libface 0.0.1-prealpha1 Tutorial

Purpose of the document

This document intends to explain

- what features are currently (0.0.1-prealpha1) implemented in the libface system
- how you can use them
- what does not work in this version (why is it called pre-Alpha)

In case you did not understand one or more issues discussed in this document (or are not sure whether you understood it, or not), feel free to contact me, Dimitri A. Pissarenko.
This document does not describe, how the eigenface face recognition algorithm works. If you have no idea about what eigenfaces are, please read the section *2, How does it work?* of the paper *Eigenface-based facial recognition*, located at the libface homepage.

Initialization

Before you can perform any sensible experiments with libface, you have to create the eigenfaces. This is the purpose of the program called *faceinit*. It calculates the eigenfaces of a given set of images and stores these eigenfaces in a *parameter file*.

Imagine, you have some images from the *Yale face database*, which are located in directory *yale-img*. You want to create 2 most important eigenfaces and store them in a file *yale-img-ef.dat*. Then, you should use the following command line.

```
faceinit.exe -ef 2 -o yale-img-ef.dat yale-img\subject01.centerlight.png
yale-img\subject01.glasses.png yale-img\subject01.happy.png
```

After *-ef* you write the number of eigenfaces you want to extract, then the name of the file, where they should be stored (*yale-img-ef.dat*) and at the end all the files of the image set (note that under Unix and Unix-like shells you may be able to use the wildcards, and so you would not need to write the names of all your image files).

If *faceinit* is executed successfully, you will see a screen like this one


C:\WINDOWS\System32\cmd.exe
C:\work\faces\artefacts\2003_11_03_tutorial\sw>faceinit-yale.cmd
C:\work\faces\artefacts\2003_11_03_tutorial\sw>faceinit.exe -ef
.dat yale-img\subject01.centerlight.png yale-img\subject01.glass
subject01.happy.png yale-img\subject01.leftlight.png yale-img\s
ses.png yale-img\subject01.normal.png yale-img\subject01.right
img\subject01.sad.png yale-img\subject01.sleepy.png yale-img\su
d.png yale-img\subject01.wink.png yale-img\subject02.center
mg\subject02.glasses.png yale-img\subject02.happy.png yale-
eftlight.png yale-img\subject02.noglasses.png yale-img\subje
yale-img\subject02.rightlight.png yale-img\subject02.sad.png
ect02.sleepy.png yale-img\subject02.surprised.png yale-img\su
Recognition parameter setup.
22 input images will be used, 2 eigenfaces will be generated.
loaded yale-img\subject01.centerlight.png.
loaded yale-img\subject01.glasses.png.
loaded yale-img\subject01.happy.png.
loaded yale-img\subject01.leftlight.png.
loaded yale-img\subject01.noglasses.png.
loaded yale-img\subject01.normal.png.
loaded yale-img\subject01.rightlight.png.
loaded yale-img\subject01.sad.png.
loaded yale-img\subject01.sleepy.png.
loaded yale-img\subject01.surprised.png.
loaded yale-img\subject01.wink.png.
loaded yale-img\subject02.centerlight.png.
loaded yale-img\subject02.glasses.png.
loaded yale-img\subject02.happy.png.
loaded yale-img\subject02.leftlight.png.
loaded yale-img\subject02.noglasses.png.
loaded yale-img\subject02.normal.png.
loaded yale-img\subject02.rightlight.png.
loaded yale-img\subject02.sad.png.
loaded yale-img\subject02.sleepy.png.
loaded yale-img\subject02.surprised.png.
loaded yale-img\subject02.wink.png.
Parameters saved to yale-img-ef.dat successfully.
C:\work\faces\artefacts\2003_11_03_tutorial\sw>

You may wish not only to calculate the eigenfaces, but also to look at them visually. For this purpose, you should use the *-v* argument of *faceinit*, so that the above command line call becomes

```
faceinit.exe -v -ef 2 -o yale-img-ef.dat yale-img\subject01.centerlight.png
yale-img\subject01.glasses.png yale-img\subject01.happy.png
```

This call generates three images

1. vis0.png - the average face
2. vis1.png - the first eigenface (the eigenface with the highest eigenvalue)
3. vis2.png - the second eigenface (the eigenface with the second-highest eigenvalue)

Comparing faces

After you created the eigenfaces (the *parameter file*), you can compare two images and calculate the similarity between these images. This is done using the program *facecmp*. It takes three parameters - the parameter file (where the eigenfaces are stored) and two image files, which should be compared.

For instance, if you want to compare files *yale-i mg\subject01.centerlight.png* and *yale-img\subject02.wink.png*, using the parameter file *yale-img-ef.dat*, you should enter following command

```
facecmp -p yale-img-ef.dat yale-img\subject01.centerlight.png yale-img\subject02.
```

This call leads to following output:

```
C:\work\faces\artefacts\2003_11_03_tutorial\sw>facecmp -p yale-img-ef.dat yale-i
mg\subject01.centerlight.png yale-img\subject02.wink.png
Image compare.
Comparing yale-img\subject01.centerlight.png with yale-img\subject02.wink.png.
Factor for ef 1:  ***
Factor for ef 0:  ***
distance is 26.782839
image match probability = 0.000000

C:\work\faces\artefacts\2003_11_03_tutorial\sw>
```

The only value, which is interesting for you at the moment, is the line

```
distance is 26.782839
```

The larger the distance, the larger the difference between these images.

Reconstructing original images from eigenfaces

If you have the eigenfaces, it is possible to reconstruct the original image from the parameter file and some face image (of a person, whose other images are among the images, from which the eigenfaces were calculated).

For this purpose, following command-line call is used

```
faceview -p yale-img-ef.dat yale-img\subject02.rightlight.png
```

This program reconstructs the image, given the eigenfaces in *yale-img-ef.dat* and the image *yale-img\subject02.rightlight.png* and creates several files, called *traitvis0.png, traitvis1.png, ..., traitvisN.png*. The last file, *traitvisN.png* is the reconstructed image.

Why is libface 0.0.1-prealpha1 buggy?

Because

- images, reconstructed by *faceview* look totally different from original images (in theory, they should look very similar, as described in the original paper by Turk and Pentland)
- sometimes, distance between two images of different people is *less* than distance between two images of the *same* person.

At the moment, we are working at correcting these problems.

We appreciate any hints concerning potential causes of these (or other) problems.

2003-11-03 20:08 Dimitri A. Pissarenko