L3 - *« Computer Vision »* *L. Mascarilla,*	**TP n°5 -** **Haar**	

L'objectif de ce TP n°5 est d'expérimenter les principes de la détection d'objets vus en cours. Nous allons suivre des « objets » détectés par une cascade de Haar.

Détection par cascade de Haar

1) Vous utiliserez comme point de départ les détecteurs par cascade de Haar. Vous détecterez un visage dans le flux de la JeVois en utilisant le filtre préentrainé disponible dans 'haarcascade_frontalface_default.xml' et 'haarcascade_eye_tree_eyeglasses.xml' qui se trouvent sur ce site :

https://github.com/opencv/opencv/tree/master/data/haarcascades

Vous vous inspirerez du tutoriel suivant :

https://docs.opencv.org/master/db/d28/tutorial_cascade_classifier.html

et tracerez un rectangle autour des ROI détectées.

2) Les cascades de Haar peuvent détecter des parties du visages : yeux, bouches, nez…

Pour cela, vous trouverez d'autres fichiers xml :

https://github.com/opencv/opencv_attic/tree/master/opencv/data/haarcascades

a) Détectez la bouche et le nez

b) En vous inspirant du code « insertion_image_alpha.py », que vous trouverez sur Moodle, insérez une moustache sur le visage.

c) Il est également possible de détecter des émotions, par exemple en utilisant le fichier

https://github.com/opencv/opencv/tree/master/data/haarcascades/haarcascade_smile.xml

Affichez un smiley (ou le texte « sourire ») sur l'image lorsqu'un sourire est détecté.

3) Les cascades peuvent également servir à détecter des gestes : vous en trouverez une liste dans l'archive déposée sur Moodle ou sur ce site :

https://github.com/Sandeep-Sthapit/HandGestureDetection

a) Choisissez un geste (okay, vicky, aGest…) puis affichez un point au centre de la ROI détectée.

b) Une amélioration possible est d'afficher les positions successives de ce point. Vous pourrez obtenir un résultat visuellement agréable en utilisant ce code :

```
def draw_track(frame,pts,color=(0,255,0)):
    for i in range(1, len(pts)):
        if pts[i - 1] is None or pts[i] is None:
            continue
        thickness = int(np.sqrt(pts.maxlen/ (len(pts)-i + 1)) * 2.5)
        cv2.line(frame, pts[i - 1], pts[i], color, thickness)
```

pts est une file à double entrée : elle contient un nombre maximum d'éléments, quand ce nombre (ici 64) est dépassé, les éléments les plus anciens sont supprimés.

pts = deque(maxlen=64)
L'ajout d'éléments se fait par append.

Remarques :
- la listes des points pourra être utilisée par la suite pour détecter des symboles, des lettres …
- pour les plus curieux, il est également possible de construire son propre détecteur : https://docs.opencv.org/4.5.1/dc/d88/tutorial_traincascade.html

Quand vos algo sont validés sur les PCs, n'oubliez pas de les tester sur la JeVois !