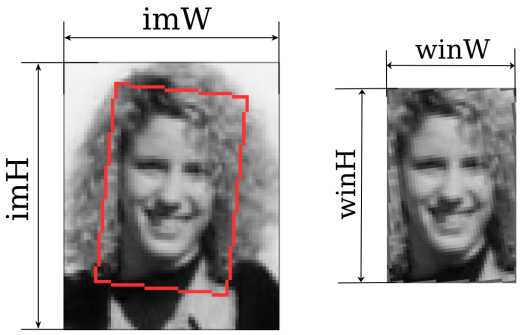
3 Proposed method

In this chapter we describe the proposed structural classifier for gender recognition and a supervised algorithm for learning its parameters from examples. The chapter is organised as follows. In section 3.1, we describe our structural classifier. In section 3.2, we formulate learning of the structural classifier from examples as a convex optimization problem. Our learning algorithm is a variant of the latent Structured Output SVM [11]. In section 3.3, we compare our approach with other methods and describe its advantages. In section 3.4, we outline the Bundle Method for Risk Minimization[10] (BMRM) which is an optimization algorithm suitable for solving the learning problem defined in section 3.2. An improved variant of the BMRM algorithm (called P-BMRM where P stands for parameter P of this algorithm) is proposed in section 3.5. Finally, in section 3.6 we describe three image feature representations suitable for our classification model. In particular, we consider Local Binary Patterns (LBP) [6], Histogram of Gradients (HOG)[2] and Histogram of LBP (HLBP).

3.1 Proposed structural classifier

Let X be a finite set of all input images, $Y = \{-1, +1\}$ be a set of class labels and let $z \in Z$ denotes a hidden state representing an unknown pose of the classified object. Let us introduce a concept of a *search-window* (see figure 3.1) which defines a sub-image inside the image $x \in X$ at the pose (position, orientation, scale and aspect ratio) $z \in Z$. Our goal is to design a classifier $h : X \to Y$ predicting label $y \in Y$ for an input image $x \in X$. Classification amounts to computing

$$h(x) = \arg\max_{y \in Y} \max_{z \in \mathbb{Z}} \langle \boldsymbol{w}, \Phi(x, y, z) \rangle, \tag{3.1}$$



a) search-window (red region) **b)** base-window

Figure 3.1 Example of a detection. *search-window*: accurate position of face, *base-window*: already transformed search-window to fixed size of $[winW \times winH]$

where $\boldsymbol{w} \in \mathbb{R}^{2d}$ is a parameter vector and an operator $\Phi\colon X \times Y \times Z \to \mathbb{R}^{2d}$ maps each combination of x, y and z onto the parameter space $\mathbb{R}^{2d}$. In the equation 3.1, the classifier finds the most compatible combination of class label y and the *search-window* pose z for given input image x. The compatibility of

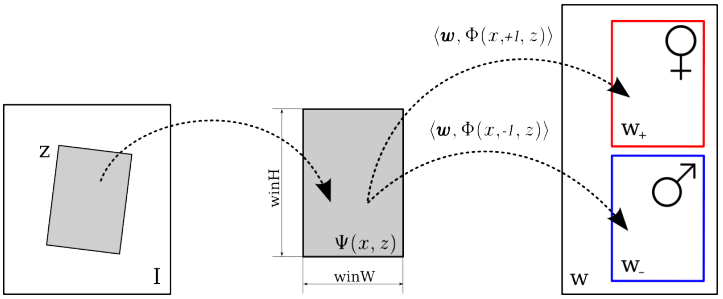


Figure 3.2 Example of $\Phi(x, y, z)$ operator

the triplet (x, y, z) is evaluated by a score function $\langle \boldsymbol{w}, \Phi(x, y, z) \rangle$. The parameter vector $\boldsymbol{w} \in \mathbb{R}^{2d}$ is a union of two prototype vectors $\boldsymbol{w}_- \in \mathbb{R}^d$ and $\boldsymbol{w}_+ \in \mathbb{R}^d$ for negative and positive class, respectively. That is the vector $\boldsymbol{w}$ reads

$$\boldsymbol{w} = \begin{bmatrix} \boldsymbol{w}_- \\ \boldsymbol{w}_+ \end{bmatrix} \in \mathbb{R}^{2d}. \tag{3.2}$$

Before we describe operator Φ precisely, we introduce a mapping $\Psi \colon X \times Z \to \mathbb{R}^d$ and a concept of *base-window* (see figure 3.1). The vector $\Psi(x, z)$ returns a feature description of the search-window at pose z inside the image x. Computation of $\Psi(x, z)$ involves two steps. First, a sub-image defined by *search-window* at pose z is affinely transformed into a *base-window* of a fixed size $[winW \times winH]$. Second, a selected features are computed on the *base-window* which produces a feature description of a fixed size. Particular feature descriptors used in this thesis are listed in section 3.6. Finally, we can define the mapping Φ implicitly by following expression

$$\langle \boldsymbol{w}, \Phi(x, y, z) \rangle = \begin{cases} \langle \boldsymbol{w}_+, \Psi(x, z) \rangle & \textbf{for} \quad y = +1, \\ \langle \boldsymbol{w}_-, \Psi(x, z) \rangle & \textbf{for} \quad y = -1. \end{cases}$$

To put it in words, the compatibility of a triplet (x,z,y) is defined by a dot-product between the prototype vector $\boldsymbol{w}_y$ and a feature vector $\Psi(x, z)$ describing a region below the *search-window* at pose z inside the input image x.

Figure 3.2 visualizes the whole process of computing the score function of the classifier 3.1.

So far we have described a general structural classifier for categorization of unregistered objects into two classes. Now, we describe a particular instance of the structural classifier applied to the gender recognition. We assume that the input x is a near-frontal facial image of size $[imW \times imH]$ pixels represented by 256 gray-scale levels. Our face images come along with manual annotation of the gender and the face pose. The face pose is determined by the position of the center of the left eye, the right eye and the mouth. We use the face pose to define the ground truth position of the search-window. The hidden parameter $z \in Z$ defines a geometric transformations which transforms the *search-window* into the *base-window*. Specifically, we consider five basic transformations: a translation in both axis $x \in Z_x$ and $y \in Z_y$, a rotation $\alpha \in Z_\alpha$, a scale $k \in Z_{scale}$ and the

aspect ratio $a \in Z_{ratio}$. The final geometric transformation is a combination of these basic transformations, that is,

$$Z = Z_x \times Z_y \times Z_\alpha \times Z_{scale} \times Z_{ratio}. \tag{3.3}$$

Let $\tau(I)$ denotes following transformation of input image I: scale by $\frac{1}{k}$, changing aspect ratio by $\frac{1}{a}$ and rotation by $-\alpha$. The straight-forward approach how to get *base-window* from *search-window* is to crop sub-image J at pose z and apply transformation $\tau(J)$. Classification (see equation 3.1) is based on searching whole set of transformations Z which biggest part is formed by translation transformation, i.e. Z_x, Z_y. By this approach we must compute all transformations for each combination of transformations $(x, y, \alpha, k, a) \in (Z_x, Z_y, Z_\alpha, Z_{scale}, Z_{ratio})$. To optimize this procedure we first apply transformation $I' = \tau(I)$ on whole input image I and cut off a sub-image of size $[winW \times winH]$ at position $[x', y']$ in image I' that corresponds to original position $[x, y]$ in untransformed image I. Thanks to that we radically reduce number of expensive transformation like rotation, scale or changing of aspect ratio. We use following order of applying transformations:

1. Scale *search-window* by $\frac{1}{k}$.
2. Change *search-window* aspect ratio by $\frac{1}{a}$.
3. Rotate *search-window* by $-\alpha$.
4. Cut off a sub-image of size $[winW \times winH]$ at position $[x', y']$ that corresponds to original position $[x, y]$ in untransformed input image.

To clarify all kinds of transformations, we must exactly define how to change an aspect ratio of *search-window*. When we apply this transformation, we will keep the *search-window* width and change only the scale of the *search-window* height by k.

3.2 Learning of the structural classifier

Given training dataset

$$T = \{(x_1, y_1, z_1), \ldots, (x_m, y_m, z_m)\} \in (X \times Y \times Z)^m$$

i.i.d. sampled from $p(x, y, z)$. Let $h\colon X \to Y$ denote a hypothesis for estimating hidden state y from an observation x and let $\mathcal{L}_{01}\colon Y \times Y \to \mathbb{R}$ be a zero-one loss function that expresses penalty which occurs when estimated label $h(x)$ is not equal to true class label y, that is,

$$\mathcal{L}_{01}(y, h(x)) = \begin{cases} 0 & \textbf{for} \quad y = h(x), \\ 1 & \textbf{for} \quad y \neq h(x). \end{cases} \tag{3.4}$$

Our goal is to find a hypothesis h^* which minimizes the expected risk $R(h)$, i.e., we want to solve

$$h^* \leftarrow \arg\min_h R(h), \tag{3.5}$$

$$R(h) = \sum_{X \times Y} \mathcal{L}_{01}(y, h(x)) \sum_{z \in Z} p(x, y, z). \tag{3.6}$$

There are two main paradigms used to learn the classifier from examples – generative and discriminative approach. First, the generative approach is based on estimating the joint distribution $p(x, y, z)$ and using the estimate to construct a plug-in Bayes classifier. A notable disadvantage of the generative approach, especially if applied to image classification, is the fact that estimation of the image generating distribution is often a very hard problem. Second, the discriminative approach is based on directly learning the parameters of the classifier by minimizing a surrogate to the expected risk which does not involve the unknown distribution $p(x, y, z)$. In this thesis, we use a representative of the generative paradigm when the regularized empirical risk is used as the surrogate for the expected risk.

Recall that our classifier (3.1) is determined up to the parameter vector $\boldsymbol{w} \in \mathbb{R}^{2d}$. We formulate learning of the parameters $\boldsymbol{w}$ as the regularized risk minimization problem

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w} \in \mathbb{R}^{2d}} \left[\frac{\lambda}{2} \|\boldsymbol{w}\|^2 + R(\boldsymbol{w}) \right]. \tag{3.7}$$

The term $\frac{\lambda}{2}||\boldsymbol{w}||^2$ is called a regularizer and it allows to control over-fitting by tuning the hyper-parameter $\lambda \in \mathbb{R}^+$. The function $R(\boldsymbol{w})$ measures the performance of the classifier with parameter $\boldsymbol{w}$ on the training examples T. The function $R(\boldsymbol{w})$ is an approximation of the real empirical risk $R_{emp}(h, T)$ defined with the 0/1-loss, that is,

$$R_{emp}(h) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}_{01}(y_i, h(x_i)). \tag{3.8}$$

We cannot use the empirical risk in the minimization problem (3.7) because it is a non-convex function hard for optimization. For this reason we replace the original loss function $\mathcal{L}_{01}(y_i, h(x_i))$ by its convex upper bound $\widehat{\mathcal{L}}_{01}(\boldsymbol{w}; x_i, y_i, z_i)$ derived as follows

$$\begin{aligned}
\mathcal{L}_{01}(y_i, h(x_i)) &\leq \max_{y \in Y \setminus \{y_i\}} \mathcal{L}_{01}(y_i, y) \left[\langle \boldsymbol{w}, \Phi(x_i, y_i, z_i) \rangle - \max_{z \in Z} \langle \boldsymbol{w}, \Phi(x_i, y, z) \rangle \leq 0 \right] \\
&\leq \max_{y \in Y \setminus \{y_i\}} \max \left\{ 0, \mathcal{L}_{01}(y, y_i) + \max_{z \in Z} \langle \boldsymbol{w}, \Phi(I_i, y, z) \rangle \right\} \\
&= \max \left\{ 0, \max_{z \in \mathbb{Z}} \left[1 - \langle \boldsymbol{w}, \Phi(x_i, z_i, y_i) \rangle + \langle \boldsymbol{w}, \Phi(x_i, z, \bar{y}_i) \rangle \right] \right\} \\
&= \widehat{\mathcal{L}}_{01}(\boldsymbol{w}; x_i, y_i, z_i)\,,
\end{aligned} \tag{3.9}$$

where $\bar{y}_i$ denotes the complementary label to y_i. Having the approximation for the single loss, we define the approximation of the empirical risk $R_{emp}(h)$ as

$$\begin{aligned}
R(\boldsymbol{w}) &= \frac{1}{m} \sum_{i=1}^{m} \widehat{\mathcal{L}}_{01}(\boldsymbol{w}; x_i, y_i, z_i) \\
&= \frac{1}{m} \sum_{i=1}^{m} \max \left\{ 0, \max_{z \in \mathbb{Z}} \left[1 - \langle \boldsymbol{w}, \Phi(x_i, z_i, y_i) \rangle + \langle \boldsymbol{w}, \Phi(x_i, z, \bar{y}_i) \rangle \right] \right\}.
\end{aligned} \tag{3.10}$$

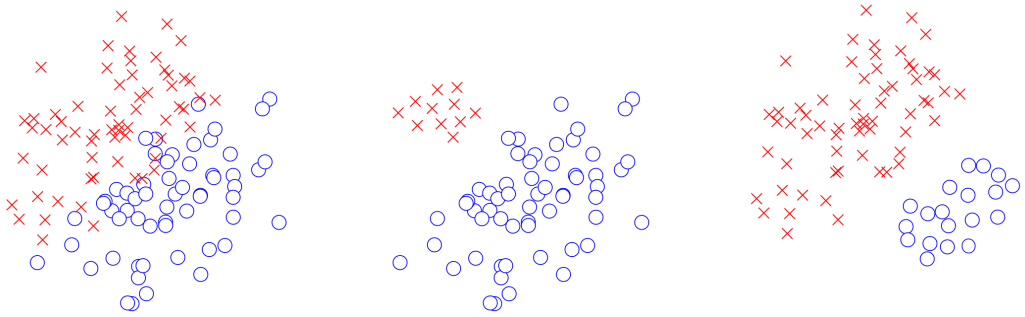
Note that the function $R(\boldsymbol{w})$ is convex in the optimized parameter $\boldsymbol{w}$. Moreover, it is easy to see that the value of $R(\boldsymbol{w})$ is an upper bound of the true empirical risk $R_{emp}(h)$, that is, $R(\boldsymbol{w})$ upper bounds the number of mistakes the classifier with parameter $\boldsymbol{w}$.

3.3 Comparison to other approaches

As mentioned in chapter 2, there are two methods to cope with the unaligned face images both based on two-class classification. In this section we describe difference between these methods and our structural classifier.

In the two-class classifier setting, male and female classes must be separated in the feature space. The accuracy of the two-class classifier depends on the intra-class variance which can be in the case of image classification quite high. The unknown pose of the object is one of the major sources of the intra-class variance. Registration of the object prior to classification is one way how to decrease the intra-class variance. However, the object registration of a complex objects like faces (it is a 3D object by nature) is never perfect. On the other hand, the proposed structural classifier searches for the best possible pose of the object during the classification process which effectively drives the intra-class variance caused by the unknown pose to its minimum.

The difference is illustrated in figures 3.3b and 3.3c.



a) Two-class classifier based methods

b) For prototype $\boldsymbol{w}_-$ of our model

c) For prototype $\boldsymbol{w}_+$ of our model

Figure 3.3 Illustration of the difference between the two-class classifier based methods and the proposed structural classifier modeling the face pose explicitly. (red crosses represent examples of the negative class and blue circles represent the positive class).

Another notable advantage of the proposed approach is that the learning objective function is clearly related to the expected risk being the true objective in this application. In particular, learning of our classifier is formulated as minimization of an upper bound on the empirical risk. In the case of the method of VE, a statistical interpretation of the learning objective function is far not that clear. In the case of two stage methods combining registration and classification, deriving a compact statistical model of the two stages is also not trivial.

3.4 Standard BMRM algorithm

The Bundle Method for Risk Minimization (BMRM) [10] is a recently published optimization algorithm designed for solving the following convex problem

$$\min_{\boldsymbol{w}\in\mathbb{R}^D} F(\boldsymbol{w}) := \frac{\lambda}{2}||\boldsymbol{w}||^2 + R(\boldsymbol{w})\,, \tag{3.11}$$

where $R\colon \mathbb{R}^D \to \mathbb{R}$ is an arbitrary convex function. The risk term R is often the complex part of the objective function which makes the optimization hard. The basic idea of this algorithm is to replace the original risk function $R(\boldsymbol{w})$ by its simpler approximation $R_t(\boldsymbol{w})$. The approximation used by the BMRM algorithm is based on the cutting plane model

$$\begin{aligned} R_t(\boldsymbol{w}) &= \max_{i=0,1,\ldots,t-1} \left[R(\boldsymbol{w}_i) + \langle \nabla R(\boldsymbol{w}_i), \boldsymbol{w} - \boldsymbol{w}_i\rangle\right] = \\ &= \max_{i=0,1,\ldots,t-1} \left[b_i + \langle a_i, \boldsymbol{w}\rangle\right]. \end{aligned} \tag{3.12}$$

where $\nabla R(\boldsymbol{w})$ denotes a sub-gradient of function R at point $\boldsymbol{w}$. We use shorthands $a_i = \nabla R(\boldsymbol{w}_i)$ and $b_i = R(\boldsymbol{w}) - \langle \nabla R(\boldsymbol{w}_i), \boldsymbol{w}_i\rangle$ are introduced to simplify the notation. The cutting plane model tries to approximate risk function $R(\boldsymbol{w})$ by a convex piecewise linear function $R_t(\boldsymbol{w})$ defined as a maximum over t cutting planes. The i-th linear term $R(\boldsymbol{w}_i) + \langle \nabla R(\boldsymbol{w}_i), \boldsymbol{w} - \boldsymbol{w}_i\rangle$, called the cutting plane due to its geometrical interpretation, is a liner lower bound of a function R which is tight at the point $\boldsymbol{w}_i$. Having the risk approximation $R_t(\boldsymbol{w})$, the original objective function $F(\boldsymbol{w})$ is replaced by

$$F_t(\boldsymbol{w}) = \frac{\lambda}{2}||\boldsymbol{w}||^2 + R_t(\boldsymbol{w})\,,$$

being a lower bound on the original objective function $F(\boldsymbol{w})$. The core of the BMRM algorithm lies in replacing the original hard minimization problem (3.11) by a simpler approximated minimization problem

$$\min_{\boldsymbol{w}\in\mathbb{R}^D} F_t(\boldsymbol{w}) := \frac{\lambda}{2}||\boldsymbol{w}||^2 + R_t(\boldsymbol{w})\,. \tag{3.13}$$

The problem (3.13) is in the optimization literature as the *reduced problem*.

The BMRM Algorithm 1 iteratively solves the reduced problem (3.13) and uses the intermediate solution to improve the cutting plane model $R_t(\boldsymbol{w})$. This means that the cutting plane model $R_t(\boldsymbol{w})$, and by that also the reduced problem objective function $F_t(\boldsymbol{w})$, becomes progressively more accurate lower bound approximation of the of risk function $R(\boldsymbol{w})$ and the original objective $F(\boldsymbol{w})$, respectively. In Figure 3.4 you can see an example of a few iterations of this algorithm. Let use define $\epsilon_t = F(\boldsymbol{w}_t) - F_t(\boldsymbol{w}_t)$ which is a gap between the original and the reduced problems' objective functions. Clearly, the size of the gap ϵ_t bounds the difference between the current objective value $F(\boldsymbol{w}_t)$ and the optimal one $\min_{\boldsymbol{w}} F(\boldsymbol{w})$. The BMRM algorithm uses two stopping conditions. First stopping condition holds at the moment when gap ϵ_t drops below a predefined constant ϵ_{abs}. The second

stopping condition is satisfied if the rate $(\epsilon_t / F(\boldsymbol{w}_t))$ is lower than the predefined threshold ϵ_{rel}. The second stopping condition is more suitable in practice as it does not depend on the scale of the objective function. It can be proved that for arbitrary non-zero precession ϵ_{abs} (or ϵ_{rel}) the BMRM algorithm stops in a finite number of iterations [10].

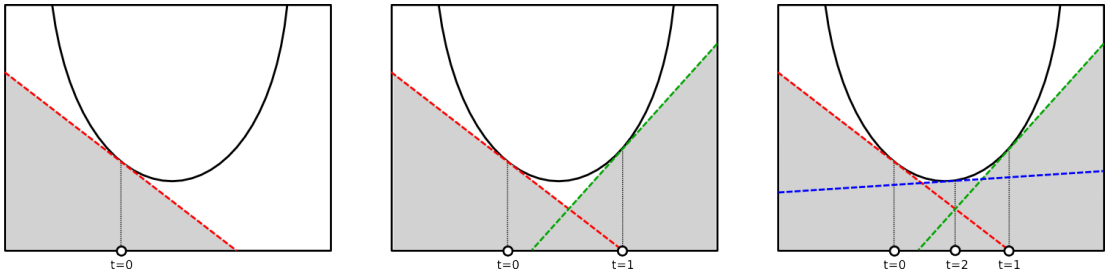


Figure 3.4 Example of BMRM algorithm iterations (black line = risk function, dashed lines = cutting planes, gray area = cutted off space, white area = solving space)

Algorithm 1 Bundle Method for Risk Minimization (BMRM) algorithm

Input: ϵ_{abs}, ϵ_{rel}, a method pointer $R(\boldsymbol{w})$, a method pointer $\nabla R(\boldsymbol{w})$

1: **initialization:** $w \leftarrow 0$, $t \leftarrow 0$
2: **repeat**
3: $t \leftarrow t + 1$
4: Compute $R(\boldsymbol{w}_t)$, $\nabla R(\boldsymbol{w}_t)$
5: Update $R_t(\boldsymbol{w}_t)$
6: Solve $\boldsymbol{w}_t \leftarrow \arg\min F_t(\boldsymbol{w})$
7: $\epsilon_t \leftarrow F(\boldsymbol{w}_t) - F_t(\boldsymbol{w}_t)$
8: **until** $\epsilon_t \leq \epsilon_{abs} \vee (\epsilon_t / F(\boldsymbol{w}_t)) \leq \epsilon_{rel}$

The BMRM algorithm transforms the original hard minimization into minimization of a simpler reduced problem (3.13) which is solved repeatedly in the main loop of the algorithm. The reduced problem (3.13) is equivalent to the following quadratic program

$$\boldsymbol{w}_t = \underset{\boldsymbol{w} \in \mathbb{R}^D}{\arg\min} \left[\frac{\lambda}{2} ||\boldsymbol{w}||^2 + \xi \right], \tag{3.14}$$

subject to

$$\langle a_i, \boldsymbol{w} \rangle + b_i \leq \xi \quad i = 0, 1, \ldots, t-1\,. \tag{3.15}$$

It is better to solve this quadratic program in its Lagrange dual formulation which we derive in the rest of this section. We create a Lagrange function $L(\boldsymbol{w}, \xi, \alpha)$ of equations 3.14 and 3.15

$$L(\boldsymbol{w}, \xi, \alpha) = \frac{\lambda}{2} ||\boldsymbol{w}||^2 + \xi - \sum_{i=0}^{t-1} \alpha_i \left[b_i + \langle a_i, \boldsymbol{w} \rangle - \xi \right]. \tag{3.16}$$

We set the partial derivatives of the Lagrangian to zero and we solve for the primal variables, that is,

$$\frac{\partial L(\boldsymbol{w},\xi,\alpha)}{\partial \boldsymbol{w}} = \lambda \boldsymbol{w} + \sum_{i=0}^{t-1} \alpha_i a_i = 0 \quad \Rightarrow \quad \boldsymbol{w} = -\frac{1}{\lambda}\sum_{i=0}^{t-1} \alpha_i a_i, \tag{3.17a}$$

$$\frac{\partial L(\boldsymbol{w},\xi,\alpha)}{\partial \xi} = 1 - \sum_{i=0}^{t-1} \alpha_i = 0 \quad \Rightarrow \quad \sum_{i=0}^{t-1} \alpha_i = 1, \tag{3.17b}$$

$$\frac{\partial L(\boldsymbol{w},\xi,\alpha)}{\partial \alpha_i} = b_i + \langle a_i, \boldsymbol{w}\rangle - \xi \quad \Rightarrow \quad \xi = b_i + \langle a_i, \boldsymbol{w}\rangle \quad i = 0, 1, \ldots, t-1. \tag{3.17c}$$

In the next step we substitute formulas for $\boldsymbol{w}$ and $\boldsymbol{\xi}$, i.e. the equations (3.17a) and (3.17b), into the minimized function and we derive the Lagrange dual quadratic program

$$\alpha_t = \arg\max_{\alpha \in \mathbb{R}^t} \left[\sum_{i=0}^{t-1} \alpha_i b_i - \frac{1}{2\lambda} \left\| \sum_{i=0}^{t-1} \alpha_i a_i \right\|^2 \right], \tag{3.18}$$

subject to:

$$\sum_{i=0}^{t-1} \alpha_i = 1,$$
$$\alpha_i \geq 0\,, \qquad i = 0, 1, \ldots t-1.$$

We can obtain the solution $\boldsymbol{w}_t$ of the primal formulation of the reduced problem (3.14) from the solution $\boldsymbol{\alpha}_t$ of the dual problem (3.18) using the equation (3.17a). The advantage of solving the dual program lies in the smaller number of variables which, in the case of the dual program, corresponds to the iteration number of the BMRM algorithm. In practice, the number of iterations of the BMRM algorithm is much lower than the number of variables of the primal reduced problem being equal to the dimensionality of the parameter space.

This algorithm has been already implemented in STPR Toolbox for MATLAB[4]. To run this implementation of the algorithm you must provide to it methods for computation of value of risk $R(\boldsymbol{w})$ and for a computation of subgradient of risk $\nabla_w R(\boldsymbol{w})$ at point $\boldsymbol{w}$. For our proposed classifier method for computation of risk function is represented by equation 3.10 and computation of its subgradient has following form

$$\begin{aligned} \nabla_w R(\boldsymbol{w}) &= \frac{1}{m}\sum_{i=1}^{m} \left\{ \Phi(I_i, \hat{z}_i, \bar{y}_i) - \Phi(I_i, z_i, y_i) \right\}, \\ \hat{z}_i &= \arg\max_{z \in \mathbb{Z}} \langle \boldsymbol{w}, \Phi(I_i, z, \bar{y}_i) \rangle. \end{aligned} \tag{3.19}$$

3.5 Improved BMRM algorithm (P-BMRM)

The computational time required by the BMRM algorithm can be large especially in vision problems we are interested in. The bottle neck of the algorithm

is computation of the risk $R(\boldsymbol{w})$ and its subgradient $\nabla R(\boldsymbol{w})$. In this situation every saved iteration of the BMRM algorithm can significantly reduced the over computational time. We propose an improvement of the BMRM algorithm which reduces the number of iterations without increasing per iteration time. The improvement lies in modifying of the cutting plane model $R_t(\boldsymbol{w})$ by its partitioning into P terms, that is,

$$R_t(\boldsymbol{w}) = \sum_{i=1}^{P} R_t(\boldsymbol{w}, p), \tag{3.20}$$

$$\begin{aligned} R_t(\boldsymbol{w}, p) &= \max_{i=0,1,\ldots,t-1} [R(\boldsymbol{w}_i, p) + \langle \nabla R(\boldsymbol{w}_i, p), \boldsymbol{w} - \boldsymbol{w}_i \rangle] = \\ &= \max_{i=0,1,\ldots,t-1} [b_i + \langle a_i, \boldsymbol{w} \rangle] \qquad p = 1, \ldots, P, \end{aligned} \tag{3.21}$$

$$R(\boldsymbol{w}) = \sum_{p=1}^{P} R(\boldsymbol{w}, p), \quad R(\boldsymbol{w}, p) = \sum_{i}^{M_p} \mathcal{L}(y_i, h(x_i)), \tag{3.22}$$

where function $R(\boldsymbol{w}, p)$ denote value of the term p of the risk function at point $\boldsymbol{w}$ and $\nabla R(\boldsymbol{w}, p)$ denote a sub-gradient of function $R(\boldsymbol{w}, p)$ at point $\boldsymbol{w}$. M_p denote a set of examples belonging into term p and $\mathcal{L}\colon Y \times Y \to \mathbb{R}$ denote a general loss function. We denote the modified BMRM algorithm as the P-BMRM algorithm.

The standard BMRM algorithm improves in each iteration the cutting plane model by adding a single linear term which require one pass over the whole training examples. The proposed P-BMRM algorithm adds a single linear term to each of the P partial cutting plane models which still requires a single pass over the examples. Thanks to that, the P-BMRM algorithm uses P times more cutting planes for compared to the standard BMRM algorithm. Clearly, more cutting planes better approximates the risk function.

Modifications of the BMRM algorithm incurred by the proposed finer cutting plane model are only minor. We described them below. For notational simplicity we introduce variables $a_{i,p}$ and $b_{i,p}$ instead of a_i, b_i

$$\begin{aligned} a_{i,p} &= \nabla R(\boldsymbol{w}_i, p), \\ b_{i,p} &= R(\boldsymbol{w}_i, p) - \langle \boldsymbol{w}_i, \nabla R(\boldsymbol{w}_i, p) \rangle. \end{aligned}$$

The pseudo-code of the proposed P-BMRM algorithm is outlined in Algorithm 2.

The reduced problem of the P-BMRM algorithm leads to solving the following convex quadratic program

$$\boldsymbol{w}_t = \underset{\boldsymbol{w} \in \mathbb{R}^D}{\arg\min} \left[\frac{\lambda}{2} ||\boldsymbol{w}||^2 + \sum_{p=1}^{P} \xi_p \right] \tag{3.23}$$

subject to

$$\begin{aligned} b_{i,p} + \langle a_{i,p}, \boldsymbol{w} \rangle \leq \xi_p \quad & i = 0, 1, \ldots, t-1, \\ & p = 1, 2, \ldots, P. \end{aligned}$$

Algorithm 2 Proposed P-BMRM algorithm

Input: ϵ_{abs}, ϵ_{rel}, method pointer $R(\boldsymbol{w}, p)$, method pointer $\nabla R(\boldsymbol{w}, p)$

1: **initialization:** $w \leftarrow 0$, $t \leftarrow 0$
2: **repeat**
3: $t \leftarrow t + 1$
4: Compute $R(\boldsymbol{w}_t, p)$, $\nabla R(\boldsymbol{w}_t, p)$ $p = 1, 2, \ldots, P$
5: Update $R_t(\boldsymbol{w}_t, p)$ $p = 1, 2, \ldots, P$
6: Solve $\boldsymbol{w}_t \leftarrow \arg\min F_t(\boldsymbol{w})$
7: $\epsilon_t \leftarrow F(\boldsymbol{w}_t) - F_t(\boldsymbol{w}_t)$
8: **until** $\epsilon_t \leq \epsilon_{abs} \vee (\epsilon_t / F(\boldsymbol{w}_t)) \leq \epsilon_{rel}$

We again use Lagrange multipliers to derive the dual formulation of the reduced problem (3.23). After applying similar procedure as in the previous section we get the following dual quadratic program

$$\alpha_t = \arg\max_{\alpha \in \mathbb{R}^t} \left[\sum_{i=0}^{t-1} \sum_{p=1}^{P} \alpha_{i,p} b_{i,p} - \frac{1}{2\lambda} \left\| \sum_{i=0}^{t-1} \sum_{p=1}^{P} \alpha_{i,p} a_{i,p} \right\|^2 \right] \tag{3.24}$$

subject to

$$\begin{aligned} \sum_{i=0}^{t-1} \alpha_{i,p} &= 1 \qquad p = 1, 2, \ldots, P, \\ \alpha_{i,p} &\geq 0 \qquad i = 0, 1, \ldots t-1 \quad , p = 1, 2, \ldots, P. \end{aligned}$$

The transformation of the dual solution to the primal solution is given by the formula

$$\boldsymbol{w}_t^* = -\frac{1}{\lambda} \sum_{i=0}^{t-1} \sum_{p=1}^{P} \alpha_{i,p}^* a_{i,p}. \tag{3.25}$$

By using the finer cutting plane approximation we reduce number of iterations without increasing the time-complexity. However, the finer approximation comes at the cost of higher memory requirements needed for storing the cutting plane model. The memory complexity of the proposed P-BMRM algorithm is $\Theta(nIter \cdot D \cdot P)$ compare to $\Theta(nIter \cdot D)$ of the standard BMRM. It is seen, that the memory complexity grows linearly with the parameter P.

Analogously to standard BMRM, you must provide to algorithm two methods. First, for computation term p of risk function $R(\boldsymbol{w}, p)$ and second its subgradient

$\nabla_w R(\boldsymbol{w}, p)$. For our classifier this functions have following form

$$R(\boldsymbol{w}, p) = \frac{1}{m} \sum_{i}^{M_p} \max \left\{ 0, \max_{z \in \mathbb{Z}} \left[1 - \langle \boldsymbol{w}, \Phi(x_i, z_i, y_i) \rangle + \langle \boldsymbol{w}, \Phi(x_i, z, \bar{y}_i) \rangle \right] \right\}, \tag{3.26}$$

$$\begin{aligned} \nabla_w R(\boldsymbol{w}, p) &= \frac{1}{m} \sum_{i}^{M_p} \left\{ \Phi(I_i, \widehat{z}_i, \bar{y}_i) - \Phi(I_i, z_i, y_i) \right\}, \\ \widehat{z}_i &= \arg\max_{z \in \mathbb{Z}} \langle \boldsymbol{w}, \Phi(I_i, z, \bar{y}_i) \rangle. \end{aligned} \tag{3.27}$$

3.6 Image feature descriptors

Our structural classifier requires a feature descriptor of the image transformed to the fixed size *base-window*. In this section, we list the image feature descriptors we experimented with in this thesis. There is a lot of approaches in the literature how to represent image data. A naive representation using directly the image intensity values usually fails because it drastically changes with varying lighting conditions. Here we consider only the image representations which are known to be in large extent invariant to the change of the lighting conditions.

Because we need to represent an input image that contains a near-frontal face a good candidate could be features originally designed for texture recognition. In particular, we use two descriptors based on the Local Binary Patterns (LBP) and a descriptor using the Histogram Of Gradients (HOG).

3.6.1 Pyramid of Local Binary Patterns (LBP-pyramid)

A lot of approaches dealing with face images are based on the LBP descriptor originally published in [6]. These descriptors have been proved successful in texture recognition and face recognition.

This method thresholds a brightness value of neighbouring pixels. In our case, we use an eight-neighbourhood for computation features for a given pixel. The procedure computing LBP on one pixel can be divided into several stages (see Figure 3.5 for graphical illustration of the computation):

- Thresholding intensity values of the eight-neighbourhood with a intensity value of the center pixel x,y
- Representing the thresholding result in a single byte
- Encoding the byte (0 - 255) by the one-hot code into a 256 bit long logical vector
- Repeating the previous steps for each pixel in the image and stacking the results into a single feature vector Ψ representing the image

Because each pixel is encoded by the one-hot code, it is clear that vector Ψ contains mostly zeros. Precisely, only 1/256 of vector values are ones. Hence it is rational to choose a sparse representation for the features. We will denote the feature vector in sparse representation by Ψ^{sparse}. In particular, we store only

indexes of ones in this feature vector. This approach allows us to evaluate the dot-product $\langle \Psi, x \rangle$ just by summing up the elements of x addressed by the indexes in Ψ^{sparse}, that is,

$$\langle \Psi, \boldsymbol{x} \rangle = \sum_{i=1}^{D/256} \boldsymbol{x}_{\Psi_i^{sparse}} .$$

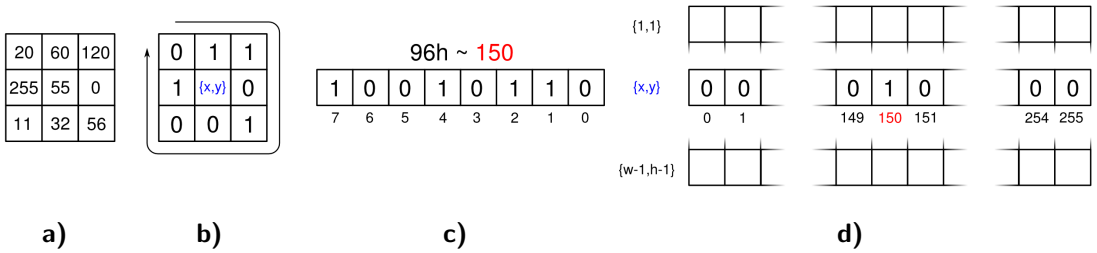


Figure 3.5 Computation of the LBP for one pixel, a) input pixel with its eight-neighbourhood, b) thresholding brightness values, c) rewriting texture into a byte, d) encode texture byte by one-hot code

When we use the previous approach we loose a relation between pixels that are not directly neighbouring. In order to increase discriminative power of the image descriptor we compute the LBP features for an image pyramid derived from the input image and we stack the LBP descriptors to a single vector. The number of levels in the pyramid is $nPyramids$. The first level of the pyramid is the original image. A higher level in the pyramid is obtained by downscaling the nearest lower level by factor of two. That is, to create all levels of the image pyramid we downscale the original image by factor of $\frac{1}{2}, \frac{1}{4} \ldots \frac{1}{nPyramids-1}$. In the Algorithm 3 you can see a pseudo-code of the LBP-pyramid feature descriptor used in the experiments.

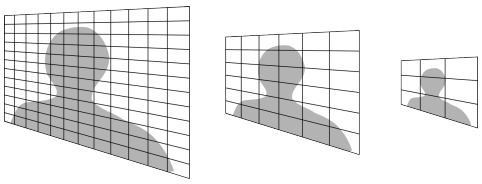


Figure 3.6 Image pyramid of height $nPyramids = 3$.

The LBP-pyramid feature descriptor has a single parameter to be tuned, i.e. $nPyramids$ which is the number of levels in the pyramid.

3.6.2 Histogram Of Gradients (HOG)

The Histogram Of Gradients (HOG) descriptor [2] was first used in the state-of-the-art detector of pedestrians. Our choice for using HOG descriptor in our application was mainly motivated by a need to have a lower dimensional alternative to the high dimensional LBP descriptor.

Implementation of this method can be divided into following stages:

Image normalization It is suitable to normalize the input image in order to make the descriptor less sensitive to illumination changes. We apply a linear

Algorithm 3 LBP-pyramid image feature descriptor

Input: $I \in \mathbb{Z}^{imH \times imW}$, $nPyramids \in \mathbb{Z}$

initialization: $\Psi = \emptyset$
for $p = 0 \rightarrow (nPyramids - 1)$ **do**
  for $x = 1 \rightarrow (\lfloor imW/2^p \rfloor - 1)$ **do**
    for $y = 1 \rightarrow (\lfloor imH/2^p \rfloor - 1)$ **do**
      $\psi \leftarrow$ LBP label at pixel x, y in image I
      $\Psi' = \Psi$
      $\Psi = \begin{bmatrix} \Psi' \\ \psi \end{bmatrix}$
    end for
  end for
  $I \leftarrow$ shink image I to a half in both axis
end for

function to the intensity values so that the whole range of intensity values (i.e, values from 0 to 255) is represented in the normalized image.

Computing gradient We compute a gradient by applying convolution between the normalized image and the filter $[-1 \quad 0 \quad 1]$ in both axis. The gradient is represented by a magnitude $M(x, y)$ and an angle $\alpha(x, y)$. Because we do not care if the gradient is growing or descending we use only unsigned angles, i.e. $\alpha(x, y) \in \langle 0; \pi)$.

Creating cells The whole image is divided into cells of square grid whose edges are formed by $cellSize$ pixels (see Figure 3.7a).

Computing cell histograms The gradients in each cell are histogramed (see Figure 3.7b). Let h_i where $i = 1, 2, \ldots, nBins$ denote histogram bins and $(h_n, h_m\rangle$ denote the smallest possible interval where angle $\alpha(x, y)$ fits. Every pixel with an angle $\alpha(x, y)$ and a magnitude $M(x, y)$ votes into histogram bins h_n, h_m by a linear combination

$$H(n) = H(n) + \frac{\alpha(x,y) - h_n}{h_m - h_n} M(x,y),$$
$$H(m) = H(m) + \frac{h_m - \alpha(x,y)}{h_m - h_n} M(x,y) \quad \textbf{,where } h_n \leq \alpha(x,y) < h_m.$$

Creating blocks Blocks have a squared shape and their edge is formed by $blockSize$ cells. These blocks are overlapping. In our case every block overlaps a half of its neighbouring blocks (see Figure 3.7c). This means that each cell (excluding the boundary cells) is present in four blocks.

Normalize block histogram A block histogram is formed by all its cell histograms. In our case we normalize this block histogram so that the votes sum is equal to one.

Stacking histograms together All block histograms are stacked together into one feature vector Ψ. This vector represents the input image (see the Figure 3.7d).

Quadratic features We extend a general HOG descriptor by appending a vector of squared values of the original HOG. That is, let Ψ' be the HOG descriptor of a given image, then we construct a vector

$$\Psi = \begin{bmatrix} \Psi' \\ \Psi'^2 \end{bmatrix}.$$

The quadratic features extend the discriminative power of the linear score function to a diagonal quadratic function.

The HOG descriptor requires to tune the following parameters: *cellSize*, *blockSize* and $nBins$.

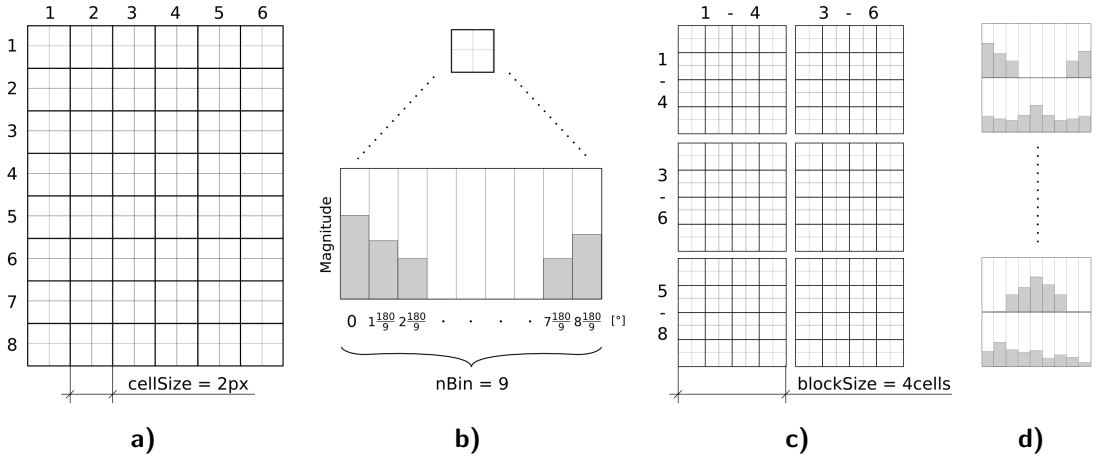


Figure 3.7 Computing a HOG description, a) dividing image into cells, b) Computing cell histograms, c) creating blocks, d) linking block histograms together

3.6.3 Histogram Of LBP (HLBP)

The last image descriptor we use in this thesis is the Histogram of LBPs. This descriptor is in a sense a fusion of descriptors described above. At the beginning we divide image into cells like the HOG descriptor does. We compute LBP byte codes. The LBP codes are in each cell are histogramed. The final descriptor is created by stacking the cell histograms to a single vector. Compared to the LBP-pyramid feature descriptor, the descriptor based on the histogram of LBPs has much lower dimension.

This descriptor has a single parameter to tune, i.e. the *cellSize*.

4 Experiments

In this chapter we experimentally evaluate the proposed structural classifier and we compare our method against two different approaches. This chapter is organized as follows. In section 4.1 we describe testing protocol used in all experiments. Image database, its partitioning and pre-processing methods are described in section 4.2. In section 4.3 we evaluate one base-line method based on the linear SVM classifier using LBP-pyramid features and one strong competitor, namely, the SVM classifier trained from the virtual examples which has been previously proved successful for gender recognition. Experimental tuning of the parameters of the proposed structural classifier and its evaluation is described in section 4.4. The main results are summarized in section 4.5. Finally, in section 4.6, we present a preliminary comparison of the proposed P-BMRM learning algorithm against the standard BMRM.

4.1 Testing protocol

Goal of the testing protocol is to estimate the expected classification error of the gender classifier learned from a given set of annotated examples. All investigated methods have two kinds of parameters. The first kind of parameters are trained from training set of examples using a given learning algorithm. The second kind of parameters (also called hyper-parameters) are tuned (i.e., selected from a finite set of candidates) on a validation set. The estimate of the expected classification error is computed on an independent testing set. To sum up, the testing protocol involves the following three components:

Training In the training stage the parameters of the classifier are learned from the *training set* for given setting of the hyper-parameters.

Validation In the validation stage the classifiers trained for different setting of the hyper-parameters are evaluated on the *validation set* in terms of the classification error. The classifier with the minimal classification error estimated on the validation set is selected to be the resulting classifier.

Testing The classification error of the resulting classifier selected in the validation stage is computed on the *testing set*. The test error is our estimate of the

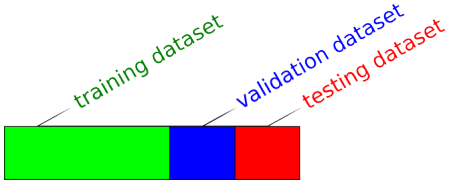


Figure 4.1 Partitioning of the examples.

true (expected) classification error.

In order to implement the test protocol, we randomly split the database of example images into three independent parts: training, validation and testing set (see Figure 4.1). The splits are generated randomly to guarantee that the distribution of the examples is similar.

We use the following notation to describe the results:

$e_{trn}, e_{val}, e_{tst}$	stands for the training, validation and testing error, respectively.
$\Rightarrow$	Indicates the setting with the minimal validation error.
M	Indicates the experiment which did not finish due to lack of memory.

4.2 Database description

In the experiments we use a proprietary image database provided by courtesy of the Eyedea recognition Ltd [8]. The database contains images of faces along with their manual annotation. The annotation is provided for gender (male/female) and position of the main facial features: centers of the eyes, tip of the nose and the center of the mouth. The database is composed from several sources like images downloaded from the Internet, images captured by web cameras, images form standard face recognition databases (FERET, LFW, M2VTS) etc. Images in the database exhibits a large diversity in their quality (different resolution, noise, sharpness), background, lighting conditions, race of the people, age of the people etc.

According to the testing protocol, we divide this database into training, validation and testing set. The training dataset consists of 14,000 near-frontal facial images. The validation and testing sets have 6,000 images each. The distribution of male and female examples is exactly uniform. Table 4.1 summarizes the distribution of the images into the training, validation and testing set as well as the size of faces (the parameter Δe defined below).

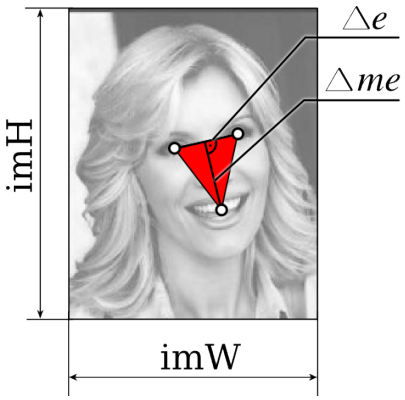


Figure 4.2 Illustration of variables for alignment (red area denote $\triangle EEM$)

Now we describe the procedure used to generate the input images for the classifier. We define the input image size $[imW \times imH]$ to be $[64 \times 80]$px, that is, after processing all example images in the training/validation/testing sets have this fixed size. In the experiments we use input images with faces aligned by three different methods. Prior to describing the alignment methods, we introduces several auxiliary variables (see figure 4.2). We define Δe to be a distance between the centers of the left and the right eye. Next, we define distance, denoted by Δme, between the center of the mouth and the closest point on the line connecting

Table 4.1 Statistics of the image database

Δe [px]	Dataset Training	Validation	Testing
$0 < \Delta e \leq 15$	1	0	0
$15 < \Delta e \leq 30$	824	328	297
$30 < \Delta e \leq 60$	8 145	3 482	3 554
$60 < \Delta e \leq 90$	2 648	1 149	1 152
$90 < \Delta e \leq 180$	2 178	955	912
$180 < \Delta e \leq 360$	183	75	76
$360 < \Delta e \leq 720$	20	11	9
$720 < \Delta e$	1	0	0
male	7,000	3,000	3,000
female	7,000	3,000	3,000
total	14,000	6,000	6, 000

the eyes. By $\triangle EMM$ we denote the triangle defined by the centers of the eyes and the mouth. Finally, we use $C_{\triangle EEM}$ to denote the centroid of the triangle $\triangle EMM$.

From the image database we estimated the average distances

$$E(\Delta e) = 14.62\,, \tag{4.1}$$

$$E(\Delta me) = 16.61\,, \tag{4.2}$$

which are used to define the aligned faces.

Now we introduce three methods used in the experiments to generated example images with different face alignment:

Unaligned To create unaligned examples, we use bounding boxes generated by a real AdaBoost-based face detector. To ensure that the whole face is contained in the input image we enlarge the bounding box by a constant 1.6. Next, the height of the enlarge bounding box is increased to match the aspect ration of the input image size $[imW \times imH]$. Finally, the image defined by the resulting bounding box is rescaled to the input image size $[imW \times imH]$.

Aligned position, scale and rotation (PSR) We define a normalized pose of the face in the input image so that: i) the center of the input image coincides with the centroid $C_{\triangle EEM}$ and ii) the line connecting the eyes is parallel to the horizontal edge of the input image and iii) the distance Δe is set to 14.62. The input PSR aligned images are generated by applying an affine transform which brings the original face to its normalized pose.

Fully aligned This method extends the PSR alignment by enforcing the aligned faces to have the distance Δme set to 16.61.

The unaligned images (the first method) simulate a realistic input to the gender classifier. Hence, the unaligned images are used in computation of the validation and the test errors.

4.3 Competing methods

In this section we describe methods used to compare with our approach. We use one base-line approach described in subsection 4.3.1 and one strong competitor outlined in subsection 4.3.2.

4.3.1 Baseline SVM classifier

Our baseline is a standard two-class linear SVM classifier. This approach does not care about the fact that the input faces are not aligned. The feature vector is generated by applying the LBP-pyramid descriptor on the base-window cropped from the input image. The centers of the base-window and the input image coincide. The number of levels of the LBP pyramid was set to $nPyramids = 4$. The classifier is trained on the unaligned images. The validation and the test errors are computed on the unaligned images as well. The classifier has two hyper-parameters to be tuned on the validation set: i) the SVM regularization constant C and ii) the size of the base window. To find the best C, we fixed the base-window size to $[40 \times 60]$ pixels and we computed the validation error for a range of C. The results for tuning C are listed in Table 4.2.

Table 4.2 Tuning parameter C with the *base-window* size set to $[40 \times 60]$px

	C	$e_{trn}[\%]$	$e_{val}[\%]$
	1×10^{-4}	7.02	14.20
$\Rightarrow$	1×10^{-3}	0.01	11.32
	1×10^{-2}	0.00	11.45

To tune the optimal base-window size, we used the constant $C = 1 \times 10^{-3}$ obtained from the first experiment, and we computed the validation error for various base-window sizes. For the base-window width we tried values from 34 to 48 and for its height the values from 44 to 66. The results are summarized in Table 4.3. Finally, we repeated tuning of the parameter C with the best base-window size $[46 \times 58]$ obtained in the previous experiment. The optimal value for C remained the same, i.e. $C = 1 \times 10^{-3}$. Thus, the best configuration of the hyper-parameters was $C = 1 \times 10^{-3}$ and the base-window size $[46 \times 58]$.

Table 4.3 Five *base-window* sizes with the lowest validation errors for $C = 1 \times 10^{-3}$

	height [px]	width [px]	$e_{trn}[\%]$	$e_{val}[\%]$	$e_{tst}[\%]$
$\Rightarrow$	58	46	0.01	10.63	10.77
	66	44	0.00	10.65	
	64	44	0.00	10.77	
	64	46	0.00	10.82	
	66	48	0.00	10.82	

Table 4.4 Confusion matrix of classifier learned by baseline method

	ESTIMATED CLASS		ROW NORMALIZED	
TRUE CLASS	Male	Female	Male	Female
Male	45.27%	4.73%	90.53%	9.47%
Female	6.03%	43.97%	12.07%	87.93%

The testing classification error computed for the best hyper-parameter configuration was $e_{tst} = 10.77\%$.

4.3.2 SVM classifier trained from virtual examples

We used the COFFIN framework [7] for large-scale learning of two-class linear SVM classifier from virtual examples. In this approach, a set of transformations which do not change the class membership are applied on the PSR aligned images to generate synthetic (so called virtual) examples. The synthetically generated virtual examples are then used to learn the two-class linear SVM classifier. We consider set of transformations which exactly corresponds to the face pose parameter $z \in Z$ used in our structured classifier. The set of poses Z is composed as a Cartesian product of Z_x, Z_y, Z_α, Z_{scale} and Z_{ratio} describing change in position (Z_x, Z_y), face rotation (Z_α) and scale (Z_{scale}). As a result the original $m = 14,000$ training examples are expanded to $m \cdot |Z| = 10,290,000$ virtual examples. In particular, we use the following setting

$$\begin{aligned} Z_\alpha &= \{-\frac{\pi}{30}, -\frac{\pi}{60}, 0, \frac{\pi}{60}, \frac{\pi}{30}\} \\ Z_{scale} &= \{0.9, 1, 1.1\} \\ Z_x \times Z_y &= \{-3; -2; \ldots, 2; 3\} \times \{-3; -2; \ldots; 2; 3\} \\ Z &= (Z_x \times Z_y \times Z_\alpha \times Z_{scale}) \end{aligned}$$

The feature vectors were computed by applying the LBP-pyramid descriptor on the base-window transformed by the corresponding $z \in Z$. We used the base-window of the size $[40 \times 60]$ pixels. The number of levels of the LBP pyramid was set to $nPyramids = 4$.

Table 4.5 Testing error of classifier learned by VE method

	C	$e_{trn}[\%]$	$e_{val}[\%]$	$e_{tst}[\%]$
	10^-5	6.27	7.92	
$\Rightarrow$	10^-4	4.05	7.13	8.20
	10^-3	3.11	10.03	
	10^-2	1.81	8.75	

Table 4.6 Confusion matrix of classifier learned by VE method

	ESTIMATED CLASS		ROW NORMALIZED	
TRUE CLASS	Male	Female	Male	Female
Male	44.42%	5.58%	88.83%	11.17%
Female	2.62%	47.38%	5.23%	94.77%

4.4 Evaluation of the proposed structural classifier

This section is split into two parts. First, in section 4.4.1 we describe tuning of the hyper-parameters of the structural classifier. Second, in section 4.4.2, we evaluate the benefits of the structural classifier coming from explicitly modeling the unknown face pose.

4.4.1 Tuning of the parameters

Experiments described in this section 4.4.1 use the standard two-class linear SVM classifier to evaluate setting of various hyper-parameters of the structural SVM classifier. We used the two-class linear SVM because its training is much more time efficient compared to the structural classifier. We train and evaluate the two-class SVM classifier on (PSR or fully) aligned examples because we assume that the variance due to the unknown face pose will be removed when using the found hyper-parameter setting in the structural classifier.

4.4.1.1 LBP-pyramid descriptor

The LBP-pyramid descriptor (c.f. section 3.6.1) has two parameters. The size of the base-window and the number of levels of the LBP pyramid. We described tuning of these two parameters in two independent paragraphs.

***Base-window* size of the LBP-pyramid descriptor** The goal here is to find the optimal base-widow size to separate aligned female and male images when using the LBP-pyramid descriptor. The experimental procedure is identical to the one used for the baseline SVM classifier in section 4.3.1. The only difference is that for training and validation we use images with aligned faces. We consider the PSR aligned and fully aligned images separately. Table 4.7 shows results for the PSR aligned examples. Results for the fully aligned examples are in Table 4.8. We use notation e_{val}^{PSR} and e_{val}^{full} to distinguish from the validation error e_{val} evaluated on unaligned examples.

Table 4.7 Results of tuning the *Base-window* size of the LPB-pyramid descriptor on PSR aligned faces.

	C	$e_{trn}[\%]$	$e_{val}^{PSR}[\%]$		height [px]	width [px]	$e_{trn}[\%]$	$e_{val}^{PSR}[\%]$
	10^{-4}	4.14	9.17	⇒	64	36	0.00	6.26
	10^{-3}	0.01	6.37		62	38	0.00	6.50
⇒	10^{-2}	0.00	6.27		64	38	0.00	6.58
	10^{-1}	0.00	6.28		56	38	0.00	6.58

Table 4.8 Results of tuning the *Base-window* size of the LPB-pyramid descriptor on fully aligned faces.

	C	$e_{trn}[\%]$	$e_{val}^{full}[\%]$		height [px]	width [px]	$e_{trn}[\%]$	$e_{val}^{full}[\%]$
	10^{-4}	3.21	8.03	⇒	56	46	0.00	5.80
⇒	10^{-3}	0.00	5.80		50	46	0.01	5.97
	10^{-2}	0.00	5.88		48	44	0.02	5.98
	10^{-1}	0.00	5.86		56	44	0.01	5.98

Number of levels $nPyramid$ of the LBP-pyramid descriptor The LBP-pyramid descriptor stacks LBP features computed on a image pyramid with $nPyramid$ levels. The image on the i-th level of the pyramid is obtained by downscaling the original image by factor of 2^{i-1} (c.f. section 3.6.1). The goal of this section is to find the optimal number of levels of the LBP pyramid. We used PSR aligned examples for training and validation. The SVM regularization constant C was set to 0.01. The base-window of size $[36 \times 64]$ was used.

Table 4.9 Tuning parameter $nPyramids$ of improved LBP feature

	$nPyramids$	$e_{trn}[\%]$	$e_{val}^{PSR}[\%]$
	1	0.02	7.73
	2	0.00	6.98
	3	0.00	6.43
⇒	4	0.00	6.27

We varied the number of levels $nPyramid$ from 1 to 4. The setting $nPyramid = 4$ for the base-window $[36 \times 64]$ is the limit value because the upper most level of the pyramid is of the size $[4 \times 8]$. Recall that the LBP feature is computed from a window $[3 \times 3]$.

Results summarized in table 4.9 show that the classifier using LBP features computed only on the original image ($nPyramid = 1$) has the lowest accuracy. Adding LBP features computed on other images in the pyramid gradually improve the accuracy. The highest improvement was achieved by adding the second level ($nPyramid = 2$). For higher levels the improvement is not that significant. In the experiments with the structural classifier described below we use the best setting $nPyramid = 4$.

4.4.1.2 HOG parameters

HOG descriptor itself (see section 3.6.2) has four parameters excluding the *base-window* size. Tuning all HOG parameters is a problem with six degrees of freedom (DoF): *cellSize*, *blockSize*, *nBins*, *useSquare* and *base-window* size. For tuning we used again linear SVM but this time with HOG feature descriptor. This fact adds to whole problem another DoF. Firstly, we examined parameter C of SVM model. Similar to tuning *base-window* size of LBP, we continued with a *base-window* size tuning.

Table 4.10 *Base-window* size tuning experiment for HOG

	C	e_{trn}[%]	e_{val}[%]		height [px]	width [px]	e_{trn}[%]	e_{val}[%]
	10^{-4}	8.42	10.18	⇒	64	40	3.40	7.45
⇒	10^{-3}	3.61	7.68		64	42	3.03	7.55
	10^{-2}	0.30	9.37		60	42	3.33	7.57
	10^{-1}	0.00	9.92		64	44	2.99	7.57

To optimize remaining parameters we search them in four dimensional parameter space. Let $P = (P_{cellSize}, P_{blockSize}, P_{nBins}, P_{useSquare})$ be this searching parameter space. So for this searching we used following value.

$$\begin{aligned} P_{cellSize} &= \{2, 3, 4, 5, 10\} \\ P_{blockSize} &= \{2, 3, 4, 5, 8, 10\} \\ P_{nBins} &= \{6, 9, 12, 15\} \\ P_{useSquare} &= \{\neg useSquare, useSquare\} \end{aligned}$$

Table 4.11 Tuning of HOG descriptor parameters

$\neg useSquare$					
	cellSize	*blockSize*	*nBins*	e_{trn}	e_{val}
	2	3	6	5.89	8.25
	2	3	9	5.39	8.28
	2	3	12	4.96	8.35
	2	2	6	4.76	8.79
useSquare					
	cellSize	*blockSize*	*nBins*	e_{trn}	e_{val}
⇒	2	3	6	5.24	7.97
	2	3	9	4.79	8.18
	2	3	12	4.51	8.18
	2	2	6	5.43	8.35

Results in Table 4.11 shown the advantage of using *useSquare* extension which maps problem on parabola. In the paper [2] describing HOG descriptor for pedestrian description authors estimate the best number *nBins* to 9 by measuring. It

is interesting that for face description a lower resolution of this parameter is more suitable than a higher one used in the person detection.

By this procedure we reach the best results with *base-window* size $[40 \times 64]$ pixels, parameters $cellSize = 2$, $blockSize = 3$, $nBins = 6$ and enabled $useSquare$ parameter.

4.4.1.3 HLBP parameters

The Histogram of LBP (HLBP) descriptor (c.f. section 3.6.3) has two parameters to be tuned: *cellSize* and the *base-window* size. For tuning *base-window* we use the same procedure as for the previous descriptors. Results are summarized in table 4.12.

Table 4.12 *Base-window* tuning experiment for HLBP descriptor with cellSize = 2.

	C	$e_{trn}[\%]$	$e_{val}[\%]$		height [px]	width [px]	$e_{trn}[\%]$	$e_{val}[\%]$
	10^{-4}	4.64	8.48	⇒	64	40	0.00	7.11
⇒	10^{-3}	0.12	7.45		64	42	0.00	7.13
	10^{-2}	0.00	7.62		62	40	0.01	7.20
	10^{-1}	0.00	7.63		66	38	0.01	7.30

As we want to reduce the feature dimension we choose values of the parameter *cellSize* to be greater than 3. Specifically, we tried values $4, 5, 6, 8, 10$. Table 4.13 summarizes the results.

Table 4.13 Tuning parameter *cellSize* of HLBP descriptor

	cellSize	$e_{trn}[\%]$	$e_{val}[\%]$
⇒	4	0.56	8.47
	5	1.10	8.78
	6	1.79	9.55
	8	3.81	10.67
	10	5.54	12.27

4.4.2 Evaluation of the proposed structural classifier

In this section we first evaluate impact of modeling the unknown faces pose on the classification accuracy. In this evaluation we use only the LBP-pyramid descriptor which was found better than the other considered descriptors (more details below). In section 4.4.2.1 we examine the structural classifier modeling only the translation, in section 4.4.2.2 we model translation, rotation and scale and finally, in section 4.4.2.3, we model all pose transformations including the aspect-ratio.

In section 4.4.2.4 we evaluate the structural classifier using the HOG descriptor and, in section 4.4.2.5, the structural classifier with HLBP descriptor.

4.4.2.1 Structural classifier invariant to translation

In this experiment we evaluate the proposed structural classifier which models only the translation of the recognized face. This means, that the set of the pose parameter $z \in Z$ is formed by all possible positions of the *search-window*. To speed up learning, we chose only even positions in both axes of the translation transformations. However, in the validation and testing stages all possible positions of the search-window are considered. We used the PSR aligned examples for training and unaligned examples for validation and testing. The setting for the base-window size was taken from the experiment in section 4.4.1.1. In particular, the settings used in this experiment were as follows:

$$[winH, winW] = [64, 36]\text{px} \qquad Z_\alpha = \{0\}$$
$$I \in \mathbb{R}^{80x64xm} \qquad Z_{scale} = \{1\}$$
$$nPyramids = 4 \qquad Z_{ratio} = \{1\}$$
$$\textbf{training:} \quad Z_x = Z_y = \{\ldots; -2; 0; 2; \ldots\}$$
$$\textbf{valid./testing:} \quad Z_x = Z_y = \{\ldots; -1; 0; 1; \ldots\}$$

Table 4.14 Tuning of the structural classifier invariant to translation.

	λ	$nIter$	$e_{trn}[\%]$	$e_{val}[\%]$	$e_{tst}[\%]$
	10	12	17.69	23.95	
	1	44	5.09	15.67	
$\Rightarrow$	0.1	211	0.93	11.48	11.58
M	0.01	> 1000	–.–	–.–	

Table 4.15 Confusion matrix of the structural classifier invariant to translation.

	ESTIMATED CLASS		ROW NORMALIZED	
TRUE CLASS	Male	Female	Male	Female
Male	47.52%	2.48%	95.03%	4.97%
Female	9.10%	40.90%	18.20%	81.80%

Table 4.14 shows results for tuning the regularization parameter λ on the validation set as well as the test error for the best setting of λ. Unfortunately, the found λ does not need to be optimal as the experiment for its lowest value did not finish due to lack of memory and time to fix the problem. However, based on analyzing obtained validation errors we do not expect significant improvement.

Table 4.15 shows the confusions matrix estimated on the testing examples. Responses of the structural classifier obtained on a sample of testing examples are visualized in figure 4.3. The figure shows both the estimated gender (color of the search-window) as well as the estimated pose of the search-window.

Figure 4.3 Visualization of responses of the structural classifier invariant to translation on a sample of testing examples. The box shown corresponds to the estimated face pose. Color of the box corresponds to the predicted gender.

4.4.2.2 Structural classifier invariant to translation, scale and rotation

In this section we examine the impact of modeling rotation and scale in addition to the translation of the face. In particular, we use the following setting of the structural classifier:

$$[winH, winW] = [64, 36]\text{px} \qquad Z_\alpha = \{-\frac{\pi}{30}, -\frac{\pi}{60}, 0, \frac{\pi}{60}, \frac{\pi}{30}\}$$

$$I \in \mathbb{R}^{80x64xm} \qquad Z_{scale} = \{0.8, 1, 1.2\}$$

$$nPyramids = 4 \qquad Z_{ratio} = \{1\}$$

$$\textbf{training:} \quad Z_x = Z_y = \{\ldots; -2; 0; 2; \ldots\}$$

$$\textbf{valid./testing:} \quad Z_x = Z_y = \{\ldots; -1; 0; 1; \ldots\}$$

Table 4.16 Tuning of the structural classifier invariant to translation, scale and rotation.

	λ	$nIter$	$e_{trn}[\%]$	$e_{val}[\%]$	$e_{tst}[\%]$
	10	15	14.43	20.00	
	1	39	3.11	10.03	
$\Rightarrow$	0.1	211	0.00	7.40	7.60
M	0.01	> 1000	–.–	–.–	

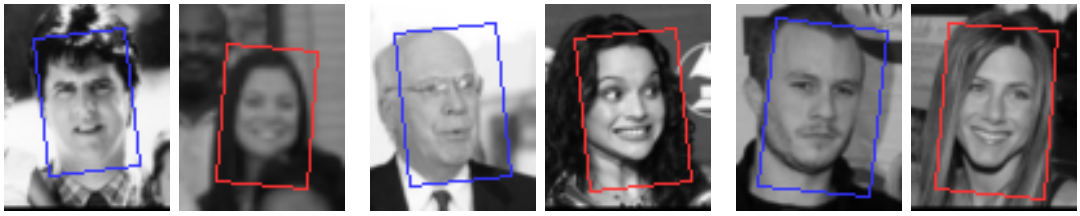
Table 4.17 Confusion matrix of the structural classifier invariant to translation, scale, rotation.

	ESTIMATED CLASS		ROW NORMALIZED	
TRUE CLASS	Male	Female	Male	Female
Male	48.00%	2.00%	96.00%	4.00%
Female	9.00%	41.00%	18.00%	82.00%

Table 4.16 shows results of tuning the regularization parameter λ on the validation set as well as the test error for the best setting of λ. We again had problems to finish the training for the lowest value of λ. It is seen from the results that modeling rotation and scale in addition to the translation increases the absolute

classification accuracy by almost 4%. We also see that the achieved test error improves the SVM classifier trained from virtual examples by 0.6% (c.f. section 4.3.2). Note that both methods model exactly the same face transformations and they use exactly the same image descriptor.

Table 4.17 shows the confusions matrix estimated on the testing examples. Responses of the structural classifier obtained on a sample of testing examples are visualized in figure 4.4. It is seen that the estimated face pose are relatively accurate.



a) lower scale **b)** middle scale **c)** larger scale

Figure 4.4 Visualization of responses of the structural classifier invariant to translation, rotation and scale on a sample of testing examples. The box shown corresponds to the estimated face pose. Color of the box corresponds to the predicted gender.

4.4.2.3 Structural classifier invariant to translation, scale, rotation and aspect-ratio

In this experiment we use the structural classifier modeling the whole range of transformations, that is, translation, rotation, scale and the aspect-ratio of the face. In particular, we use the following setting of the structural classifier:

$$[winH, winW] = [56, 46]\text{px} \qquad Z_\alpha = \{-\frac{\pi}{30}, -\frac{\pi}{60}, 0, \frac{\pi}{60}, \frac{\pi}{30}\}$$

$$I \in \mathbb{R}^{80x64xm} \qquad Z_{scale} = \{0.8, 1, 1.2\}$$

$$nPyramids = 4 \qquad Z_{ratio} = \{0.9, 1, 1.1\}$$

$$\textbf{training:} \quad Z_x = Z_y = \{\ldots; -2; 0; 2; \ldots\}$$

$$\textbf{valid./testing:} \quad Z_x = Z_y = \{\ldots; -1; 0; 1; \ldots\}$$

Table 4.18 Tuning of the structural classifier invariant to translation, scale, rotation and aspect-ratio.

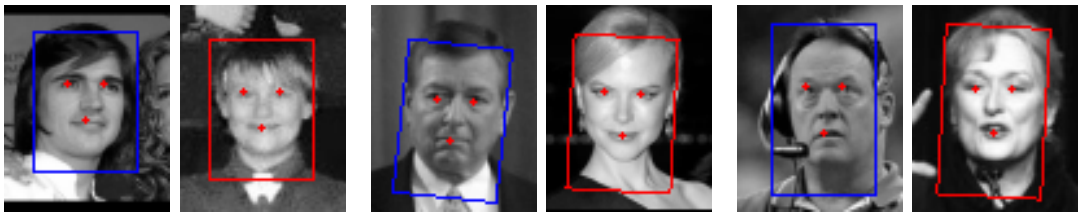
	λ	$nIter$	$e_{trn}[\%]$	$e_{val}[\%]$	$e_{tst}[\%]$
	10	16	16.28	20.73	
	1	44	3.14	9.35	
$\Rightarrow$	0.1	237	0.00	7.33	7.38
M	0.01	–.–	–.–	–.–	

Table 4.19 Confusion matrix of the structural classifier invariant to translation, scale, rotation and aspect-ratio.

	ESTIMATED CLASS		ROW NORMALIZED	
TRUE CLASS	Male	Female	Male	Female
Male	47.80%	2.20%	95.60%	4.40%
Female	5.18%	44.82%	10.37%	89.63%

Table 4.18 shows results of tuning the regularization parameter λ on the validation set as well as the test error for the best setting of λ. Even in this experiment we had problems to finish the training for the lowest value of λ. As one could expect modeling the full range of transformation produces the highest classification accuracy, in particular, 7.38% which was the best result we have achieved. Compared to the experiment from section 4.4.2.2 adding the aspect-ratio yields relatively minor improvement of 0.22%. This could be explained by a relatively small number of babies in our database having faces with significantly different aspect-ratio compared to adults.

Table 4.19 shows the confusions matrix estimated on the testing examples. Responses of the structural classifier obtained on a sample of testing examples are visualized in figure 4.5. We also marked estimated positions of the eyes and the mouth. It is seen that the estimates are relatively precise though the precision of the pose estimate is not explicitly expressed by the used loss function.



a) small aspect ratio **b)** medium aspect ratio **c)** high aspect ratio

Figure 4.5 Visualization of responses of the structural classifier invariant to translation, rotation, scale and the aspect-ratio on a sample of testing examples. The box shown corresponds to the estimated face pose. We also marked estimated positions of the eyes and the mouths. Color of the box corresponds to the predicted gender.

4.4.2.4 Structural classifier using the HOG descriptor

The goal of this experiment is to evaluate the HOG descriptor when used in the structural classifier. We used the same setting of the pose parameter set Z as for the structural classifier with the LBP-pyramid features for fair comparison. We consider the structural classifier modeling the translation, rotation and scale. The setting of HOG parameters was taken from the experiment in section 4.4.1.2.

In particular, we used the following setting:

$$[winH, winW] = [64, 40]\text{px} \qquad Z_\alpha = \{-\frac{\pi}{30}, -\frac{\pi}{60}, 0, \frac{\pi}{60}, \frac{\pi}{30}\}$$
$$I \in \mathbb{R}^{80x64xm} \qquad Z_{scale} = \{0.8, 1, 1.2\}$$
$$cellSize = 2 \qquad Z_{ratio} = \{1\}$$
$$nBins = 6 \qquad \textbf{training: } Z_x = Z_y = \{\dots; -2; 0; 2; \dots\}$$
$$blockSize = 3 \qquad \textbf{valid./testing: } Z_x = Z_y = \{\dots; -1; 0; 1; \dots\}$$
$$useSquare$$
$$P = 160$$

Table 4.20 Tuning the structural classifier invariant to translation, scale and rotation using the HOG descriptor.

	λ	$nIter$	$e_{trn}[\%]$	$e_{val}[\%]$	$e_{tst}[\%]$
	1	17	10.38	12.15	
	0.1	33	5.54	8.30	
$\Rightarrow$	0.01	69	1.48	7.78	8.60
M	0.001	–.–	–.–	–.–	

Table 4.21 Confusion matrix of the structural classifier invariant to translation, scale, rotation using the HOG descriptor.

	ESTIMATED CLASS		ROW NORMALIZED	
TRUE CLASS	Male	Female	Male	Female
Male	47.00%	3.00%	93.00%	7.00%
Female	5.00%	45.00%	10.00%	90.00%

Table 4.20 shows results of tuning the regularization parameter λ on the validation set as well as the test error for the best setting of λ. Training of the HOG descriptor is less time demanding thanks to its lower dimensional feature vector when compared to the LBP-pyramid. Unfortunately, classification accuracy for the HOG descriptor is considerably lower than the one achieved with LBP-pyramid: 8.6% compared to 7.6% for the same setting of Z. Table 4.21 shows the confusion matrix estimated on the test examples.

4.4.2.5 Structural classifier using the HLBP descriptor

In this section we experimentally evaluate the structural classifier with the Histogram of LBP (HLBP) descriptor. Compared to the LBP-pyramid descriptor, the HLBP descriptor losses some information by counting only the occurrences of the LBP codes. On the other hand, the HLBP descriptor has much lower dimension which makes learning of the structural classifier less time demanding. We

used the same setting for the pose parameter $z \in Z$ as previously to allow for fair comparison, i.e. we modeled translation, rotation and scale. The base-window size was taken from the experiment in section 4.4.1.3. In particular, we used the following setting:

$$[winH, winW] = [64, 36]\text{px} \qquad Z_\alpha = \{-\frac{\pi}{30}, -\frac{\pi}{60}, 0, \frac{\pi}{60}, \frac{\pi}{30}\}$$
$$I \in \mathbb{R}^{80x64xm} \qquad Z_{scale} = \{0.8, 1, 1.2\}$$
$$cellSize = 4 \qquad Z_{ratio} = \{1\}$$
$$P = 160 \qquad \textbf{training: } Z_x = Z_y = \{\dots; -2; 0; 2; \dots\}$$
$$\textbf{valid./testing: } Z_x = Z_y = \{\dots; -1; 0; 1; \dots\}$$

Table 4.22 Tuning of the structural classifier invariant to translation, scale and rotation using the HLBP descriptor.

	λ	$nIter$	$e_{trn}[\%]$	$e_{val}[\%]$	$e_{tst}[\%]$
	10	19	10.52	13.20	
$\Rightarrow$	1	33	4.43	8.82	9.30
	0.1	63	0.37	9.38	

Table 4.23 Confusion matrix of the structural classifier invariant to translation, scale, rotation suing the HLBP descriptor.

	ESTIMATED CLASS		ROW NORMALIZED	
TRUE CLASS	Male	Female	Male	Female
Male	46.88%	3.12%	93.77%	6.23%
Female	6.18%	43.82%	12.37%	87.63%

Table 4.22 shows results of tuning the regularization parameter λ on the validation set as well as the test error for the best setting of λ. The results show that the HLBP descriptor is significantly worse than the LBP-pyramid and the HOG descriptor. We completeness we also report the confusion matrix in Table 4.23.

4.5 Results summary

In this section, we summarize results obtained in the previous sections. We report test classification errors for the best hyper-parameter setting of the corresponding classifier. In particular, the results are reported for the baseline SVM classifier (section 4.3.1), the SVM classifier trained from virtual examples (section 4.3.2) and for several variants of the proposed structural classifier (section 4.4).

The results are summarized in Table 4.24. It is seen that the methods trying to cope with the unknown face pose, either by using the virtual examples or by modeling the pose explicitly in the structural classifier, improve the classification

accuracy compared to the baseline approach. The accuracy of the baseline SVM classifier is relatively good which is due to the large number of examples we use for training. Further, we see that more pose transformations are modeled the higher accuracy is achieved. We also see that accuracy of the structural classifier modeling only the translation is worse than the baseline SVM classifier. This is not surprising because the learning algorithm for the structural classifier in this case does not take into account the rotation and scale which are significant sources of the image variation.

The overall best results were obtained for the structural classifier using the LBP-pyramid features and modeling the full range of the pose transformations. Comparison against the SVM classifier trained from VE shows that the structural classifier improves the accuracy from 8.2% to 7.38%, that is, relative improvement of 10%. As for the feature descriptors, the LBP-pyramid features significantly outperformed the HOG features as well as the histogram of the LBPs.

Table 4.24 Summary table.

			Classifier invariant to				
	Classifier	Descriptor	transl.	scale	rot.	asp. rat.	e_{tst}[%]
	Baseline SVM	LBP-pyr.					10.77
	SVM from VE	LBP-pyr.	x	x	x		8.20
	Structural classifier	LBP-pyr.	x				11.58
	Structural classifier	LBP-pyr.	x	x	x		7.60
⇒	Structural classifier	LBP-pyr.	x	x	x	x	7.38
	Structural classifier	HOG	x	x	x		8.60
	Structural classifier	HLBP	x	x	x		9.30

4.6 Comparison of the proposed P-BMRM against the standard BMRM algorithm

In this section we present a preliminary experiment comparing the standard BMRM algorithm (c.f. section 3.4) with the proposed P-BMRM algorithm (c.f. section 3.5).

For testing we consider the problem of learning the structural classifier with the HOG features. The HOG features are low-dimensional representation suitable for the single machine implementation of the P-BMRM algorithm which has a higher memory demands due to the need to store P cutting planes per iteration. We trained only on 500 examples (250 male and 250 female) randomly sub-sampled from the original 14,000 examples. We did not experiment on the full dataset due to lack of time.

We run the standard BMRM and proposed P-BMRM until the relative precision ϵ_{rel} dropped below 10^{-2}. For both algorithms we measured the objective value, i.e. the objective of the learning problem (3.7), the memory demands and the number of iterations. We used the number of iterations as the measure of computational time independent to the particular implementation and used computer. This is

possible because both algorithms have essential the same per iteration complexity dominated by computation of the sub-gradient(s) and evaluation of the risk. In the case of the P-BMRM, we also varied the number of cutting plane models P. Note that P-BMRM with $P = 1$ is equivalent to the standard BMRM.

Table 4.25 summarizes the number of iterations as a function of the regularization parameter λ. The column for $P = 1$ shows results for the standard BMRM algorithm. It is seen that using P-BMRM algorithm becomes beneficial for lower values of λ. It is also seen that the number of iterations decreases as the number of cutting plane models P grows. This behavior perfectly fits the intuition. The lower λ means less contribution of the quadratic term which implies less smoothness and thus more "complex" objective function. A complex objective function requires finer cutting plane model which is provided by the P-BMRM algorithm. Similarly, more cutting plane models can faster approximate the objective function. The line before the last shows accumulated number of iterations needed to train the classifier for all λs, i.e. the accumulated iterations corresponds to the time needed for validation of λ. The last line shows the speed-up computed from the accumulated numbers of iterations. The convergence curves for varying λ and P are visualized in Figure 4.6.

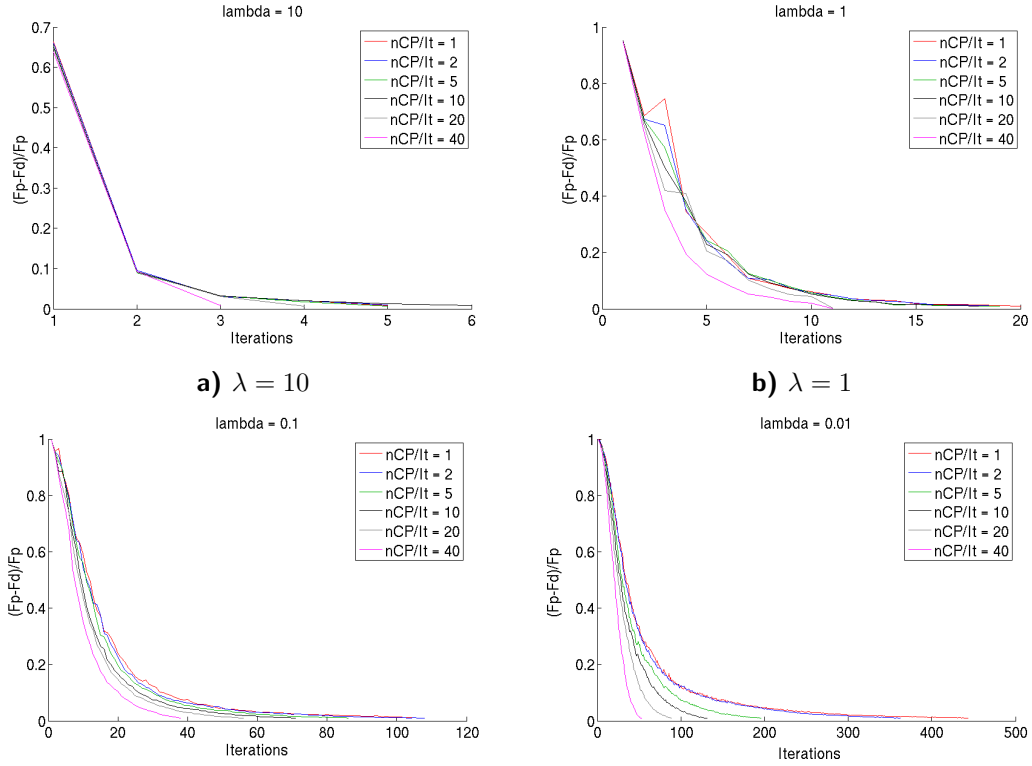
The speed-up gained from using more cutting plane models in the P-BMRM algorithm is paid by its higher memory requirements. Table 4.26 shows the memory requirements for varying λ and P. The memory requirements are measured relatively to the requirements of the standard BMRM corresponding to the column for $P = 1$. The memory problems can be alleviated by using multiple computes each responsible for computation and storing a single cutting plane model.

Table 4.25 Number of iterations required by the proposed P-BMRM algorithm as a function of λ and the number of cutting plane models P. Note that $P = 1$ corresponds to the standard BMRM algorithm.

$\lambda \setminus \frac{nCP}{It.}$	1	2	5	10	20	40
10	5	5	5	6	4	3
1	20	19	19	18	11	11
0.1	107	108	86	71	56	38
0.01	443	362	196	131	89	53
accum. iter.	575	494	288	226	160	105
speed-up	0%	16%	100%	154%	259%	448%
color in figure 4.6	red	blue	green	black	gray	violet

Table 4.26 Memory requirements as a function of λ and P. The memory requirements are measured relatively to the standard BMRM which corresponds to setting $P = 1$.

$\lambda \setminus \frac{nCP}{It.}$	1	2	5	10	20	40
10	1.00	2.00	5.00	12.00	16.00	24.00
1	1.00	1.90	4.75	9.00	11.00	22.00
0.1	1.00	2.02	4.02	6.64	10.47	14.21
0.01	1.00	1.63	2.21	2.96	4.02	4.79



a) $\lambda = 10$ **b)** $\lambda = 1$

c) $\lambda = 0.1$ **d)** $\lambda = 0.01$

Figure 4.6 Convergence curves of the P-BMRM algorithm for varying λ and P. Note that $P = 1$ corresponds to the standard BMRM algorithm. The vertical axis corresponds to the relative gap between the primal and the reduced objective functions while the horizontal shows the number of iterations. Each of the four figures corresponds to different value of λ. Curves for varying P are distinguished by color.

5 Implemented library

We implemented an open source library for Gender Recognition with Structural Model (libGRSM). The libGRMS is licensed under GNU GPLv3. Despite its name the libGRSM can be applied for visual classification of arbitrary object into two classes.

The code is written in two programming languages. The core time demanding functions are written in C. Learning algorithm is implemented in MATLAB which is more suitable for fast development. Moreover, MATLAB provides a wealth of useful functions for image processing. The MATLAB code calls the time demanding functions implemented in C via the MEX interface. The structural classifier is implemented both in MATLAB and C. Compilation of the C library providing the basic API and the MEX functions is done separately.

5.1 Main building blocks of the library

The library is structured into the following five building blocks

- Classifier
- Descriptors
- MEX interfaces
- MATLAB function
- Examples

Figure 5.1 visualizes the building blocks and relations between them.

Classifier

This is the main part of the library. This part is formed only by files `libgrsm.h` and `libgrsm.c`. Besides the structural classifier these files also implement computation of the risk function $\widehat{R}(\boldsymbol{w})$ and its subgradient $\nabla R(\boldsymbol{w})$ which are procedures called from the learning algorithm. These computations require functions for image transformation. In order to minimize dependencies on other libraries we implemented the image transformation functions ourselves. The transformations use the bilinear interpolation of the image function.

Descriptors

We implemented a simple interface which allows to use different image descriptors in the structural classifier. Inserting a new descriptor is possible without modifying other pieces of the code like e.g. the learning algorithm. This feature makes experimentation with new image descriptors pretty simple. Because the library is implemented mainly in C using of objects is not possible. For this reason the

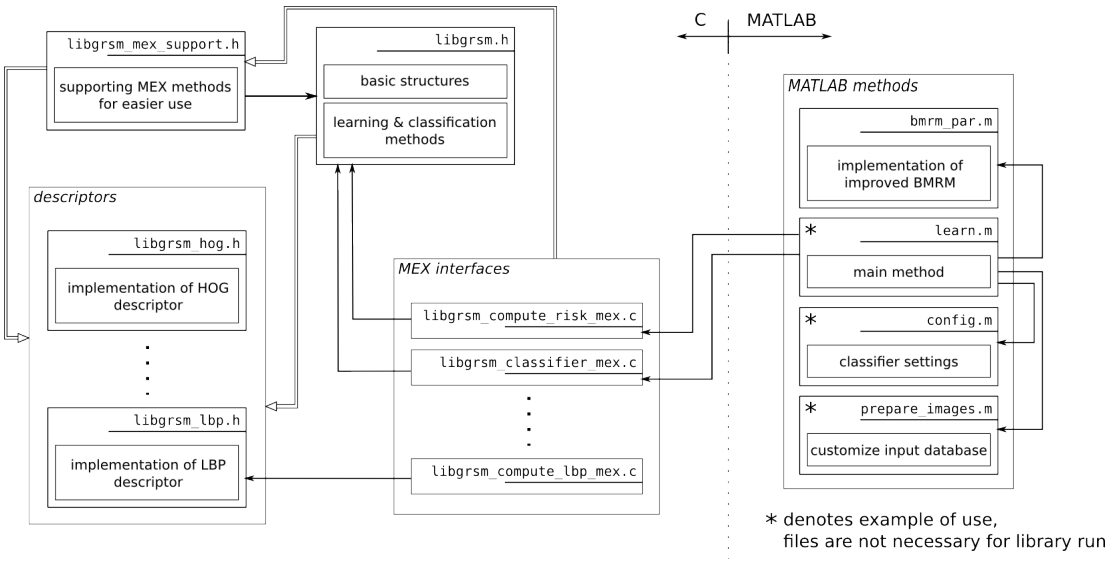


Figure 5.1 Simplifier schema of implemented library

interface uses functions pointers. The interface is formed by the following suit of functions:

constructor	Initialization of the descriptor. Implementation is optional.
destructor	Releases the descriptor data from memory. Implementation is optional.
addition function	For given real vector $x \in \mathbb{R}^d$ and feature vector Ψ it returns sum $\boldsymbol{x} + \Psi$.
subtraction function	For given real vector $x \in \mathbb{R}^d$ and feature vector Ψ it returns difference $\boldsymbol{x} - \Psi$.
dot prod function	For given real vector $x \in \mathbb{R}^d$ and feature vector Ψ it returns the dot product $\langle \boldsymbol{x}, \Psi \rangle$.
Ψ **function**	Constructs feature vector Ψ for given transformation.
dimension function	Returns dimension d of the feature vector $\Psi \in \mathbb{R}^d$.
saving function	Saves descriptor structures into file.
loading function	Loads and initializes descriptor structures from file.
joiner	This function connects a descriptor with the model.

The library also provides an example implemation of an empty feature descriptor (file `libgrsm_empty_if.h`) which uses only the important functions of the interface. It is also shown how to inasert a new descriptor to the library. This example is intended for users who want to experiment with new image descriptors.

MEX interface

This part of the libGRSM library provides an interface between MATLAB and core function implemented in C. To simplify implementation of the MEX functions we wrote a set of helper functions for converting MATLAB structures to C structures. Moreover these functions check consistency of the input model. The set of helper functions is available in files `libgrsm_mex_support.h/c`.

MATLAB functions

This part of the libGRSM library is formed by an implementation of the P-BMRM algorithm in file `pbmrm.m`. There are also example scripts demonstrating usage of the library. In particular, scripts `libgrsm_prepare_datasets.m` and `libgrsm_prepare_images` perform pre-processing of a given image database and script `libgrsm_learn.m` which implements typical learning procedure.

5.2 Parallel computations

Standard computers used nowadays are equipped with multi-core CPUs or even multiple CPUs. Such hardware asks for development of parallel algorithms. In this section we described three ways of using parallel computations to speed up learning as well as classification.

Method 1: Parallel computation of cutting planes The bottle neck of the P-BMRM algorithm used for learning is computation of the new cutting planes involving evaluation of the objective function $R(\boldsymbol{w})$ and its sub-gradient $\nabla R(\boldsymbol{w}, p)$. The evaluation of the sub-gradient $\nabla R(\boldsymbol{w}, p)$ is illustrated by algorithm 4. We can speed-up the evaluation by creating as many threads as we have cutting plane models (determined by the parameter P). In this case, each thread computes one of P sub-gradinets $\nabla R(\boldsymbol{w}, p)$. This method can be used to speed-up only the learning algorithm.

Method 2: Parallel computation of examples In this case, we parallelize the sequential computation of the loop on line 2 of Algorithm 4. This method can speed-up only the learning algorithm.

Method 3: Parallel evaluation of the *search-windows* In this case loops started on line 4 are forked and run in parallel. Advantage of this method is in possibility to use more CPUs for classification.

We implemented parallel computations using POSIX threads and measured the speed-up achieved when using the method 2 and 3 both implemented in C. We skipped evaluation of the method 1 which was implemented only in Matlab. For measuring we used a notebook with the Intel® i5 Core CPU/4GB RAM which

Algorithm 4 Pseudo-code of $\nabla R(\boldsymbol{w}, p)$ method

Input: $I^p \in \mathbb{Z}^{imH \times imW \times m_p}$, $y^p \in \{-1, +1\}^m$, $z^p \in (Z_x \times Z_y \times Z_\alpha \times Z_{scale} \times Z_{ratio})^m$, $\boldsymbol{w}^D$, *model*

Output: $\nabla R(\boldsymbol{w}, p)$

initialization: $\nabla R(\boldsymbol{w}, p) = \emptyset^D$
for $i = 1 \rightarrow m_p$ **do**
  $\widehat{z} = \{\}$
  for all $z \in Z$ **do**
    if $\langle \boldsymbol{w}, \Phi(I_i^p, z, \neg y_i^p) \rangle > \langle \boldsymbol{w}, \Phi(I_i^p, \widehat{z}, \neg y_i^p) \rangle$ **then**
      $\widehat{z} = z$
    end if
  end for
  $\nabla R(\boldsymbol{w}, p) \leftarrow \nabla R(\boldsymbol{w}, p) + \Phi(I_i^p, \widehat{z}, \neg y_i^p) - \Phi(I_i^p, z_i^p, y_i^p)$
end for
$\nabla R(\boldsymbol{w}, p) \leftarrow \nabla R(\boldsymbol{w}, p)/m$

has two cores and HyperThreading technology. We use the following setting:

$$F \in \mathbb{R}^{60x40} \qquad Z_\alpha = \{-\frac{\pi}{30}, -\frac{\pi}{60}, 0, \frac{\pi}{60}, \frac{\pi}{30}\}$$
$$I \in \mathbb{R}^{80x64xm} \qquad Z_{scale} = \{0.8, 1, 1.2\}$$
$$m = 100 \qquad Z_{ratio} = \{1\}$$
$$nPyramids = 4 \qquad Z_x \times Z_y = \{\ldots; -2; 0; 2; \ldots\} \times \{\ldots; -2; 0; 2; \ldots\}$$

Table 5.1 Contribution achieved for different methods of parallelization.

	1 thread [ms/image]	2 thread [ms/image]	4 thread [ms/image]	speed-up for 2 threads [%]
Method 2	232.4	130.9	111.7	178
Method 3	230.4	214.9	207.6	107

Table 5.1 summarizes the results. It is seen that the parallelization method 2, i.e. on the level of the whole examples, is considerably more efficient than method 3 which parallelizes on the level of search-window computation. It is caused by hidden overhands for managing the threads (construction and releasing the threads). These overheads kill the speed-up from faster calculations. The method 2 speeds the computation two times when 2 threads were used. The speed-up achieved for 4 threads is surprising because the computed has only two CPUs. This speed-up, more than 30%, is solely due to the HyperThreading. Unfortunately the method 2, being the most efficient one, is not useful for classification in the standard setting when only one faces is to be processed.

5.3 Classification time

In this section we measure the classification time of the structural classifier for the three tested image descriptors. The classification time is mainly determined

by the size of the set of transformations Z and the size of the *base-window*. In the experiment we used the same computer and parameter setting as in section 5.2 which measures the effect of parallelization. In particular, the setting was:

$$
\begin{aligned}
F \in \mathbb{R}^{60\times 40} \qquad & Z_\alpha = \{-\frac{\pi}{30}; -\frac{\pi}{60}; 0; \frac{\pi}{60}; \frac{\pi}{30}\} \\
I \in \mathbb{R}^{80\times 64} \qquad & Z_{scale} = \{0{,}8; 1; 1{,}2\} \\
Z_{ratio} \qquad & = \{1\} \\
& Z_{transl} = \{\ldots; -2; 0; 2; \ldots\} \times \{\ldots; -2; 0; 2; \ldots\}
\end{aligned}
$$

Table 5.2 Average classification time per example required by the structural classifier using different feature descriptors.

	Descriptor	$t[ms]$	Settings
$\Rightarrow$	LBP-pyramid	222	$nPyramids = 4$
	HOG	796	$cellSize = 2px$, $blockSize = 4$, $nBins = 9$, $useSquare$
	HLBP	348	$cellSize = 5px$

Table 5.2 reports classification times per a single image. The times are computed as averages over 100 examples. The shortest classification time was achieved for the LBP-pyramid descriptor whose computation involves only a lot of $if - condition$ evaluations. On the other hand, the HOG descriptor requires more complex operations like computation of the image gradient. The HOG and HLBP descriptors are also slowed down by computations of histograms.

5.4 Learning method

The library contains an implementation of the P-BMRM algorithm proposed in section 3.5. Our implementation modifies existing code of the BMRM algorithm from the Statistical Pattern Recognition Toolbox for MATLAB [4]. The P-BMRM algorithm requires solving an instance of the Quadratic Program in its inner loop. To solve the QP we use the Library for Quadratic Programming (libQP) [6]. The libQP is involved into the library, i.e. library is ready for compilation immediately.

5.5 Proposal of further improvements not yet implemented

We tried to speed-up learning algorithm by proposing the P-BMRM algorithm and by paralleling its core procedures. However, we have been using a single multi-core machines so far. The next step would be to distribute computation over several connected machines. Besides increasing the computational power

using the distributed computing would also increase the available memory which is currently the major bottle-neck of the P-BMRM algorithm.

This extension can be achieved easily by using the MPI protocol. The MPI protocol has became widespread due to its large support for major programming languages including C and MATLAB which are of the main interest for us.

6 Conclusion

In this thesis we propose a structural classifier for a gender recognition from facial images. The structural classifier explicitly models the unknown pose of the recognized faces which is considered as an additional hidden parameter. The structural classifier performs the search for the face pose and the estimation of the gender simultaneously. This is the main feature distinguishing the proposed method from existing approaches all being based on two-class classifiers.

We formulate learning of the structural classifier from annotated examples as a convex optimization problem. The learning problem is as a variant of the Latent Support Vector Machines algorithm.

We experimentally evaluate the impact of modeling the face pose on the classification accuracy. We found that modeling the translation, rotation and scale brings the major improvement compared to the base-line SVM classifier. Including the aspect-ratio among the modeled transformation results in additional improvement which is, however, smaller than expected. By comparing various image representations we found that the LBP-pyramid significantly outperforms the HOG and the Histogram of LBPs based descriptors. The overall best classification accuracy 7.38% was obtained for the structural classifier modeling the translation, rotation, scale and the aspect-ration of the face and using the LBP-pyramid image descriptor. This result significantly improves accuracy of the baseline SVM classifier being 10.77% as well as so-far-the-best SVM classifier trained from the virtual examples having the accuracy 8.20%.

In addition, we proposed an improved of the BMRM algorithm for solving the large-scale convex optimization task required for learning the structural classifier. The proposed algorithm, called P-BMRM, uses P cutting planes to approximate the optimized objective unlike the original BMRM which uses only one cutting plane model. Preliminary results show that the proposed P-BMRM algorithm can significantly reduced the number of iterations while the per iteration cost remains the same as in the original BMRM.

Last but not least we implemented an open source library called Library for Gender Recognition with Structural Model (libGRSM). The libGRSM contains an implementation of the structural classifier and the P-BMRM algorithm for training its parameters from annotated examples. The structural classifier as well as all time demanding procedures called by the learning algorithm are implemented in C. The P-BMRM algorithm itself is written in Matlab. The library comes with a suit of MEX functions which provide an interface between C and Matlab.

A CD content

The thesis comes with a compact disk which contains the library libGRMS, source codes of the experiments presented in the thesis and electronic version of the thesis. The CD has the following structure:

doc This folder contains electronic version of the thesis.

experiments This folder contains Matlab scripts implementing the experiments which are split into two parts: i) parameter tuning by the two-class linear SVM (involves also evaluation of the baseline SVM classifier) and ii) experiments with the structural classifier.

i. Parameter tuning by the linear SVM

exp_lbp_c Tuning the parameter C of the linear SVM with LBP-pyramid descriptor. The same script used for evaluation of the baseline method.

exp_lbp_bw Tuning the size of the *base-window* by linear SVM with LBP-pyramid feature descriptor. The same script used for evaluation of the baseline method.

exp_lbp_npyr Tuning the parameter $nPyramids$ of LBP-pyramid descriptor using the linear SVM.

exp_hog_c Tuning the parameter C of the linear SVM with the HOG descriptor.

exp_hog_bw Tuning the size of *base-window* by the linear SVM with the HOG descriptor.

exp_hog_parameters Tuning $cellSize$, $blockSize$, $nBins$ and $useSquare$ parameters of the HOG descriptor by the linear SVM.

exp_hlbp_c Tuning the parameter C of the linear SVM with the HLBP descriptor.

exp_hlbp_bw Tuning the size of the *base-window* by the linear SVM with the HLBP descriptor.

exp_hlbp_cellSize Tuning the parameter $cellSize$ of the HLBP descriptor by the linear SVM.

ii. Experiments with the structural classifier

struct_lbp_POS Classifier invariant against translation using the LBP descriptor.

struct_lbp_PSR Classifier invariant against translation, rotation and scale using the LBP-pyramid descriptor.

struct_lbp_PSR Classifier invariant against translation, rotation, scale and aspect ratio using the LBP-pyramid descriptor.

struct_hog_PSR Classifier invariant against translation, rotation and scale using the HOG descriptor. This classifier was also used for comparison of the proposed P-BMRM against the standard BMRM.

struct_hlbp_PSR Classifier invariant against translation, rotation and scale transformations using the HLBP descriptor.

releases This folder contains the final release of the libGRSM library.

Bibliography

[1] J. Bekios-Calfa, J.M. Buenaposada, and L. Baumela. Revisiting linear discriminant techniques in gender recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(4):858 –864, april 2011. 8

[2] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886 –893 vol. 1, 2005. 10, 21, 31

[3] Y. Fang and Z. Wang. Improving lbp features for gender classification. In *Wavelet Analysis and Pattern Recognition, 2008. ICWAPR '08. International Conference on*, volume 1, pages 373 –377, aug. 2008. 8

[4] V. Franc. *Statistical Pattern Recognition Toolbox for MATLAB*. CTU in Prague, 2000-2011. http://cmp.felk.cvut.cz/cmp/software/stprtool/. 17, 46

[5] B. Moghaddam and Ming-Hsuan Yang. Gender classification with support vector machines. In *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*, pages 306 –311, 2000. 8

[6] T. Ojala, M. Pietikainen, and D. Harwood. Performance evaluation of texture measures with classification based on kullback discrimination of distributions. In *Proceedings of the 12th IAPR International Conference on Computer Vision and Image Processing*, volume 1, pages 582 –585 vol.1, October 1994. 10, 20, 46

[7] S. Sonnenburg and V. Franc. COFFIN: A computational framework for linear SVMs. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML'10)*, pages 999–1006, Haifa, Israel, June 2010. Omnipress. 9, 28

[8] Eyedea Recognition s.r.o. http://www.eyedea.cz/. 25

[9] Z. Sun, X. Yuan, G. Bebis, and S.J. Louis. Neural-network-based gender classification using genetic search for eigen-feature selection. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'02)*, volume 3, pages 2433 –2438, 2002. 8

[10] C. H. Teo, S.V.N. Vishwanthan, A. Smola, and V. Le Quoc. Bundle methods for regularized risk minimization. In *Journal of Machine Learning Research*, volume 11, pages 311 –365 vol.11, January 2010. 6, 10, 15, 16

[11] C. J. Yu and T. Joachims. Learning structural SVMs with latent variables. In *International Conference on Machine Learning (ICML'09)*, 2009. 6, 10