PFI - PLS Face Identification

version 0.1.1

Manual by William Robson Schwartz and Jonghyun Choi
http://www.dcc.ufmg.br/~william/softwares.html

April 23, 2012

Contents

Chapter 1

Introduction

This manual describes some functionalities of the software **PFI - PLS Face Identification**, which contains an implementation of face identification method proposed in the following papers (the second is an extension of the first by adding new feature descriptors).

> [1] *A Robust and Scalable Approach to Face Identification.*
> W. R. Schwartz and H. Guo and L. S. Davis.
> European Conference on Computer Vision. 2010. [PDF].
>
> [2] *Face Identification Using Large Feature Sets.*
> W. R. Schwartz, H. Guo, J. Choi, L. S. Davis.
> IEEE Transactions on Image Processing. 2012. [PDF].

if you use this software in your research, please cite the papers above. **Note:** this software cannot be used for commercial purposes. Further information about this research might be found on the author's webpage: `http://www.dcc.ufmg.br/~william/research.html`.

In [1, 2], we evaluate the proposed method using two face data sets: FERET [3] and FRGC [4]. The goal of this implementation is to allow researchers to evaluate our approach on other face identification datasets.

Upon request (e-mail to william@dcc.ufmg.br), we make available the cropped samples and partition files which were used during our experimental validation in [1, 2].

If you have found bugs or problems in this software or you have suggestions to improve or make it more user friendly, please send an e-mail to william@dcc.ufmg.br.

Chapter 2

Executing Face Identification

In this chapter, we provide a step-by-step help about how to use the PFI software. To facilitate the software usage, we provide a small data set (a subset of the FERET dataset), inside the directory *SampleData*. Along the text, we use this data set as example.

2.1 Setting Samples

Before performing face identification in your dataset, the following steps need to be executed. (i) gallery and probe samples should be placed in directories identified by the ID of the subject; (ii) files listing samples in the gallery and probe samples need to be created.

1. Add all samples belonging to a subject in a directory named with the subjects' ID.

2. Create two files: gallery.txt and probe.txt. The first will contain samples used to create the gallery and the second will contain probe samples. Each line of these files have the format shown bellow, where ID$_x$ indicates the unique identifier for the subject and filename$_k$ has the image name (this image must be placed inside the directory created for the subject, e.g., images filename1, filename2 and filename3 must be inside the directory ID1). Note: a subject may have more than one sample associated to it.

```
ID1 filename1
ID1 filename2
ID1 filename3
ID2 filename4
...
IDx filenameK
```

Examples of partition files can be found in the directory test/partitions. These files are used to indicate samples belonging to the gallery and to the probe (images are in the directory test/images).

2.2 Setting up Feature Descriptors

Feature descriptors need to be set to define the features to be used, block sizes and specific parameters used for each feature extraction method.

2.3 Configuration File

The configuration file is required to setup how the new object model should be build. It specifies the feature extraction methods, their parameters, sample size, object size, and block size and strides. Figure 2.1 shows an example of a configuration file.

In this example, the features considered are HOG, color frequency (implemented in HOG for computational efficiency) and co-occurrence matrix for color bands H, S, and V (`COOCH`, `COOCS`, `COOCV`). The features computed using co-occurrence matrix are been cached (`params <COOCPARAM, cache, 1>`). The block sizes considered for both HOG and co-occurrence matrix are 16×16 and 32×32 pixels with stride of 8, respectively. The face size is 110×110. Note that # is a directive for comment.

The feature extraction methods to be used are defined as following.

```
method <method name, method ID, cache, parameters ID>
```

```
# internal identifier (ignore this)
ModelID <FaceModel>

# feature methods used
method <HOG,HOG,0,HOGPARAM>
method <COOCH,COOC,1,COOCPARAM>
method <COOCS,COOC,1,COOCPARAM>
method <COOCV,COOC,1,COOCPARAM>

# feature parameters
params <COOCPARAM,bins,16>
params <COOCPARAM,distance,1>
params <COOCPARAM,cache,1>

params <HOGPARAM,UseColorFrequency,1>
params <HOGPARAM,UseHOG,1>
params <HOGPARAM,cache,0>

# block sizes and strides for each feature
block <HOG,16,16,8,8,5>
block <HOG,32,32,8,8,5>

block <COOC,16,16,8,8,5>
block <COOC,32,32,8,8,5>

# detection window and object size
window <85,110>
object <0,0,85,110>

# region inside the detection window to be considered
region <0,0,84,109>

# method specific parameters
params <HOGPARAM,UseColorFrequency,1>

# deprecated options  (not used, but do not remove)
classifier <qda>
factors <20>
```

Figure 2.1: Example of the configuration file used to learn an object model considering HOG, co-occurrence matrix, and color frequency.

where

method name	Name of the feature extraction method. These names are defined in the implementation. Currently available methods are COOCH, COOCS, COOCV (co-occurrence matrix for color bands H, S, and V) and HOG.
method ID	ID used to link this method to the block setup (see `blocks`).
cache	Obsolete parameter.
parameters ID	ID used to setup the parameters for this method (see `params`).

Each feature extraction method has a few parameters that can be set by the user. They have the following format (the specific parameters for each feature extraction method implemented will be described in Section 2.4).

```
params <parameters ID, name, value>
```

where

parameters ID	identifier to link to the feature extraction method (same as in `method`).
name	parameter name.
value	value for the current parameter.

The block setup is done by the following command.

```
block <method ID, width, height, stride_X, stride_Y, factors>
```

where

method ID	identifier to link to the feature extraction method (same as in `method`).
width	width of the block.
height	height of the block.
stride_X	stride of the block in the horizontal direction.
stride_Y	stride of the block in the vertical direction.
factors	obsolete parameter.

The following command defines the size of the sub-window to be used to extract the feature (this option can be override by command line. See Section 2.5).

```
window <width, height>
```

Usually, the sub-window size for feature extraction is determined by the size of fiducial component in pixel, depending on the size of the entire face image. Note that the various combination of sub-window configuration can be specified simultaneously to make the feature descriptor rich in information.

```
object <x, y, width, height>
```

Sometimes one is interested in extracting features at a specific locations of sub-window. The command allows the user to specify the regions. Also, multiple entries of this command can be set to covering different regions of the detection window (usually, the region is set as `<0, 0, width-1 and height-1>` so that the entire window is considered, where width and height are the window size).

```
region <x, y, width, height>
```

Each feature extraction method has its own parameters. They are defined using the command `params`, where methodID represents the `parameters ID` defined in command `method`, parameter indicates the parameter name (next Section presents the parameters available for each feature extraction methods) and value contains the parameter value.

```
params <methodID, parameter, value>
```

2.4 Implemented Feature Extraction Methods

This section describe the parameters of the available feature extraction methods. All these feature descriptors are extracted from image blocks with size specified as described in the previous section.

2.4.1 Histogram of Oriented Gradients

It is an implementation of [5] that extract and concatenate HOG descriptor at specified pixel position within a specified sub-window. We also implement additional color frequency feature as a useful additional information due to computational efficiency. Another reason to pair the features is that since HOG is purely edge information the color information can help to improve the performance in complementary manner. The method ID for this feature is `HOG`.

The parameters available for HOG method are: UseColorFrequency, UseHOG, cache. If UseColorFrequency has value 1, the color frequency feature will be computed. If UseHOG has value 1, HOG will be compute. If cache is set to 1, the HOG features are cached (**Note:** since HOG uses integral histograms, the cache does not save much computational time, but, in fact, it only increases the memory consumption).

2.4.2 Gray Level Co-occurrence Matrices

Implementation of the statistical descriptors extracted from the co-occurrence matrices [6]. The method ID for this feature is `COOCH`,`COOCS` and `COOCV` for co-occurrence matrices for hue, saturation and value in HSV color space, respectively.

The parameters available for co-occurrence matrix are: bins, distance, cache. Please, do not change the first two parameters. If the value of cache is 1, the features extracted will be cached, avoiding multiple computations of features for a given block.

2.4.3 Local Binary Patterns

It is an implementation of the [7]. It is an original version of LBP that is extracted in 3×3 window and encoded as a byte (0-255) and packed into a histogram of bin ranged 0-255. The method ID for this feature is `LBP`.

The only parameter available for LBP method is cache. If cache is set to 1, the LBP features are cached.

2.4.4 Multi-Scale Local Binary Patterns

It is a a multi-scale implementation of the [8] as an extension of LBP [7]. It is extracted within 5×5 and 9×9 windows for radii 2 and 4 scales respectively.

The method ID for this feature is MSLBP.

Since we fix the number of radius as [8] did, there is also only one parameter, cache, for this feature. If cache is set to 1, the MSLBP features are cached.

2.4.5 Gabor Filters

It is an implementation of the [9]. The Gabor filter is a set of Gaussian kernels that need a specified support window to be convoluted. We fix the support as the size of the image and this implementation does not fetch the image information dynamically (for checking dataset sanity check), the user need to fix the size of the image manually. The method ID for this feature is GABOR.

So the parameters for Gabor filter are w, h and cache, where w and h stand for width and height of the image, respectively. If cache is set to 1, the Gabor features will be cached. (**Note:** the size of the block (parameter block defined in the previous section) for the Gabor feature has to have the same size of the image).

2.4.6 Intensity Average within a Sub-window

It is an implementation of intensity average of a sub-window. This feature is to capture the blurred color information that is complementary to edge information captured by HOG, LBP and Gabor. The method ID for this feature is MEAN.

The only parameter available for MEAN method is cache. If cache is set to 1, the MEAN features are cached.

2.5 Command Line Parameters

The command line parameters provide information regarding the input and output of the detector. An example is shown below.

```
./PFI -r  --facePartitionQuery probefile \
 --faceFactors 3,5,7 --faceSampleDir "C:\data\images" \
 --facePartitionFile  "C:\data\partitions\gallery.txt" \
 --featureModels "C:\data\features.txt" --output results
```

In this example, the face identification is performed using as gallery samples defined in file *galley.txt* (–facePartitionFile) and as probe samples defined in *probefile.txt* (–facePartitionQuery) (note that the extension and the path is not used for this file). The face identification is performed considering 3, 5 and 7 PLS factors (–faceFactors) and using feature extraction methods defined in file *features.txt* (–featureModels). The results will be stored in the directory *results* (note that this directory must exist before the execution).

The general command line with its possible parameters has the following format.

```
./PFI -r [-o <directory>] [--threads <number>] \
   [--faceWidth <number>] [--faceHeight <number>] \
   [--facePartitionQuery <filenames>] \
   [--faceFactors <list of number>] \
   [--faceSampleDir <directory>] \
   [--facePartitionTraining <filename>] \
   [--facePartitionFile <filename>] \
   [--extraSamples <directory>] \
   [--featureModels <filename>] [--version] [-h]
```

The following table describe the user options for the detection (the second column indicates if an argument is required (r) or optional (o)).

`-r`	r	Indicates that face identification should be performed.
`-o`	r	Directory where the results will be stored (for more information regarding the output, please refer to Section 2.6). Note: this directory must exist prior to the execution.
`--threads`	o	Number of threads used to build the one-against-all PLS models.
`--faceWidth`	o	Width of the face sample. If not set, the value in the configuration file is used.
`--faceHeight`	o	Height of the face sample. If not set, the value in the configuration file is used.

`--facePartitionQuery`	r	Partition file containing the probe samples, in the format showed in Section 2.1. This file is assumed to be in the same directory of the file defining the gallery samples. Therefore, its path is not added in this parameter. Note: this file is assumed to have extension .txt and this extension cannot be added.
`--facePartitionTraining`	r	Partition file containing the gallery samples, in the format showed in Section 2.1. The full path for this file is required for this parameter.
`--faceSampleDir`	r	Directory containing sample images. Note: this directory contains subdirectories named with subject's ID (one subdirectory per subject). Please refer to the directory tree accompanying the software (subset of FERET in directory ????).
`--extraSamples`	o	Directory containing extra samples (please refer to the paper to see a discussion regarding the use of extra samples).
`--featureModels`	r	Location of the configuration file described in Section 2.3.
`--version`	o	Version of the software.
`-h`	o	Help message.

2.6 Results and Output

The method creates a set of files as output in the directory defined by parameter -o. The first file with name ?????.txt, where ????? is a sequential number padding with zeros, contains setup information, such as command line, feature descriptors used, and CMC values for each number of PLS factors. An example of this file is shown below.

```
#------------------
# command: command line used...
#------------------
# gallery and partition files
trainID:fa_small (100 subjects);
testID:fb_small;
nlevels:1
# feature descriptor parameters
featureModel:level=0;
  (HOG,MEAN);
  HOG:<16,16,4,4>;<32,32,8,8>;
  MEANFEATURE:<110,110,1,1>;
  HOGPARAM:<UseColorFrequency: 1><UseHOG: 1><cache: 0>
  MEANFEATUREPARAM:<cache: 0>
meanNumFeatures:26365;
SampSubjTrain:-1;
SampSubjTest:-1;
faceHeight:110;
faceWidth:110;
secLevelFeatures:false;
# CMC according to the number of factors
[
factors:3;
CMC:0.960,0.990,0.990,1.000,1.000,1.000,1.000,1.000,1.000, ...
];
factors:5;
CMC:0.970,0.990,0.990,1.000,1.000,1.000,1.000,1.000,1.000, ...
];
```

The remaining output files contain the response scores, each file for a different number of factors. These files have the name: scores_?????_factors_??.txt, where the identifier ????? is the same as the first file and ?? indicates the number of factors. Each line of these files has the following format.

```
<probe ID> <gallery ID1> <score1> <gallery ID2> <score2> ...
```

Each line contains the identification results obtained for a probe sample identified by ID. Then, the gallery subjects most similar to that probe sample are listed in descending order, given by the score. Therefore, the gallery ID1 is, according to the face identification method, the best matching to the probe sample ID (if both IDs are the same, this match will be considered a correct match).

A fragment of one of these files containing the scores is shown below. An example of incorrect match returned by the identification method is shown in the last line, in which candidate with the highest score has ID 00052 but the probe ID is 00006.

```
00001 00001 0.333 00007 0.139 00013 0.107 00012 0.079 ...
00002 00002 0.419 00028 0.119 00059 0.107 00088 0.085 ...
00003 00003 0.627 00009 0.067 00002 0.063 00016 0.057 ...
00004 00004 0.330 00019 0.104 00015 0.104 00032 0.093 ...
00005 00005 0.285 00028 0.091 00053 0.081 00059 0.078 ...
00006 00052 0.620 00006 0.084 00008 0.082 00030 0.062 ...
```

2.7 Execution Example

In this section, we provide an execution example based on a subset of the FERET dataset provided with the software. These steps are also described in the example.txt file that comes in the package.

The package contains the following directories: doc (this documentation), SampleData (a subset of the FERET dataset), software (directory containing the executable).

The following command is used to perform face identification considering the subset of the FERET dataset sent with the software. The file features.txt defines the feature extraction methods used to perform the identification (this set of features is a subset of the features used in the TIP paper [2]).

```
./PFI.exe -r --facePartitionQuery fb_small --faceFactors 3,5,7 \
--faceSampleDir ../SampleData/smallFERET/images/ \
--facePartitionFile ../SampleData/smallFERET/partitions/fa_small.txt  \
--featureModels ../SampleData/smallFERET/parameters/features.txt \
--output results/
```

This example must be executed from the `software` directory. It executes face identification considering partition `fa_small` as gallery and `fb_small` as the probe samples. **Note:** the directory `results` must be created beforehand, inside the directory `software`.

Note: To obtain the data (cropped images and partition files) used in our ECCV [1] and TIP [2] papers, please send an e-mail to william@dcc.ufmg.br

Bibliography

[1] W. R. Schwartz, H. Guo, and L. S. Davis, "A Robust and Scalable Approach to Face Identification," in *European Conference on Computer Vision*, ser. Lecture Notes in Computer Science, vol. 6316, 2010, pp. 476–489.

[2] W. R. Schwartz, H. Guo, J. Choi, and L. S. Davis, "Face Identification Using Large Feature Set," *IEEE Transactions on Image Processing*, 2012, in press.

[3] P. J. Phillips, H. Moon, S. A. Rizvi, and P. J. Rauss, "The FERET Evaluation Methodology for Face-Recognition Algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 1090–1104, 2000.

[4] P. J. Phillips, P. J. Flynn, T. Scruggs, K. W. Bowyer, J. C., K. Hoffman, J. Marques, J. Min, and W. Worek, "Overview of the Face Recognition Grand Challenge," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 947–954.

[5] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *CVPR*, 2005.

[6] R. M. Haralick, K. Shanmugam, and Dinstein, "Textural features for image classification," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 3, no. 6, pp. 610–621, 1973.

[7] T. Ahonen, A. Hadid, and M. Pietikainen, "Face Description with Local Binary Patterns: Application to Face Recognition," *IEEE T. PAMI*, vol. 28, no. 12, pp. 2037–2041, 2006.

[8] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *PAMI*, vol. 24, no. 7, pp. 971–987, 2002.

[9] L. Shen and L. Bai, "A review on Gabor wavelets for face recognition," *Pattern Anal. Appl.*, vol. 9, pp. 273–292, 2006.