

BLIN Sébastien,
COLLIN Pierre-Henri,
LOUARN Amaury



Université de Rennes 1

Campus de Beaulieu

Licence STS
Cycle Préparatoire Ingénieur Rennes 1 - Informatique et Télécommunications

Rapport de Travail d'Initiative Personnelle Encadrée (TIPE)

Comment la reconnaissance faciale du conducteur peut-elle
améliorer sa sécurité au volant ?

Sous l'encadrement de :

Johanne Bézy-Wendling

Maître de Conférences

Responsable cycle préparatoire ingénieur de l'Université de Rennes I
(spécialité informatique et télécommunications)

Finn Jørgensen

Responsable L3 d'informatique - ISTIC

Résumé

Introduction

Pourquoi ce projet

Partie 1

Les limites de la sécurité routière

Dans cette première partie, nous allons aborder les limites de la sécurité routière, qui sont à l'origine de notre problématique. D'abord, nous verrons les moyens mis en oeuvre pour protéger l'automobiliste. Puis, nous verrons les comportements à risque qui nuisent encore à sa sécurité.

1.1 Les moyens mis en place pour la sécurité du conducteur en France.

Dans le but de visualiser les moyens déployés par les pouvoirs publics, nous ferons dans un premier temps un historique de la sécurité routière depuis le début des années 70 (période qui correspond à l'adoption des premières mesures pour diminuer le nombre de morts sur la route). Ensuite, nous verrons l'impact induit par ces moyens sur l'augmentation de la sécurité au volant.

1.1.1 Historique de la sécurité routière depuis 40 ans

Après la seconde guerre mondiale, et en particulier dès le début des années 50, le nombre d'accidents mortels sur la route a fortement augmenté. Plusieurs facteurs sont en cause : l'expansion du parc automobile, un réseau routier inadapté, ainsi que l'insuffisante formation des conducteurs. Le premier dénombrement en 1954 recensa 7166 tués en 3 jours. à cette époque, la sécurité routière était de loin un problème prioritaire pour le gouvernement car il n'y avait pas encore de politique publique.

à partir des années 60, ce fut le début des opérations de traitement des points noirs ##(peut-être des précisions à apporter mais pas encore trouvé)##. Entre 1960 et 1970, la mortalité augmenta de 55,7% et le trafic est multiplié par 2,3.

En 1972, le Comité Interministériel de la Sécurité Routière (C.I.S.R) fut créé afin de définir la politique de sécurité routière en France. Cette année fut aussi celle qui aura fait le plus de victimes sur les routes avec 16 545 morts. Durant la décennie qui suivie, le gouvernement instaura plusieurs mesures telles que : les limitations de vitesse et l'obligation du port de la ceinture à l'avant. En dix ans, la mortalité diminua de 30% tandis que le trafic global fut multiplié par 1,6.

Au début des années 80, les pouvoirs publics constatèrent une stabilisation de la baisse de la mortalité routière. Ils instaurèrent des plans départementaux de sécurité routière ainsi que

le programme R.E.A.G.I.R (Réagir pour les Enquêtes sur les Accidents Graves et les Initiatives pour y Remédier). Ce fut également le début de la politique locale de sécurité routière. Entre 1980 et 1990, le seuil d'alcoolémie autorisé fut abaissé de 1,2 à 0,8 g/l d'alcool dans le sang, la plupart des véhicules furent équipés d'un système anti-blocage des roues, le nombre de carrefours giratoires augmenta (diminution notable du nombre d'accidents mortels dans les carrefours).

à la fin des années 80, un livre blanc de la sécurité routière fut publié dans le but d'énoncer les orientations majeures des futures politiques de sécurité routière. Entre 1990 et 2000, de nombreuses mesures furent donc mises en oeuvre : en 1990, la vitesse maximale autorisée en agglomération fut fixée à 50km/h et un permis à points fut instauré, le taux d'alcool autorisé dans le sang se limita à 0,5 g/l, l'essentiel du réseau autoroutier était quasiment achevé, la plupart des véhicules furent équipés d'airbags. De plus, le continuum éducatif est mis en place. D'après le site gouvernemental de la sécurité routière, le continuum éducatif exprima l'idée que "l'éducation à la sécurité routière ne se fait pas seulement lors du passage du permis de conduire, mais tout au long de sa vie".

En dix ans, le trafic global augmenta de 20%, alors que la mortalité routière diminua d'autant.

En 2002, la sécurité routière était l'un des principaux chantiers du Président de la République. Un an plus tard, les premiers radars de contrôle/sanction automatiques //—systèmes de sanction automatique—// arrivèrent au bord des routes. La même année, le Conseil National de la Sécurité Routière (C.N.S.R) s'installa ##préciser ses missions##. En 2004, le permis probatoire fit son apparition. Les sanctions sont devenues plus importantes pour les conducteurs en état d'ébriété. En effet, un dépassement du taux légal d'alcool dans le sang entraîne un retrait de six points sur le permis de conduire. En 2000 et 2010, la mortalité baissa de 51,1% alors que le trafic global augmenta de 7%.

Après un bref aperçu sur l'étendu des décisions prises par les gouvernements depuis 1972 en matière de sécurité routière, nous allons maintenant voir quel est le bilan de ces mesures.

1.1.2 Bilan de quarante années de mesures

Premièrement, le nombre de tués sur la route a fortement diminué depuis 1972 (année qui marque la fin de la hausse constante de la mortalité routière). Comme nous pouvons le constater sur le graphique ci-dessus/ci-dessous, ce nombre a été divisé par plus de quatre en quarante ans malgré un doublement du parc automobile. En 2012, 3 653 personnes ont été tuées en 30 jours contre plus de 18 000 en 1972.

Ces progrès observés en matière de sécurité routière ont été obtenus en agissant sur plusieurs facteurs. Lors d'un accident, quatre facteurs essentiels doivent être pris en compte : tout d'abord les facteurs liés à l'infrastructure (conception, entretien et exploitation), les facteurs liés aux véhicules (sécurité passive et active), les facteurs liés aux comportements des usagers (formation, communication, répression), et le progrès des services de secours et de soin. Cependant, il est très difficile de définir la part de chacun de ces facteurs dans l'amélioration de la sécurité routière.

Concernant les facteurs liés à l'infrastructure, nous pouvons retenir plusieurs points qui ont participé à l'amélioration de la sécurité routière : comme la construction d'un réseau autoroutier, le déploiement de barrières de sécurité le long des routes potentiellement dangereuses, les protections anti-éblouissements, des bandes sonores anti-endormissement, les bandes d'arrêts

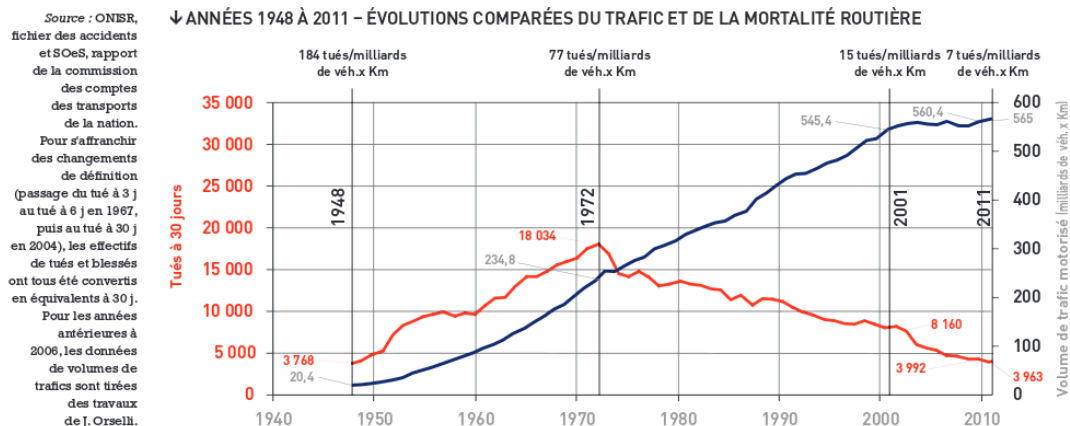


FIGURE 1.1 – Évolution comparée du trafic et de la mortalité routière entre 1948 et 2011

d'urgences.

Ensuite parmi les facteurs liés aux véhicules, la situation actuelle n'a plus rien à voir avec celle d'il y a quarante ans. Le port obligatoire de la ceinture, l'obligation pour les enfants de s'asseoir à l'arrière du véhicule, l'apparition de différents coussins gonflables de sécurité dans le poste conducteur (airbags), la structure des véhicules qui absorbent mieux les chocs (à l'aide des pare-chocs par exemple) ont eu un impact considérable sur la diminution de la gravité des accidents. Nous pouvons aussi évoquer la mise en place progressive dans tous les véhicules de systèmes d'aide à la conduite (ABS par exemple).

Les facteurs liés aux comportements des usagers sont sûrement ceux qui ont eu le plus d'effets sur la diminution du nombre de tués sur la route. Tout d'abord, la formation des usagers qui n'a cessé de croître au fil des années a permis de leur donner une meilleure appréhension de la route. Cette formation s'inscrit dans un processus progressif et continu. Que ce soit en famille, à l'école, lors du passage du permis de conduire, ou pendant le reste de leur vie. Ensuite, la répression a incité les usagers à respecter les nouvelles mesures de sécurité mises en place. Par exemple, sur le respect des limitations de vitesse, du taux d'alcoolémie présent dans le sang, du port de la ceinture, etc. Au fil du temps, la répression s'est durcie avec l'augmentation des points retirés en cas de délit. Enfin, la communication sur la sécurité routière a été primordiale pour faire changer les moeurs. Des campagnes publicitaires (voir ci-dessous) parfois chocs ont fait prendre conscience aux usagers que nous sommes "tous responsables" (slogan utilisé lors des campagnes de prévention).

Suite à cette présentation des moyens mis en oeuvre par les gouvernements successifs et de leurs impacts sur la sécurité routière, nous allons à présent aborder les limites de ces mesures, c'est-à-dire les comportements qui mettent en péril cette sécurité.



FIGURE 1.2 – Karl Lagerfeld pour une affiche de préventions routière

1.2 Les comportements à risques

Malgré toutes les mesures prises pour protéger le conducteur, de nombreux accidents mortels ont encore lieu. La plupart du temps, cet accident n'est pas dû à une défaillance du véhicule, une route endommagée ou encore au mauvais temps, mais à une défaillance humaine. C'est pourquoi, nous analyserons d'abord les principales causes des accidents, puis un type de comportement dangereux en particulier : la conduite à l'aveugle.

1.2.1 Les principales causes d'accidents

Les accidents corporels sont généralement déterminés par plusieurs facteurs. Ces facteurs peuvent influencer sur l'occurrence des accidents, mais aussi sur leur gravité. Ils sont souvent liés entre eux, et il est particulièrement difficile de déterminer le facteur principal, habituellement appelé la "cause" de l'accident.

La vitesse est un facteur prépondérant dans les accidents corporels. C'est également un facteur transversal, car la vitesse est presque toujours présente comme facteur d'occurrence et/ou facteur de gravité. En 2011, en France, au moins 26% des accidents mortels ont pour cause identifiée la vitesse d'après les forces de l'ordre. En Suisse et en Allemagne la vitesse, seule ou associée, représente 40% des accidents mortels.

L'alcool est également un facteur majeur présent dans les accidents. En effet, s'il est présent dans le sang en quantité trop importante, il entraîne une augmentation de la vitesse, la somnolence, l'oubli du port de la ceinture de sécurité. Le taux d'implication de l'alcool dans la mortalité routière reste constant : environ 31%. Les 875 accidents mortels avec au moins un

conducteur ayant une alcoolémie supérieur au seuil autorisé ont provoqué 964 victimes. Parmi les victimes des accidents mortels avec comme facteur l'alcool, les conducteurs ainsi que leurs passagers représentent 70% des personnes tuées.

Les autres facteurs sont :

- L'usage des stupéfiants (en 2011, 455 accidents mortels avec au moins un conducteur testé positivement aux stupéfiants). Ces accidents ont entraînés le décès de 499 personnes (soit 13% de la mortalité routière).
- La prise de médicaments : une étude réalisée par une équipe de l'INSERM (projet CESIR-A) a montré que 3% des accidents étaient attribuables à la prise de produits médicamenteux.
- le respect des distances de sécurité. D'après le Code de la Route : « Lorsque deux véhicules se suivent, le conducteur du second doit maintenir une distance de sécurité suffisante pour pouvoir éviter une collision en cas de ralentissement brusque ou d'arrêt subit du véhicule qui le précède. Cette distance est d'autant plus grande que la vitesse est plus élevée. Elle correspond à la distance parcourue par le véhicule pendant un délai d'au moins deux secondes ». Globalement, plus de 50% des conducteurs ne respectent pas cette règle. Or, le non-respect des distances de sécurité qui entraînent des collisions par l'arrière et des collisions en chaîne totalise 6.2% de la mortalité routière.
- Le port de la ceinture : en 2011, 22% des personnes tuées n'étaient pas ceinturées.

Comme nous venons de le voir, les principales causes d'accidents mettent en avant le comportement de l'usager. Nous allons nous attarder sur une cause d'accident qui n'a pas été évoquée précédemment : la distraction du conducteur.

1.2.2 La conduite à l'aveugle

La conduite à l'aveugle fait référence au relâchement d'attention du conducteur pour effectuer d'autres tâches annexes à l'intérieur du véhicule. Ainsi, ses capacités d'analyse de la circulation et de réaction sont fortement diminuées.

Il existe trois types de distraction : visuelle (regarder autre chose que la route), manuelle (détacher ses mains du volant), cognitive (ne pas être concentré sur la route). Concrètement, cela signifie : utiliser son téléphone portable ou un smartphone, envoyer un sms, manger et boire, parler aux passagers, se maquiller, lire (y compris les cartes), utiliser un système de navigation, regarder une vidéo, régler la radio, un lecteur CD ou mp3, etc. Envoyer des sms est la distraction la plus dangereuse, car elle combine les trois types de distraction. De plus, envoyer ou lire un texte oblige à quitter des yeux la route pendant 4,6 s en moyenne. A 55 mph (environ 88 km/s), c'est comme conduire la longueur d'un terrain de football, les yeux bandés.

En France, certaines études ont montré que 25% à 50% des accidents corporels étaient dus à la distraction du conducteur. A l'étranger, la conduite à l'aveugle est également un facteur de plus en plus important dans la mortalité routière. Par exemple, chaque jour aux Etats-Unis, plus de 9 personnes sont tuées et plus de 1,060 personnes sont blessées dans des accidents où sont impliqués un conducteur distrait. Par ailleurs, beaucoup d'usagers (en particuliers les jeunes) ne semblent pas encore avoir pris conscience du danger de ces pratiques. En effet, d'après un sondage réalisé en 2011 aux Etats-Unis, 69% des conducteurs âgés de 18 à 64 ans avaient téléphoné en conduisant (En Europe, ce taux varie entre 21% au Royaume-Uni à 59% au Portugal) et 31%

avaient lu ou envoyés un sms en conduisant dans les 30 derniers jours avant d'être sondés.

Partie 2

La reconnaissance faciale

Dans cette deuxième partie, nous nous pencherons sur la reconnaissance faciale. Dans un premier temps, nous étudierons l'aspect théorique de la reconnaissance, puis l'aspect expérimental.

2.1 Théorie

Afin d'étudier l'aspect théorique de la reconnaissance faciale, nous allons d'abord l'aborder de façon général. Ensuite, nous analyserons plus en détail différentes méthodes de reconnaissance : Eigenface, Fisherface et LBPH.

2.1.1 Généralités

Après des recherches approfondies, nous avons pu constater qu'il existe énormément de méthodes possibles pour reconnaître un visage. Ces méthodes peuvent également être combinées entre elles (par exemple, la mise en place d'un réseau neuronal ##explication peut-être?##). Cependant, en trois mois, nous nous n'aurions pas eu le temps de mettre en place une telle méthode.

Par ailleurs, nous avons choisi de traiter uniquement des images en noir et blanc, car la colorimétrie a très peu d'impact dans le processus de reconnaissance.

2.1.2 La méthode Eigenface

La méthode de reconnaissance faciale Eigenface a pour particularité de se baser sur des « eigen vectors », c'est-à-dire des vecteurs propres.

L'algorithme est le suivant : chaque image est considérée comme un vecteur avec pour dimension son nombre de pixels. Puis, un ou plusieurs algorithmes recherchent les principales composantes //—plus de précision sur composante?—//. A l'aide de plusieurs images du même visage, nous pouvons alors composer une image du visage « moyen » qui contiendra également les principales composantes (l'eigenface). La méthode de reconnaissance des axes principaux est détaillée ici : Matthew Turk and Alex Pentland. Eigenfaces for recognition. J. Cognitive Neuroscience. 3(1) :7186, 1991. Enfin, il suffit de comparer l'eigenface et les composantes principales

d'une capture d'un visage.

L'avantage de cette méthode est la connaissance de son existence depuis longtemps. Cependant, elle présente également des désavantages non-négligeables. En effet, comme chaque pixel est une dimension, une image 100×100 donne 10000 vecteurs à traiter. Le nombre de données à analyser est donc trop important.

2.1.3 La méthode Fisherface

La méthode de reconnaissance faciale Fisherface se base sur les travaux de Sir R.A. Fisher. L'idée de base de l'algorithme est la suivante : toutes les images qui se ressemblent, se retrouvent proches. Concrètement, imaginons un espace d'images comprenant une échelle représentant les visages. Si nous projetons les images sur l'échelle, nous pouvons dire que //– je n'ai pas compris la suite –//

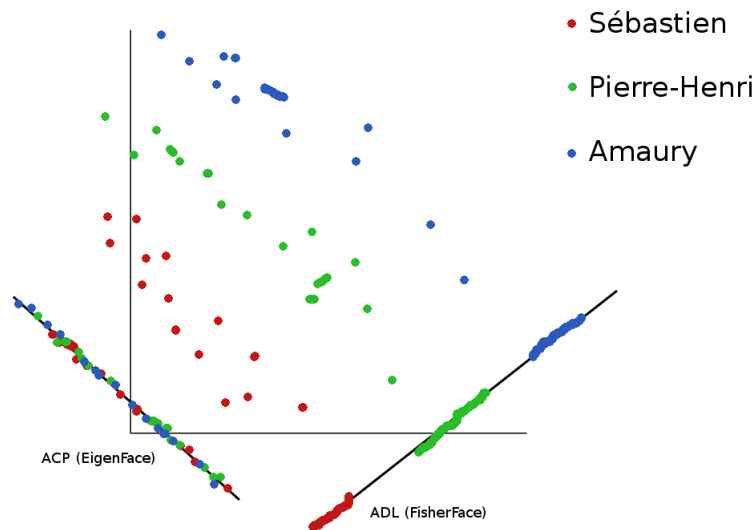


FIGURE 2.1 – Distribution Fisherface

Pour plus de détails : http://docs.opencv.org/modules/contrib/doc/facerec/facerec_tutorial.html#fisherfaces.

Globalement, cette méthode présente les mêmes avantages et inconvénients que la méthode Eigenface. Cependant, elle est moins sensible à la lumière et à la déformation des visages car elle ne se base pas sur des composantes discriminatoires.

2.1.4 La méthode LBPH

La méthode de reconnaissance faciale LBPH (Local Binary Patterns Histogram) consiste à visualiser la valeur d'un pixel (moyenne des trois composantes RGB) par rapport aux pixels voisins.

Pour commencer, l'image est divisée en groupe de pixels. Chaque groupe de pixels correspond à une matrice carrée contenant les valeurs des pixels. Puis, le pixel placé au centre de la matrice est choisi comme valeur de référence. Ensuite, toutes les valeurs de la matrice sont remplacées soit par 0, soit par 1 en fonction de leur valeur. La fonction d'Heaviside, nous dit que

$$\forall x \in \mathbb{R}, H(x) = \begin{cases} 0 & \text{si } x \leq 0 \\ 1 & \text{sinon} \end{cases}$$

Ici, si nous attribuons la valeur 0 si la valeur du pixel est inférieure à la valeur du pixel de référence, 1 sinon. Après cette opération, chaque pixel du groupe est pondéré avec un poids plus ou moins fort (le pixel en haut à gauche a le poids le plus faible, tandis que le pixel en bas à droite a le poids le plus fort). Ainsi, nous obtenons un nombre binaire qui donne une certaine valeur en base 10. Tous les groupes de l'image sont soumis à ce processus pour finalement obtenir un histogramme de l'image. Enfin, il ne reste plus qu'à faire la différence entre deux histogrammes pour comparer deux images.

Cet algorithme est très utilisé comme nous pouvons le voir dans : Facial expression recognition based on Local Binary Patterns :A comprehensive study de Caifeng Shan a, *, Shaogang Gong b, Peter W. McOwan b, où il est combiné avec Adaboost (plugin d'opencv) pour obtenir de meilleurs résultats, et un réseau neuronal. Mais aussi dans : Face Recognition with Local Binary Patterns, Spatial Pyramid Histograms and Naive Bayes Nearest Neighbor (NBNN) classification de Daniel Maturana, Domingo Mery and Alvaro Soto où les auteurs essayent d'améliorer l'algorithme dans le but d'en créer un nouveau : NBNN. Nous n'utiliserons pas cet algorithme par manque de temps, mais également car LBPH est déjà présent dans openCV.

Comme nous venons de le voir, il existe différentes méthodes

2.2 Expérimentations

Dans cette partie, nous expérimenterons les méthodes de reconnaissance de visage évoquées dans la première partie. D'abord, nous verrons le protocole expérimental, puis la réalisation des expériences et enfin, les résultats et analyses de ces manipulations.

2.2.1 Protocole

Le but de l'expérimentation était de comparer les trois méthodes (Eigenface, Fisherface et LBPH) implémentées par la librairie OpenCV de façon à en choisir une pour notre application finale, puis d'étudier sa robustesse. Nous avons réalisé deux expériences : la première consistait à observer la vitesse de reconnaissance en fonction du nombre d'images présentes dans la base de données et de comparer le pourcentage de réussite. La seconde expérience cherchait à étudier la robustesse de la méthode, c'est-à-dire si elle marchait également pour des cas particuliers comme un visage partiellement caché ou quand l'utilisateur réalisait grimace.

2.2.2 Réalisation des expériences

Lors de la première expérience, nous avons mesuré deux vitesses d'exécution : le temps d'importation des images pour réaliser l'entraînement de l'algorithme et l'entraînement de l'algorithme (premier temps) et le temps pour réaliser la reconnaissance (second temps). La base de données était composée de trois individus. Nous avons d'abord capturé une photo d'un des individus présents dans la base. Nous avons ensuite utilisée cette photo pour tester les trois algorithmes (Eigenface, Fisherface et LBPH). Pour chaque méthode, l'opération a été répétée cent fois sur le même ordinateur (afin d'éviter les imprécisions dues au processeur qui ne travaille pas toujours de manière identique). Nous avons également varié le nombre de photos présentes dans la base de données : 1, 2, 4, 6, 8, 10, puis 12 photos par individu.

Dans la deuxième partie de l'expérience, nous avons pris une vidéo composée de cent images pendant laquelle le sujet bougeait doucement la tête dans toutes les directions. Par ailleurs, chaque sujet était enregistré dans la base de données à l'aide de douze photos prises dans différentes postures. Pour finir, chaque algorithme était appliqué aux cent images de la vidéo pour observer le pourcentage de réussite.

Pour la seconde expérience qui cherche à étudier la robustesse d'un algorithme, nous avons privilégié la méthode LBPH qui montrait de meilleurs résultats lors de notre première expérience. Nous avons demandé à plusieurs personnes de s'enregistrer dans la base de données. Ensuite, pour chacune de ces personnes, une vidéo composée de 100 images a été prise. Par ailleurs, les sujets ont été filmés sous différentes conditions. Nous avons varié la luminosité (faible, naturelle, élevée), l'orientation et l'inclinaison de la tête. Nous avons aussi accentué la déformation du visage, caché certaines parties du visage et modifié la distance par rapport à la caméra. La caméra utilisée était la même que celle utilisée lors de l'enregistrement de l'individu dans la base de données pour éviter tout changement de résolution possible.

2.2.3 Résultats et analyses

Concernant l'expérience n° 1, l'algorithme Fisherface a été le plus rapide dans l'élaboration du modèle, suivi par Eigenface puis LBPH (voir Figure 2.2). Nous pouvons noter qu'Eigenface devient de plus en plus lent à mesure que le nombre d'images dans la base de données augmente (au bout de 36 images, la méthode est aussi lente que LBPH).

Pour la reconnaissance des visages, la durée est quasiment identique pour les trois algorithmes. Elle est comprise entre 680 et 690 ms pour un processeur Intel Core i5 de deuxième génération. Nous pouvons donc conclure que le temps n'est pas un facteur important dans le choix de l'algorithme. Si nous choisissons de prendre LBPH, l'initialisation prendra plus le temps, mais il n'y aura pas d'impact sur le temps de reconnaissance.

Nous déduisons de la seconde partie de l'expérience n° 1 les résultats affichés Figure 2.3. Nous remarquons que l'algorithme LBPH est globalement meilleur que les deux autres. Fisherface arrive deuxième, suivi de loin par Eigenface.

Nous pouvons conclure de la seconde partie que la luminosité peut être un problème lorsque les yeux sont cachés par la lumière. Cela pose aussi un souci lorsque la limite entre le visage et l'arrière-plan est difficilement perceptible (ce qui arrive assez souvent avec une caméra de faible qualité). L'orientation et l'inclinaison de la tête sont limitées à un certain angle (environ 20°), car au-delà les deux yeux ne sont pas bien visibles. La déformation du visage ne perturbe pas la reconnaissance, en outre le port de lunettes de vue ne modifie que très légèrement le résultat. Pour pallier cette difficulté, la meilleure solution serait d'inclure des photos avec accessoires dans

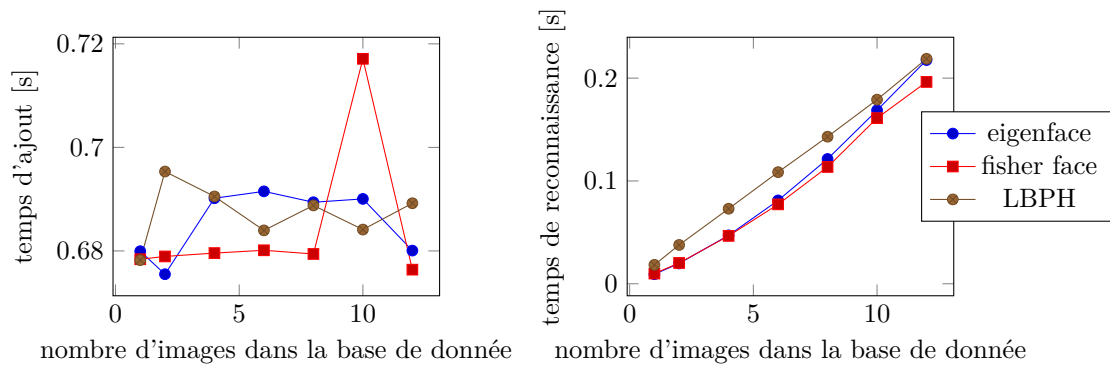


FIGURE 2.2 – Durées d'initialisation, et de reconnaissance, par algorithme

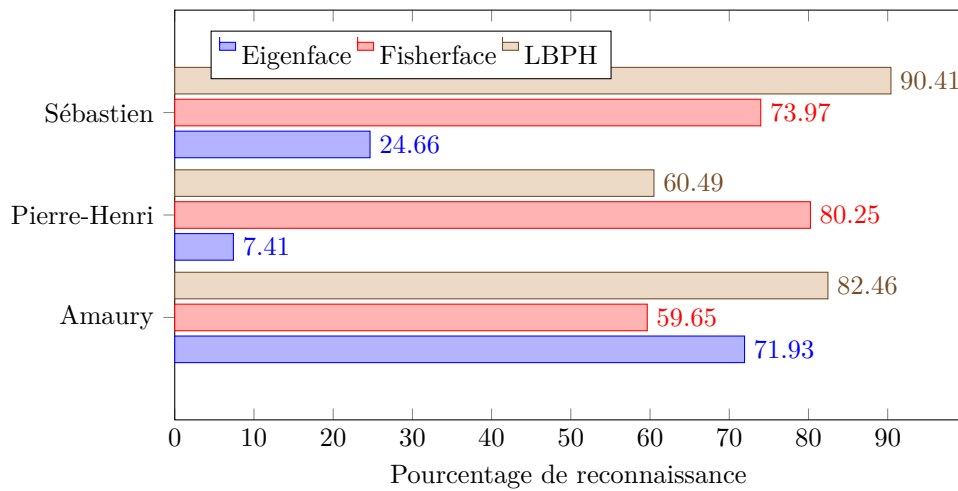


FIGURE 2.3 – Pourcentage de reconnaissance, en fonction du cobaye et de l'algorithme

la base de données. Masquer certaines parties du visage est beaucoup plus problématique. En effet, lorsque la bouche est cachée, la reconnaissance est réalisable, mais plus difficilement. En revanche, si les yeux sont cachés, la reconnaissance est impossible. Ce qui pose un problème pour les personnes aux cheveux longs. La distance visage-caméra peut être également un obstacle à la reconnaissance dans le cas où la taille du visage reconnue est plus petite que celle dans la base de données. La réalisation de nouveaux classifieurs pourraient être intéressants pour améliorer la reconnaissance.

Pour conclure, nous pouvons dire que les expérimentations ont montré que la méthode de reconnaissance faciale LBPH est la plus efficace. Cependant, nous avons également vu que de nombreux facteurs extérieurs influent sur la qualité de la reconnaissance et peuvent mettre en échec celle-ci.

Dans cette partie, nous avons comparé différentes méthodes théoriques de reconnaissance faciale. Puis, nous avons réalisé des expériences sur chacun des algorithmes pour finalement en retenir un : LBPH. Nous allons maintenant passer à l'étape suivante de notre démarche qui est

la reconnaissance des émotions. //–Trouver une meilleure transition–//

Partie 3

Reconnaissance des émotions

3.1 Théorie

3.1.1 Différentes solutions

3.1.2 Solution choisie

3.2 Expérimentations

3.2.1 Protocole

3.2.2 Réalisation

3.2.3 Résultats et analyse

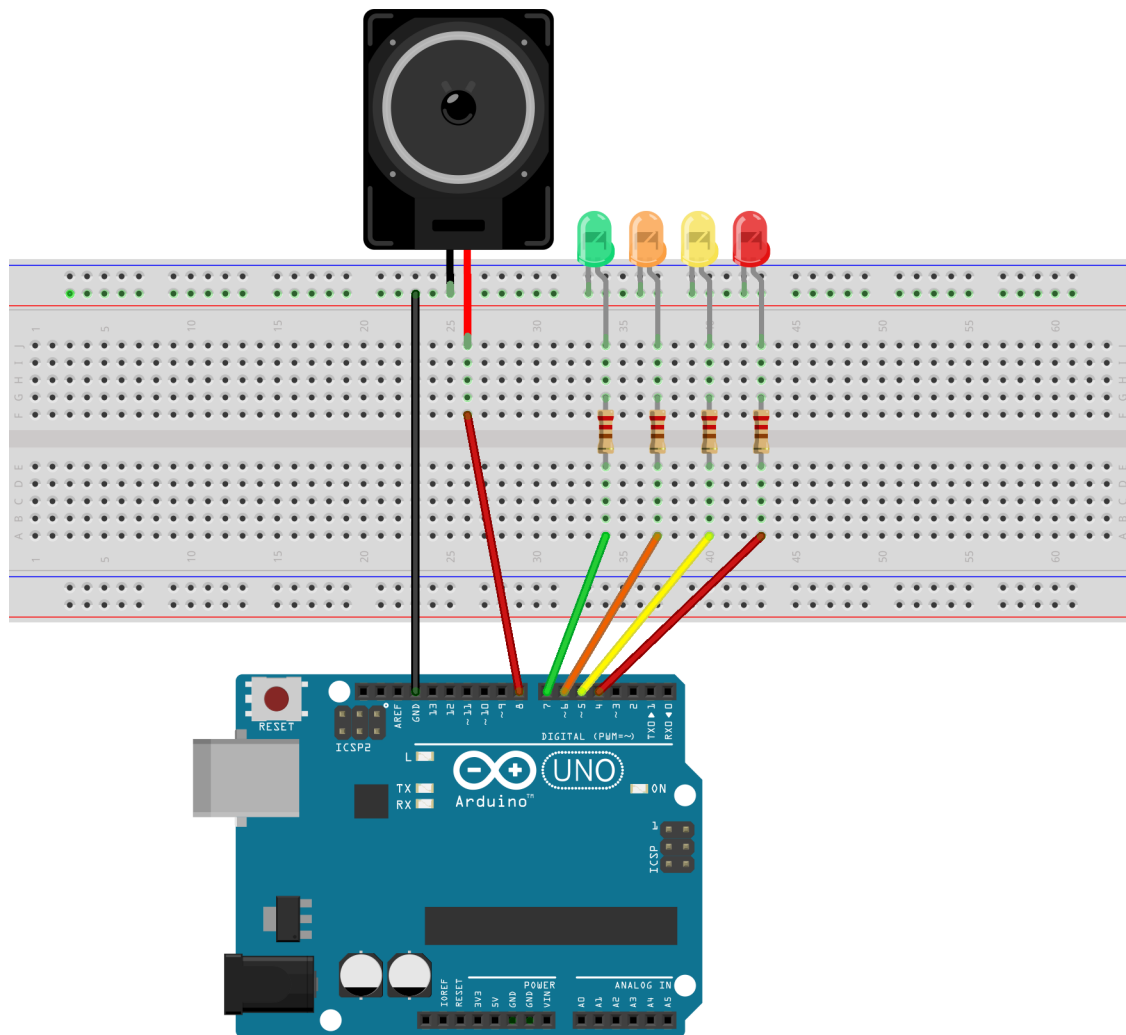
Partie 4

Production

4.1 Prototype final

4.2 Tests finaux

4.3 Discussion et analyse



fritzing

FIGURE 4.1 – Schéma du prototype de test

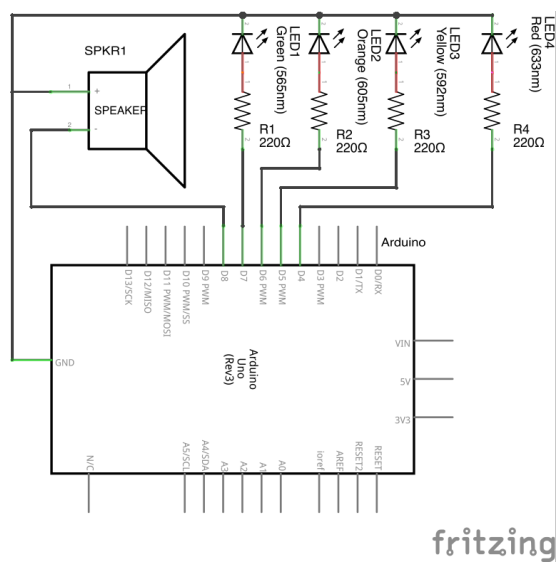


FIGURE 4.2 – Schéma électrique du prototype de test

Conclusion

Annexe A

Code des applications

A.1 Application principale

```
1  #!/usr/bin/env python2
2  # -*- coding: utf-8 -*-
3
4  #      oooooooooo ooooo oooooooooo ooooooooooooo      o      oooooooooo      oooooooooo8
5  #      88  888  88  888  888      888 888      888      888      888 888 888
6  #          888      888  888oooo88  888ooo8      8  88      888oooo88  888ooooooo
7  #          888      888  888      888      8oooo88  888      888      888
8  #          o888o      o888o o888o      o888ooo8888 o88o  o888o o888o      o88oooo888
9
10 #Imports
11 import cv2
12 import numpy
13 import time
14 import os
15 import sys
16 import math
17 import serial
18 import glob
19 from collections import defaultdict
20
21 #Constantes
22 TRAINSET = "lbpcascade_frontalface.xml" #Fichier de reconnaissance
23 IMAGE_SIZE = 170 #Normalisation des images de base
24 NUMBER_OF_CAPTURE = 10 #Nombre de captures a realiser pour la base de donnees
25 THRESHOLD = 90 #Seuil de reconnaissance
26 CAMERA = 1 #La camera
27 ARDUINO = False #Utiliser l'arduino ?
28
29 INDIVIDUS = []
30
31 #####
32 def sendSerial(ser , command):
33     """Envoie command a l'arduino"""
34     if(ARDUINO):
35         ser.write(command)
36
37 class CreateDataBase():
38     def __init__(self , imagePath , ident):
39         self.rval = False
40         self.camera = cv2.VideoCapture(CAMERA)
```

```

41         self.classifier = cv2.CascadeClassifier(TRAINSET)
42         self.faceFrame = None
43         self.identity = ident
44         self.imagesPath = imgPath
45
46     def getFacesPos(self, frame):
47         """Retourne la position des visages detectes de la forme [[x y w h]] """
48         faces = self.classifier.detectMultiScale(frame)
49         return faces
50
51     def drawDetectedFace(self, frame, faces):
52         """Dessine un rectangle autour du visage detecte"""
53         for f in faces:
54             x,y,w,h = [v for v in f]
55             cv2.rectangle(frame, (x,y), (x+w, y+h), (0,140,255))
56             self.LBPHBaselImage = self.getFaceFrame(frame, x, y, w, h)
57         return frame
58
59     def getFaceFrame(self, frame, x, y, w, h):
60         """On recupere un rectangle (largeur, hauteur) (centreX, centreY)"""
61         cropped = cv2.getRectSubPix(frame, (w, h), (x + w / 2, y + h / 2))
62         grayscale = cv2.cvtColor(cropped, cv2.COLOR_BGR2GRAY)
63         self.faceFrame = cv2.resize(grayscale, (IMAGE_SIZE, IMAGE_SIZE))
64         return self.faceFrame
65
66     def collectFace(self, frame):
67         """On enregistre le visage recupere"""
68         imageCreated = False
69         captureNum = 0
70         #Cree le dossier s'il n'existe pas
71         try:
72             os.makedirs("{0}/{1}".format(self.imagesPath, self.identity))
73         except OSError:
74             print("ecriture dans dossier existant")
75         #Cree l'image a la suite
76         while not imageCreated:
77             if not os.path.isfile("{0}/{1}/{2}.jpg".format(self.imagesPath, self.identity, captureNum)):
78                 cv2.imwrite("{0}/{1}/{2}.jpg".format(self.imagesPath, self.identity, captureNum), frame)
79                 imageCreated = True
80             else:
81                 captureNum += 1
82
83     def capture(self):
84         """Recupere le flux video"""
85         if self.camera.isOpened():
86             (rval, frame) = self.camera.read()
87         else:
88             rval = False
89
90         while rval:
91             (rval, frame) = self.camera.read()
92             frame = self.drawDetectedFace(frame, self.getFacesPos(frame))
93             #Affichage du texte
94             cv2.putText(frame, "Appuyez sur c pour collecter", (0,20), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0,0,255))
95             cv2.imshow("Creation de la BDD", frame)
96             key = cv2.waitKey(20)
97             if key in [27, ord('Q'), ord('q')]: #esc / Q
98                 break
99             if key in [ord('C'), ord('c')] and self.faceFrame != None:
100                 self.collectFace(self.faceFrame)
101
102     class Recognize():

```

```

103     def __init__(self, imgPath):
104         self.rval = False
105         self.camera = cv2.VideoCapture(CAMERA)
106         self.classifier = cv2.CascadeClassifier(TRAINSET)
107         self.faceFrame = None
108         self.identities = []
109         self.imagesPath = imgPath
110         self.images = []
111         self.imagesIndex = []
112         self.time = time.time()
113
114         self.eyeWide = 0
115         self.eyeHeight = 0
116         self.grayMouthClosed = 0
117         self.thresholdEyeClosed = 0
118
119     def getFacesPos(self, frame):
120         """Retourne la position des visages detectes de la forme [[x y w h]] """
121         faces = self.classifier.detectMultiScale(frame)
122         return faces
123
124     def drawDetected(self, frame, detected, color):
125         """Dessine un rectangle autour du visage detecte"""
126         if detected is None:
127             return frame
128         for d in detected:
129             x,y,w,h = [v for v in d]
130             cv2.rectangle(frame, (x,y), (x+w, y+h), color)
131         return frame
132
133     def getFaceFrame(self, frame, x, y, w, h):
134         """On recupere un rectangle (largeur, hauteur) (centreX, centreY)"""
135         cropped = cv2.getRectSubPix(frame, (w, h), (x + w / 2, y + h / 2))
136         grayscale = cv2.cvtColor(cropped, cv2.COLOR_BGR2GRAY)
137         self.faceFrame = cv2.resize(grayscale, (IMAGE_SIZE, IMAGE_SIZE))
138         return self.faceFrame
139
140     def getCroppedEyesPos(self, croppedFrame):
141         """Retourne la position des bouches detectes de la forme [[x y w h]] """
142         cascade = cv2.CascadeClassifier('haarcascade_lefteye_2splits.xml')
143         rects = cascade.detectMultiScale(croppedFrame)
144         if len(rects) == 0:
145             return rects
146         final = None
147         x1 = 0
148         x2 = 0 + len(croppedFrame)
149         y1 = 0
150         y2 = 0 + len(croppedFrame[0])*1/2
151
152         #Prend la partie inferieure de la tete pour le traitement
153         for rect in rects:
154             if rect[0] > x1 and rect[0] + rect[2] < x2 and rect[1] > y1 and rect[1] + rect[3] < y2:
155                 if final is None:
156                     final = [rect]
157                 else:
158                     final += [rect]
159         return final
160
161     def getCroppedMouthPos(self, croppedFrame):
162         """Retourne la position des bouches detectes de la forme [[x y w h]] """
163         cascade = cv2.CascadeClassifier('mouth_classifier.xml')
164         rects = cascade.detectMultiScale(croppedFrame)

```

```

165         if len(rects) == 0:
166             return rects
167         final = None
168         x1 = 0
169         x2 = 0 + len(croppedFrame)
170         y1 = 0 + len(croppedFrame[0])*5/8
171         y2 = 0 + len(croppedFrame[0])
172
173         #Prend la partie inferieure de la tete pour le traitement
174         for rect in rects:
175             if rect[0] > x1 and rect[0] + rect[2] < x2 and rect[1] > y1 and rect[1] + rect[3] < y2:
176                 if final is None:
177                     final = [rect]
178                 else:
179                     final += [rect]
180         return final
181
182     def extractAndResize(self, frame, x, y, w, h):
183         """On recupere juste la tete en noir et blanc"""
184         cropped = cv2.getRectSubPix(frame, (w, h), (x + w / 2, y + h / 2))
185         grayscale = cv2.cvtColor(cropped, cv2.COLOR_BGR2GRAY)
186         resized = cv2.resize(grayscale, (IMAGE_SIZE, IMAGE_SIZE))
187         return resized
188
189     def cropFromFace(self, frame, facePos):
190         """Garde seulement la partie "tete" de la frame"""
191         #X,Y,W,H
192         if facePos is None:
193             return frame
194         if len(facePos) == 0 :
195             return frame
196         else :
197             x1 = facePos[0][0]
198             x2 = x1 + facePos[0][2]
199             y1 = facePos[0][1]
200             y2 = y1 + facePos[0][3]
201             return frame[y1:y2, x1:x2]
202
203     def readImages(self):
204         """Recupere les images de bases pour effectuer la reconnaissance des visages"""
205         c = 0
206         self.images = []
207         self.imagesIndex = []
208         for dirname, dirnames, filenames in os.walk(self.imagesPath):
209             for subdirname in dirnames:
210                 self.identities.append(subdirname)
211                 subject_path = os.path.join(dirname, subdirname)
212                 for filename in os.listdir(subject_path):
213                     try:
214                         im = cv2.imread(os.path.join(subject_path, filename), 0)
215                         self.images.append(numpy.asarray(im, dtype=numpy.uint8))
216                         self.imagesIndex.append(c)
217                     except IOError, (errno, strerror):
218                         print "I/O error ({0}): {1}".format(errno, strerror)
219                     except:
220                         print "Unexpected error:", sys.exc_info()[0]
221                         raise
222                     c += 1
223
224     def recognizeLBPHFace(self):
225         """Reconnait par la methode LBPH"""
226         self.model = cv2.createLBPHFaceRecognizer()

```



```

227         self.model.train(numpy.asarray(self.images), numpy.asarray(self.imagesIndex))
228
229     def recognize(self):
230         """On choisit la methode de reconnaissance et on construit la base de donnee"""
231         self.readImages()
232         self.recognizeLBPHFace()
233         if not self.camera.isOpened():
234             return
235         self.capture()
236
237     def identify(self, image):
238         """On reconnait l'identite de la personne si enregistree"""
239         [p_index, p_confidence] = self.model.predict(image)
240         found_identity = self.identities[p_index]
241         return found_identity, p_confidence
242
243     def initNeutral(self, neutralImg):
244         """Initialise les thresholds + les largeurs/hauteurs pour la detection des emotions"""
245         frame = neutralImg
246         facePos = self.getFacesPos(frame)
247         cropped = self.cropFromFace(frame, facePos)
248         mouthPos = self.getCroppedMouthPos(cropped)
249         mouthFrame = self.cropFromFace(frame, mouthPos)
250         gray = cv2.cvtColor(mouthFrame, cv2.COLOR_BGR2GRAY)
251         ret, thresh = cv2.threshold(gray, 50, 255, cv2.THRESH_BINARY)
252         self.grayMouthClosed = numpy.count_nonzero(thresh)
253         eyePos = self.getCroppedEyesPos(cropped)
254         eyeFrame = self.cropFromFace(frame, eyePos)
255         hsv = cv2.cvtColor(eyeFrame, cv2.COLOR_BGR2HSV)
256         lowerColor = numpy.array([80, 0, 0])
257         upperColor = numpy.array([160, 100, 100])
258         mask = cv2.inRange(hsv, lowerColor, upperColor)
259         self.thresholdEyeClosed = numpy.count_nonzero(mask)
260         self.eyeWide = eyePos[0][2]
261         self.eyeHeight = eyePos[0][3]
262         return 0
263
264
265     def isMouthOpen(self, mouthFrame):
266         gray = cv2.cvtColor(mouthFrame, cv2.COLOR_BGR2GRAY)
267         ret, thresh = cv2.threshold(gray, 50, 255, cv2.THRESH_BINARY)
268         return self.grayMouthClosed < numpy.count_nonzero(thresh) - 300
269
270
271     def EyeNotHeightThanNeutral(self, EyeHeight, threshold):
272         """Determine si les yeux sont fronces"""
273         return EyeHeight < self.eyeHeight - threshold
274
275     def EyeHeightThanNeutral(self, EyeHeight, threshold):
276         """Determine si les yeux sont equarquilles"""
277         return EyeHeight > self.eyeHeight + threshold
278
279     def isEyeClosed(self, eyeFrame):
280         hsv = cv2.cvtColor(eyeFrame, cv2.COLOR_BGR2HSV)
281         lowerColor = numpy.array([80, 0, 0])
282         upperColor = numpy.array([160, 100, 100])
283         mask = cv2.inRange(hsv, lowerColor, upperColor)
284         return self.thresholdEyeClosed * 2 < numpy.count_nonzero(mask)
285
286
287     def emotions(self):
288         """Recupere le flux video"""

```

```

289     interval = 0
290     dontlook = 0
291     sleep = 0
292     mouthOpen = 0
293     eyeClose = 0
294     eyeBigger = 0
295     eyeNotBigger = 0
296     error = 0
297     if self.camera.isOpened():
298         (rval, frame) = self.camera.read()
299     else:
300         rval = False
301     i = 0
302     while rval:
303         (rval, frame) = self.camera.read()
304         facePos = self.getFacesPos(frame)
305         if len(facePos) is 0 or facePos is None:
306             dontlook += 1
307             if dontlook % 20 is 0:
308                 print('Conducteur inattentif')
309                 sendSerial(ser, 's')
310         else:
311             dontlook = 0
312             if i < 10:
313                 i += 1
314             if i is 10:
315                 self.initNeutral(frame)
316                 i += 1
317             frame = self.drawDetected(frame, facePos, (0,140,255))
318             cropped = self.cropFromFace(frame, facePos)
319             eyePos = self.getCroppedEyesPos(cropped)
320             sendSerial(ser, 'd')
321             if eyePos is None:
322                 #print('yeux fermes ou yeux non detectes')
323                 sleep += 1
324                 error = 1
325             elif error is not 1:
326                 sleep = 0
327                 sendSerial(ser, 'd')
328                 sendSerial(ser, 'r')
329             else:
330                 error = 0
331             if sleep > 3:
332                 print('Endormi')
333                 sendSerial(ser, 'w')
334                 sendSerial(ser, 's')
335             if eyePos is not None and i > 10:
336                 cropped = self.drawDetected(cropped, eyePos, (255,0,255))
337                 eyeFrame = self.cropFromFace(frame, eyePos)
338                 if self.isEyeClosed(eyeFrame):
339                     #print('yeux fermes ou yeux non detectes')
340                     sleep += 1
341                     error = 1
342                 elif error is not 1:
343                     sleep = 0
344                 else:
345                     error = 0
346                 if len(eyePos) > 0 and self.EyeHeightThanNeutral(eyePos[0][3], 5):
347                     #print('plus grand')
348                     eyeBigger += 1
349                     error = 1
350                 elif error is not 1:

```

```

351         eyeBigger = 0
352     else:
353         error = 0
354     if len(eyePos) > 0 and self.EyeNotHeightThanNeutral(eyePos[0][3], 4):
355         #print('plus petit')
356         eyeNotBigger += 1
357         error = 1
358     elif error is not 1:
359         eyeNotBigger = 0
360     else:
361         error = 0
362     mouthPos = self.getCroppedMouthPos(cropped)
363     cropped = self.drawDetected(cropped, mouthPos, (0,0,255))
364     if mouthPos is not None and self.isMouthOpen(self.cropFromFace(frame, mouthPos)) and
365         #print('bouche ouverte')
366         mouthOpen += 1
367         error = 1
368     elif not self.isMouthOpen(self.cropFromFace(frame, mouthPos)) and error is not 1:
369         moutOpen = 0
370     else:
371         error = 0
372
373     if mouthOpen > 5 and eyeBigger > 5:
374         print('Surpris')
375         sendSerial(ser, 'b')
376     else:
377         sendSerial(ser, 'n')
378     if eyeNotBigger > 5:
379         print('Enerve')
380         sendSerial(ser, 'a')
381     else:
382         sendSerial(ser, 'q')
383     #if mouthOpen <= 5 and eyeNotBigger > 5:
384     #    print('enerve')
385     #    sendSerial(ser, 'a')
386     cv2.imshow("Tete", cropped)
387     cv2.imshow("TIPEAPS", frame)
388     key = cv2.waitKey(20)
389     if key in [27, ord('Q'), ord('q')]:
390         break
391
392 def capture(self):
393     """Recupere le flux video"""
394     interval = 0
395     if self.camera.isOpened():
396         (rval, frame) = self.camera.read()
397     else:
398         rval = False
399     i = 0
400     while i < 55:
401         i+=1
402         (rval, frame) = self.camera.read()
403         self.time = time.time()
404         facePos = self.getFacesPos(frame)
405         frame = self.drawDetected(frame, facePos, (0,140,255))
406         for f in facePos:
407             x,y,w,h = [v for v in f]
408             resized = self.extractAndResize(frame, x, y, w, h)
409             identity, confidence = self.identify(resized)
410             if confidence > THRESHOLD:
411                 identity = "INCONNU"
412             INDIVIDUS.append(identity)

```

```

413         print(identity + "└───┘" + str(confidence))
414         cv2.putText(frame, "%s_("%s)"%(identity, confidence), (x,y), cv2.FONT_HERSHEY_PLAIN,
415         cv2.imshow("TIPEAPS", frame)
416         key = cv2.waitKey(20)
417         if key in [27, ord('Q'), ord('q')]: #esc / Q
418             break
419
420 def getSerialName():
421     """Retourne le fichier correspondant a l'arduino"""
422     serialName = '/dev/null'
423     osname = sys.platform.lower()
424     if 'darwin' in osname: #si mac OS X
425         for tty in glob.glob('/dev/tty*'):
426             if 'usbmodem' in tty:
427                 serialName = tty
428     elif 'linux' in osname: #si linux
429         for tty in glob.glob('/dev/tty*'):
430             if 'ACM' in tty:
431                 serialName = tty
432     return serialName
433
434 if __name__ == "__main__":
435     mode = 0
436     for i in range(1, len(sys.argv)):
437         if sys.argv[i] == '-n' and i < len(sys.argv):
438             individu = sys.argv[i + 1]
439         if sys.argv[i] == '-a':
440             ARDUINO = True
441         ser = serial.Serial(getSerialName(), 9600)
442         if sys.argv[i] == '-c':
443             mode = 1
444     if mode is 0:
445         recognize = Recognize("images")
446         recognize.recognize()
447         d = defaultdict(int)
448         for i in INDIVIDUS:
449             d[i] += 1
450         result = max(d.iteritems(), key=lambda x: x[1])
451         individu = result[0]
452         if result[1] > 25:
453             print('Bienvenue_┘' + individu)
454             if ARDUINO:
455                 sendSerial(ser, 'e')
456             recognize.emotions()
457     if mode is 1:
458         createDB = CreateDataBase("images", individu)
459         createDB.capture()

```

A.2 Code Arduino pour les tests

```

1  /**
2  * 0000000000 00000 0000000000 0000000000 0 000000000 000000008
3  * 88 888 88 888 888 888 888 888 888 888 888
4  * 888 888 888 888000088 8880008 8 88 888000088 888000000
5  * 888 888 888 888 888 8000088 888 888
6  * 08880 08880 08880 08880008888 0880 08880 08880 0880000888
7  */
8
9 int pinEngine = 7;
10 int pinWarning = 6;
11 int pinAccelerationLimit = 5;

```

```

12  int pinBrake = 4;
13  int pinSound = 8;
14
15  boolean engine = false;
16
17  char command = 0;
18
19  void Exit(){
20      engine = false;
21      digitalWrite(pinEngine,LOW);
22      digitalWrite(pinWarning,LOW);
23      digitalWrite(pinAccelerationLimit,LOW);
24      digitalWrite(pinBrake,LOW);
25
26      noTone(pinSound);
27  }
28
29  void Engine(){
30      engine = true;
31      digitalWrite(pinEngine, HIGH);
32  }
33
34  void Warning(){
35      digitalWrite(pinWarning, HIGH);
36  }
37
38  void StopWarning(){
39      digitalWrite(pinWarning, LOW);
40  }
41
42  void AccelerationLimit(){
43      digitalWrite(pinAccelerationLimit, HIGH);
44  }
45
46  void StopAccelerationLimit(){
47      digitalWrite(pinAccelerationLimit, LOW);
48  }
49
50  void Brake(){
51      digitalWrite(pinBrake, HIGH);
52  }
53
54  void StopBrake(){
55      digitalWrite(pinBrake, LOW);
56  }
57
58  void Sound(){
59      tone(pinSound, 666);
60  }
61
62  void StopSound(){
63      noTone(pinSound);
64  }
65
66  void setup()
67  {
68      pinMode(pinEngine, OUTPUT);
69      pinMode(pinWarning,OUTPUT);
70      pinMode(pinAccelerationLimit,OUTPUT);
71      pinMode(pinBrake,OUTPUT);
72
73      Serial.begin(9600); //On démarre la connexion serie

```

```

74  while(!Serial){} // on attend que la connexion serie démarre
75
76  /**
77   *  verification du fonctionnement des systemes
78   *  (1/2 seconde)
79   */
80  digitalWrite(pinEngine, HIGH);
81  Warning();
82  AccelerationLimit();
83  Brake();
84  Sound();
85
86  delay(500);
87
88  digitalWrite(pinEngine, LOW);
89  StopWarning();
90  StopAccelerationLimit();
91  StopBrake();
92  StopSound();
93  }
94
95  void loop()
96  {
97      if(!engine)//si non démarre
98      {
99          command = Serial.read();
100         if(command == 'e')
101             Engine();
102         command = 0;
103     }
104     else
105     {
106         if(Serial.available() > 0) //si on recoit une donnée sur le port serie
107         {
108             command = Serial.read();
109             switch(command){
110                 case 'x':
111                     Exit();
112                     break;
113                 case 'w':
114                     Warning();
115                     break;
116                 case 'r':
117                     StopWarning();
118                     break;
119                 case 'b':
120                     Brake();
121                     break;
122                 case 'n':
123                     StopBrake();
124                     break;
125                 case 'a':
126                     AccelerationLimit();
127                     break;
128                 case 'q':
129                     StopAccelerationLimit();
130                     break;
131                 case 's':
132                     Sound();
133                     break;
134                 case 'd':
135                     StopSound();

```

```
136         break;
137     }
138     command = 0;
139 }
140 }
141 }
```