University of West Bohemia

Faculty of Applied Sciences

Dept. of Computer Science & Engineering

and

New Technologies for the Information Society

AutoFaceRec

Automatic Face Recognition System

Manual

Ladislav Lenc, Pavel Král

Pilsen 2013

Chapter 1

Introduction

Automatic Face Recognition System (AutoFaceRec) is a tool-kit designed for face detection and automatic recognition from real-world photographs. It means recognizing people in ordinary photographs that are not acquired in controlled environment. The quality of such photographs is significantly lower than in the case of photographs usually used for testing of Automatic Face Recognition (AFR) methods. The face in these photographs is often rotated, tilted or occluded and the pose is not uniform. Therefore, the recognition from such photographs is very difficult. This tool-kit allows creating a fully automated face recognition system from the following modules depending on the needs of the users. Five main modules are implemented:

- Face extractor
- Corpus cleaner
- Face representation creator
- Face model creator
- Recognizer

The face extraction module detects and extracts the face region from photographs. The detection is done using the Viola-Jones algorithm. After face localization, the eyes are detected. If the algorithm detects both eyes the face is rotated so that the eyes were in horizontal line and the face is placed to the center of the image.

Corpus cleaning is an application specific task. In our case, many images for each person are present. But not all extracted images have sufficient quality to create face models. Corpus cleaning means that we choose only the best face examples and these are later used to create the face model.

Face representation creator applies the Scale Invariant Feature Transform (SIFT) on the facial images. It uses the software developed by the SIFT author David G. Lowe. The SIFT features are extracted from the image and stored in a XML file for further processing.

Face model creator prepares data for recognizer module. From all images of one person one example is randomly chosen and it is saved as a test image. The rest of images is used to create a face model.

The Recognizer module tests the prepared test faces against the gallery models.

Chapter 2

Face Recognition System

Many papers presented in the AFR domain concentrate only on the recognition task itself. But if we consider fully automated AFR system we have to include also face detection task In some few cases, the process of face localization (or detection) is considered. Because the faces in the original images are usually sufficiently well aligned, this task is usually reduced to define the face position more accurately.

One example of such system that addresses the issue of imprecisely localized faces is proposed in [1]. The system compensates the face position and also solves partial occlusion and different facial expressions. Only one training example per person is used.

A complete face recognition system is described in [2]. The training images are well aligned (acquired in controlled conditions) whereas the recognized images are real-world photos. The system is based on the Sparse Representation and Classification (SRC) [3] algorithm. It achieves very good results on the FERET database.

Our implemented system is based on real-world photographs. Moreover we need to use multiple images for each person and the number of images is not constant. Therefore, our system has some modules which are not usually implemented in other systems.

2.1 Face Detection

For face detection, we use an implementation of the Viola-Jones algorithm [4] which uses a boosted cascade of simple classifiers. This algorithm was chosen because it is one of the most successful face detection method. Great advantage of this approach is its speed.

Unfortunately, a certain number of incorrectly detected faces occurs in the output. A verification of the detected faces is thus indispensable. In order to avoid the manual processing, we propose utilizing the eye detection for this task. The eyes are detected also by the Viola-Jones algorithm. Only the face images with two successfully detected eyes are kept while the other ones are discarded.

If both eyes are detected successfully, the face is rotated according to the detected positions so that the eyes were in horizontal line. The face is further placed to the image centre. This transformation cannot completely resolve the issue of the variations in the pose and tilt. Nevertheless, the face variations are significantly reduced.

2.2 Corpus Cleaning

We have obtained a large number of face images for each individual. However, these numbers differ significantly. In our previous work [5], we have proved that more training examples play a crucial role for a correct estimation of the face models. Nevertheless, we have also shown that this number should be balanced and the quality of the faces should be as high as possible.

Therefore, we perform this step called "*corpus cleaning*" in order to chose the same number of the most representative face images. We have manually verified on a randomly chosen small face sub-set that a majority of the face images is correct and the erroneous examples differ substantially from this representative set. Our algorithm used to chose only the representative faces is based on this observation.

Let S be a set of all face images extracted by the three steps described above. Let S_i be a set of n face images $I_1, .., I_n \in S_i$ representing one individual. Each face image I_j is represented by the feature vector F_j computed by the SIFT algorithm. The pseudo-code of the proposed algorithm is given below.

N = face image number/individual
K = required face image number/individual
for all $S_i \in S$ **do**
  while $N > K$ **do**
    for all $F_j \in F$ **do**
      compute the face model M using the feature set $F \setminus F_j$
      compute the similarity FS_j between F_j and M by the Equation 2.2
    end for
    compute an average value av_{FS}
    compute a standard deviation sd_{FS}
    compute a similarity threshold $ST = av_{FS} + \frac{sd_{FS}}{\sigma}$
    if $\exists I_j : FS_j < ST$ **then**

for all $FS_k < ST$ **do**
remove the face image I_k from the face corpus
$N \leftarrow N - 1$
end for
else
remove the face image I_{min} with the minimal similarity FS_{min} from the face corpus
$N \leftarrow N - 1$
end if
end while
end for

2.3 Face Representation

The face representation is created using a set of Scale invariant feature transform (SIFT) features. The algorithm was proposed by David Lowe in [6] for object recognition. However, this algorithm proved to be suitable also for the face recognition task [7].

The first step is the determination of extrema in the image filtered by the Difference of Gaussian (DoG) filter. The input image is gradually down-sampled and the filtering is performed in several scales.

In the next step, the detected key-points are further examined to choose the "best" candidates. Only points with high enough contrast are used and also points near edges are discarded. Then, orientation is assigned to each key-point. The resulting set of points is then used for creation of feature vectors (descriptors). Each descriptor contains a vector of the length 128 and also the coordinates of the point.

2.4 Face Model Creation

To create the face model so called composed face is used. The representations of each example image of one person are put together and create the composed face. This ensures better robustness of the whole system. Using more face examples in one model allows more variations in recognized images.

The face model creation module simplifies the testing. It randomly chooses one example image of each person which is used for testing. The remaining images are used for creation of the composed model. The models are then used as gallery.

2.5 Recognizer

The SIFT based Kepenekci method is utilized for recognition in our system. This approach combines two methods of matching and uses a weighted sum of the two values as a result as follows.

Let T be a test image and G a gallery image. For each feature vector t of the face T we determine a set of relevant vectors g of the face G. A vector g is relevant iff:

$$\sqrt{(x_t - x_g)^2 + (y_t - y_g)^2} < distanceThreshold \tag{2.1}$$

where x and y are coordinates of the feature vector points.

If no relevant vector to vector t is identified, the vector t is excluded from the comparison procedure. The overall similarity of two faces OS is computed as an average of similarities between each pair of the corresponding vectors as:

$$OS_{T,G} = mean\left\{S(t,g), t \in T, g \in G\right\} \tag{2.2}$$

Then, the face with the most similar vector to each of the test face vectors is determined. The C_i value informs how many times the gallery face G_i was the closest one to some of the vectors of test face T. The similarity is computed as C_i/N_i where N_i is the total number of feature vectors in G_i. Weighted sum of these two similarities is used as similarity measure:

$$FS_{T,G} = \alpha OS_{T,G} + \beta \frac{C_G}{N_G} \tag{2.3}$$

The face is recognized as follows:

$$\hat{FS}_{T,G} = \arg\max_{G}(FS_{T,G}) \tag{2.4}$$

The cosine similarity is used for vector comparison.

Chapter 3

Programmer's Documentation

The program is written in the C++ language. It is based upon the Qt and OpenCV libraries.

3.1 Program description

A brief description of the classes is presented below.

3.1.1 FaceExtractor

This class performs the face detection task and also the eye detection and rotation. It utilizes the Viola-Jones detector for both face and eyes detection. The constructor has two arguments *inputFile* and *outputDirectory*. It calls the method *detect()* which performs the face detection,eyes detection and rotation tasks.

3.1.2 CleanCorpus

The class is used for corpus cleaning. The constructor takes three arguments: *cleanedDirectory* and *discardedDirectory* and *imagesCount*. It walks through all subdirectories in the *cleanedDirectory* and applies method *cleanOneDirectory* on them. *imagesCount* specifies the number of images to be left in each directory. The discarded images are moved to the *discardedDirectory*.

3.1.3 Sift

It applies the SIFT algorithm on given image. The binary created by David G. Lowe is called for this task.

3.1.4 Face

This class represents one face. It has constructor which creates the face representation from given image. Method *saveFace* is used to save the representation in XML format. This class utilizes the *Sift* class

3.1.5 CreateModels

This class is used to create the face models and test files from all examples of one person. It takes the set of XML files and creates the *model.xml* and *test.xml* from them

3.1.6 Recognizer

This class performs the testing on the prepared data. It loads the gallery (models) and test images. Each test image is compared with the gallery images using SIFT based Kepenekci method.

Chapter 4

User's Guide

4.1 Requirements

This program is intended to be executed on Linux systems. To compile the source codes following programs and libraries are required:

- g++ compiler
- Qt library
- OpenCV library

4.2 Compilation

The compilation has two steps. The first one is the creation of the *Makefile* using the *qmake* utility. To perform this task use the following command:

- qmake -o Makefile recognition_sw.pro

The second step is the compilation using program *make*:

- make

The program is created in the *bin* directory.

4.3 Usage

The program is a command line utility. Execution of the program with no arguments shows the man page with explanation of possible usage. The first

argument specifies the task to be done. Possible values are: extract, clean, process, models, test.

4.3.1 Face Extractor

It takes one image as input. The image must be in PNG, PGM or JPEG format. It detects faces in the image and stores them into specified directory. Output images are in PGM format and are numbered from 1.

Usage:

- ./bin/recognizer -extract inputImage outputDirectory

4.3.2 Corpus Cleaner

It is assumed that we have sorted the images of distinct people in directory structure. Each directory contains images of one person. First parameter of the program is the directory containing these subdirectories. Second argument is the directory into which are moved the discarded images. The last argument is required number of images which will be left in the cleaned directory.

Usage:

- ./bin/recognizer -clean cleanedDirectory discardedDirectory imagesCount

4.3.3 Face Representation Creator

This program takes the input image and applies the SIFT algorithm on it. The resulting SIFT descriptors are stored into XML file for further processing. First argument is the input image path. The second argument is the number of the person (For simplicity only integer numbers are used instead of person names). The third one is the output XML file.

Usage:

- ./bin/recognizer -process inputImage number outputFile

4.3.4 Face Model Creator

This module prepares data for testing. It takes all images of one person, chose randomly one of them and stores it as *test.xml* file. This one is used for testing. The rest of images is used for creating a face model and is stored as *model.xml* int the same directory. First argument is the directory

containing subdirectories with images of one person. The second one is the output directory. Same subdirectories as in the input directory are created. Each contains the *test.xml* and *model.xml* files.

Usage:

- ./bin/recognizer -models inputDirectory outputDirectory

4.3.5 Recognizer

This program performs tests on the prepared data from previous stages. It loads the XML files created by Face model creator. The *model.xml* are used as a gallery and the *test.xml* are compared with the models in the gallery. First argument is the input directory. The second one, threshold, is optional. It is the *distance threshold* used in the Kepenekci matching. The default value used when no threshold specified is 0. Output of the program is the recognition rate.

Usage:

- ./bin/recognizer -test inputDirectory [threshold]

Acknowledgements

This work has been partly supported by the UWB grant SGS-2013-029 Advanced Computer and Information Systems and by the European Regional Development Fund (ERDF), project "NTIS - New Technologies for Information Society", European Centre of Excellence, CZ.1.05/1.1.00/02.0090. We also would like to thank Czech News Agency[1] for support and for providing the photo data[2].

[1] http://www.ctk.eu

[2] http://multimedia.ctk.cz/en/foto/

Bibliography

[1] Aleix M. Martinez, "Recognizing imprecisely localized, partially occluded and expression variant faces from a single sample per class," 2002.

[2] A. Wagner, J. Wright, A. Ganesh, Zihan Zhou, H. Mobahi, and Yi Ma, "Toward a practical face recognition system: Robust alignment and illumination by sparse representation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 2, pp. 372–386, 2012.

[3] Ke Huang and Selin Aviyente, "Sparse representation for signal classification," *Advances in neural information processing systems*, vol. 19, pp. 609, 2007.

[4] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Conference on Computer Vision and Pattern Recognition*, 2001.

[5] L. Lenc and P. Král, "Face recognition under real-world conditions," in *International Conference on Agents and Artifitial Intelligence*, Barcelona, Spain, February 14-18 2013.

[6] David G. Lowe, "Object recognition from local scale-invariant features," in *International Conference on Computer Vision*, 1999.

[7] M. Aly, "Face recognition using sift features," 2006.