Facial Recognition

using different methods and comparing them

By Josua Philippot
Félix Yriarte

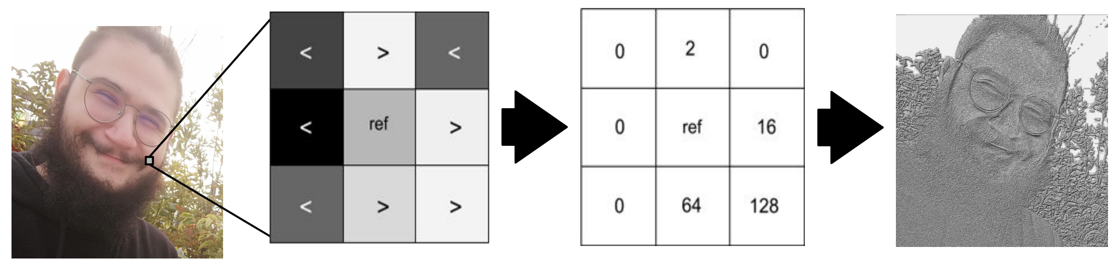
UNIVERSITÉ DE MONTPELLIER
FACULTÉ DES SCIENCES MONTPELLIER

Facial recognition consists of :

1. Identifying discriminating features
2. Matching these features against a database

Local Binary Patterns Histograms

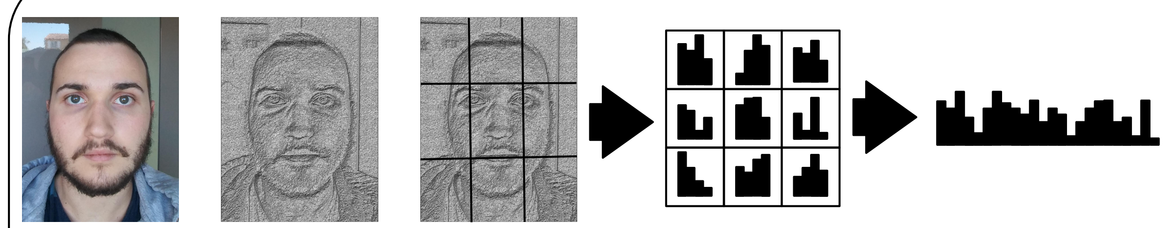
LPB consist in categorizing pixels depending on their neighbourhood values. We then try to match a histogram of theses values with a database containing other images' histograms.



We assign a byte that represents their relative neighbour configuration to every pixel

The reference pixel is assigned 11010010 => 210

A new image is made using our assigned values



We then split the LBP image into equal parts and compute a histogram for each block

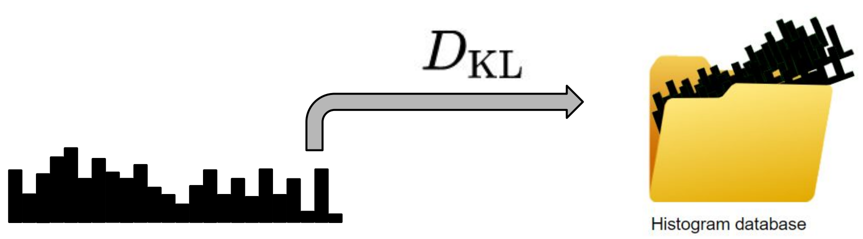
The concatenation of these histograms makes up our face identifier !

Matching a query image with our database

Computing the relative entropy of two concatenations (Kullback-Leibler Divergence) gives a good sense of whether or not the images were similar. Let P and Q be 2 concatenated histograms :

$$D_{\mathrm{KL}}(P\|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

The greater the divergence, the least similar the images.



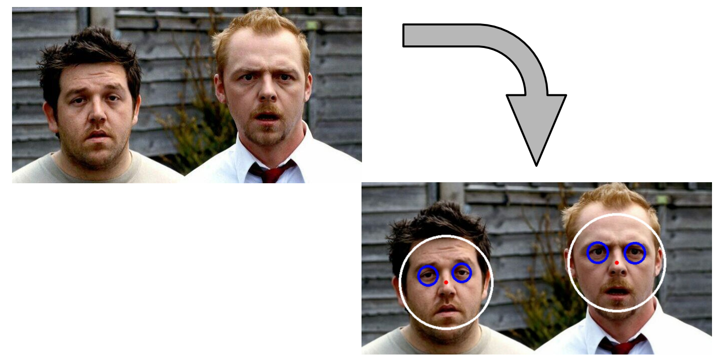
To find a match in our database, the query concatenated histogram is matched against every entry using KL divergence.
If no entry is close enough to the query, no match is found.

Haar cascade classifier

In real life, the input images of our systems are very likely to be off-centered, filled with "noise" and only a very small area is useful to us : the face. That is why we needed to find a way to detect and isolate faces in an image, we decided to use a Haar cascade classifier.

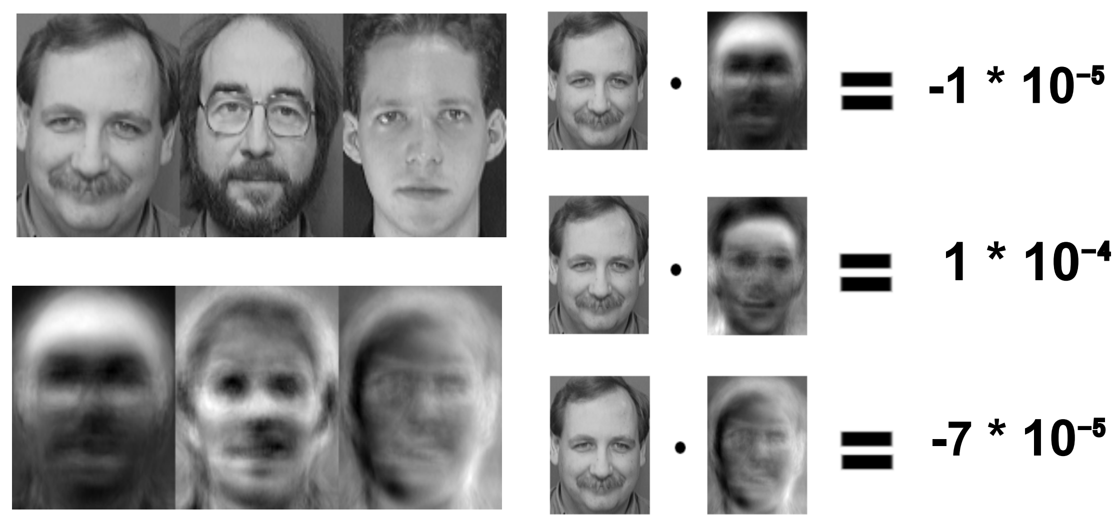
In a nutshell, Haar works as follow :

- Set-up a database containing "positive" images (which contain faces) and "negative" ones (which don't).
- Extract the pseudo-Haar features.
- Train the classifier on our database.
- Detect faces on other images (not in our database).



Eigenfaces

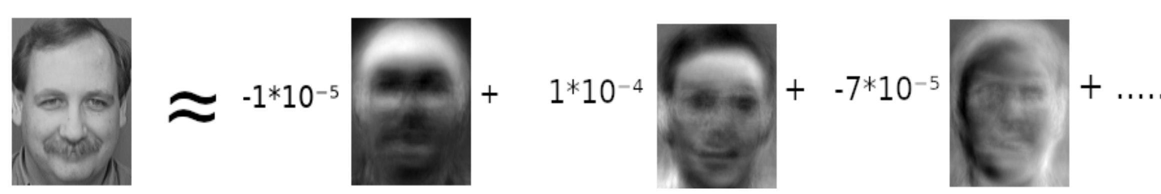
Computing an eigenface set



Eigenfaces are the eigenvectors of the covariance matrix of a set of face images. Eigenfaces are the principal components of the imageset.

A face image can be projected onto an eigenface (using dot product). The resulting float indicates how similar the face and eigenface are.

Authentication using eigenfaces



By projecting a face image on a set of eigenfaces, one gets a vector of floats (that is, a float per eigenface). These floats can be used to describe a face, *i.e.* a face image can be seen as a linear combination of eigenfaces.

Two linear combinations can be compared to deduce a metric of resemblance between two faces.

By adding together the eigenfaces (multiplied by the float factors), one gets a reconstruction of the projected face.

CNNs

Sources / Special Thanks

Special thanks to Pauline Puteaux and William Puech for their continued help and guidance.