Projet de Découverte de la Recherche
Suivi de visage et détection de saillances

Julien NOWAK, Stéphane RIMLINGER
Élèves ingénieurs
TELECOM NANCY
julien.nowak@telecomnancy.eu, stephane.rimlinger@telecomnancy.eu

Alain DUTECH, Amine BOUMAZA,Yann BONIFACE
Chargés de recherche
INRIA
alain.dutech@loria.fr, amine.boumaza@loria.fr, yann.boniface@loria.fr

***Résumé*—Dans le but d'étudier les interactions non-verbales entre l'homme et la machine, nous avons développé un programme permettant de détecter sur un flux d'images, les visages des personnes présentes et leurs saillances, c'est à dire les éléments distinctifs du visage tels que les yeux, la bouche ou encore les sourcils.**

Ce programme sera par la suite implanté sur une lampe robotisée équipée d'une caméra qui analysera le comportement des sujets lors des mouvements de cette dernière.

Mots clés : Détection de visage ; Reconnaissance de saillances ; Python ; OpenCV ; Webcam ; Dlib ; Imutils ; Lampe robotisée ; Étude du comportement ;

I. Introduction

A. Contexte

De nos jour, la détection de visage est de plus en plus utilisée dans des domaines très variés. On peut par exemple citer de nombreuses applications mobiles comme Facebook, Snapchat ou Instagram qui permettent de détecter en temps réel le visage de l'utilisateur afin d'y appliquer des filtres ludiques. La détection de visage peut également servir à déverrouiller son téléphone portable ou encore à déterminer si le conducteur d'un véhicule est entrain de s'assoupir. Le monde de la photographie aussi s'est emparé de ces technologies afin d'implémenter des systèmes de mise au point automatique sur les visages des sujets pris en photo. Les applications sont vraiment nombreuses et dans des domaines très variés comme la sécurité, la publicité, l'automobile, les réseaux sociaux ou encore la photographie. Il y a donc un réel enjeu commercial autour de ces outils de détection de visage pour les années à venir.

B. Problématique

Dans le cadre d'une étude sur les interactions non-verbales entre les hommes et les machines, en l'occurrence une lampe robotisée équipée d'une caméra, ce projet avait pour but d'améliorer les logiciels existants permettant de détecter et d'observer le visage de la personne se trouvant face à la lampe robotisée. Afin de pouvoir, a terme, faire réagir la lampe en fonction du comportement du sujet face à la caméra. Pour ce faire, il fallait donc mettre en place un système de détection rapide et précis permettant de détecter le visage d'une ou plusieurs personnes en temps réel et de façon suffisamment précise afin d'obtenir un maximum d'informations concernant le visage des sujets.

C. Contributions

Afin de répondre à cette problématique, nous avons décidé d'utiliser une nouvelle librairie Python pour d'obtenir une détection de visage plus efficace que la version précédente du programme. En effet, la librairie OpenCV précédemment utilisée avait montré ses limites avec une détection hasardeuse et peu évolutive. Nous avons donc choisi d'utiliser les librairies "Dlib" et "Imutils" que nous avons trouvé très complètes. Nous avons ensuite testé la robustesse de notre détection avant de se pencher sur l'interprétation des visages détectés.

D. Plan de l'article

Dans la suite de cet article, nous allons tout d'abord vous présenter une étude de l'existant puis aborder la méthodologie mise en place dans le but de répondre à la problématique et enfin présenter les résultats obtenus.

II. État de l'art

La détection de visage par ordinateur est un sujet qui intéresse depuis longtemps. Les premiers essais remontent aux années 70. Limités par les technologies de l'époque, ces essais étaient basés sur des méthodes heuristiques très peu robustes. Les conditions au niveau de la prise de vue devaient être optimales : visage de face, bien éclairé. Dans les années 90, les gains en puissance de calcul permettent d'accélérer grandement le procédé et se s'appuyer sur des heuristiques moins strictes. Le procédé reste néanmoins trop imprécis.

Une grande évolution dans le domaine a lieu en 2001 avec la publication de la méthode de Viola et Jones, du nom de ses auteurs, Paul Viola et Michael Jones. Cette méthode consiste à parcourir l'intégralité de l'image pour extraire un certain nombre de caractéristiques. Ces caractéristiques sont calculées en faisant la différence de la somme des pixels de deux ou plusieurs zones (figure 1).

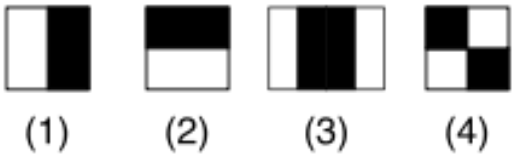

FIGURE 1: Exemple de caractéristiques utilisées par Viola et Jones

Ces caractéristiques sont "glissées" sur toute l'image jusqu'à ce que la différence de valeur entre les pixels de la zone blanche et les pixels de la zone noire soient suffisamment importantes pour qu'une caractéristique soit retenue. Par exemple, si la caractéristiques composée de deux rectangles verticaux (1) de la figure 1 montre une grande différence de valeurs entre les deux rectangles, cela signifie que la caractéristique à détecté le bord gauche ou droit d'un objet de l'image. Chaque caractéristique permet de détecter une information différente sur l'objet présent sur l'image.

La méthode combine donc ces caractéristiques pour détecter les propriétés qui se retrouvent sur toute image d'un visage vu de face, illuminé d'un éclairage naturel. Par exemple, le bout du nez sera plus clair que ses bords, une caractéristique composée de trois rectangles verticaux doit donc détecter, en passant sur la zone du nez, que le centre est plus clair que les bords (figure 2). De même, la région des yeux est plus sombre que la partie du visage juste en dessous, composé du nez et des joues. Une caractéristique composée de deux rectangles horizontaux permet donc de détecter cette propriété du visage (figure 3). Si toutes les propriétés du visages ont été reconnues avec les caractéristiques correspondantes, l'algorithme renvoie les coordonnées de ce visage dans l'image.

FIGURE 2: Détection du nez par une des caractéristiques

FIGURE 3: Détection des yeux par une des caractéristiques

L'algorithme de Viola et Jones est implémenté dans de nombreuses librairies de détection de visages et notamment OpenCV qui sera utilisé pour le projet.

La détection de visage n'est toutefois qu'une partie de la problématique. Une fois le visage détecté, il faut en extraire les saillances de façon précise. L'utilisation des caractéristiques rectangulaires de la méthode précédente permet de détecter les yeux ou la bouche sur un visage, et cette fonction est disponible dans OpenCV. Cette méthode est cependant trop imprécise pour cette application car elle ne renvoie qu'un rectangle entourant les yeux ou la bouche. De nombreux faux-positifs sont aussi à déplorer.

Depuis les années 2000 on peut noter une amélioration et un engouement considérable pour les méthodes d'apprentissage profond. Aussi appelées deep-learning, ces technologies reposent sur l'apprentissage, par un ordinateur, d'un modèle de données servant à remplir une tâche. Pour ce faire, un réseau de neurones est modélisé dans l'ordinateur. Celui-ci reçoit alors des images choisies au préalable, dans notre cas il s'agirait de visages, et doit en extraire les saillances. S'il n'y parvient pas, les paramètres de connexions entre les différents

neurones sont modifiés et on réitère l'opération. Si un résultat satisfaisant est observé, les connexions utilisées sont renforcées. Le résultat final est une masse de données plus ou moins compréhensible par un humain, mais permettant à n'importe quel ordinateur de remplir cette tâche.

Un certain nombre de bases de données permettant de détecter les saillances d'un visage ont ainsi été réalisées au cours de la dernière décennie. On note par exemple la base de données HELEN, composée de 2330 images de visages, qui place 194 points sur la saillances des visages détectés (figure 4). Cette détection détaillée des saillances est cependant parfois imprécise. A l'inverse, une autre base de données nommée AFLW est composée de près de 26000 visages, aux expressions variées. Seuls 21 points de saillances sont cependant placés, trop peu pour notre application (figure 5).

FIGURE 4: Points de saillances avec la méthode HELEN

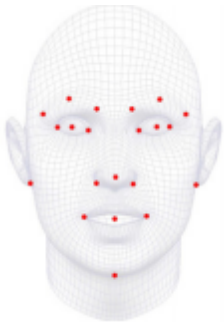

FIGURE 5: Points de saillances avec la méthode AFLW

Dans l'optique de lier les différentes bases de données et de proposer un compromis entre le nombre de points de saillances et la précision de la détection, la base de données 300-W a été mise au point. Celle-ci utilise les images de 8 bases de données de reconnaissance de visages existantes, dont les deux mentionnées précédemment. Cela permet de détecter les visages avec un angle compris entre -45° et 45°, de nombreuses expressions faciales et place 68 points de saillances (figure 6).

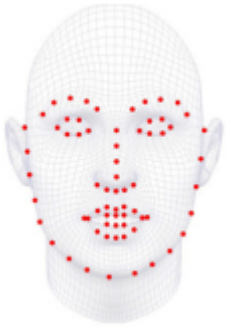

FIGURE 6: Points de saillances placés avec la méthode 300-W

III. MÉTHODOLOGIE

A. *Analyse du problème*

La tâche consistait donc en la création d'un programme python capable de détecter les saillances d'un visage, c'est à dire les caractéristiques marquantes d'un visage telles que les yeux, les sourcils, la bouche ou le nez. Ce programme doit aussi s'inscrire dans l'optique de la lampe autonome et doit donc tenir comptes des caractéristiques particulières de la situation, telles que la résolution de la caméra intégrée à la lampe, ou la puissance de calcul limitée de l'ordinateur qui l'accompagne. Le programme doit aussi être compatible avec Linux.

La tâche de détection des saillances en elle-même est toutefois assez indépendante du reste du programme de la lampe. Cela nous permet de répondre à la problématique dans un programme séparé du code principal de la lampe, nous épargnant des soucis éventuels de compréhension du code pré-existant, ou de compatibilité, puisque nous n'avions pas accès à ladite lampe durant ce projet. La création d'une base de données de détection de saillances étant un travail considérable et impossible à réaliser dans le cadre de ce projet, nous utiliseront une librairie existante pour les détections.

Cette inaccessibilité de la lampe durant la conception du programme nous a obligé à le rendre le plus efficient possible, dans les limites de nos capacités, de sorte qu'il soit capable de fonctionner quelque soit le matériel utilisé.

B. *Solutions proposées*

Le choix principal a été celui des librairies à utiliser pour la détection de visages et de saillances. La détection de visages était déjà présente dans le programme de la lampe et utilisait la librairie OpenCV.

La comparaison avec d'autres librairies de détection de visages ne présentait pas de différences quant au temps de détection, nous avons donc conservé cette librairie afin de limiter le nombre d'imports à effectuer.

Concernant le choix de la base de données de détection de saillances. La plus complète et la plus utilisée pour le détection de visages en python s'est trouvé être la 300-W évoquée précédemment. Le code utilisant cette base de données étant en C++, il nous a fallu installer la librairie Dlib pour lier les deux langages. La dernière librairie importée nommée imutils comporte des fonctions pour faciliter la liaison entre Dlib et OpenCV. Des fonctions permettant l'affichage des points de saillances en utilisant des couleurs différents pour chaque élément (oeil, bouche, sourcil, ...) s'y trouvent aussi et nous ont aidés lors de la prise en main de la détection en début de projet.

C. Matériel

Le programme devant être compatible avec Linux, nous avons tous deux mis en place une machine virtuelle sur nos ordinateurs portables personnels ayant Windows comme système d'exploitation intégré. Cette solution est plus agréable à l'utilisation qu'un dual boot puisque cela nous permet de garder les applications Windows ouvertes durant notre avancement. La phase de mise en place de cette machine virtuelle a cependant été quelque peu laborieuse, notamment pour la détection de la webcam intégrée à l'ordinateur portable. Nous avons également utilisé l'outil de gestion de version GIT. En effet, le projet ayant déjà été commencé lors des précédentes années par d'autres élèves de TELECOM Nancy, une base de projet était déjà présente sur la plateforme GIT du Loria. Cet outil très pratique nous à permis de développer l'application en parallèle sans se soucier des différents conflits qui auraient pu être créés sans un outil comme celui-ci.

Enfin, l'application doit, à terme, servir d'interface de contrôle pour une lampe robotisée équipée d'une caméra. Cette caméra remplacera simplement la Webcam de l'ordinateur que nous avons utilisé.

Comme le montre la figure 7, il s'agit d'une lampe équipée de 3 petits moteurs électrique (en noir sur l'image) commandés à distance qui vont permettre aux chercheurs de bouger la lampe dans n'importe quel sens.

FIGURE 7: Lampe robotisée

D. Conception et implémentation

Plusieurs versions du programme ont été crées, afin de pouvoir prendre une vidéo pré-enregistrée en entrée, un photo, ou un flux vidéo en temps réel. Chacun d'entre eux possède cependant la même routine. L'image est dans un premier temps lue sur l'entrée (figure 8). Elle est ensuite redimensionnée selon une valeur prise en argument (ou 400 pixels de large si aucune valeur n'est précisée).

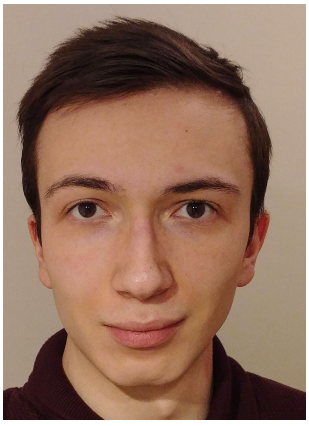

FIGURE 8: Image d'origine, extraite un flux vidéo

Une copie de l'image est crée et celle-ci est passée en noir et blanc, ceci afin de réduire la quantité d'information à traiter. Le détecteur de visages n'a

pas besoin de connaître les couleurs, mais seulement le niveau de gris de chaque pixel. L'étape suivante consiste donc à appeler la fonction de détection de visage sur cette image. Si un ou plusieurs visages ont été détectés, les coordonnées des rectangles englobants chacun des visages seront renvoyées et pourront être tracées (en bleu sur la figure 9). Si aucun visage n'est détecté, le programme passe à l'image suivante et la boucle recommence.

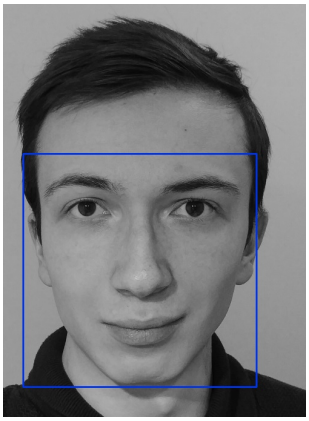

FIGURE 9: Image dont les bords du visage détecté ont été tracés

Sur chacun des visages détectés, la fonction utilisant la base de données de détection de saillances va être appelée. Celle-ci va renvoyer les coordonnées des 68 points du modèle 300-W. Ces coordonnées sont alors utilisables à souhait. Chaque point est en effet identifié par un numéro entre 0 et 67, qui correspond à son indice dans le tableau de coordonnées renvoyé par la détection de saillances. Ces points sont alors placés (figure 10) ou reliés selon l'élément auquel il correspondent (figure 11) sur une image transparente vierge de taille identique à celle redimensionnée.

Ce masque de points est ensuite superposé à l'image en couleurs initialement copiée pour obtenir l'image finale qui sera affichée par le programme (figure 12). Le programme passe ensuite à l'image suivante et réitère le procédé, ou s'arrête si aucune autre image n'est disponible (fin d'une vidéo par exemple).

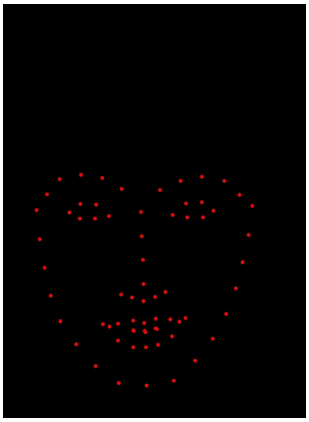

FIGURE 10: Position des 68 points de saillances détectés

FIGURE 11: Les points correspondants à un même élément ont été reliées entre eux

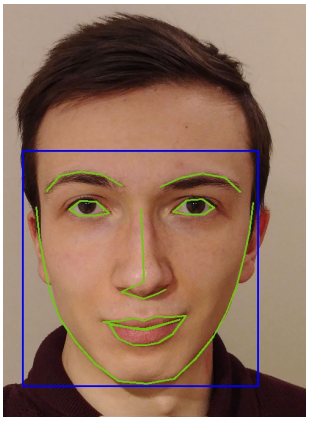

FIGURE 12: Image finale, les points de saillances ont été superposés sur l'image couleur

IV. Résultats

A. La détection

Ce projet nous a permis de développer une application de détection de visage et de saillances très performante, bien plus performante que la version en place avant notre arrivée. En effet, dans la version précédente, l'application avait du mal à détecter correctement les saillances.

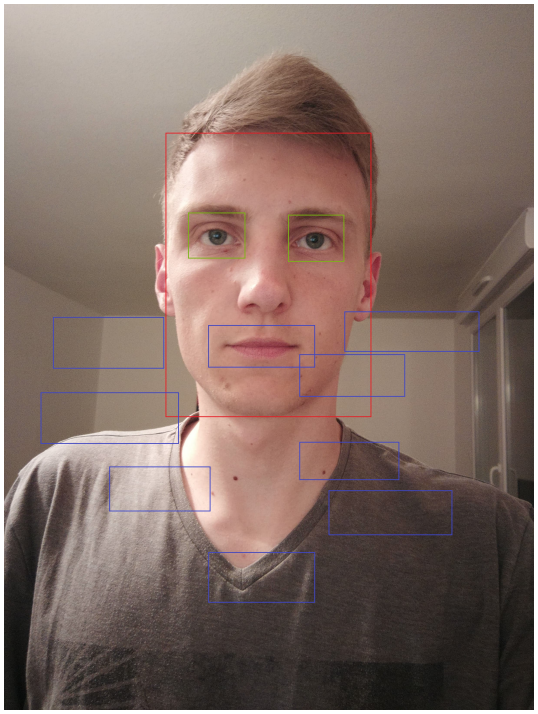

Figure 13: Ancienne version de la détection

On peut voir sur la figure 13 que dans des conditions de luminosité où le visage est plutôt bien éclairé, l'application ne permettait pas une bonne détection efficace. En effet, la librairie OpenCV précédemment utilisée permettait d'afficher un cadre autour des différentes saillances du sujet. le visage et les yeux étaient plutôt bien détectés mais ce n'était pas le cas de la bouche qu'il était impossible de détecter correctement. De plus, on ne pouvait absolument pas avoir d'informations sur les saillances à part leur position.

La nouvelle version de l'application est bien plus complète. Sur la figure 14, on peut voir que la nouvelle méthode de détection est bien plus précise et ce même avec une qualité de caméra et d'éclairage bien moindre (cf. partie Robustesse). En effet, les nouvelles libraires utilisées permettent d'avoir un affichage précis des saillances en temps réel. Les contours du visage, des yeux, de la bouche sont parfaitement détectés et même le nez et les sourcils sont désormais identifiables. De plus, cette application va permettre de suivre le visage du sujet en temps réel et de façon très précise.

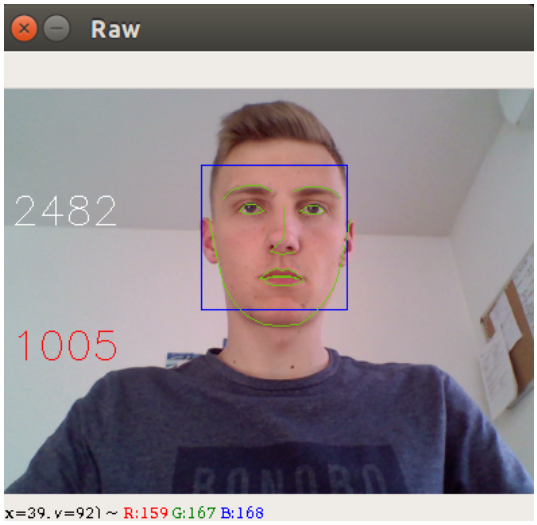

Figure 14: Dernière version

B. Performances

Qui dit détection en temps réel dit performances élevées. En effet, il faut que le programme soit capable de calculer les coordonnées de chacun des points de la détection avant que l'image suivante n'arrive du flux vidéo. Nous avons donc effectué des tests de performances afin de vérifier qu'il n'y avait pas de perte d'informations. Voici les résultats obtenus, en ordres de grandeur :

Action	Temps de traitement
Lecture de l'image	10ms
Redimensionnement de l'image	10ms
Suppression de couleurs	0,1ms
Détection des visages	100ms
Détection des saillances des visages	1ms
Conversion des points de saillances en coordonnées python	1ms

La détection de visages est donc le processus le plus coûteux dans le programme. Source de

nombreux ralentissements lors des premières version du programme. La mise en place d'un thread ayant la seule tâche d'effectuer la détection du visage a été réalisée. En parallèle, le reste du programme effectue la détection de saillances et l'affichage de l'image précédente.

Un des choix ayant un gros impact sur les performances est la taille du redimensionnement de l'image. En effet, la fonction de détection de visages doit parcourir l'intégralité de l'image de façon précise. Plus l'image comporte de pixels, plus celle-ci sera longue à parcourir. Ce paramètre est modifiable facilement dans le programme final puisqu'il s'agit d'un simple argument du programme exécutable. A l'origine, l'image capturée par la webcam était redimensionnée à 400 pixels de large (soit 90 000 pixels en tout puisque l'image est en 16/9). Des tests sur le gain de performance selon la largeur de l'image (figure 15) montrent que la diminution de la largeur entraîne un gain de performance considérable de plus de 50% si l'on diminue de moitié la largeur de l'image, le gain est encore supérieur si l'on passe à 150 pixels de large.

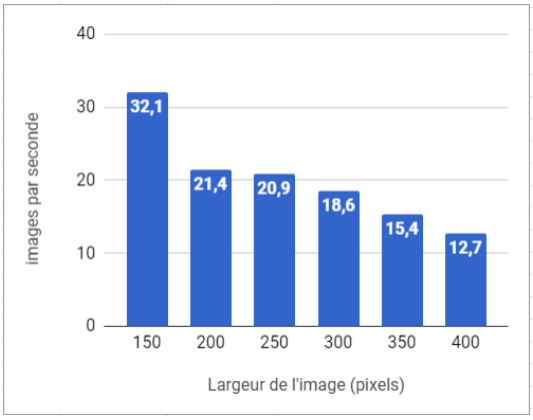

FIGURE 15: Impact de la largeur de l'image sur le nombre d'images traitées par seconde

Cependant, il faut aussi prendre en compte la perte de qualité liée à ce rétrécissement de l'image. Les tests sur l'impact du redimensionnement sur le taux de détection du visage (figure 16) indiquent qu'à partir de 150 pixels la perte de qualité est trop grande pour que la détection se fasse correctement. Cela explique aussi le gain de performance considérable : les images où le visage n'est pas détecté ne passent pas par la détection de saillances et gagnent donc en temps de calcul.

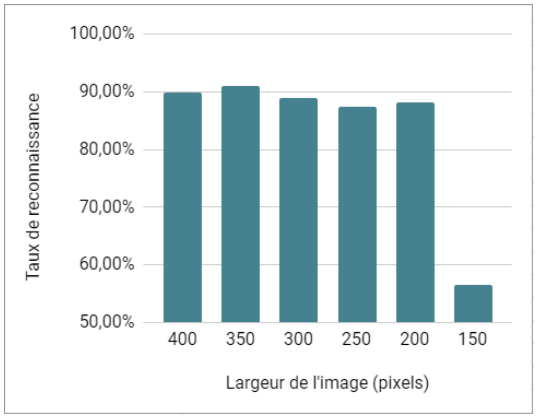

FIGURE 16: Impact de la largeur de l'image sur le taux de détection du visage

Les mesures ont été réalisés en prenant la même vidéo en entrée. Celle-ci qui présente un visage de face, à une distance d'environ 40 centimètres de l'écran avec des changements d'expression et des rotations de la tête. L'ordinateur de test possédait un processeurs 4 coeurs à 2.5 GHz. Une machine au processeur moins performant observera un nombre d'images par secondes moindre, et inversement.

C. Robustesse

Nous avons également testé la robustesse de notre programme. C'est à dire que nous avons fait tourné notre programme dans différentes conditions qui peuvent très bien se reproduire en condition réelles afin de déterminer précisément la plage d'utilisation de notre algorithme. Plusieurs facteurs ont été étudiés comme les conditions de luminosité, l'angle de rotation de la caméra, l'angle de rotation du sujet etc... Voici les résultats obtenus :

1) Impact de la lumière: Dans un premier temps, nous avons testé l'impact des conditions de luminosité sur notre détection de visage :

- En extérieur, avec la lumière du soleil face au sujet (luminosité $\approx 30000lux$), la détection se fait parfaitement.

- En extérieur, avec la lumière du soleil dans le dos du sujet, la détection à du mal à se faire dans les zones très claires de l'image où la mise au point n'est pas faite correctement à cause du contre-jour (Figure 17).

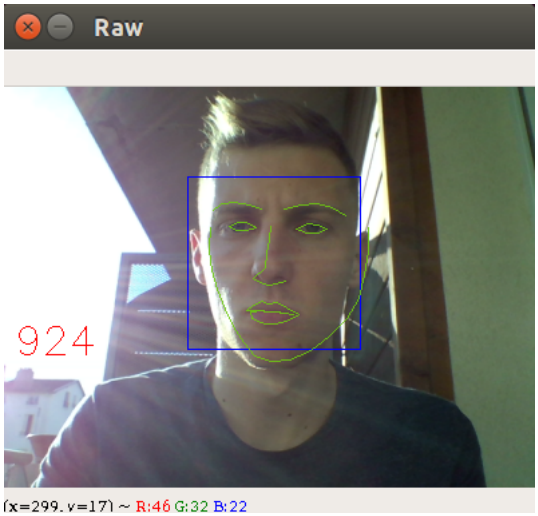

FIGURE 17: Détection en extérieur avec contre-jour

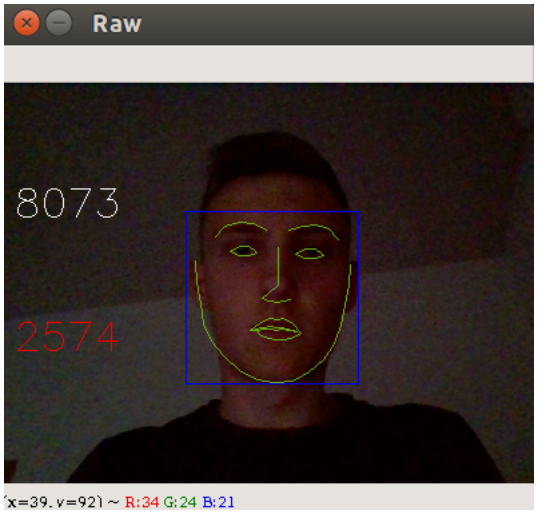

FIGURE 18: Détection en condition de faible luminosité

- En intérieur, avec une lumière naturelle d'environ 500 lux, dos à la source de lumière, on obtient le même problème de clignotement lors de la détection dû aux contre-jour.

- En intérieur, face à la source de lumière, le programme permet une détection parfaite pour des luminosités comprises entre 500 et 20 lux.

- En intérieur, pour des luminosité allant de 10 à 5 lux, la détection en temps réel commence à connaître quelques soucis lorsque le sujet est en mouvement mais la détection reste globalement correcte (Figure 18).

- En intérieur, dans le noir complet le sujet est uniquement éclairé par l'écran de l'ordinateur devant lui : la détection des yeux est décalée vers le bas, la précision n'est pas du tout au rendez-vous.

- En intérieur, éclairé par une source de lumière artificielle : La détection se fait parfaitement lorsque le sujet se trouve face à la source de lumière. On retrouve toujours le même problème de précisions lorsque ce dernier se trouve dos à la lumière.

Vous trouverez en annexe un tableau récapitulatif qui reprend toutes les données des tests de luminosité.

2) Impact des angles de vue: Angle du PC sur la table : Pour ce test, le sujet reste assis bien droit sur sa chaise et on fait pivoter l'ordinateur sur lui-même. La détection se fait correctement pour des angles allant jusqu'à 30°. Au dessus de cette valeur, on observe des clignotements jusqu'à 45°. A partir d'un angle de 50°, la détection est très aléatoire et au dessus de 60° le visage n'est plus du tout détecté.

- Angle du sujet par rapport à la caméra : Cette fois, le PC reste fixe et le sujet se penche à droite ou à gauche (figure 19). La détection se fait correctement jusqu'à un angle de 40°, au dessus de cette valeur le visage n'est plus du tout détecté. A noter : la détection ne fonctionne pas lorsque le sujet à la tête à l'envers.

- Angle de la caméra haut-bas : Lorsque le sujet est filmé de haut, la détection permet très peu d'angle avant de ne plus détecter le visage. Lorsque le sujet est filmé par le dessous, c'est un peu plus permissif, il est cependant très difficile de mesurer des valeurs précises.

- Rotation de la tête : Lorsque le sujet est face à l'ordinateur et qu'il tourne la tête de gauche à droite, la détection ne se fait correctement que dans un angle compris entre 30° et 150° : voir figure 20.

3) Autres conditions particulières: Pour finir, nous avons testé notre détection de visage sur des sujets ayant des particularités physiques ou des accessoires, voici les résultats obtenus :

- Lunettes de vue : Détection parfaite.

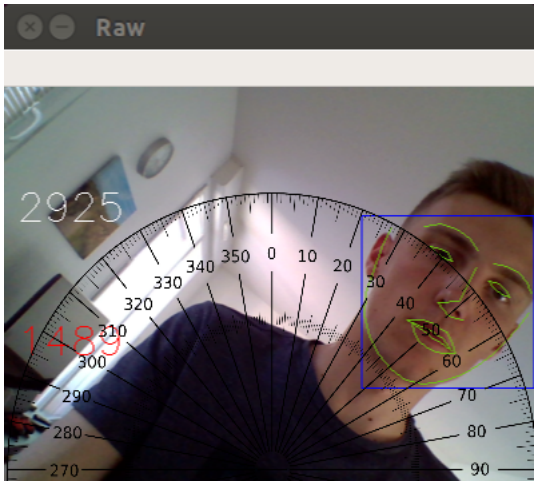

FIGURE 19: Angle du sujet par rapport à la caméra

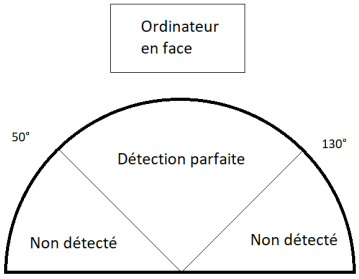

FIGURE 20: Rotation de la tête de droite à gauche

- Lunettes de soleil : Visage non détecté.

- Casquette : Le visage est détecté si le sujet se trouve bien face à la caméra.

- Barbe : Détection parfaite.

- Yeux fermés : Détection parfaite.

- Bouche ouverte : La détection prend un peu plus de temps à se faire mais elle est globalement correcte.

- Langue tirée : Détection parfaite.

- Expressions du visage (Joie, colère, tristesse) : Détection parfaite.

- Grimaces : Beaucoup de cas où le visage n'est pas détecté.

Encore une fois vous trouverez le détail des résultats en annexe.

D. Applications possibles

La position des points de saillances peut a terme être exploitée pour déterminer le comportement de l'individu. Un exemple pourrait être la détection de l'ouverture des yeux. Pour cela, il suffit de calculer la hauteur de l'oeil (en soustrayant les coordonnées des points en bas des yeux avec les coordonnées des points en haut des yeux) et de la comparer avec la largeur afin d'obtenir une valeur quantifiant l'ouverture de l'oeil. Afin de vérifier que cela fonctionne, un programme rapide à été mis au point pour tracer la courbe d'ouverture des yeux en fonction du temps (figure 21).

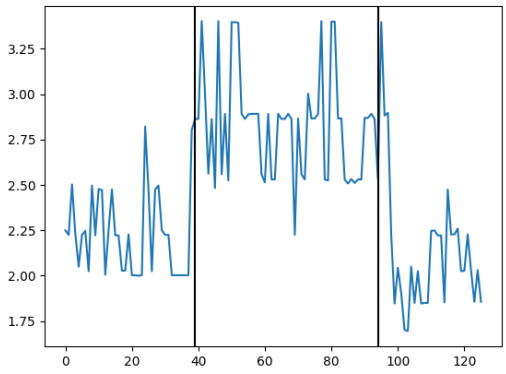

FIGURE 21: Ouverture des yeux en fonction du temps

Un appui sur la barre d'espace permet de placer un trait vertical afin que l'utilisateur puisse indiquer quand il ferme et ouvre les yeux. Le graphe obtenu montre bien que cette valeur augmente quand les yeux se ferment (première barre verticale) puis diminue quand les yeux s'ouvrent (deuxième barre verticale). Certains pics sont cependant à déplorer et les mesures pourraient être affinées, mais le manque de temps nous a empêché d'approfondir ces applications autant que nous l'aurions souhaités.

V. Conclusion

L'application de détection de visage et de saillances qui à été développée permet une grande plage d'utilisation. En effet, on a vu que la détection des saillances est très précises. Les contours des yeux, de la bouche, du nez, des sourcils et du visage sont parfaitement dessinés et ce même dans des conditions assez difficiles comme en faible luminosité par exemple. Le suivi des visages en temps réel est également possible grâce aux très bonnes performances de l'algorithme.

On peut maintenant envisager d'aller plus loin dans la détection en s'appuyant sur ces premiers résultats concluants. On peut par exemple imaginer détecter une émotion particulière chez le sujet, par exemple un sourire ou de la fatigue en mesurant par exemple l'ouverture de la bouche ou encore l'angle des sourcils. On pourra déterminer le sexe ou estimer l'âge d'une personne en fonction de la forme du visage ou des yeux. Il existe tout un tas de possibilités engendrées par une détection précise du visage d'une personne.

Toutes ces applications vont permettre, à terme, de faire réagir la lampe motorisée en fonction de l'attitude du sujet qui se trouve en face et ainsi de pouvoir étudier le comportement de la personne face à un objets qui semble "vivant".

Remerciements

Nous tenons à adresser nos remerciements à l'ensemble des personnes nous ayant permis de réaliser ce projet.

Dans un premier temps, nous souhaitons remercier Monsieur Jean-François Scheid, responsable du module de ce projet à TELECOM Nancy.

Dans un second temps, nous remercions messieurs Alain DUTECH, Amine BOUMAZA et Yann BONIFACE, les encadrants de ce projet qui nous ont apportés de nombreux et précieux conseils tout au long du développement.

Références

[1] C. Sagonas, E. Antonakos, G, Tzimiropoulos, S. Zafeiriou, M. Pantic. *300 faces In-the-wild challenge : Database and results.* Special Issue on Facial Landmark Localisation "In-The-Wild". 2016.
https ://ibug.doc.ic.ac.uk/media/uploads/documents/sagonas_2016_imavis.pdf

[2] P. Viola M. Jones *Robust Real-time Object Detection* Second international workshop on statistical and computational theories of vision, 2001
http ://citeseerx.ist.psu.edu/viewdoc/summary ?doi=10.1.1.110.4868

[3] A. Rosebrock *Facial landmarks with dlib, OpenCV, and Python* https ://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/

Lieu	Conditions	Luminosité	Résultat
Extérieur	Naturelle, plein soleil	30000 lux	Détection parfaite
	Naturelle, contre-jour	17000 lux	Léger souci de détection (clignotement) sur les zones très claires où le focus n'est pas fait
Intérieur	Naturelle, dos fenêtre	500 lux	Toujours le même problème de clignotement du au contre-jour
	Naturelle, face fenêtre	de 500 à 20 lux	Détection parfaite
		de 10 à 5 lux	Léger bug pour la détection en temps réel, la détection met un peu plus de temps à trouver et se caler sur le visage lorsque le sujet bouge mais globalement cela reste très bien.
	Noir complet, eclairage uniquement grâce à l'écran du PC	Environ 2 lux	Détection des yeux décalée vers le bas, le reste du visage est bien détecté.
	Artificielle, dos à la lumière	Environ 50 lux	Léger bug au niveau du contour du visage car la lumière vient du dessus : léger contre-jour du coup.

FIGURE 22: Rotation de la tête de droite à gauche

Condition	Résultat
Lunettes de vue	Détection parfaite
Lunettes de soleil	Visage non détecté
Casquette	Visage détecté tant que le sujet est bien face à la caméra. Si la caméra filme de haut le visage ne sera plus détecté du tout.
Barbe	A priori détection parfaite sur les quelques cas testés
Yeux fermés	Détection parfaite
Bouche ouverte	La détection prend un peu plus de temps à se caler sur la bouche du sujet quand celui-ci bouge
Joues gonflées (grimace)	Visage non détecté
Langue tirée	Détection parfaite
Expressions du visage exagérées (joie, colère, surprise..)	Détection parfaite
Grimaces	Beaucoup de cas ou le visage n'est pas détecté

FIGURE 23: Rotation de la tête de droite à gauche