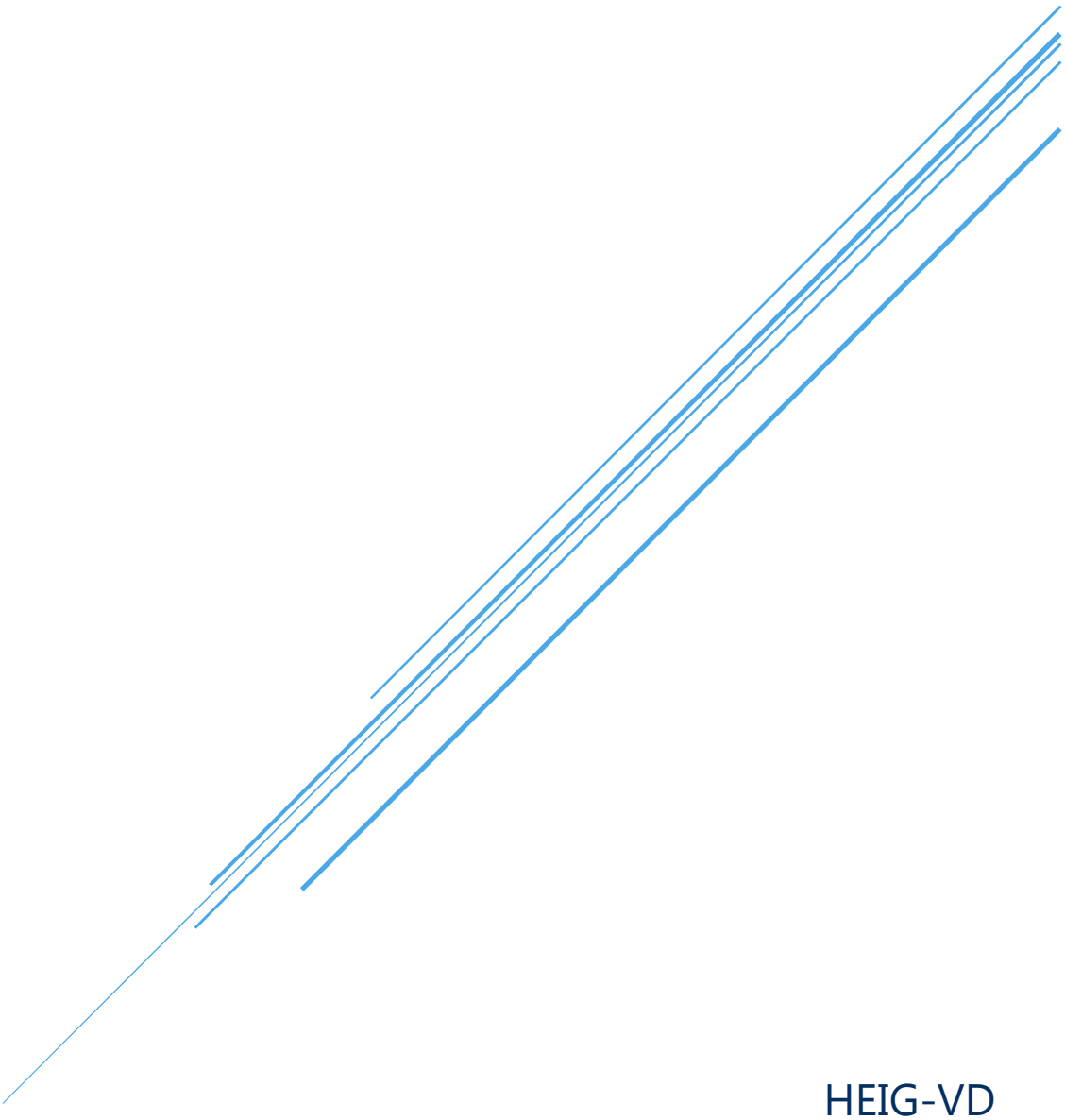
RAPPORT PROJET IUTIL

Détection avec OpenCV



HEIG-VD
Julien PICCALUGA & François WAGNER // IL4

Introduction

Dans le cadre du cours IUTIL nous devons utiliser OpenCV pour réaliser une application de reconnaissance des gestes de la main. Pour plus d'information concernant les objectifs et le cahier des charges de ce projet, veuillez prendre connaissance du fichier « Projet_IUTIL_2013.pdf ».

Ayant eu des soucis avec l'utilisation de Visual Studio, ceux-ci principalement dû à notre inexpérience avec ce logiciel, nous avons choisi d'utiliser Eclipse qui nous est plus familier et que nous connaissons mieux.

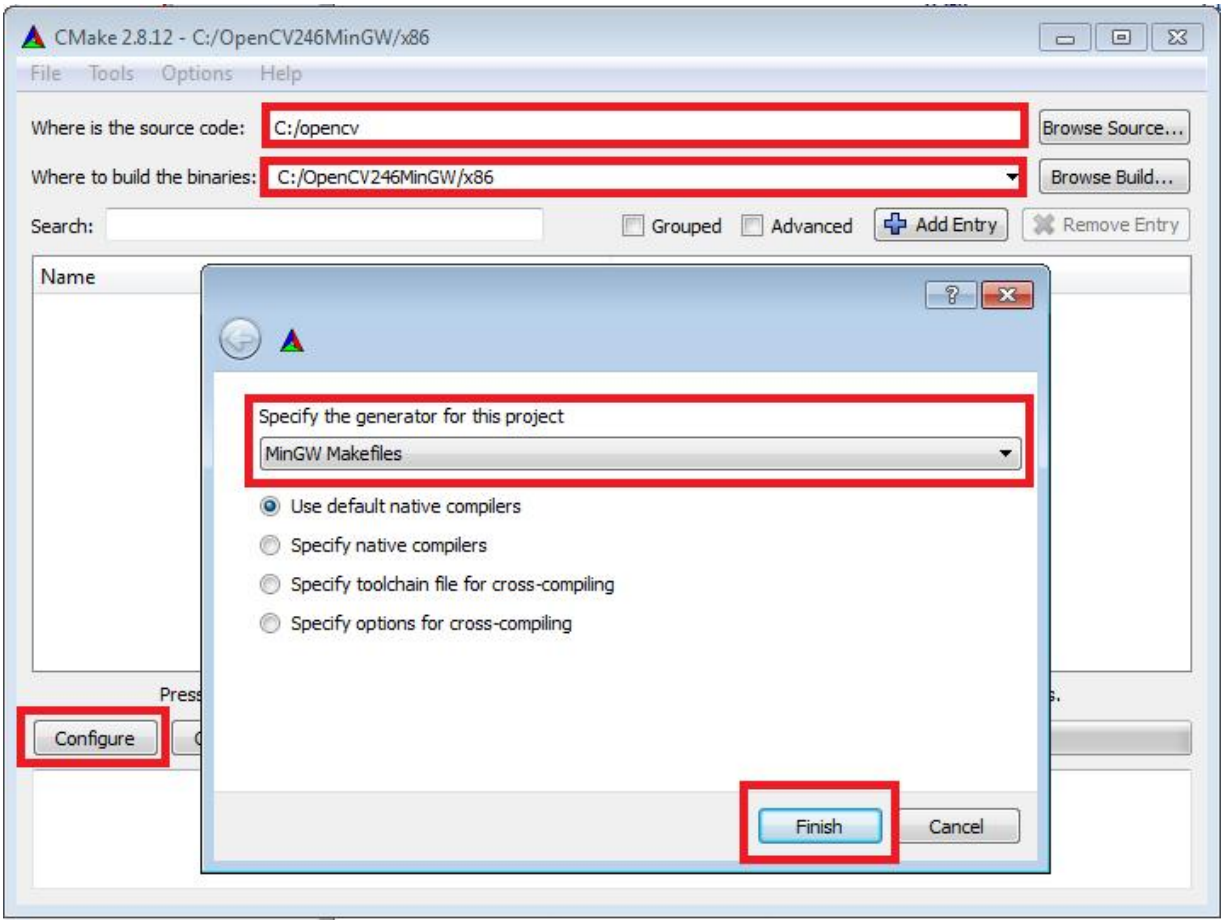
Prérequis

Installation des outils

- Eclipse pour C++: http://www.eclipse.org/downloads/packages/eclipse-ide-cc-developers/keplersr1
    - Extraire l'archive dans un répertoire, par exemple : C:\eclipse4cpp\
- CMake: http://www.eclipse.org/downloads/moreinfo/c.php
    - Faire l'installation par défaut
- OpenCV: http://sourceforge.net/projects/opencvlibrary/files/opencv-win/2.4.6/OpenCV-2.4.6.0.exe/download
    - Installer/extraire OpenCV dans le répertoire suivant : C:\opencv
- MinGW: http://sourceforge.net/projects/mingw/files/Installer/
    - Faire l'installation par défaut et installer les packages "mingw32-base" et mingw32-gcc-g++"
    - Ajouter « C:\MinGW\bin » à la variable d'environnement PATH

Compiler OpenCV pour le compilateur G++

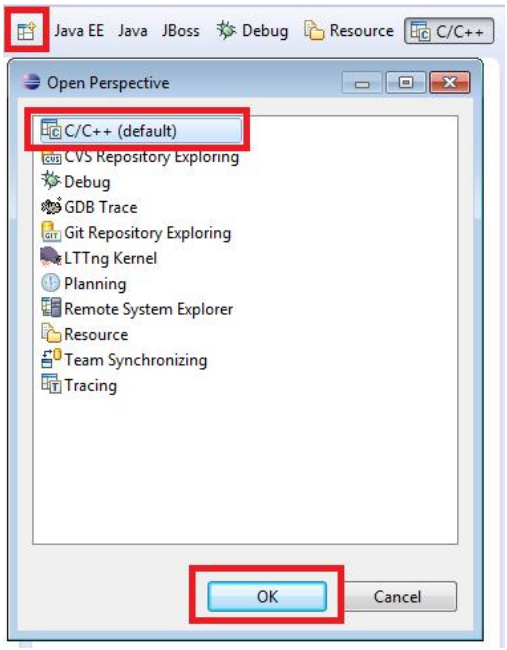
- Exécuter CMake-gui et le configurer comme sur l'image ci-dessous



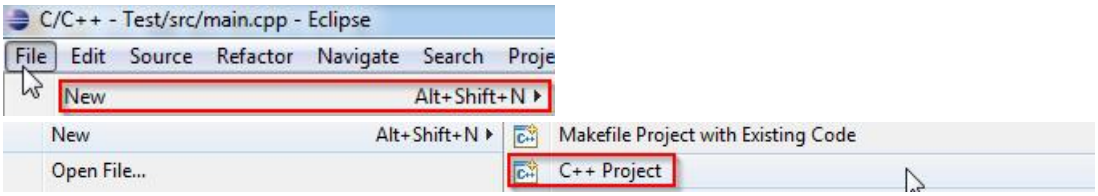
- Laisser par défaut et cliquer sur « Generate »
- Ensuite lancer une commande line et aller dans le répertoire C:\OpenCV246MinGW\x86
- Taper la commande « mingw32-make » pour lancer la compilation2
- Lorsque la compilation est finie, ajouter le répertoire « C:\OpenCV246MinGW\x86\bin » dans la variable d'environnement PATH
- Redémarrer votre machine

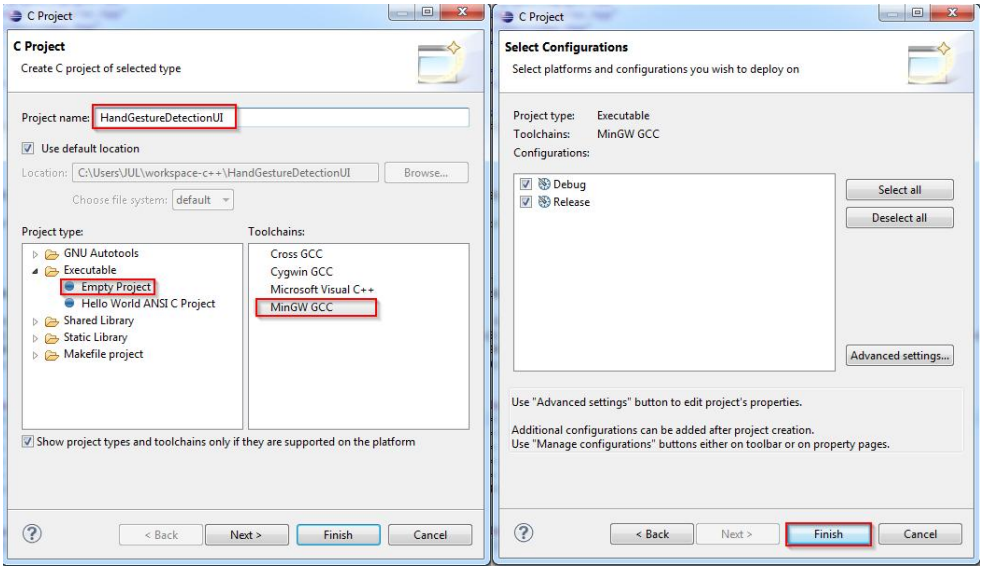
Configurer un projet C dans Eclipse

- Lancer Eclipse : C:\eclipse4cpp\eclipse.exe
- Ouvrir la perspective C/C++

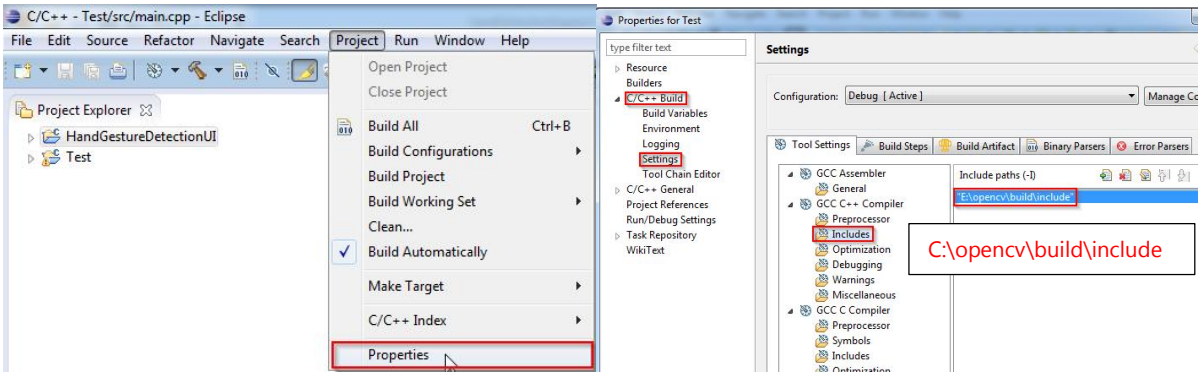


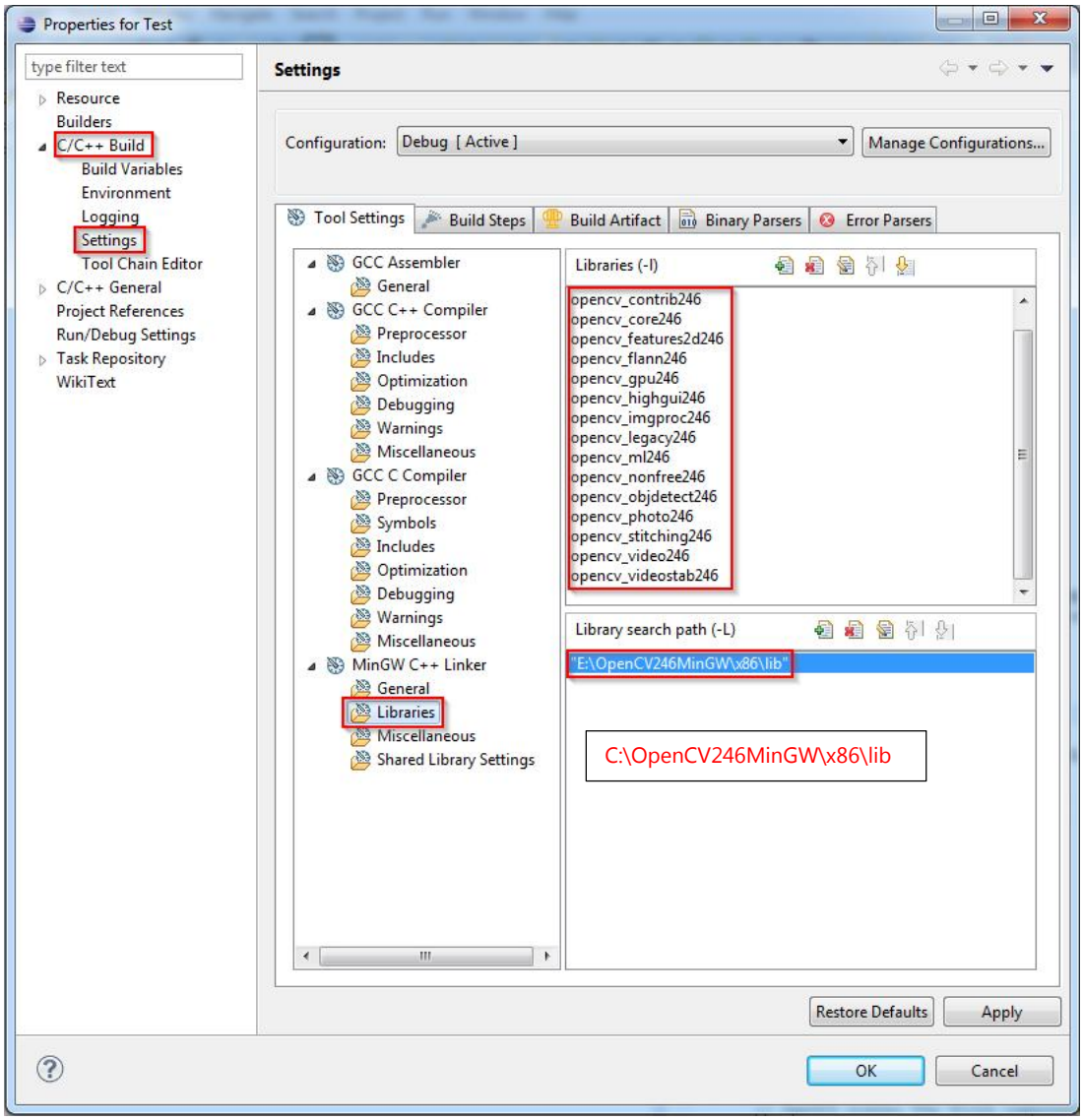
- Créer un projet C





- Configurer le projet





Réalisation

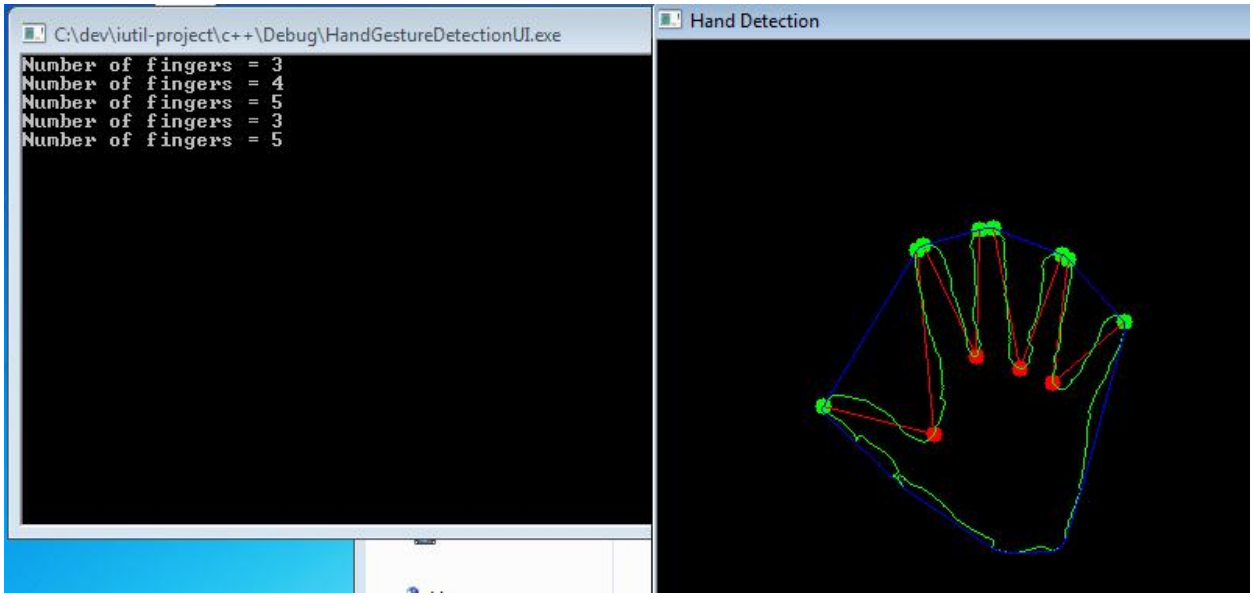
Processus

Lors du lancement du programme un flux vidéo est ouvert avec la caméra de l'ordinateur et le processus boucle en effectuant les étapes suivantes sur chaque image :

1. « Flip image » : une symétrie horizontale de l'image est effectuée pour avoir un effet « miroir » qui se trouve être plus logique pour nous
2. « Face detection » : on détecte le ou les visages se trouvant sur l'image grâce à un fichier « haar » mis à disposition par la libraire et on le passe en noir pour ne pas qu'il nous dérange lors de la détection de la main
3. « Blur » : un appel à la méthode « blur » est effectué afin de flouter l'image pour la lisser et réduire le bruit sur l'image
4. « Skin detection » : la détection de la peau se fait en plusieurs étapes :
    a. Passage de l'image de BGR à HSV
    b. Séparation des canaux HSV en trois matrices distinctes
    c. Un « treshold » est effectué sur chacune des trois matrices, de type CV_TRESH_BINARY_INV pour le canal Hue et CV_TRESH_BINARY pour les canaux Saturation et Value
        i. Matrice H : chaque pixel étant supérieur au seuil (dans notre cas 18) est passé à 0, sinon à 255
        ii. Matrice S : chaque pixel étant supérieur au seuil (dans notre cas 50) est passé à 255, sinon à 0
        iii. Matrice V : chaque pixel étant supérieur au seuil (dans notre cas 80) est passé à 255, sinon à 0
    d. On applique une fonction logique AND pour réunir nos trois canaux en une seule matrice et celle-ci est renvoyée au processus

RÉSULTAT APRÈS DÉTECTION DE LA PEAU

5. « Contour detection » : les différentes étapes de la fonction « contourDetect » sont les suivantes :
    a. Détection des contours grâce à la fonction « findContours » de la libraire
    b. On boucle sur les contours trouvés et on traite tous les contours dont la surface fait plus de 5000, ce qui nous permet d'éliminer les taches qui se trouvent sur notre image (surface < 5000)
        i. Récupération de l'enveloppe convexe de la forme créée par le contour
        ii. Récupération des « defects » de l'enveloppe et création de cercles à chacun des 3 points de chaque « defect » ainsi que des lignes qui les relient
        iii. Calcul du nombre de doigts et affichage du résultat final à l'écran



RÉSULTAT FINAL

Problèmes rencontrés

Lors de ce projet nous nous sommes confrontés à plusieurs problématiques :

- Manque de connaissance dans le traitement d'image
- Problème de la détection de la peau dû aux différentes luminosités par rapport aux endroits où nous nous situons pour développer, car la lumière ne vient pas toujours du même côté
- Egalement un problème pour la détection de la peau à cause des différences dues à la caméra utilisée
- Recherche d'un algorithme permettant de diminuer les 2 problèmes précédents
- Compréhension des différents algorithmes de traitement d'images
- Le comptage du nombre de doigt ne fonctionne pas à tous les coups et mériterait d'être améliorer

Améliorations

L'algorithme de la détection des doigts doit être amélioré afin de supprimer les « defects » n'appartenant aux doigts. Pour cela nous pourrions calculer l'angle créé par nos deux vecteurs partant du creux du « defect » (point rouge) et si celui-ci est plus grand que 90° (angle maximum défini par l'angle formé par le pouce et l'index) on le supprimer.

Une autre amélioration pourrait être de nommer chaque doigt de la main (pouce, index, etc.). Ensuite nous pourrions essayer de détecter certains gestes bien précis (signe victoire, ok, etc.).

Et au niveau du code il nous reste à trouver un moyen de mettre le chemin du fichier haar en relatif plutôt qu'en absolu.