

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

Remerciements

Table des matières

1	Les limites de la sécurité routière	2
1.1	Les moyens mis en place pour la sécurité du conducteur en France	2
1.1.1	Historique de la sécurité routière depuis 40 ans	2
1.1.2	Bilan de quarante années de mesures	3
1.2	Les comportements à risques	5
1.2.1	Les principales causes d'accidents	5
1.2.2	La conduite à l'aveugle	6
2	La reconnaissance faciale	8
2.1	Théorie	8
2.1.1	Généralités	8
2.1.2	La méthode Eigenface	8
2.1.3	La méthode Fisherface	9
2.1.4	La méthode LBPH	10
2.2	Expérimentations	11
2.2.1	Protocole	11
2.2.2	Réalisation des expériences	11
2.2.3	Résultats et analyses	12
3	Reconnaissance des émotions	15
3.1	Théorie	15
3.1.1	Différentes solutions	15
3.1.2	Solution choisie	16
3.2	Expérimentations	18
3.2.1	Protocole	18
3.2.2	Réalisation	18
3.2.3	Résultats et analyse	18
4	Production	19
4.1	Prototype final	19
4.2	Protocole du test final	19
4.3	Discussion et analyse des résultats	21
A	Code des applications	26
A.1	Application principale	26
A.2	Code Arduino pour les tests	33

B Documents utilisés en CST	37
B.1 check-list	37
B.2 Matrice RACI	38
B.3 Diagramme de GANTT	39

Introduction

. caser [12] qui nous a donne l'idee . caser [21] en projet similaire

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 entre 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 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'étendue 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-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 d'urgences.

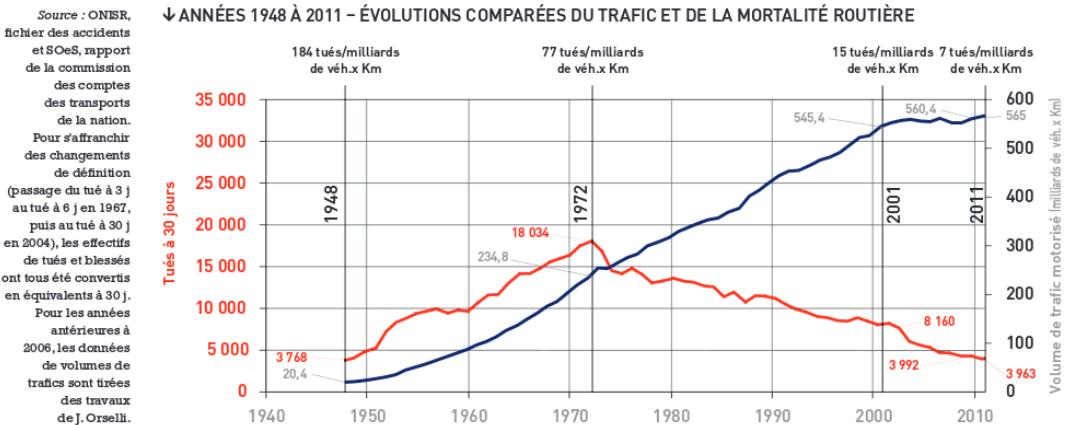


FIGURE 1.1 – Évolution comparée du traffic et de la mortalité routière entre 1948 et 2011

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 choquantes 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 influer 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 un 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. À 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. À 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 particulier 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 trois différents algorithmes couramment utilisé dans le domaine de la reconnaissance faciale : 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 en place d'un réseau neuronal [22] ou l'implémentation de modèles cachés de Markov [2], [14], [16]). Cependant, en trois mois, nous nous n'aurions pas eu le temps de mettre en place une telle méthode. De plus la mise en place d'un réseau neuronal nécessite beaucoup de calculs. En effet, un réseau neuronal est composé d'une suite de couches de neurones. On donne la matrice de l'image en sortie et le réseau nous donne en sortie le nom de la personne identifiée. La couche de sortie est donc composée d'un neurone par personne enregistrée, mais la couche d'entrée nécessite autant de neurones que de pixels dans l'image. Pour une image 100×100 il faut donc 10000 neurones pour la couche d'entrée.

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. Elle a été introduite en 1991 par Turk et Pentland (cf [17])

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 (ces composantes peuvent être diverses comme le montre la figure 2.1). À 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 10 000 vecteurs à traiter. Le nombre de données à analyser est donc très important. De plus, cette méthode est sensible à la luminosité et à la déformation des visages (grimaces, accessoires, ...).

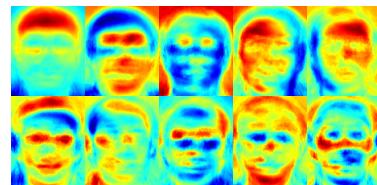


FIGURE 2.1 – Ensemble d'eigenfaces

2.1.3 La méthode Fisherface

La méthode de reconnaissance faciale Fisherface se base sur les travaux de Sir R.A. Fisher. Il s'agit d'une évolution de l'algorithme Eigenface présenté précédemment. Alors que l'algorithme Eigenface se base sur l'Analyse des Composantes Principales (ACP), l'algorithme Fisherface utilise l'Analyse Discriminante Linéaire (ADL) qui cherche le meilleur sous espace minimisant la distance entre les images d'une même classe (ici une classe = un visage) en maximisant la distance inter-classe.

Tout d'abord, l'algorithme calcul l'image moyenne de la classe :

$$\frac{1}{n} \times \sum_{i=1}^n x_i$$

avec

$$x_0, x_1, \dots, x_n$$

les images de la classe.

Puis, on calcule les matrice de dispersion intra et inter classe (disons respectivement M_i et M_e). Puis on trouve la matrice de projection (P) en résolvant

$$M_i \times x = \lambda \times M_e \times x$$

Pour trouver la meilleure projection, il suffit de résoudre

$$\frac{P^{-1} \times M_e \times P}{P^{-1} \times M_i \times P} = \text{argmax}(P)$$

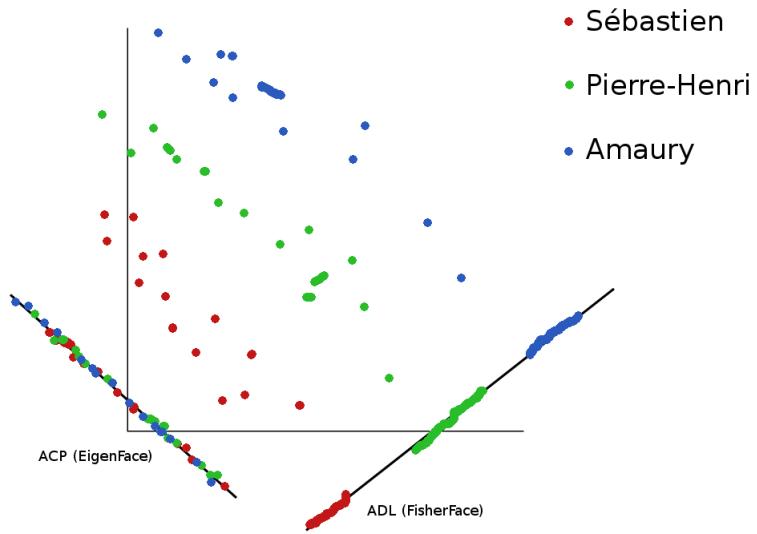


FIGURE 2.2 – Distribution Fisherface

Vous pouvez retrouver plus de détails dans la documentation d'OpenCV ([10] et [9])

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 divisé en groupe de pixels. Chaque groupe de pixels correspond à une matrice carré 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 inférieur 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 : [5] où il est combiné avec Adaboost (plugin d'*OpenCV*) pour obtenir de meilleurs résultats et un réseau neuronnal. Mais aussi dans [8] où les auteurs essayent d'améliorer l'algorithme dans le but d'en créer un nouveau : NBNN (*Naive Bayes Nearest Neighbor*). 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 différents 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 filmées 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 2.3). 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.

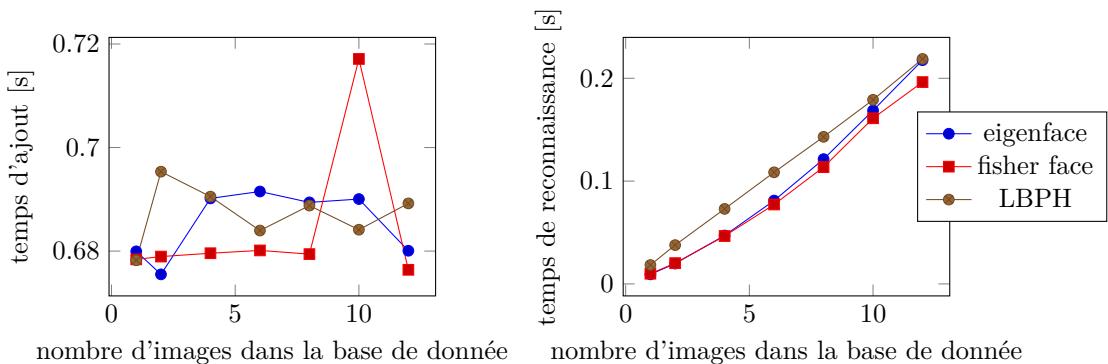


FIGURE 2.3 – Durées d'initialisation, et de reconnaissance, par algorithme

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 de 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 2.4. 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 (voir Figure 2.5). 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 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

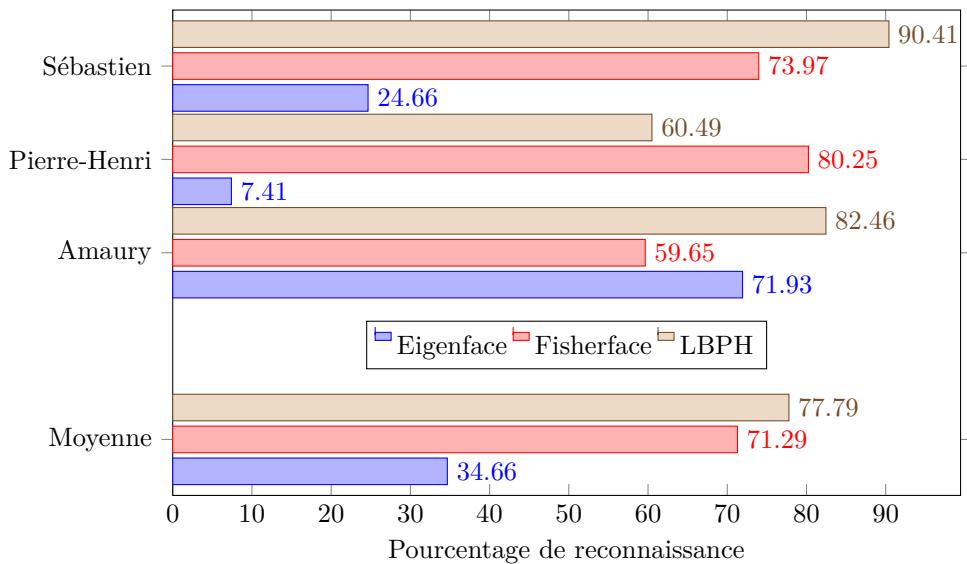


FIGURE 2.4 – Pourcentage de reconnaissance, en fonction du cobaye et de l’algorithme

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 classifiers 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 la partie principale de notre application qui est la reconnaissance des émotions.

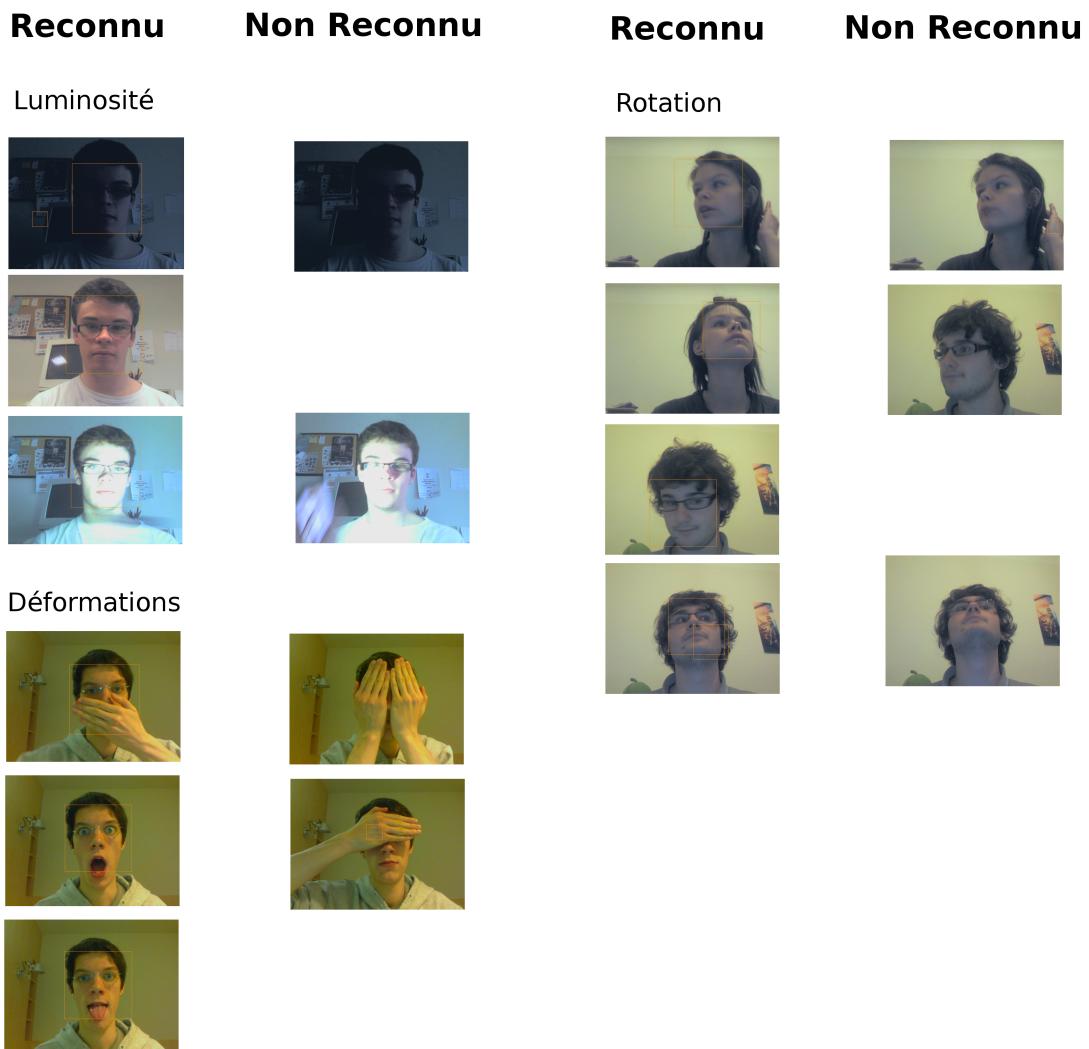


FIGURE 2.5 – Tests de robustesse pour la reconnaissance des visages.

Partie 3

Reconnaissance des émotions

Dans cette dernière partie, nous allons aborder la reconnaissance des émotions. Comme pour la reconnaissance faciale, nous étudierons d'abord l'aspect théorique, puis nous passerons aux expérimentations.

3.1 Théorie

Avant de passer à l'expérimentation, nous avons besoin de connaître la méthode la plus adaptée à notre application. C'est pourquoi, dans un premier temps nous allons voir les différentes solutions possibles, puis nous allons choisir celle qui correspond le mieux à nos besoins.

3.1.1 Différentes solutions

Avant toute chose, nous étudions les expressions faciales et non directement les émotions. Les expressions faciales sont brèves : entre 250ms et 5s [4]. Il existe différentes façons de récupérer des expressions faciales : l'approche holistique (c'est-à-dire que la tête est analysée de façon globale) ou l'approche locale qui consiste à prendre des parties de la tête et analyser ces parties indépendamment les unes des autres.

Ensuite, pour « trouver » les émotions, il existe également différentes manières de procéder. Nous avons l'approche basée sur un modèle : c'est la technique utilisée précédemment pour effectuer la reconnaissance des visages, mais à la place de séparer la base de données par individus, on sépare les images par individus et par émotions. On peut imaginer par exemple une arborescence avec un dossier nommé Xavier contenant les sous-dossiers `Xavier_Content`, `Xavier_Peur`, `Xavier_Degout`, ... Le problème de cette méthode est qu'il faut que l'individu soit connu dans la base de données pour reconnaître les émotions (ce qui explique pourquoi nous n'avons pas choisi cette méthode). Nous avons aussi l'approche basée sur l'image : à partir d'une image, nous essayons de trouver des expressions faciales qui s'apparentent aux émotions primaires ([3]) définies par Ekman et Friesen. Tous les humains ont la même façon d'exprimer ces émotions primaires, peu importe son origine ou sa culture [13]. Cette méthode présente l'avantage d'être plus simple et plus rapide à mettre en place. Cependant, elle est moins robuste, notamment aux changements de position de la tête.

Enfin, pour visualiser les changements //à préciser//, nous pouvons regarder les déformations. Pour commencer, cela consiste à classer les émotions primaires en termes de mouvements,



FIGURE 3.1 – Les 6 émotions primaires

translations, rotations, etc. Puis, nous prenons des images à différents instants et nous analysons les mouvements, translations, rotations, etc... afin de les comparer à ceux des émotions primaires. Une autre solution est le suivi des points du visage : sur chaque image, nous relevons la position de certains points et nous essayons de les aligner sur le modèle de chaque expression primaire. Finalement, l'émotion retenue est celle dont le modèle de points est le plus proche de celui de l'image. C'est la méthode la plus robuste, mais aussi la plus couteuse en calcul et plus difficile à mettre en place que la précédente méthode.

3.1.2 Solution choisie

Avant de présenter la solution choisie, nous avons testé plusieurs autres méthodes que nous avons abandonné. Par exemple, pour détecter la forme de la bouche, nous avons utilisé une fonction présente dans *OpenCV* : `findContours` //—INSERER CODE FIND CONTOURS + LIEN—// qui permet d'obtenir des résultats plutôt bons 3.2.

Cependant, dans certains cas, le résultat n'était pas du tout exploitable (cf 3.3).

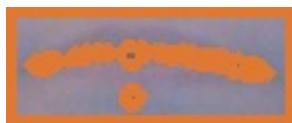


FIGURE 3.2 – Utilisation de la méthode `findContours` sur une bouche

Par ailleurs, la détection des couleurs //—INSERER CODE COULEUR + LIEN—// a donné de bien meilleurs résultats (3.4).

Utiliser `findContours` aurait été plus précis, mais son manque de fiabilité et la nécessité de traiter les images rend son utilisation contraignante. Une autre solution consistait à détecter les expressions à l'aide des couleurs et de l'utilisation de seuils spécifiques. C'était la solution la plus rapide et la plus simple à mettre en place, mais elle est imprécise et difficile à utiliser pour savoir si les



FIGURE 3.3 – Utilisation de la méthode findContours sur un oeil



FIGURE 3.4 – Détection d'une plage de couleur sur une bouche

yeux sont froncés ou écarquillés par exemple.

La solution choisie est une méthode légèrement différente de la méthode de détection de couleur. Le suivi des points (cf [11], [20], [16] et [19]) ou les réseaux neuronaux ont pour désavantage de demander énormément de calculs. Comme notre application a but d'avoir pour support un ordinateur de bord d'une voiture vraisemblablement moins puissant qu'un ordinateur portable, il est nécessaire de trouver une solution portable. Pour simplifier les calculs, nous avons défini la méthode suivante pour effectuer la reconnaissance des émotions : la première étape est l'obtention d'un visage. L'identité de la personne nous importe peu (elle est uniquement utilisée pour le démarrage du véhicule). Ensuite, à l'aide d'un second classifieur, nous obtenons toutes les positions susceptibles de correspondre à des yeux. Les yeux situés dans la partie basse du visage sont éliminés. La hauteur du classifieur nous indique si les yeux sont en position normale, froncés ou écarquillés. Pour détecter les yeux fermés, il suffit de regarder le nombre de pixels ayant une couleur comprise entre [80,0,0] et [160,100,100] avec la première composante correspondant au rouge, la seconde au vert et la troisième au bleu (on effectue donc une détection plutôt à dominance rouge pour ce point). Concernant la bouche, la méthode est presque identique à celle des yeux. Avec un classifier spécifique, toutes les bouches possibles sont récupérées, puis les erreurs sont écartées (bouches situées dans la partie haute du visage ou trop près du bord). Ensuite, la bouche est convertie en niveaux de gris et un seuil est appliquée pour n'obtenir que l'intérieur de la bouche. En effet, on peut remarquer que la couleur à l'intérieur d'une bouche ouverte tend vers le noir, alors que les lèvres sont plus d'une couleur rouge. Ainsi, on a plus qu'à comparer le taux de noir de l'image au taux obtenu à l'initialisation du programme. Si le taux de noir de l'image est supérieur à celui obtenu à l'initialisation plus un seuil, la bouche est ouverte.

Après cet aperçu des différentes solutions possibles pour la reconnaissance faciale et du choix de notre méthode, nous allons à présent l'expérimenter.

3.2 Expérimentations

Dans cette partie, nous allons présenter les expérimentations, nous allons d'abord exposer le protocole expérimental, puis la réalisation en elle-même, et enfin les conclusions que nous en avons tiré.

3.2.1 Protocole

Dans le but de régler les seuils de façon précise, il a fallu définir une distance entre le sujet et l'objectif de la caméra (cette distance a été fixée à 50 cm), et effectuer une initialisation avec une émotion neutre. Puis, pour vérifier la robustesse de la détection, le sujet devait froncer les sourcils, écarquiller et fermer les yeux et ouvrir la bouche. Ces déformations particulières du visage correspondent aux expressions faciales (la colère, la surprise et si le conducteur était endormi) que nous souhaitons récupérer pour notre application. Le processus est répété jusqu'à la détection de l'expression faciale.

3.2.2 Réalisation

//-A COMPLETER-//
/-SEB : Tu veux mettre quoi à part qu'on a réalisé le protocole-//

3.2.3 Résultats et analyse

Nos expérimentations nous ont montré que dans un lieu où l'utilisateur de l'application ne bouge pas et où la luminosité ne change pas, on arrive à obtenir une détection correcte des émotions. On pouvait tout de même observer des problèmes lorsque l'utilisateur s'éloignait ou reculait de la caméra. En effet les seuils et les tailles devenaient incorrects. Il fallait donc réinitialiser les seuils ou se remettre à la distance que l'utilisateur avait à la précédente initialisation.

Partie 4

Production

Dans cette dernière partie, nous présenterons la réalisation de notre application. Nous parlerons d'abord du prototype final, puis des tests réalisés en situation réelle et enfin, des résultats de ces tests.

4.1 Prototype final

Afin de vérifier l'efficacité de notre application (voir code source en annexe), nous avons décidé de la tester en situation réelle, c'est-à-dire dans un véhicule en mouvement. Cependant, nous ne pouvions pas relier directement notre programme à un véhicule et modifier en temps réel son comportement (le freinage par exemple). C'est pourquoi, à l'aide d'un jeu de DELs et d'un montage électronique réalisé à l'aide d'une carte Arduino Uno (voir Figure 4.1), nous avons simulé les différentes réactions du véhicule en fonction du comportement du conducteur. Ainsi, nous avons choisi d'utiliser quatre DELs :

- Une DEL rouge pour simuler le frein
- Une DEL orange clignotante pour simuler les warnings
- Une DEL jaune pour simuler une limitation de l'accélération
- Une DEL verte pour simuler le droit de démarrer

De plus, une enceinte était reliée à l'Arduino pour émettre un signal sonore en cas d'inattention de la part du conducteur ou de son endormissement. Le prototype est fonctionnel sous Linux, Windows, et Mac OS (il est à noter qu'il y a quelques problèmes de dépendances liés à *OpenCV*). Nous voulions porter l'application sur une carte électronique de type Raspberry Pi ou Beaglebone, afin de se rapprocher au plus d'un ordinateur de bord, cependant une des fonctions d'*OpenCV* que nous utilisons n'est pas disponible (*createLBPHFaceRecognizer*, qui permet l'utilisation de la méthode de reconnaissance faciale LBPH). N'ayant ni le temps, ni les moyens de recréer cette fonction sur ces cartes, nous avons abandonné cette idée.

4.2 Protocole du test final

Tout d'abord, la première partie du test était réalisée sur le passager avant. Nous avons effectué l'initialisation à l'arrêt, ainsi qu'une détection des émotions pour vérifier le fonctionnement correct de l'application. Puis, un trajet complètement random a été réalisé pour obtenir trois endroits avec des luminosités différentes (faible, naturelle et forte) afin de vérifier la robustesse

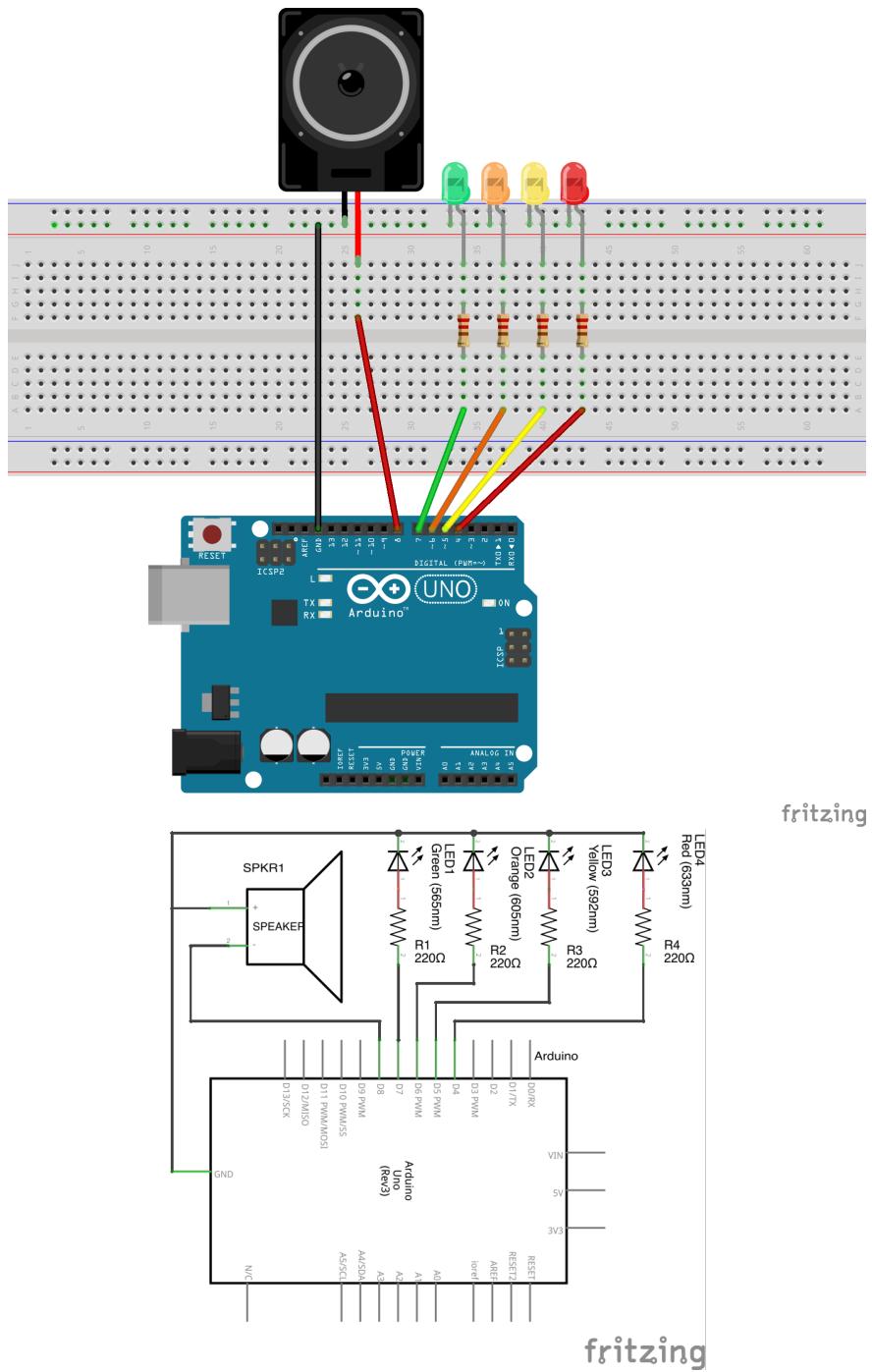


FIGURE 4.1 – Prototype de test

du programme. Lors de la seconde partie du trajet, une caméra a été placée en face du conducteur (au niveau du volant pour ne pas gêner sa visibilité). Nous avons roulé de façon continue pour simuler une situation réelle et confirmer le bon fonctionnement de l'application et visualiser d'éventuels problèmes.

4.3 Discussion et analyse des résultats

Premièrement, l'application fonctionne bien dans le cas où le conducteur est inattentif. Quand nous avons simulé un conducteur endormi, la détection s'effectuait dans un temps compris entre 5 et 15 secondes. Le programme marche également pour les expressions faciales « énervé » et « surpris » mais comme précédemment, la réaction du programme n'est pas instantanée et les émotions doivent être sur-jouée la plupart du temps. Nous en avons conclu que la qualité de la caméra était à revoir, ainsi que son placement dans l'habitacle. En effet, le volant empêchait la caméra de bien visualiser le visage du conducteur.

Le point le plus problématique concerne le changement de luminosité ; lorsque la luminosité est trop importante, le visage du conducteur devient blanc et se confond avec l'arrière-plan. Il est alors impossible d'identifier une quelconque expression faciale, ni même de reconnaître le visage.

Nous venons donc de voir qu'une intégration de notre prototype est possible dans une voiture de tourisme. L'application détecte bien les expressions recherchées malgré un temps de réponse important. Cependant, plusieurs paramètres doivent encore être améliorés comme une meilleure adaptation au changement de luminosité ou le choix d'une caméra avec une résolution plus élevée.

Conclusion

La détection et la reconnaissance d'individus et d'émotions peut s'effectuer à l'aide d'une multitude d'algorithmes résultant de dizaines d'années de recherche (LBPH, FisherFace, EigenFace, Réseaux neuronaux, Modèle de Markov Cachés, ...) dont beaucoup demandent énormément de calculs. Certains ont une approche plus simple et sont facilement intégrables à un projet tel que LBPH à l'aide de la librairie OpenCV. L'application développée tout au long de ce TIPE nous montre que la reconnaissance faciale du conducteur peut être utile pour éviter un vol de voiture, même si la reconnaissance faciale n'est pas un facteur de sécurité fort. En effet, cette sécurité se contourne simplement en utilisant une photo du conducteur.

L'application permet aussi de détecter une personne endormie (seulement après quelques secondes) ou inattentive et permet donc d'éviter une conduite à l'aveugle. Par contre, l'application est très peu fiable pour la détection de la surprise ou de l'énervement. De plus, la surprise et l'énervement peuvent provenir d'une multitude de sources tels que la présence d'un bouchon, une sortie ratée, un enfant qui traverse au mauvais moment, etc. La détection des émotions n'est donc pas forcément la meilleure approche à avoir pour savoir comment la voiture doit réagir. De plus, cette détection peut être gênée par de multiples facteurs tels que la luminosité, une caméra de mauvaise qualité, son placement (la tête peut-être cachée par le volant), etc.

Enfin, il faudrait plutôt prendre en compte le mouvement des points du visage plutôt que des seuils imprécis qui peuvent s'avérer faux avec la modification de l'environnement ou trop peu réactif (il faut parfois sur-jouer une émotion pour obtenir une détection, ce qui n'est pas envisageable dans une application réelle).

Ce projet réalisé dans le cadre du TIPE nous a permis d'acquérir de nombreuses compétences. Nous avons appris à travailler en groupe avec des outils tels que git (pour le partage des ressources, des images et des sources de l'application) ainsi que l'utilisation de pad collaboratif pour rédiger à plusieurs facilement (Framapad). Nous avons aussi appris des méthodes de gestion de projets tels que la Check List (pour connaître les actions à réaliser), la méthode RACI (pour se répartir les tâches), le diagramme de Gantt (pour planifier l'emploi du temps) ou la méthode Scrum (pour se diviser l'écriture de l'application). Par ailleurs, nous avons appris à programmer à l'aide de Python 2.7 (la norme python 3 est sortie, mais elle est encore mal supportée par OpenCV pour les méthodes de reconnaissance faciale) ainsi qu'à maîtriser la bibliothèque OpenCV (pour les parties reconnaissance faciale et traitement d'images). Pour analyser facilement nos résultats donnés sous forme de log, nous avons utilisé le langage PHP. Enfin pour rédiger le rapport et générer les graphes, nous avons utilisé LaTeX. De plus, nous avons développé notre sens de l'autonomie en répartissant les tâches entre les différents membres du groupe que ce soit pour les recherches, ou la rédaction du rapport. Nous avons également développé notre rigueur en suivant des protocoles précis lors de nos expériences. Nous avons pris

énormément de plaisir à réaliser notre TIPE, ce qui nous a permis de confirmer notre projet professionnel qui est la poursuite d'études dans le domaine de l'informatique.

Malgré le fait que notre application fonctionne, nous pouvons l'améliorer de plusieurs façons. Tout d'abord, en reliant directement l'application à la voiture et non à des LEDs qui s'allument. Pour le choix des technologies associées, nous pouvons imaginer un programme plutôt écrit en C ou C++ (le python étant plus lent) tirant mieux parti de la programmation parallèle afin d'effectuer des tâches simultanément. Le programme devrait tourner sur une architecture dédiée (système allégé type linux/BSD allégé). Enfin des algorithmes plus précis devraient être envisagés tels que le suivi des points du visage. Même si cette méthode nécessite bien plus de calculs, l'application serait plus précise. Pour pallier au problème de non information sur la cause de la surprise et de l'énerverment de l'utilisateur, nous pouvons imaginer une amélioration du système actuel avec des radars et des caméras orientées vers l'extérieur de la voiture. Nous pouvons également renforcer la détection d'un conducteur inattentif en générant des classifiers pour détecter lorsque le conducteur utilise son téléphone au volant. Enfin, la reconnaissance faciale du conducteur peut servir, en plus de la protection anti-vol à générer des profils de conduite (consommation d'essence, style de conduite, style de musique, ...). En outre, notre programme pourrait être implémenté dans d'autres types de véhicules comme des camions ou des bus. Si tous les véhicules de tourisme étaient équipés de ce type de programme, nous pouvons envisager une réduction significative du nombre de morts et de blessés sur les routes. Sachant que le coût des accidents corporels en 2011 est estimé à 9,7 milliards d'euros, cela représenterait une économie non négligeable. Enfin, la présence permanente d'une caméra dans l'habitacle pose le problème du respect de la vie privée. Il faudrait à veiller à ce que son utilisation soit réservé uniquement à des fins de protection du conducteur et pas à un moyen supplémentaire d'espionner des individus.

dans les ameliorations : [7] pour detecter via reco vocale. et ca pour le coeur : [6] entrée biblio [15] <- Amaury [23]<- je vois pas ou le caser, c'est plus simuler expression / dans l'intro, c'est un point de départ qu'on a eu... Ph -> [1] [18]

Bibliographie

- [1] Délégation à la sécurité et à la circulation routière ministère de l'intérieur. L'essentiel de la sécurité routière.
- [2] Pascal Caplier Alice. Big brother is watching but helping you : analyse et interprétation de mouvements humains. Technical report, Institut National Polytechnique de Grenoble, 2005.
- [3] Young Andrew, Rowland Duncan, Calder Andrew, Etcoff Nancy, Seth Anil, and Perrett David. Facial expression megamix : Tests of dimensional and category accounts of emotion recognition. *Cognition*, 63 :271–313, 1997.
- [4] Fasel B. and Luettin Juergen. Automatic facial expression analysis : a survey. *Pattern Recognition*, 36, 2003.
- [5] Shan Caifeng, Gong Shaogang, and McOwan Peter. Facial expression recognition based on local binary patterns : a comprehensive study. *Image and Visin Computing*, 27 :803–816, 2009.
- [6] Lisetti Chistine and Nasoz Fatma. Affective intelligent car interfaces with emotion recognition. In *Proceedings of 11th International Conference on Human Computer Interaction*, Juillet 2005.
- [7] Nass Clifford, Jonsson Ing-Marie, Harris Helen, Reaves Ben, Endo Jack, Brave Scott, and Takayama Leila. Improing automotive safety by pairing driver emotion and car voice emotion. In *CHI 2005 : Late Breaking Results : Short papers*, Avril 2005.
- [8] Maturana Daniel, Mery Domingo, and Soto Álvaro. Face recognition with local binary patterns, spatial pyramid histograms and naive bayes nearest neighbor classification. *2011 30th International Conference of the Chilean Computer Science Society*, 0 :125–132, 2009.
- [9] OpenCV dev team. Facerecognizer, Février 2013. http://docs.opencv.org/modules/contrib/doc/facerec/facerec_api.html.
- [10] OpenCV dev team. Face recognition with fisherface, Février 2014. http://docs.opencv.org/modules/contrib/doc/facerec/facerec_tutorial.html#fisherfaces.
- [11] Abdat F., Maaoui C., and Pruski A. Real time facial feature points tracking with pyramidal lucas-kanade algorithm. In Chugo Daisuke, editor, *Human-Robot Interaction*, chapter 13. InTech, Février 2010.
- [12] Le Roy Frédéric. Reconnaissance faciale facile avec opencv et python. *Linux magazine*, 166, Décembre 2013.
- [13] Elfenbein Hillary Anger and Ambady Nalini. On the universality and cultural specificity of emotion recognition : a meta-analysis. *Psychological Bulletin*, 128(2) :203–235, 2002.
- [14] Cohen Ira, Sebe Nicu, Garg Ashutosh, Chen Lawrence, and Huang Thomas. Facial expression recognition from video sequences : temporal and static modeling. *Computer Vision and Image Understanding*, 91 :160–187, 2003.

- [15] Tabari Karmia, Boukadoum Mounir, Chartier Sylvain, and Lounis Hakim. Reconnaissance d'expressions faciales à l'aide d'une mémoire associative bidirectionnelle à fonction de sortie chaotique. In *Maghrébian Conference on Software Engineering and Artificial Intelligence*, décembre 2006.
- [16] DeSilva Liyanage and Ng Pei Chi. Bimodal emotion recognition. In *Fourth IEEE International Conference on Automatic Face and Gesture Recognition*. National University of Singapore, mars 2000.
- [17] Turk Matthew and Pentland Alex. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3 :71–86, 1991.
- [18] Hetier Mickaël. *Analyse et quantification des comportements des conducteurs automobiles lors des phases de pré-crash : contribution au développement d'un modèle de détection des postures de conduite en temps réel*. PhD thesis, Université de Valenciennes et du Hainaut-Cambrésis, 2008.
- [19] Sebe N., Lew M.S., Y. Sun, Cohen I., Gevers T., and Huang T.s. Authentic facial expression analysis. *Image and Vision Computing*, 25 :1856–1863, 2007.
- [20] Sebe Nicu, Lew Michael, Cohen Ira, Garg Ashutosh, and Huang Thomas. Emotion recognition using a cauchy naive bayes classifier. In *Sixteenth international conference on pattern recognition*, volume 1, pages 17–20, 2002.
- [21] Gomez Rafael, Popovic Vesna, and Bucolo Sam. Driving : the emotional experience and automotive design. In *Fourth international conference on design and emotion*. Middle East Technical University, Ankara, juillet 2004.
- [22] Ioannou Spiros, Raouzaiou Amaryllis, Tzouvaras Vasilis, Mailis Theofilos, Karpouzis Kostas, and Kollias Stefanos. Emotion recognition through facial expression analysis based on a neurofuzzy network. *Neural Networks*, 18 :423–435, 2005.
- [23] Wan Vincent, Anderson Robert, Blokland Art, Braunschweiler Norbert, Chen Langzhou, Kolluru BalaKrishna, Latorre Javier, Maia Ranniery, Stenger Björn, Yanagisawa Kayoko, Stylianou Yannis, Akamine Masami, Gales Mark, and Cipolla Roberto. Photo-realistic expressive text to talking head synthesis. In *Interspeech*, août 2013.

Annexe A

Code des applications

A.1 Application principale

```
1 #!/usr/bin/env python2
2 # -*- coding: utf-8 -*-
3
4 #     oooooooooooooo  oooooo  oooooooooooooo  oooooooooooooo      o      oooooooooooooo  ooooooooooooo8
5 #     88  888  88  888  888  888  888  888      888      888  888  888  888
6 #           888      888  888oooo88  888ooo88      8  88      888oooo88  888oooo88
7 #           888      888  888      888      888oooo88      888      888
8 #           08880      08880  08880      0888oooo8888  0880  08880  08880      088oooo888
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 à réaliser pour la base de données
25 THRESHOLD = 90                                    #Seuil de reconnaissance
26 CAMERA = 1                                       #La caméra
27 ARDUINO = False                                 #Utiliser l'arduino ?
28
29 INDIVIDUS = []
30
31 #####
32 def sendSerial(ser, command):
33     """Envoie command à l'arduino"""
34     if(ARDUINO):
35         ser.write(command)
36
37 class CreateDataBase():
38     def __init__(self, imgPath, 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.LBPHBaseImage = 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 recuperé"""
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 à 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)
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 reconnaît l'identité de la personne si enregistrée"""
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 froncés"""
273         return EyeHeight < self.eyeHeight - threshold
274
275     def EyeHeightThanNeutral(self, EyeHeight, threshold):
276         """Determine si les yeux sont écarquillés"""
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("TIEPAPS", 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.items(), 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   * 0oooooooooooo  oooooo  ooooooooooooo  oooooooooooooo      o      ooooooooooooo  ooooooooooooo8
3   * 88 888 88    888  888    888 888           888    888 888 888
4   *          888    888  888oooo88  888ooo8     8 88    888oooo88  888oooo888
5   *          888    888  888           888    8ooooo88  888    888
6   *          o888o   o888o  o888o       o888oooo8888  o88o   o888o  o888o       o88oooo888
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 demarre la connexion serie

```

```

74     while (!Serial){} // on attend que la connexion serie demarre
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 demarre
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 donnee 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 }
```

Annexe B

Documents utilisés en CST

B.1 check-list

Tâches :

- | | Réalisation : |
|---|---|
| 1. Recherche sur des projets similaires déjà existants | <input type="checkbox"/> oui <input type="checkbox"/> non |
| 2. Recherche bibliographique sur la sécurité routière | <input type="checkbox"/> oui <input type="checkbox"/> non |
| 3. Recherche bibliographique sur la reconnaissance faciale | <input type="checkbox"/> oui <input type="checkbox"/> non |
| 4. Recherche bibliographique sur la reconnaissance des émotions | <input type="checkbox"/> oui <input type="checkbox"/> non |
| 5. Contacter des professionnels | <input type="checkbox"/> oui <input type="checkbox"/> non |
| 6. Apprendre Python | <input type="checkbox"/> oui <input type="checkbox"/> non |
| 7. Apprendre à utiliser git | <input type="checkbox"/> oui <input type="checkbox"/> non |
| 8. Créer le dépôt git | <input type="checkbox"/> oui <input type="checkbox"/> non |
| 9. Apprendre à utiliser OpenCV | <input type="checkbox"/> oui <input type="checkbox"/> non |
| 10. Comparer les différents systèmes de reconnaissance faciale | <input type="checkbox"/> oui <input type="checkbox"/> non |
| — vitesse d'exécution | <input type="checkbox"/> oui <input type="checkbox"/> non |
| — fiabilité | <input type="checkbox"/> oui <input type="checkbox"/> non |
| 11. Réaliser l'application | <input type="checkbox"/> oui <input type="checkbox"/> non |
| — module de reconnaissance faciale | <input type="checkbox"/> oui <input type="checkbox"/> non |
| — module de reconnaissance des émotions | <input type="checkbox"/> oui <input type="checkbox"/> non |
| 12. Créer le prototype de test | <input type="checkbox"/> oui <input type="checkbox"/> non |
| 13. Réaliser des tests | <input type="checkbox"/> oui <input type="checkbox"/> non |
| — tests en situation idéale | <input type="checkbox"/> oui <input type="checkbox"/> non |
| — tests en situation réelle | <input type="checkbox"/> oui <input type="checkbox"/> non |
| — déboggage | <input type="checkbox"/> oui <input type="checkbox"/> non |
| 14. Analyser les résultats des tests | <input type="checkbox"/> oui <input type="checkbox"/> non |
| 15. Rédiger le rapport écrit | <input type="checkbox"/> oui <input type="checkbox"/> non |
| 16. Relecture du rapport écrit | <input type="checkbox"/> oui <input type="checkbox"/> non |
| 17. Préparer l'oral | <input type="checkbox"/> oui <input type="checkbox"/> non |
| 18. Réaliser le diaporama | <input type="checkbox"/> oui <input type="checkbox"/> non |
| 19. Réaliser la fiche synoptique | <input type="checkbox"/> oui <input type="checkbox"/> non |

B.2 Matrice RACI

Tâche \ Membre	Sébastien	Pierre-Henri	Amaury	Finn	Déborah
1	C	R-C	C	I	
2	I	R-C	I	I	
3	R-C	I	I	I	
4	I	I	R-C	I	
5	C	C	C	I	
6	R	C	I		
7	R	C	C		
8	R-C	I	I		
9	R-C	C	C		
10.1	R-C	C	C	I	
10.2	R-C	C	C	I	
11.1	R-C	C	C	A	
11.2	C	C	R-C	A	
12	C	R-C	C	I	
13.1	C	R-C	C	I	
13.2	C	C	R-C	I	
13.3	R-C	C	C	I	
14	C	R-C	C	A	
15	C	C	R-C	A	I
16	I	I	R-C		
17	C	R-C	C		
18	C	C	R-C	A	I
19	C	R-C	C	A	I

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

2013-2014

Diagramme de Gantt

