DYLART : ENDORMISSEMENT AU VOLANT

Arthur GUIHARD
Dylan JERAULT

Master 1 – Électronique, Énergie Électrique et Automatisme

Année 2018/2019

Remerciements :

Nous tenons à remercier M. Oscar Acosta, notre tuteur de ce projet, pour l'aide et le temps qu'il nous a consacré depuis l'ébauche du sujet en Novembre 2018.

Nous souhaitons, tout particulièrement, remercier M. Jérémy Beaumont qui nous a apporté son soutien tout au long de ce projet sur la partie programmation sous Linux.

Nous voulons adresser nos remerciements à Nathanaël Momefo Fouobe et Yakup Memisoglu pour leur base de travail et les renseignements qu'ils nous ont apportés tout au long du semestre afin de continuer et d'améliorer leur projet initial.

Nous aimerons également remercier l'Université National Tsing Hua de Taïwan et ses chercheurs Ching-Hua Weng, Ying-Hsiu Lai, et Shang-Hong Lai qui nous ont permis d'avoir une base de données importante afin de réaliser notre projet.

Table des matières :

I. Introduction

D'après l'Observatoire National Interministériel de la Sécurité Routière (ONISR), en 2017, 3 684 personnes ont perdu la vie sur les routes de France (métropole et outre-mer). Ce chiffre a heureusement diminué en 2017 en comparaison à 2016, mais reste une préoccupation majeure du Ministère des Transports, rattaché au Ministère de la Transition Écologie et Solidaire. En effet, la volonté du gouvernement est de réduire chaque année le nombre d'accidents routiers corporels, et par conséquent le nombre de personnes tuées. Selon le rapport de l'ONISR, l'objectif est fixé à 2000 morts par an sur la route d'ici 2020, soit une baisse de quasiment 46 points.

Plusieurs causes sont mises en évidence dans ce rapport, comme la consommation d'alcool, le non-respect des limitations de vitesse et des sens de circulation, mais aussi l'inattention et la somnolence, jusqu'à l'endormissement. D'après le rapport de l'ONISR, le weekend est davantage touché par les accidents corporels sur les routes que les autres jours de la semaine. En effet, c'est durant les weekends que les automobilistes se permettent de partir en séjour, de profiter du temps libre pour rendre visite à des proches, choses plus difficiles à mettre en œuvre pendant la semaine de travail. Mais organiser ce genre d'activités peut engendrer une diminution du temps de sommeil ainsi qu'une baisse considérable des conditions de sommeil, ce qui augmente fortement le taux de fatigue, et donc les risques encourus au volant des véhicules motorisés. La fatigue, selon les chiffres du gouvernement, représente un tiers des causes d'accidents corporels sur les routes de France, et reste une des causes majoritaires, avec l'alcool (qui représente un second tiers par exemple).

Suite à l'analyse de ces chiffres, nous avons cherché un sujet pour notre projet qui rentrerait en adéquation avec les deux domaines dans lesquels nous souhaitions continuer notre formation dans le futur. L'un de nous s'est d'abord intéressé au domaine de la santé, de la médecine, et de l'accompagnement des personnes dans leur quotidien, alors que l'autre voulait continuer sa formation dans l'embarquement de systèmes dans le milieu de l'automobile. Ces deux domaines réunis nous offraient encore de multiples possibilités de sujet d'étude. Mais à la vue de ce constat plus que négatif sur la mortalité routière ainsi que sur les causes de ces accidents, nous avons fait le choix du projet DYLART, un détecteur d'endormissement des conducteurs de véhicules quatre roues, celui-ci basé sur la reconnaissance faciale. Une nécessité est alors apparue, celle du fonctionnement de notre détecteur sur un éventail très important d'individus. En effet, il nous semble essentiel que notre détecteur d'endormissement fonctionne correctement sans tenir compte de l'âge du conducteur, de son origine ethnique, de sa corpulence, de son sexe, ni de ses traits physiques. Mais d'autres contraintes se sont ajoutées aux précédentes, comme la météo. Il est vrai qu'un temps très ensoleillé et un temps pluvieux n'offrent pas les mêmes conditions de captures de l'image par notre caméra. Il nous fallait donc un système suffisamment robuste pour remplir tous ces critères, et permettant donc de ne pas négliger le moindre cas d'utilisation. Ensuite, nous voulions concevoir un système permettant une utilisation en temps réel, donc sur une vidéo et non sur une simple image, un autre fait à prendre en compte dans notre conception.

La détection de l'endormissement du conducteur d'une voiture (généralement un véhicule quatre roues), basée sur la reconnaissance faciale et le calcul de l'ouverture des yeux, remplit donc parfaitement les conditions et critères pour permettre une possible diminution de ces chiffres alarmants sur la mortalité routière, et sur la part de l'endormissement dans les causes. Nous vous présenterons donc notre travail de la manière suivante: dans un premier temps l'élaboration du cahier des charges, dans lequel nous nous posons toutes les questions nécessaires au balayement de toutes les problématiques que nous pourrons rencontrer, puis nous évoquerons la base de données que nous avons utilisé tout au long du projet, celle-ci répondant à des attentes très précises. Dans les parties suivantes, nous vous parlerons de la programmation de notre détecteur, des différents logiciels et langages de programmation que nous avons utilisé, puis de la manière dont nous utilisons toutes les données recueillies pour définir un détecteur optimal. Enfin, nous aborderons les différentes applications qu'il peut y avoir de notre détecteur.

II. Analyse fonctionnelle - Cahier des charges

1. Diagrammes

1. Bête à corne

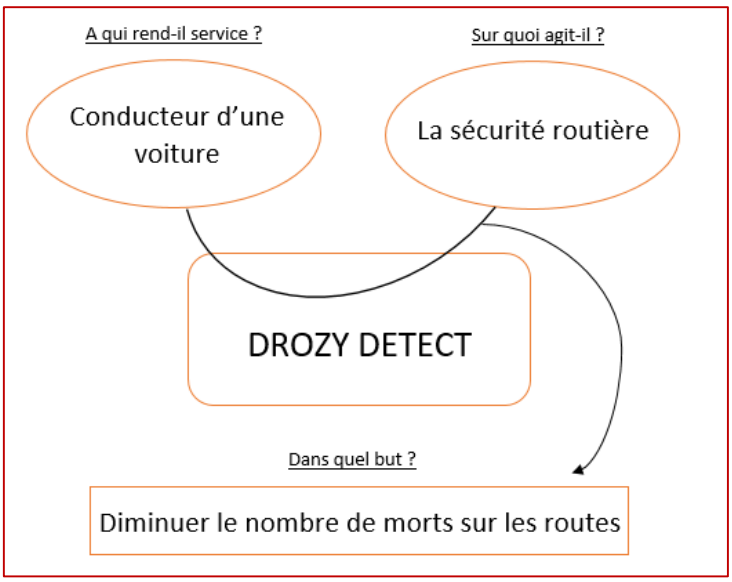


Figure 1 : Bête à corne

2. Diagramme pieuvre

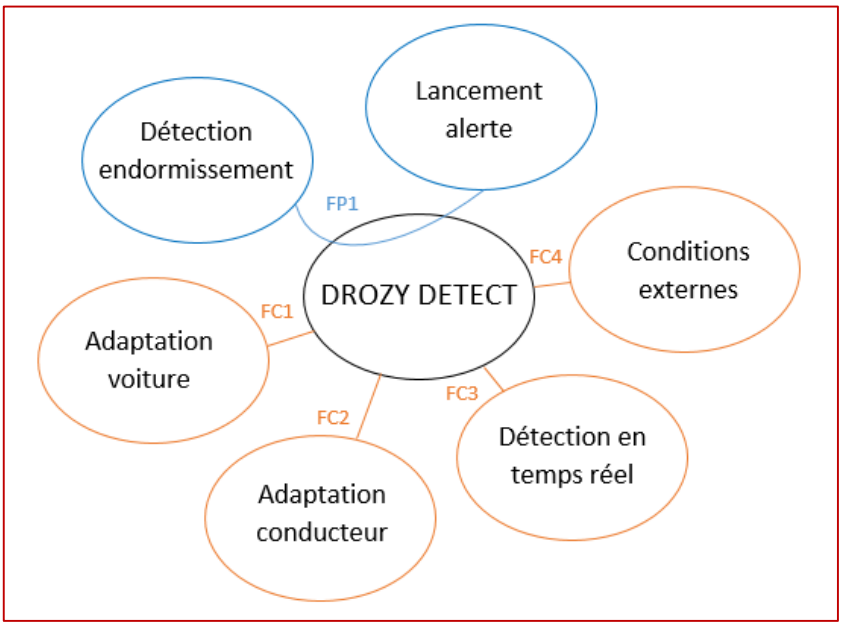


Figure 2 : Diagramme pieuvre

FP1 : Détection de la fermeture des yeux - Calcul du temps de fermeture - Lancement alerte
FC1 : Adaptation de la caméra au véhicule (taille, poids, type de caméra)
FC2 : Adaptation au différent type de conducteur (taille et élargissement des yeux différents)
FC3 : Exécuter la détection instantanément afin d'être rapide pour alerter le conducteur au plus vite
FC4 : Adaptation aux conditions externes à la détection (luminosité, lunettes…)

2. Démarche

1. Partie 1 : Fonctionnement de la détection

Tout d'abord, le cahier des charges se compose de plusieurs parties. En effet, nous commençons par le fonctionnement de la détection en général. La première partie est la détection d'un visage dans une image puis le repérage des éléments caractéristiques de ce visage. Nous avons décidé de passer par ces deux étapes pour permettre la détection de l'endormissement d'un conducteur.

Par la suite, nous devons utiliser les caractéristiques de notre visage recueillies précédemment afin de travailler sur les yeux de notre conducteur. Nous avons décidé de nous baser uniquement sur la reconnaissance des yeux, pour savoir si ces derniers sont fermés ou bien ouverts. La détection d'endormissement d'un conducteur va donc directement intervenir sur la réaction des yeux. Nous voulions travailler sur une détection basée sur le rythme cardiaque. Cependant, il était plus complexe pour nous d'étudier le rythme cardiaque d'un individu et de détecter rapidement la baisse de celui-ci afin d'éviter un accident. En effet, la variation du rythme cardiaque est trop « lente » lorsque l'on s'endort et donc le conducteur aurait déjà eu accident avant qu'il soit prévenu. De plus, après avoir travaillé sur ces caractéristiques, nous voulons envoyer une alerte au conducteur afin que ce dernier puisse se réveiller.

2. Partie 2 : Base de données

Afin de tester le programme de détection que nous allons réaliser, nous devions nous appuyer sur une base de données composée de vidéos afin de voir le bon fonctionnement de notre détecteur. Nous avons donc décidé de rechercher des données en libres service sur Internet afin de pouvoir réaliser notre propre base de données. Comme cette dernière est le pilier de toutes nos recherches, nous avons choisi de commencer ce projet par chercher des données cohérentes afin de pouvoir avancer au mieux et le plus efficacement de ce projet.

3. Partie 3 : Définition des critères seuil/durée

Cependant, de nombreux critères restent à définir. En effet, deux critères sont très importants pour une détection optimale d'une personne en train de s'endormir au volant. Ce sont les critères du seuil et de la durée. C'est-à-dire, qu'avec ces deux critères nous devons donner une réponse à la question suivante : Quand est-ce que le conducteur est endormi?

Afin de savoir si un conducteur est endormi ou non, nous devons nous baser sur différents cas afin de seuiller au plus précis notre détection. C'est pour cela que nous avons besoin de chercher une base de données variée et volumineuse afin de pouvoir tester différentes vidéos et différents individus. Ceci servant à trouver le seuil de détection et le temps maximum avant l'envoi d'une alerte au conducteur.

III. Base de données

1. L'origine, la provenance

Après de nombreuses recherches, nous sommes parvenus à trouver une base de données qui nous semblait cohérente avec notre projet. En effet, nous sommes arrivés sur le site d'une Université de Taïwan, la National Tsing Hua University. Cette dernière travaille avec un laboratoire de recherche nommé NTHU.

À la suite d'un rapport sur les accidents mortels à cause de la conduite somnolente en 2014, la National Sleep Foundation (NSF) des États-Unis à décider d'appliquer des nouvelles réglementations ainsi que de la prévention sur la somnolence et la distraction. Depuis ces nouvelles lois, le développement des systèmes de sécurité dans les véhicules est devenu très important. Ce laboratoire de recherche a alors réalisé cette base de données afin de créer un challenge ingénieur. En effet, un concours a été organisé afin de réaliser une détection la plus précise possible de l'endormissement au volant en créant une application la plus adéquate afin de contrôler la somnolence au volant. De nombreux ingénieurs ont réalisés leur application via la base de données que nous allons utiliser.

Pour acquérir celle-ci, nous avons eu le soutien de notre tuteur de projet, M. Oscar Acosta, pour la demande officielle de cette base de données. Nous avons envoyé un mail joint d'un agrément d'utilisation à cette université afin qu'il nous envoie cette dernière et qu'ils nous autorisent son utilisation. C'est la partie de notre projet qui nous a pris le plus de temps de recherches (6 semaines pour la réponse de l'Université et la réception de la base de données).

2. Le contenu

La base de données que nous avons reçue est composée de 18 sujets différents sur les 36 personnes qui ont été utilisées dans le cadre du concours d'ingénieur expliqué précédemment. Nous pouvons voir des personnes de différentes nationalités comme par exemple des personnes de type asiatique ou bien européen. Nous avons 6 personnes de sexe féminin et 12 personnes de sexe masculin. Ces personnes ont aussi les yeux complétement différents, ce qui est un atout pour notre projet. En effet, en ayant le maximum d'individus différents, nous allons pouvoir traiter le plus de cas possibles sur les caractéristiques des yeux et donc pouvoir avoir une application plus précise de notre détecteur d'endormissement.

Chaque personne est présente dans 5 scénarios. En effet, la personne en position de conduite agit selon 5 cas différents :

- De jour, sans lunettes
- De jour, avec lunettes
- De nuit, sans lunettes
- De nuit, avec lunettes
- Avec des lunettes de soleil

Dans chacune des situations précédentes, la base de données est composée de 4 vidéos :

- « Yawning » : Le conducteur fait des bâillements.
- « SlowBlinkWithNodding » : Le conducteur hoche la tête et est endormi dans la vidéo.
- « SleepyCombination » : Le conducteur ferme les yeux et s'endort durant la vidéo.
- « NonSleepyCombination » : Le conducteur rigole, parle, et regarde sur le côté.

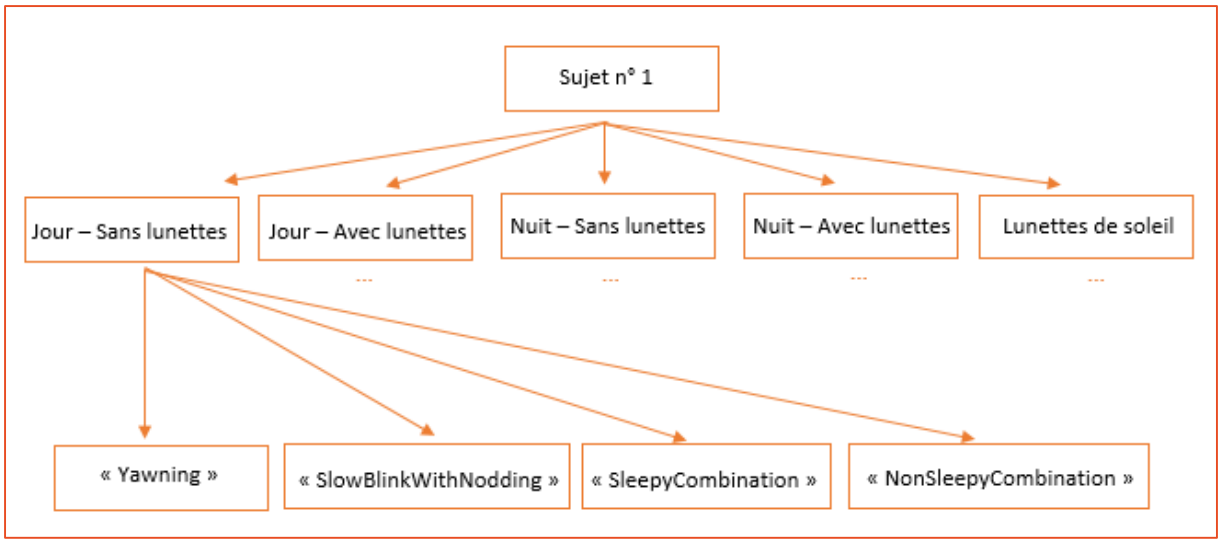


Figure 3 : Exemple d'un sujet – Base de données

Cette base de données est aussi composée d'annotations. En effet, chaque vidéo est annotée en fonction de la zone du visage que nous voulons tester. Les différents types d'annotations sont réalisées selon l'endormissement, du mouvement de la tête (de haut en bas), du mouvement de la bouche, et du mouvement des yeux. Nous allons travailler avec les dernières annotations car nous cherchons à savoir, dans notre cas, si le conducteur a les yeux ouverts ou fermés. Il faut savoir que les annotations sont composées de 0 et de 1 :

- 0 : Les yeux du conducteur sont ouverts
- 1 : Les yeux du conducteur sont fermés

De plus, fournies avec cette dernière, toutes les informations liées à la caméra qui filmait les personnes en simulations de conduite et aux fichiers fournis. En effet, nous savons que les vidéos ont été filmées grâce à une caméra Infrarouge à 30 images/seconde dans certains scénarios, et à 15 images/seconde dans d'autres scénarios. Les fichiers que nous avons reçus sont des documents .txt pour les annotations, et des fichiers .avi pour les vidéos. De plus, nous savons que la taille des vidéos est de 640*480 pixels.

À la suite de cette acquisition de données, nous avons, dans un premier temps, commencé par travailler sur la création de notre détecteur grâce aux connaissances théoriques sur la programmation que nous avons acquises durant nos études supérieures.

IV. Linux – Open CV – Dlib : Programmation du détecteur

1. Langage de programmation – Librairies

Tout d'abord, nous avons fait quelques recherches tout en se basant sur ce qui avait été fait l'année précédente par deux anciens étudiants du Master 1 Nathanaël Momefo Fouobe et Yakup Memisoglu. Nous avons eu plusieurs choix à réaliser afin de débuter notre prédicteur. En effet, nous devions choisir le langage de programmation pour créer notre application ainsi que les librairies et Open Sources nécessaires au bon fonctionnement de notre application.

En collaboration avec notre tuteur, nous avons choisi de programmer en langage C++ au lieu du langage python qui a été utilisé auparavant. Pour le traitement d'image, nous avons été conseillés d'utiliser le C++. De plus, ce dernier nous est familier car nous travaillons sur celui-ci depuis notre troisième année de Licence. Nous avons également choisi, en collaboration avec M. Jeremy Beaumont, d'installer en double boot sur un de nos PC le système d'exploitation Linux Ubuntu 18.04.01. Ce dernier permet une programmation plus lisible, sans logiciel à installer au préalable car nous pouvons directement coder notre application sous invite de commandes à partir de fichiers textes.

Durant le second semestre, nous avons choisi l'option AVIO qui consiste au traitement d'image. Grâce à cette option, nous avons découvert la librairie OpenCV qui est applicable au langage C++. Nous avons donc décidé de réaliser notre détecteur grâce à cette libraire qui est gratuite et en libre-service. Cette dernière a été développée par Intel et est spécialisée dans le traitement d'images en temps réel. C'est-à-dire que nous pouvons utiliser cette libraire afin de traiter les vidéos de notre base de données.

De plus, après quelques recherches, nous avons découvert une autre librairie qui se nomme Dlib. Cette dernière contient des algorithmes d'apprentissage automatique pouvant nous aider à créer notre application. Les licences open sources de Dlib nous permettent d'utiliser cette dernière gratuitement. Les open sources sont modifiables et utilisables par n'importe quelle personne. En effet, nous pouvons trouver Dlib sur le site « GitHub » qui regroupe les travaux de nombreux programmateurs que nous pouvons utiliser facilement sans infractions à une charte de confidentialité. Ce site est un espace de partage communautaire prônant la gratuité de la propriété intellectuelle. Cette libraire est composée de deux membres qui vont nous intéresser par la suite :

- « Shape_predictor »
- « Face_landmarks »

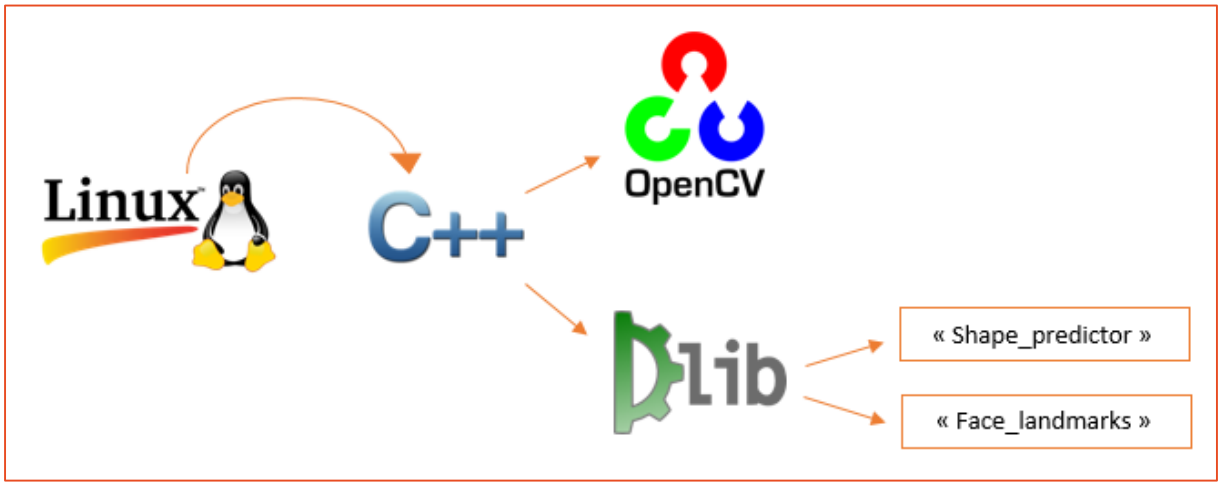


Figure 4 : Configuration outil de travail – Langage/Librairies

2. Détection du visage

1. Descripteur HOG (Histogram of Oriented Gradient)

La première étape de notre application est la détection du visage qui se trouve sur une image. Pour se faire, nous utilisons un descripteur HOG (Histogramme de Gradient Orienté). Ce dernier est une caractéristique permettant la détection d'un objet dans une image. Cette méthode consiste en la distribution de l'intensité du gradient ou la direction des contours. C'est-à-dire que l'image est divisée en plusieurs parties. Sur chacune de ces parties, un histogramme des directions du gradient ou des orientations des contours pour les pixels est calculé. L'ensemble des histogrammes configure le descripteur HOG que nous voulons utiliser. Une normalisation en fonction du contraste, en travaillant sur l'intensité des parties, est réalisée par la suite pour que les changements d'illuminations et des ombres de l'image n'interfèrent pas avec les données récoltées.

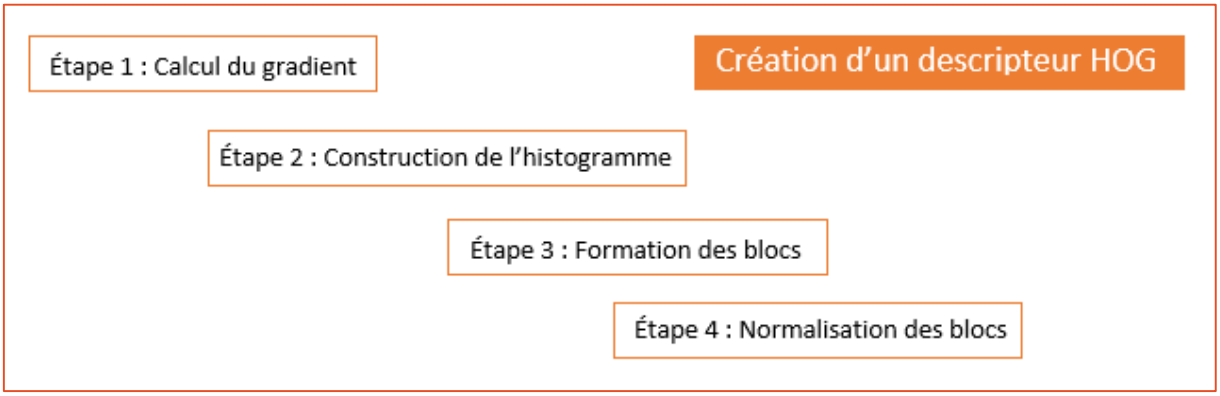


Figure 5 : Étapes d'une réalisation d'un descripteur HOG

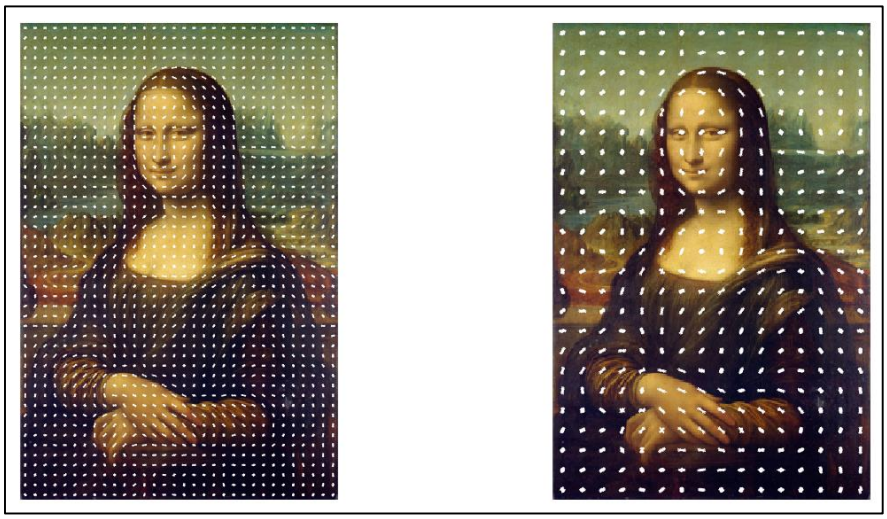


Figure 6 : Exemple descripteur HOG - Joconde

Afin de terminer notre processus de détection d'objet, nous devons utiliser nos descripteurs HOG réalisés précédemment afin de d'élaborer une classification d'objet via l'apprentissage supervisé. Cette méthode est un apprentissage automatique qui consiste à créer une prédiction à partir de nombreux exemples prédéfinis. Cette méthode existe déjà, et se nomme « Méthode de Viola et Jones ».

2. Méthode de Viola et Jones

La méthode de Viola et Jones a été publiée par les chercheurs Paul Viola et Michael Jones en 2001. Celle-ci consiste en la détection d'objet dans une image numérique. C'est un procédé d'apprentissage supervisé comme nous l'avons expliqué précédemment. Cette méthode a été configurée via des milliers d'exemples d'objets.

De plus, cette dernière, très utilisée lors de la détection d'objet, est implantée dans la bibliothèque OpenCV que nous utilisons pour notre application. Liée avec le descripteur HOG, nous n'avons donc pas besoin de créer ce dernier et nous pouvons directement l'utiliser dans notre programme de détection d'endormissement. Cependant, nous savons que des implémentations matérielles ont été réalisées sur FPGA par exemple. L'utilisation de ce dernier permet de gagner en temps d'exécution du détecteur par rapport à l'implémentation faite sous OpenCV.

3. Extraction des caractéristiques – Kazemi et Sullivan (2014)

La seconde étape de notre détection, après le repérage d'un visage dans une image, est l'extraction des caractéristiques de ce dernier. Pour se faire, nous devons utiliser la méthode de Kazemi et Sullivan de 2014. Elle consiste en la détection des repères faciaux importants d'un visage. Un apprentissage d'une cascade de régresseurs est réalisé à partir de données représentant des visages annotés manuellement en spécifiant les coordonnées des différents repères que nous voulons détecter. De plus, cet apprentissage permet d'estimer la position des différents repères faciaux grâce à l'intensité des pixels se trouvant dans l'image comme lors de la représentation HOG. Ensuite, le détecteur est utilisé dans la librairie Dlib que nous avons importée. Cette dernière permet de localiser 68 points de repères représentant la structure d'un visage comme nous pouvons le voir ci-dessous.

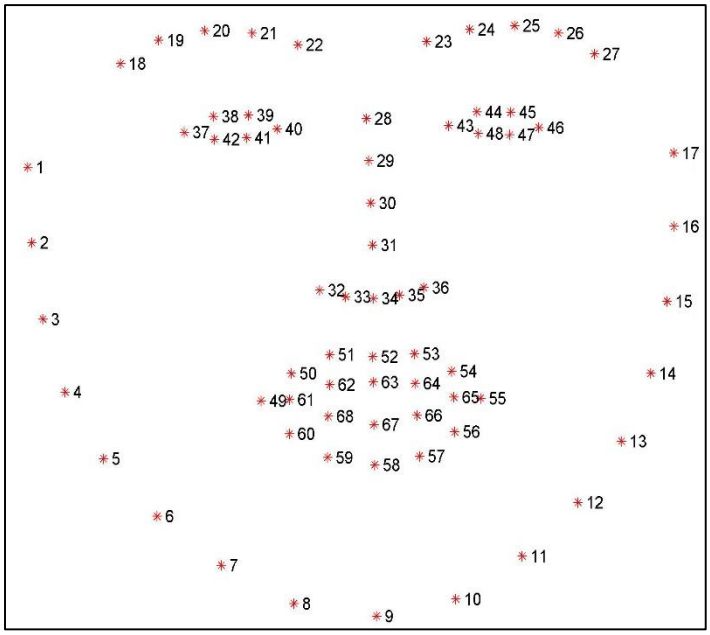


Figure 7 : 68 points de repères faciaux

Afin de réaliser ces repérages, nous devons utiliser les deux fonctions implémentées sous Dlib que nous avons évoquées précédemment. En effet, le « Shape_predictor » va nous permettre de localiser correctement notre visage en fonction de la forme de celui-ci. Ce prédicteur est superposé à la fonction « Face_landmarks ». En effet, cette dernière utilise la méthode de Kazemi et Sullivan précédente dans le but de pouvoir obtenir les coordonnées des 68 repères caractéristiques du visage. Afin d'initialiser correctement ces 68 repères, nous utilisons un fichier nommé « shape_predictor_68_face_landmarks.dat » créé suite à cette méthode.

Dans un premier temps, nous avons testé ces méthodes seulement sur une image. En effet, nous voulions travailler étape par étape sur le traitement d'une image afin de savoir si notre application fonctionnait correctement sur une simple image. Après de nombreuses modifications liées à notre code, nous avons réalisé la superposition du détecteur à notre image comme nous pouvons le voir ci-dessous avec la photo de Lena.

Figure 8 : Image de Lena avec superposition du détecteur

Dans une seconde temps, nous avons testé la même chose sur une partie de vidéo d'une personne en train de conduire. Pour se faire, nous avons modifié notre code afin d'adapter la lecture de la vidéo au lieu d'une image et de traiter la vidéo image par image. Sur chacune d'entre elles nous appliquons le détecteur réalisé auparavant ce qui recréer la séquence vidéo à l'identique mais avec le détecteur en superposition (voir les codes en Annexes).

À la suite de ces applications, nous devons extraire les caractéristiques servant à la création du prédicteur afin de travailler directement sur la détection des yeux du conducteur. Les librairies de Dlib importés précédemment, et notamment le « Shape_predictor » vont nous servir pour la suite de notre projet.

4. Traitement des caractéristiques

1. Calcul EAR (Eye Aspect Ratio)

La partie du visage qui nous intéresse le plus pour notre application est les yeux. En effet, afin de détecter l'endormissement du conducteur nous avons voulu travailler sur l'EAR (Eye Aspect Ratio). Cette méthode élaborée par Tereza Soukupova et Jan Cech. Cette dernière est un calcul de rétrécissement des yeux. En effet, en récupérant les caractéristiques particulières des yeux, à savoir leur position dans l'image, nous allons pouvoir effectuer ce calcul. Nous devons donc nous focaliser sur les 12 repères correspondant aux deux yeux suivants :

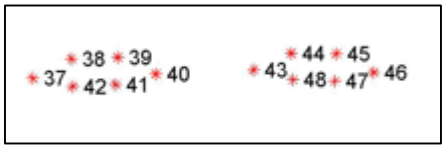
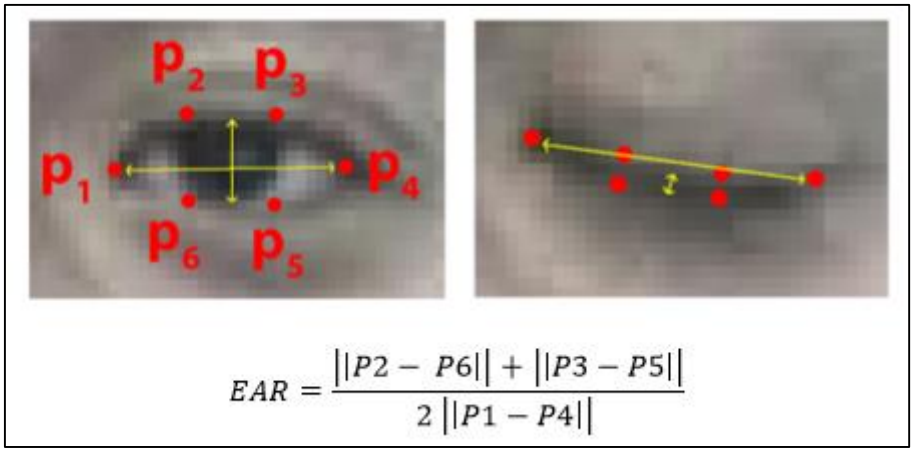


Figure 9 : Les 12 repères des yeux

Grâce au « Shape_predictor » nous pouvons récolter chaque position en x et y des repères précédents. Grâce à ces coordonnées nous pouvons appliquer la formule suivante qui correspond au calcul EAR (rétrécissement des yeux) suivant la logique de la figure 10.



$$EAR = \frac{\left||P2 - P6|\right| + \left||P3 - P5|\right|}{2\left||P1 - P4|\right|}$$

Figure 10 : Équation et description du calcul de l'EAR

L'EAR est calculé pour chaque œil et moyenné en fonction du rapport des deux yeux. C'est-à-dire que nous faisons $EAR = \frac{(EAR_{gauche} + EAR_{droit})}{2}$. D'après l'exemple de l'image de Lena précédemment, nous obtenons les rapports suivants :

```
EAR de l'oeil droite: 0.425456
EAR de l'oeil gauche: 0.519423
EAR: 0.472439
```

Figure 11 : EAR de l'image de Lena

Afin de pouvoir calculer l'EAR optimal permettant de détecter un endormissement au volant, nous devons déterminer un seuil optimal pour lequel l'EAR définira si la personne a les yeux fermés ou non. Pour se faire, nous devons travailler sur une base de données d'apprentissage. En effet, nous allons utiliser la base de données récoltée et travailler sur les vidéos qui s'y trouvent.

2. Stockage des données

La première étape permettant l'identification du seuil EAR optimal est le stockage des différentes valeurs EAR de chacune des images de chaque vidéo. Afin d'avoir un meilleur rendement, nous avons choisi de travailler sur 36 vidéos. Comme nous avons 18 sujets, nous allons appliquer notre programme sur les vidéos avec les scénarios suivants :

- La conduite se fait de jour – le conducteur ne porte pas de lunettes – SleepyCombination (Le conducteur ferme les yeux et s'endort durant la vidéo)
- La conduite se fait de nuit– le conducteur ne porte pas de lunettes – SleepyCombination (Le conducteur ferme les yeux et s'endort durant la vidéo)

Chacun des deux cas précédents est composé donc de 18 vidéos. La conduite de jour a été filmée en 30 fps alors que la conduite de nuit a été filmée en 15 fps. De plus, nous avons décidé de stocker chaque calcul EAR correspondant à chaque image de la vidéo dans un fichier .txt créé spécialement afin de pouvoir traiter ces données avec des seuils différents par la suite. En effet, le but étant de trouver le seuil optimal grâce aux valeurs de références que nous disposons dans notre base de données.

Des contraintes sont apparues à ce moment de notre projet. En effet, le travail sur ces 36 vidéos d'environ 100 secondes a pris plus de 3 jours. Cela est dû à la lenteur du prédicteur car nous passons directement par la bibliothèque OpenCV comme expliqué précédemment dans ce rapport page 11, mais aussi à la lenteur des processeurs du PC HP que nous utilisons pour notre projet.

3. Détection endormissement

En explorant les documents mis à disposition sur Internet des personnes qui ont déjà travaillé sur cette méthode de détection, la valeur du seuil varie mais tourne autour de 0,24 et 0,26. Ces études nous permettent d'imaginer le seuil que l'on va obtenir après le traitement des données que nous avons récoltés précédemment. De plus, nous savons que pour savoir si une personne s'est endormie au volant, nous allons devoir modifier notre code afin de créer une boucle de détection des yeux fermé en continu. Il nous faudra, par la suite, savoir à quel moment, au bout de combien de temps, une personne est considérée dans un état de somnolence.

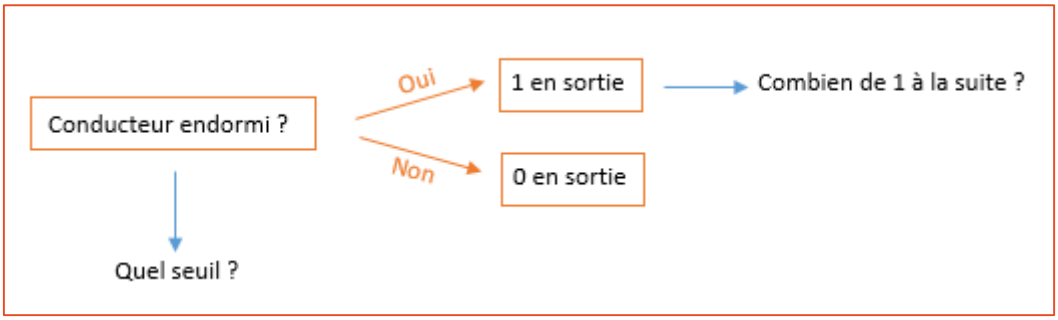


Figure 12 : Critères restants à définir

Afin de continuer notre projet, nous devons passer sur le traitement des informations obtenues, et nous avons décidé d'utiliser le logiciel Matlab que nous utilisons lors de nos cours.

V. Mesure du seuil – Matlab

1. Étapes de traitement des données

Tout d'abord, l'objectif principal est de calculer le seuil EAR optimal à la détection d'endormissement. Pour se faire, nous allons nous baser sur la sensibilité et la spécificité de nos tests en fonction d'un seuil que l'on fait varier. En effet, à la suite des tests réalisés, nous savons que les valeurs de l'EAR se situent entre 0,15 et 0,30. Nous avons donc décidé de créer 30 seuils différents afin d'avoir une plage de données assez précises des valeurs de seuil.

La première étape consiste en la modification de nos données de base (valeur de l'EAR) en 0 où 1 selon la détection. En effet, pour chacun des 30 seuils, nous regardons la valeur de l'EAR image par image et nous écrivons 0 si la valeur est au-dessus du seuil, ce qui voudrait dire que le conducteur ne ferme pas les yeux, et 1 sinon. Sous Matlab, nous écrivons la fonction suivante et nous traitons les 36 vidéos avec les 30 seuils différents, à savoir de 0,15 à 0,30 avec un pas de 0,005 (voir annexe p30).

```
function  [outputtab] = closed_eyes_bin( inputfile, seuil )

    val = 0;

for i=1:1:size(inputfile)
    val = 0;
    if inputfile(i) < seuil
        val = 1;

    else
        val = 0;
    end
    outputtab(i)= val;

end
```

Figure 13 : Fonction création fichier 0-1

Après avoir récolté ces données et stocké dans des fichiers, nous allons pouvoir travailler sur la comparaison entre les données de références que nous avons depuis notre base de données, et sur les fichiers que nous venons de créer. Pour se faire, nous importons les références et nous créons un tableau de comparaison, que nous allons appeler tableau de vérité pour la suite du projet, directement lié à la sensibilité et la spécificité.

2. Sensibilité - Spécificité

La sensibilité est un test de mesure permettant de donner un résultat positif lorsque notre hypothèse est vérifiée. La spécificité, quant à elle, correspond à l'opposé de la sensibilité. En effet, c'est un test de mesure permettant de donner un résultat négatif lorsque notre hypothèse n'est pas vérifiée. L'ensemble des deux configure une appréciation de la validité de notre test. En effet, ce que nous voulons, dans notre cas, c'est avoir une sensibilité au maximum et une spécificité nulle. Ceci correspondrait à alerter le conducteur plus souvent même si il n'était pas vraiment endormi afin de ne pas louper des somnolences probables d'un conducteur. Notre objectif est donc de trouver une spécificité quasi-nulle voire nulle.

Afin de trouver la sensibilité et la spécificité nous devons tout d'abord calculer, en fonction de chaque seuil, les VP (Vrai positif), FP (Faux positif), FN (Faux négatif), et VN (Vrai négatif) afin de remplir le tableau de la manière suivant :

	Endormi	Non endormi
Positif	VP (11)	FP (01)
Négatif	FN (10)	VN (00)

Figure 14 : Tableau de Vérité

- VP → Référence = 1 et Donnée = 1
- FP → Référence = 0 et Donnée = 1
- FN → Référence = 1 et Donnée = 1
- VN → Référence = 0 et Donnée = 0

Pour se faire, nous avons créé une fonction sous Matlab afin de calculer ces 4 composantes que nous pouvons voir ci-dessous. De plus, nous réalisons un moyennage des 36 vidéos pour chaque seuil afin d'avoir une valeur optimale en fonction de toutes les vidéos que nous avons traitées précédemment. Nous avons donc réalisé 30 tableaux comme ci-dessus en fonction des 36 vidéos de notre base de données.

```
function [ vp,fp,fn,vn,specif,sensi,tab] = tableauVerite( inputRef , inputTest)

vp = zeros(31,1);
fp = zeros(31,1);
fn = zeros(31,1);
vn = zeros(31,1);
specif = zeros(31,1);
sensi = zeros(31,1);
seuil = 0.15.* ones(31,1);
for i = 1:1:31

    for j = 1:1:size(inputRef,2)
        if (inputRef(1,j)==1 && inputTest(i,j)==1)
            vp(i)=vp(i)+1;

        elseif (inputRef(1,j)==1 && inputTest(i,j)==0)
            fn(i)=fn(i)+1;

        elseif (inputRef(1,j)==0 && inputTest(i,j)==1)
            fp(i)=fp(i)+1;

        elseif (inputRef(1,j)==0 && inputTest(i,j)==0)
            vn(i)=vn(i)+1;
        end
    end



end

for y = 2:1:31
   seuil(y) = seuil(y-1)+0.005;

end
tab = [seuil vp fp vn fn];
```

Figure 15 : Fonction création Tableau de vérité

La dernière étape est de calculer les Sensibilités et la Spécificités en fonction des 30 tableaux que nous avons créés auparavant, correspondant à chaque seuil. Nous savons que :

$$Sensibilité = \frac{VP}{VP + FN} \qquad Spécificité = \frac{VN}{VN + FP}$$

Figure 16 : Formule Sensibilité et Spécificité

Nous obtenons donc nos 30 valeurs pour chacun des cas. Ces valeurs vont nous permettre de passer à la dernière étape avant la détection du seuil EAR optimal, à savoir la courbe ROC.

3. Courbe ROC

La courbe ROC est une courbe de Sensibilité en fonction de la Spécificité. C'est-à-dire qu'on représente la mesure ROC sous forme d'une courbe qui donne le taux de vrais positifs (Sensibilité) en fonction du taux de faux positifs (1 – Spécificité).

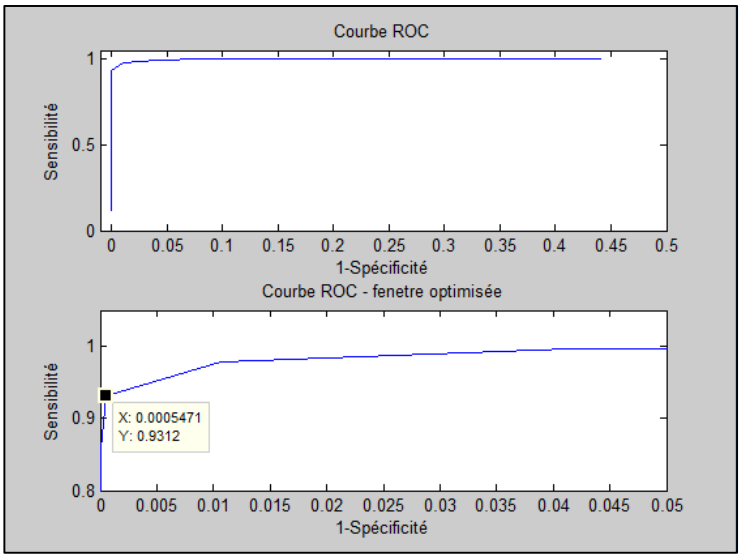


Figure 17 : Courbe ROC & Courbe ROC optimisée

En obtenant cette courbe ROC nous pouvons retourner nous référer au tableau que nous réalisé précédemment sous Matlab afin de trouver la correspondance du seuil en fonction des coordonnées que nous avons trouvées sur la courbe. Nous savons qu'à partir d'une sensibilité = 0,9312 nous pouvons repérer le seuil optimal. Nous trouvons, suite à la lecture du tableau, que le seuil moyen se trouve entre 0,25 et 0,255. Nous avons donc décidé de choisir notre seuil EAR optimal équivalent à 0,25 afin de réaliser notre application.

Seuil	moyVP	moyFP	moyFN	moyVN	Sensibilité	Spécificité
0.2500	866.9167	54.6389	2.9444	1.2655e+03	0.9966	0.9586
0.2550	869.8611	108.6111	0	1.2116e+03	1	0.9177
0.2600	869.8611	158.8056	0	1.1614e+03	1	0.8797

Figure 18 : Courbe ROC & Courbe ROC optimisée

VI. Applications

1. Exécution sur une image

Nous allons réaliser deux applications. La première sera l'exécution de notre projet sur une image. Nous avons testé différentes images du trombinoscope de la promotion 2018/2019. En effet, nous avons utilisé 6 images et nous voulons détecter si les yeux de la personne sont fermés ou non lors de la prise de la photo. Nos sujets sont Adel, Alan, Amirouche, Adrien, Olivier et Tamazgha :

Figure 19 : Application Adel

Figure 20 : Application Alan

Figure 21 : Application Amirouche

Figure 22 : Application Adrien

Figure 23 : Application Olivier

Figure 24 : Application Tamazgha

Nous avons étudié différents cas précédemment, et nous avons un taux de fiabilité de 100%. Nous pouvons donc passer à la deuxième phase d'application, à savoir le traitement d'une vidéo complète.

2. Exécution sur une vidéo

Suite à plusieurs recherches et afin de terminer notre application, nous avons décidé d'alerter le conducteur sur sa somnolence et son endormissement à partir de 20 cycles pour une vidéo réalisé de nuit. On sait qu'une seconde vaut 15 cycles, donc nous alertons le conducteur au bout 1,3 seconde. Pour se faire, nous avons juste eu le temps de créer un affichage d'alerte au niveau de notre invite de commande sous Linux comme ci-dessous :

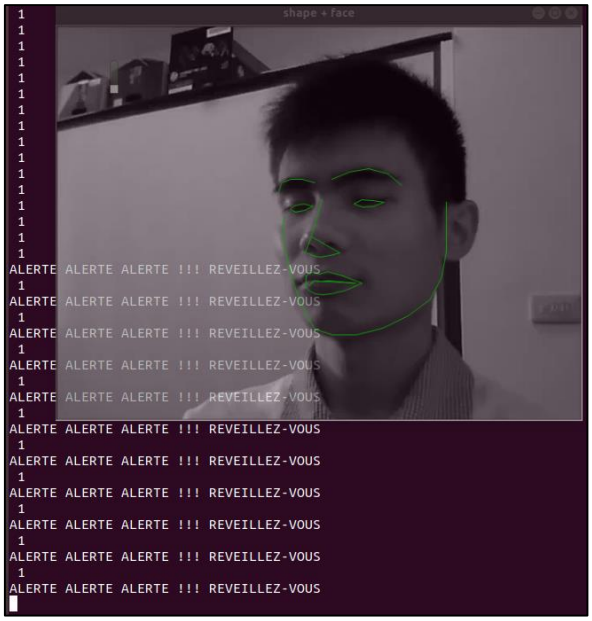


Figure 25 : Application Vidéo – Alerte endormissement

De plus, nous avons stocké les données EAR de cette même vidéo dans un fichier texte afin de voir réellement le taux de somnolence de celui-ci. Sous Matlab, nous avons défini un seuil de 0,245 pour cette vidéo afin d'avoir une approximation du taux d'endormissement.

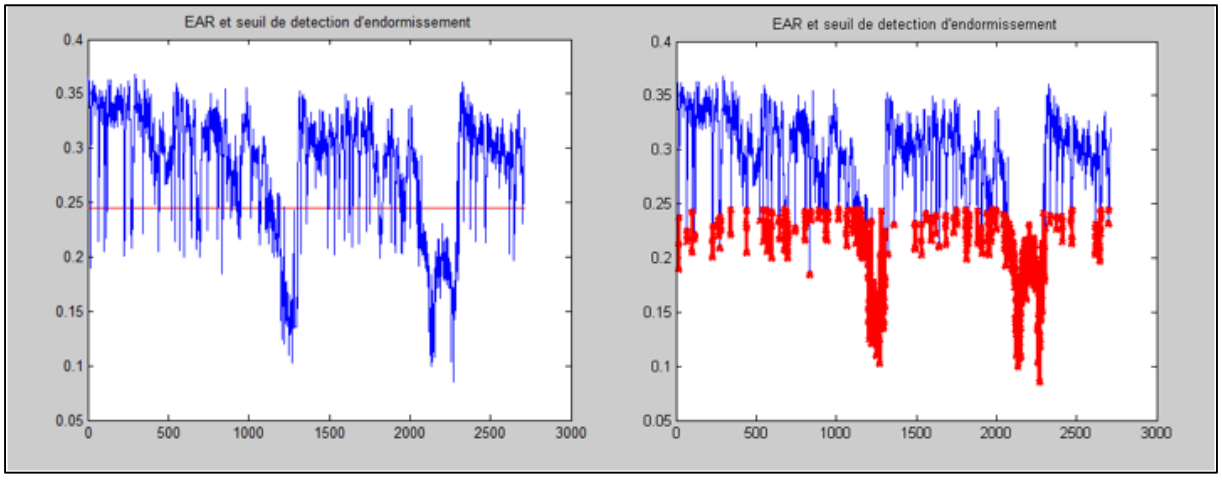


Figure 26 : Application Vidéo – Application du seuil sur une vidéo entière - Matlab

Nous pouvons voir ci-dessus deux pics important en rouge. Ces deux pics correspondent à une longue durée d'endormissement de notre conducteur. En effet, suivant l'axe x correspondant aux nombres d'images de la vidéos et l'axe y à la valeur du seuil EAR pour chaque image, nous repérons deux longues périodes en-dessous du seuil EAR prédéfini auparavant.

VII. Conclusion

Nous pouvons dire, suite à tous les travaux que nous avons effectués que notre projet est arrivé au terme de sa phase théorique, ainsi que de sa phase expérimentale. Nous avons pu générer le code nécessaire à l'utilisation du masque de Kazemi et Sullivan dans le cadre du calcul de l'EAR (Eye Aspect Ratio). Malgré cet avancement, nous constatons que le temps d'application du masque, et le temps de calcul des mesures sur les yeux est trop important pour l'utilisation en temps réel. Après l'analyse, nous en avons déduit que le délai n'était pas issu de notre programme, mais bien du masque utilisé, ainsi que de la puissance de calcul de l'ordinateur utilisé lors de l'obtention des mesures de l'EAR.

Nous pouvons toujours apporter des améliorations au projet DYLART, comme par exemple intégrer un second système permettant de réveiller le conducteur dans un laps de temps assez court pour lui éviter de causer un accident. Nous avons réfléchi à différents types d'alerte. Deux sortes nous sont alors parues envisageables, comme une alerte sonore, suffisamment puissante pour sortir le conducteur de son endormissement naissant, mais avec un volume adéquat pour ne pas le faire avoir peur et provoquer un sursaut, qui conduirait à un écart de trajectoire ou un freinage brusque. Nous avons aussi pensé à des vibrations ou des petites impulsions électriques dans une partie du corps n'aidant pas à la conduite, comme pour l'alerte sonore, pour ne pas provoquer un état de surprise ou un sursaut trop violent du conducteur. Ces améliorations combinées au projet DYLART ont pour but de diminuer le nombre de morts sur la route, non pas de l'augmenter, il est donc important de ne causer aucune peur chez le conducteur et les autres occupants du véhicule.

Nous pouvons aussi modifier les points du modèle de Kazemi et Sullivan que nous utilisons, ou en rajouter dans nos critères, pour détecter un abaissement de la tête, signe d'endormissement ou d'inattention, d'utilisation d'un téléphone portable par exemple. Nous pourrions utiliser les points définissant le menton, ou la bouche pour détecter un bâillement, ou tout autre signe précurseur de l'assoupissement.

Ce projet nous a apporté beaucoup de choses. D'une part, une autonomie quasi constante, même si M. Acosta et M. Beaumont nous ont apporté une aide précieuse, sans laquelle nous n'aurions pas réussi à obtenir un travail fini à ce point dans les délais. Nous avons appris à rechercher une base de données conséquente, à envoyer des demandes officielles pour obtenir des documents auprès d'autres universités, et de réfléchir grandement par nous même pour résoudre tous les problèmes qu'un projet de cette envergure pouvait engendrer. Nous avons acquis de nombreuses autres compétences, comme l'utilisation du langage C++, de l'invite de commande sous Linux, pour compiler et exécuter un code C++ rédigé seulement sur un logiciel de traitement de texte. Nous avons aussi découvert l'utilisation des librairies partagées, dans lesquelles nous avons puisé notre masque de Kazemi et Sullivan, ainsi que les librairies OpenCV et Dlib, sans quoi nous n'aurions pas pu traiter d'images. Ensuite, lors du traitement des données sur Matlab, nous avons appris quel était l'intérêt de créer une courbe ROC, dans l'objectif de choisir au mieux nos paramètres pour privilégier la sensibilité ou la spécificité de notre détection.

Enfin, nous pouvons assurer que travailler sur ce projet, associant le traitement de l'image, la transformation de notre programme pour traiter des vidéos et les mesures théoriques, nous a permis de choisir plus aisément notre orientation en ce qui concerne la deuxième année de Master, en nous dirigeant vers le Master 2 SISEA.

VIII. Annexes

Programmation en C++, partie OpenCV sous Linux

Détection d'un visage sur une image et calcul EAR :

```
#include <stdio.h>
#include <opencv2/opencv.hpp>
#include <dlib/image_processing/frontal_face_detector.h>
#include <dlib/image_processing/render_face_detections.h>
#include <dlib/image_processing.h>
#include <dlib/image_transforms.h>
#include <dlib/gui_widgets.h>
#include <dlib/image_io.h>
#include <iostream>
#include <fstream> // pour l'écriture dans un fichier .txt

using namespace dlib;
using namespace std;

//
------------------------------------------------------------------------------------------
---

double seuilEAR = 0.245;
double EAR = 0;


int main(int argc, char** argv)
{


    if ( argc != 3 )
    {
    std::cout << "usage: DisplayImage <Image_Path>" << std::endl;
        return EXIT_FAILURE;
    }

    cv::Mat img1;
    img1 = cv::imread( argv[1], 1 );


    cout << "Image " << argv[1] << endl;
    array2d<rgb_pixel> img;
    load_image(img, argv[1]);
     if ( !img1.data )
     {
     std::cerr << "No image data \n" << std::endl;
         return -1;
     }

     cv::namedWindow("DisplayImage", cv::WINDOW_NORMAL );
     cv::imshow("DisplayImage", img1);
     cv::resizeWindow("DisplayImage",600,600);

         //Detection du visage
     frontal_face_detector detector = get_frontal_face_detector();

         // Création d'un predicteur permettant d'initaliser les 68 repères faciaux
         grâce au
         // model provenant du fichier  shape_predictor_68_face_landmarks.dat qui a
         été créé par
         // sullivan et kazemi

         shape_predictor sp;
         deserialize(argv[2]) >> sp;
```

```
image_window win, win_faces;

std::vector<rectangle> dets = detector(img);

std::vector<full_object_detection> shapes;
for (unsigned long j = 0; j < dets.size(); ++j)
{
    full_object_detection shape = sp(img, dets[j]);
    cout << "Nombre de repères détectés: "<< shape.num_parts() << endl;


    // Determination des coordonées des points decrivant l'oeil droit
int X1 = shape.part(36)(0);
int X2 = shape.part(37)(0);
int X3 = shape.part(38)(0);
int X4 = shape.part(39)(0);
int X5 = shape.part(40)(0);
int X6 = shape.part(41)(0);

int Y1 = shape.part(36)(1);
int Y2 = shape.part(37)(1);
int Y3 = shape.part(38)(1);
int Y4 = shape.part(39)(1);
int Y5 = shape.part(40)(1);
int Y6 = shape.part(41)(1);



    // Determination des coordonées des points decrivant l'oeil gauche
int X11 = shape.part(42)(0);
int X12 = shape.part(43)(0);
int X13 = shape.part(44)(0);
int X14 = shape.part(45)(0);
int X15 = shape.part(46)(0);
int X16 = shape.part(47)(0);

int Y11 = shape.part(42)(1);
int Y12 = shape.part(43)(1);
int Y13 = shape.part(44)(1);
int Y14 = shape.part(45)(1);
int Y15 = shape.part(46)(1);
int Y16 = shape.part(47)(1);

    // Calcul des composantes de l'EAR
double num1EAR1 = sqrt(pow((X2-X6),2)+pow((Y2-Y6),2));
double num2EAR1 = sqrt(pow((X3-X5),2)+pow((Y3-Y5),2));

double numEAR1 = num1EAR1+num2EAR1;
double denumEAR1 = 2*sqrt(pow((X1-X4),2)+pow((Y1-Y4),2));

double num1EAR2 = sqrt(pow((X12-X16),2)+pow((Y12-Y16),2));
double num2EAR2 = sqrt(pow((X13-X15),2)+pow((Y13-Y15),2));

double numEAR2 = num1EAR2+num2EAR2;
double denumEAR2 = 2*sqrt(pow((X11-X14),2)+pow((Y11-Y14),2));


    // Calcul de l'EAR de chaque oeil
double EAR1 = numEAR1/denumEAR1;
double EAR2 = numEAR2/denumEAR2;
```

```
        cout << "EAR de l'oeil droite: " << EAR1 << endl;
        cout << "EAR de l'oeil gauche: " << EAR2 << endl;


            // Calcul de l'EAR global (moyenne des EAR de chaque oeil)
        EAR = (EAR1+EAR2)/2;

        cout << "EAR: " << EAR << endl;


            // On affiche le shape detector sur la figure
            shapes.push_back(shape);
        }

            // Les yeux sont fermés, alors alert, sinon écriture yeux ouvert

            if (EAR<seuilEAR){
        cout << " La personne est endormie " << endl;
            }
            else{
                cout << " La personne à les yeux ouverts " << endl;
            }



            //Affichage du visage à l'écran avec l'overlay superposé
            win.clear_overlay();
            win.set_image(img);
            win.add_overlay(render_face_detections(shapes));
            win.set_title("shape + face");


        cv::waitKey(0);                    // Attente pression d'une touche pour tout
        fermer

}
```

Traitement vidéo – La personne est-elle endormie ou non ?

```
//
-------------------------------------------------------------------------------------
---


/* arguments dans l'ordre:  0)Nom du programme
 *                          1)Nom de la vidéo
 *                          2)Shape_predictor
 *                          3)Seuil ( on ne le donne pas si on souhaite juste obtenir
 les EAR de chaque vidéo )
 *                          3) ou 4)Nom du fichier .txt
 */

        int compteur = 0;


int main(int argc, char** argv)
{

    double EAR = 0;


    if ( argc != 5 )
    {
        std::cout << "usage: DisplayImage <Image_Path>" << std::endl;
        return EXIT_FAILURE;
    }

    cv::VideoCapture cap(argv[1]); // open the default camera
```

```
if(!cap.isOpened())  // check if we succeeded
{
    std::cerr << "Error: cannot open the default camera" << std::endl;
    return EXIT_FAILURE;
}

ofstream detection(string(argv[3])+".txt",ios::out);

cv::namedWindow("DisplayImage", cv::WINDOW_NORMAL );
cv::resizeWindow("DisplayImage",600,600);


for (int i = 0; i<cap.get(CV_CAP_PROP_FRAME_COUNT);i++){
    cv::Mat img1;
    cap >> img1; // get a new frame from camera

    array2d<bgr_pixel> img;
    assign_image(img, cv_image<bgr_pixel>(img1));


    if ( !img1.data )
    {
        std::cerr << "No image data \n" << std::endl;
        return -1;
    }

    cv::imshow("DisplayImage", img1);

    //Detection du visage
    frontal_face_detector detector = get_frontal_face_detector();

    // Création d'un predicteur permettant d'initaliser les 68 repères faciaux
    grâce au
    // model provenant du fichier  shape_predictor_68_face_landmarks.dat qui a
    été créé par
    // sullivan et kazemi

    shape_predictor sp;
    deserialize(argv[2]) >> sp;

    image_window win, win_faces;

    std::vector<rectangle> dets = detector(img);

    std::vector<full_object_detection> shapes;
    for (unsigned long j = 0; j < dets.size(); ++j)
    {
        full_object_detection shape = sp(img, dets[j]);
        //        cout << "Nombre de repères détectés: "<< shape.num_parts() <<
        endl;


      int X1 = shape.part(36)(0);
      int X2 = shape.part(37)(0);
      int X3 = shape.part(38)(0);
      int X4 = shape.part(39)(0);
      int X5 = shape.part(40)(0);
      int X6 = shape.part(41)(0);

      int Y1 = shape.part(36)(1);
      int Y2 = shape.part(37)(1);
      int Y3 = shape.part(38)(1);
      int Y4 = shape.part(39)(1);
      int Y5 = shape.part(40)(1);
      int Y6 = shape.part(41)(1);
```

```
        // Determination des coordonées des points decrivant l'oeil gauche
        int X11 = shape.part(42)(0);
        int X12 = shape.part(43)(0);
        int X13 = shape.part(44)(0);
        int X14 = shape.part(45)(0);
        int X15 = shape.part(46)(0);
        int X16 = shape.part(47)(0);

        int Y11 = shape.part(42)(1);
        int Y12 = shape.part(43)(1);
        int Y13 = shape.part(44)(1);
        int Y14 = shape.part(45)(1);
        int Y15 = shape.part(46)(1);
        int Y16 = shape.part(47)(1);


        // Calcul des composantes de l'EAR
        double num1EAR1 = sqrt(pow((X2-X6),2)+pow((Y2-Y6),2));
        double num2EAR1 = sqrt(pow((X3-X5),2)+pow((Y3-Y5),2));

        double numEAR1 = num1EAR1+num2EAR1;
        double denumEAR1 = 2*sqrt(pow((X1-X4),2)+pow((Y1-Y4),2));

        double num1EAR2 = sqrt(pow((X12-X16),2)+pow((Y12-Y16),2));
        double num2EAR2 = sqrt(pow((X13-X15),2)+pow((Y13-Y15),2));

        double numEAR2 = num1EAR2+num2EAR2;
        double denumEAR2 = 2*sqrt(pow((X11-X14),2)+pow((Y11-Y14),2));


        // Calcul de l'EAR de chaque oeil
        double EAR1 = numEAR1/denumEAR1;
        double EAR2 = numEAR2/denumEAR2;



        // Calcul de l'EAR global (moyenne des EAR de chaque oeil)
        EAR = (EAR1+EAR2)/2;

        // On affiche le shape detector sur la figure
        shapes.push_back(shape);
    }


    //Affichage du visage à l'écran avec l'overlay superposé

    win.clear_overlay();
    win.set_image(img);
    win.add_overlay(render_face_detections(shapes));
    win.set_title("shape + face");


    double seuilEAR = atof(argv[3]);

    if (EAR<seuilEAR){
        cout << " 1 " << endl;
        compteur++;
        if (compteur > 20){
            cout << "ALERTE ALERTE ALERTE !!! REVEILLEZ-VOUS" << endl;
        }}
        else{
            cout << "0 " << endl;
            compteur =0;
        }
```

```
      // ALERTE ALERTE ne marche que pour ubuntu 18.04.1 !!!
      int k; // key
        k=cv::waitKey(30);
        if( 255!=k && -1!=k ) {
            break;
        }
    }

    detection.close();

    return EXIT_SUCCESS;
}
```

Traitement vidéo – Modification pour calcul de l'EAR et stockage des données dans un fichier.txt :

```
    if ( argc != 4 )
    {
        std::cout << "usage: DisplayImage <Image_Path>" << std::endl;
        return EXIT_FAILURE;
    }
```

---------- Même programme que précédemment ----------

```
       //Affichage du visage à l'écran avec l'overlay superposé

                win.clear_overlay();
                win.set_image(img);
                win.add_overlay(render_face_detections(shapes));
                win.set_title("shape + face");


      // double seuilEAR = atof(argv[3]);

       if(detection){
           /*
           if (EAR<seuilEAR){
               detection<<"1";
           }
           else{
               detection<<"0";
           }*/

           detection<<EAR<<endl; // Ecriture des différents EAR de chaque image de
           la vidéo dans un fichier texte
        }


        // ALERTE ALERTE ne marche que pour ubuntu 18.04.1 !!!
        int k; // key
        k=cv::waitKey(30);
        if( 255!=k && -1!=k ) {
            break;
        }
    }

    detection.close();

    return EXIT_SUCCESS;
}
```

Programmation Matlab, partie mesure du seuil optimal

```
%                 Calcul Moyenne de la sensibilite et specificite pour chaque seuil de detection
      sensibilite(ii) = moyVP(ii)/(moyVP(ii)+moyFN(ii));
      specificite(ii) = moyVN(ii)/(moyVN(ii)+moyFP(ii));
```

```
%                 Calcul Moyenne de VN pour chaque seuil de detection
      moyvnj = verite1(ii,4)+verite2(ii,4)+verite5(ii,4)+verite6(ii,4)+verite8(ii,4)+verite9(ii,4)
               +verite12(ii,4)+verite13(ii,4)+verite15(ii,4)+verite20(ii,4)+verite23(ii,4)+verite24(ii,4)
               +verite31(ii,4)+verite32(ii,4)+verite33(ii,4)+verite34(ii,4)+verite35(ii,4)+verite36(ii,4);

      moyvnn = verite1n(ii,4)+verite2n(ii,4)+verite5n(ii,4)+verite6n(ii,4)+verite8n(ii,4)+verite9n(ii,4)
               +verite12n(ii,4)+verite13n(ii,4)+verite15n(ii,4)+verite20n(ii,4)+verite23n(ii,4)+verite24n(ii,4)
               +verite31n(ii,4)+verite32n(ii,4)+verite33n(ii,4)+verite34n(ii,4)+verite35n(ii,4)+verite36n(ii,4);

      moyVN(ii)=(moyvnj+moyvnn)/36;

%                 Calcul Moyenne de FN pour chaque seuil de detection
      moyfnj = verite1(ii,5)+verite2(ii,5)+verite5(ii,5)+verite6(ii,5)+verite8(ii,5)+verite9(ii,5)
               +verite12(ii,5)+verite13(ii,5)+verite15(ii,5)+verite20(ii,5)+verite23(ii,5)+verite24(ii,5)
               +verite31(ii,5)+verite32(ii,5)+verite33(ii,5)+verite34(ii,5)+verite35(ii,5)+verite36(ii,5);

      moyfnn = verite1n(ii,5)+verite2n(ii,5)+verite5n(ii,5)+verite6n(ii,5)+verite8n(ii,5)+verite9n(ii,5)
               +verite12n(ii,5)+verite13n(ii,5)+verite15n(ii,5)+verite20n(ii,5)+verite23n(ii,5)+verite24n(ii,5)
               +verite31n(ii,5)+verite32n(ii,5)+verite33n(ii,5)+verite34n(ii,5)+verite35n(ii,5)+verite36n(ii,5);

      moyFN(ii)=(moyfnj+moyfnn)/36;
```

```
      %-------------calcul des moyennes de chaques composantes (VP-VN-FP-FN)----------------------------

%                 Calcul Moyenne de VP pour chaque seuil de detection
      moyvpj = verite1(ii,2)+verite2(ii,2)+verite5(ii,2)+verite6(ii,2)+verite8(ii,2)+verite9(ii,2)
               +verite12(ii,2)+verite13(ii,2)+verite15(ii,2)+verite20(ii,2)+verite23(ii,2)+verite24(ii,2)
               +verite31(ii,2)+verite32(ii,2)+verite33(ii,2)+verite34(ii,2)+verite35(ii,2)+verite36(ii,2);

      moyvpn = verite1n(ii,2)+verite2n(ii,2)+verite5n(ii,2)+verite6n(ii,2)+verite8n(ii,2)+verite9n(ii,2)
               +verite12n(ii,2)+verite13n(ii,2)+verite15n(ii,2)+verite20n(ii,2)+verite23n(ii,2)+verite24n(ii,2)
               +verite31n(ii,2)+verite32n(ii,2)+verite33n(ii,2)+verite34n(ii,2)+verite35n(ii,2)+verite36n(ii,2);

      moyVP(ii)=(moyvpj+moyvpn)/36;

%                 Calcul Moyenne de FP pour chaque seuil de detection
      moyfpj = verite1(ii,3)+verite2(ii,3)+verite5(ii,3)+verite6(ii,3)+verite8(ii,3)+verite9(ii,3)
               +verite12(ii,3)+verite13(ii,3)+verite15(ii,3)+verite20(ii,3)+verite23(ii,3)+verite24(ii,3)
               +verite31(ii,3)+verite32(ii,3)+verite33(ii,3)+verite34(ii,3)+verite35(ii,3)+verite36(ii,3);

      moyfpn = verite1n(ii,3)+verite2n(ii,3)+verite5n(ii,3)+verite6n(ii,3)+verite8n(ii,3)+verite9n(ii,3)
               +verite12n(ii,3)+verite13n(ii,3)+verite15n(ii,3)+verite20n(ii,3)+verite23n(ii,3)+verite24n(ii,3)
               +verite31n(ii,3)+verite32n(ii,3)+verite33n(ii,3)+verite34n(ii,3)+verite35n(ii,3)+verite36n(ii,3);

      moyFP(ii)=(moyfpj+moyfpn)/36;
```

Fonction Yeux fermés en fonction du seuil :

```
function  [outputtab] = closed_eyes_bin( inputfile, seuil )

    val = 0;

for i=1:1:size(inputfile)
    val = 0;
    if inputfile(i) < seuil
        val = 1;

    else
        val = 0;
    end
    outputtab(i)= val;

end
```

Fonction de création du tableau de Sensibilité/Spécificité :

```
function [ vp,fp,fn,vn,specif,sensi,tab] = tableauVerite( inputRef , inputTest)

vp = zeros(31,1);
fp = zeros(31,1);
fn = zeros(31,1);
vn = zeros(31,1);
specif = zeros(31,1);
sensi = zeros(31,1);
seuil = 0.15.* ones(31,1);
for i = 1:1:31

    for j = 1:1:size(inputRef,2)
        if (inputRef(1,j)==1 && inputTest(i,j)==1)
            vp(i)=vp(i)+1;

        elseif (inputRef(1,j)==1 && inputTest(i,j)==0)
            fn(i)=fn(i)+1;

        elseif (inputRef(1,j)==0 && inputTest(i,j)==1)
            fp(i)=fp(i)+1;

        elseif (inputRef(1,j)==0 && inputTest(i,j)==0)
            vn(i)=vn(i)+1;
        end
    end

    sensi(i)= vp(i)/(vp(i)+fn(i));
    specif(i)= vn(i)/(vn(i)+fp(i));
end

for y = 2:1:31
   seuil(y) = seuil(y-1)+0.005;

end
tab = [seuil vp fp vn fn];
```

Tableau Sensibilité/Spécificité en fonction de chaque seuil :

Seuil	moyVP	moyFP	moyFN	moyVN	Sensibilité	Spécificité
0.1500	103.4444	0	766.4167	1.3202e+03	0.1189	1
0.1550	122.9722	0	746.8889	1.3202e+03	0.1414	1
0.1600	148.0556	0	721.8056	1.3202e+03	0.1702	1
0.1650	171.7778	0	698.0833	1.3202e+03	0.1975	1
0.1700	198.4722	0	671.3889	1.3202e+03	0.2282	1
0.1750	225.7500	0	644.1111	1.3202e+03	0.2595	1
0.1800	260	0	609.8611	1.3202e+03	0.2989	1
0.1850	294.6111	0	575.2500	1.3202e+03	0.3387	1
0.1900	329.9444	0	539.9167	1.3202e+03	0.3793	1
0.1950	367.0833	0	502.7778	1.3202e+03	0.4220	1
0.2000	415.1667	0	454.6944	1.3202e+03	0.4773	1
0.2050	454.9167	0	414.9444	1.3202e+03	0.5230	1
0.2100	495.0278	0	374.8333	1.3202e+03	0.5691	1
0.2150	539.8333	0	330.0278	1.3202e+03	0.6206	1
0.2200	591.5556	0	278.3056	1.3202e+03	0.6801	1
0.2250	642.5278	0	227.3333	1.3202e+03	0.7387	1
0.2300	691	0	178.8611	1.3202e+03	0.7944	1
0.2350	746.0278	0	123.8333	1.3202e+03	0.8576	1
0.2400	810.0556	0.7222	59.8056	1.3194e+03	0.9312	0.9995
0.2450	851.5278	14.0278	18.3333	1.3061e+03	0.9789	0.9894
0.2500	866.9167	54.6389	2.9444	1.2655e+03	0.9966	0.9586
0.2550	869.8611	108.6111	0	1.2116e+03	1	0.9177
0.2600	869.8611	158.8056	0	1.1614e+03	1	0.8797
0.2650	869.8611	211.2778	0	1.1089e+03	1	0.8400
0.2700	869.8611	266.8611	0	1.0533e+03	1	0.7979
0.2750	869.8611	329.1111	0	991.0556	1	0.7507
0.2800	869.8611	381.5833	0	938.5833	1	0.7110
0.2850	869.8611	430.0556	0	890.1111	1	0.6742
0.2900	869.8611	479.6667	0	840.5000	1	0.6367
0.2950	869.8611	530.7500	0	789.4167	1	0.5980
0.3000	869.8611	583.6111	0	736.5556	1	0.5579

IX. Bibliographie

Base de données :

- Ching-Hua Weng, Ying-Hsiu Lai, Shang-Hong Lai, “Driver Drowsiness Detection via a Hierarchical Temporal Deep Belief Network”, In Asian Conference on Computer Vision Workshop on Driver Drowsiness Detection from Video, Taipei, Taiwan, Nov. 2016.

Introduction :

- https://www.preventionroutiere.asso.fr/2016/03/30/somnolence-au-volant/
- L’accidentalité routière en 2017, bilan sommaire de l’ONISR
- Accidentalité routière 2017, résultats définitifs de l’ONISR

Projet – Détection d’endormissement

- https://pymotion.com/detecter-facial-landmarks/
- http://veniceatlas.epfl.ch/atlas/gis-and-databases/objects/recognizing-copies-of-paintings-1
- One millisecond face alignment with an ensemble of regression trees de Kazemi et Sullivan (2014)
- Driver Distraction Using Visual-Based Sensors and Algorithms de Alberto Fernandez, Ruben Usamentiaga, Juan Luis Carus et Ruben Casado.
- Robust and Continuous Estimation of Driver Gaze Zone by Dynamic Analysis of Multiple Face Videos de Ashish Tawari et Mohan M. Trivedi
- Real-time SVM Classification for Drowsiness Detection Using Eye Aspect Ratio de Caio B. Souto Maior, Marcio C. Moura, Joao M. M. de Santana, Lucas M. do Nascimento, July B. Macedo, Isis D. Lins et Enrique L. Droguett.