



# Overview of Face Recognition Technology

## SPECIFICATION OF OUR FACE RECOGNITION SDK

The library we are providing performs face authentication on an image: it decides if a claimed person should be considered as a "client" or an "impostor".

## FUNCTIONALITIES

The library has 4 main functionalities. It performs:

- **Detection:** localizes one or more faces in an image or a video (frontal face detection)
- **Verification:** check a claimed face against a biometric model and tell if it is the expected person or an impostor. Biometric model adaptation is possible with new images.
- **Identification:** chooses from a list of biometric models which one corresponds to an image
  - Face identification able to work in **closed set and open set mode**
    - **closed set:** claimed person is in the database
    - **open set mode:** claimed person may not be in the database
  - Face identification algorithm can be chosen according to your needs:
    - **real time:** standard algorithm for real time calculation on a video stream with a standard PC
    - **mobile:** optimized algorithm for embedded devices; less accurate but quicker
    - **precise:** slower but more precise algorithm
- **Liveliness detection:** analyses a stream to detect if the client is a real person (detects photo attacks for example)

Security levels can be chosen according to your security needs.

## ROBUSTNESS

The library is robust to low resolution, slight light variation and face occlusion. It is also robust to beard variations, glasses changing, skin tone change... Heavy side face lighting can compromise recognition. The background has no influence for face recognition.

The library can be embedded in multi-threaded applications too. This SDK have been used in several applications for years and all known bugs have been corrected.

## PLUS

Sample codes for the main functionalities (detection, face model creation, verification and identification) are provided along with the library with a clear documentation. Error codes are handled for debugging purpose.

## INTERFACE

The API is in C language with a full documentation. You can link with the library from almost any language (C, C++, C#, Java, VB, .NET...).

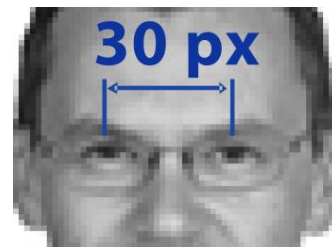
## SUPPORTED PLATFORMS

<i>Computer</i>	<i>Mobile</i>
Microsoft Windows XP, Vista, 7	Android
MacOS	iOS
Linux	



## REQUIREMENTS

- At least 30 pixels between eyes
  - Subject at about a maximum of 1m from a standard webcam with a 320x240px stream (6 m for a fullHD stream with same FOV)
- Rotation: yaw/pitch/roll  $\pm 20^\circ$  (although frontal face is required)



## COMPUTING TIME AND MEMORY INFORMATION

Localization, verification and identification are very fast. The results presented below have been realized on images with a resolution of 320x240, on an Intel Core2 Duo E8500 @ 3.16GHz CPU and on an iPhone 4.

	Computer ("real time" algorithm)	Mobile ("mobile" algorithm)
face localization	17 [ms] → 58 fps	100 [ms] → 10 fps
face verification	27 [ms] → 37 fps	30-90 [ms] → 11-33 fps
face identification (with 20 identities to check)	62 [ms] → 16 fps	Not tested yet

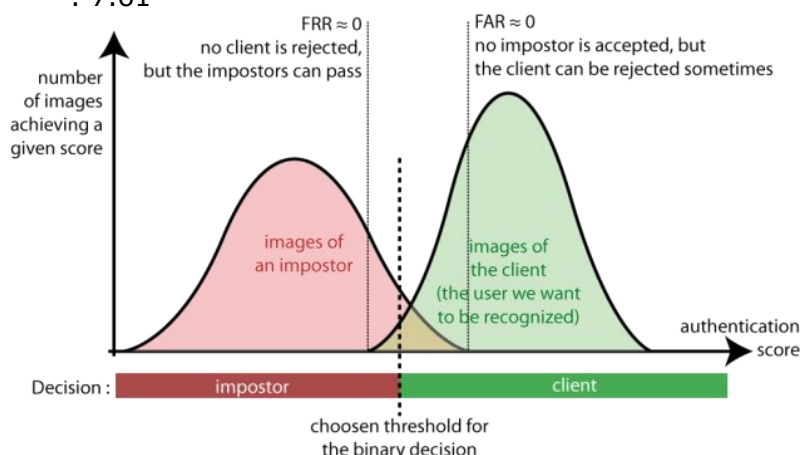
A face model takes about 70 Kb in memory.

## FUNCTIONAL PERFORMANCE

The false acceptance rate (FAR) and false rejection rate (FRR) can be balanced by adjusting the acceptance threshold according to the need (security or convenience). Three "security levels" are preconfigured in the SDK, but the threshold can also be set manually.

Here is our Half Total Error Rate  $HTER = \frac{FRR+FAR}{2}$  for several databases:

- XM2VTS (LP1) : 1.67
- BANCA (Mc) : 5.77
- BANCA (P) : 18.96
- IDIAP : 7.61



## CUSTOMIZED LIBRARY BUILD

We can provide you a library including only the functionalities you need depending on your use case. Please contact us ([info@keylemon.com](mailto:info@keylemon.com)) for more details.