Rapport de projet individuel
Les yeux nous trahissent

Projet de second semestre de Master Informatique

Janvier à Mai 2015

Auteurs :

— Elliot VANEGUE
— Gaëtan DEFLANDRE

Encadrants :
— Marius BILASCO
— José MENNESSON
— Benjamin ALLAERT

Remerciement

Résumé

Table des matières

Table des figures

1 Introduction

Pour nos études en master informatique, nous participons à un projet proposé par une équipe de recherche. Cela nous permet de découvrir le milieu de la recherche. Nous avons sélectionné ce projet, car les connaissances qu'il requiert correspondent à ce qui sera vu dans le master auquel nous avons postulé, c'est-à-dire le master IVI. Ainsi, nous consolidons les connaissances que nous acquérons durant les options enseigné par ce master.

L'application sur laquelle nous travaillons permet de localiser le visage d'un individu au travers d'un flux vidéo. Ce procédé se démocratise et il est de plus en plus utilisé dans des applications dans le domaine :

- de la sécurité, ce qui permettrait de détecter des comportements inhabituels chez un individu.
- du loisir, pour les jeux vidéo ou encore pour une interaction plus intuitive avec un ordinateur.
- de la publicité, pour que celle-ci corresponde au besoin des individus.

Ce type d'application utilise différentes technologies, mais celles-ci sont souvent trop intrusives et demandent à l'utilisateur d'utiliser du matériel assez spécifique. Or le but des algorithmes actuels est de faire en sorte que l'utilisateur ne se rende même pas compte qu'il utilise ce type de technologie. Les applications plus modernes effectuent des algorithmes de reconnaissance de formes, tels que l'algorithme de Viola et Jones, dans le but de détecter un visage à partir de simples caméras. Cela permet de mieux intégrer ces applications afin qu'il n'y est aucune contrainte pour l'utilisateur.

L'objectif de l'application sur lequel nous travaillons est de détecter les mouvements du visage de l'utilisateur afin d'effectuer de la reconnaissance d'émotions lors d'applications du type e-learning [1]. Ce procédé permettrait de détecter si un cours intéresse ou non les élèves afin de pouvoir l'améliorer. Si on voit sur une partie du cour que de nombreux élèves ont présentés des signes de fatigue ou qu'ils n'écoutaient plus, il sera alors possible aux enseignants de modifier cette partie.

1.1 Contexte

Pour la réalisation de ce projet, nous travaillons avec l'équipe FOX qui étudie l'analyse du mouvement à partir de vidéos. Leurs recherches portent sur l'extraction du comportement humain depuis les flux vidéo. Leurs travaux sont divisés en quatre grands domaines : le regard, qui est la partie sur laquelle nous travaillons, l'événement, l'émotion et la reconnaissance de personnes. La grande majorité de leurs travaux sont des applications temps réel, ce qui permet d'avoir un niveau de réactivité très élevé.

Le projet sur lequel nous travaillons est basé sur les travaux des membres de l'équipe qui se sont concentrés sur la détection de visage et des yeux. Cette détection permet ensuite de normaliser le visage, c'est-à-dire d'avoir une image statique contenant ce visage afin de réaliser des traitements sur cette image. C'est traitement peuvent par exemple être la détection de la fatigue ou de la concentration d'un individu.

1. formation en ligne

1.2 Problèmatique de l'existant

Les algorithmes utilisés dans l'application sont limités et ne permettent pas de faire un suivi correct dans toutes les situations d'une application temps réel. De nombreux cas ne sont pas traités dans ce type d'algorithme comme par exemple l'orientation du visage. Lorsque l'utilisateur effectue une rotation de la tête, les algorithmes utilisés ne sont pas capables de suivre ce mouvement. Il faut donc, grâce à différents axes présents dans le visage, détecter ce mouvement afin de garder une région d'intérêt correcte pour les traitements suivants.

Actuellement l'application n'arrive pas à suivre un visage dont les yeux sont fermés. En effet, l'algorithme de Viola et Jones repose sur la localisation de plusieurs points du visage, dont les yeux sont les points les plus importants. Si l'un des deux yeux est absent ou fermé, l'algorithme à moins de chance de détecter le visage. Cela implique que si une personne cligne des yeux l'application a des difficultées pour retrouver le visage pour la suite des traitements et cela cause de nombreuses erreurs.

De plus, les points représentant le centre des yeux ne sont pas parfaitement stables. Ce décalage a de nombreuses conséquences sur les traitements effectués par l'application, car l'application se base sur les ombres provoquées par le mouvement de certains muscles du visage. Il est donc primordial que la position de ces muscles soit stable, pour ne pas les confondre lors des traitements. Et pour cela il faut que l'image normalisé du visage soit stable.

1.3 Objectif du projet

Pour résoudre ces différentes problématiques, nous avons besoin de parfaitement localiser le centre des yeux. Pour cela, il existe différents algorithmes de détection de contours dans une image. Lorsque nous aurons détecté les contours des yeux, il nous sera alors possible de calculer le centre de la forme que nous aurons détecté. Les algorithmes de détection de forme ont besoin, pour effectuer leur traitement, d'une image binaire. Il nous faut donc déterminer quels sont les traitements à effectuer sur l'image d'origine récupérée par l'application afin de bien faire ressortir la forme d'un oeil. L'image sur laquelle nous travaillons est une image de la zone péri-oculaire de l'utilisateur, qui est calculé par l'algorithme actuel de l'application. Nous allons donc utiliser l'algorithme actuel afin d'affiner la localisation du centre de l'oeil.

Le second objectif est de calculer le centre de l'oeil lorsque celui-ci est fermé. Cette étape est plus complexe, car contrairement à un oeil ouvert, l'oeil fermé à la même couleur que la peau. Il faut donc réussir à différencier ces deux situations afin d'appliquer le bon traitement. Une première approche lors de la détection de l'oeil fermé est de travailler avec des filtres qui détectent des textures. L'idée est que la pupille a une texture différente de celle du reste du visage.

Pour répondre aux problématiques du projet, nous allons d'abord voir la structure actuelle du projet, ainsi que les algorithmes utilisés. Puis, nous détaillerons nos démarches pour résoudre les problématiques et nous expliquerons la solution finale que nous avons implémentée dans l'application.

2 Application existante

2.1 Présentation de l'application

L'application sur laquelle nous travaillons est écrite en C++ avec l'utilisation de QtCreator. Le programme a été développé par l'équipe Fox et par des étudiants. Cette application il a fait l'objet de divers améliorations auparavant.

2.2 Architecture

L'application fournis est orientée objet, celle-ci est découpé en Business Objects [2]. Ces Business Objects sont hiérarchisé dans une pyramide, les objets métiers en haut de cette pyramide sont les plus abstrés et utilise les resultats de l'étages infèrieur. Les objets métiers en bas de la pyramide sont les plus proche du flux vidéo brut.

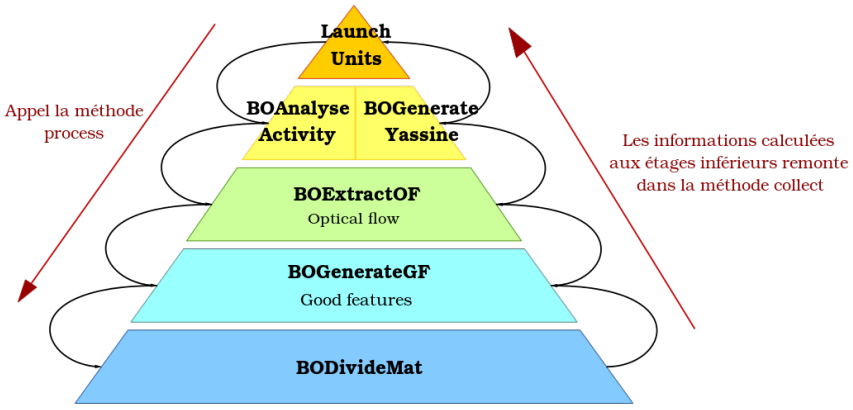

FIGURE 1 – Processus d'extraction des informations pyramidal sur le flux vidéo

2.3 Reconnaissance du visage : Viola et Jones

L'application est divisée en deux parties. La première recherche le visage grâce à l'algorithme de Viola et Jones et la seconde recherche les yeux dans la région délimitée précédemment.

L'algorithme de Viola et Jones[?] est une méthode qui a été créée pour la reconnaissance de visage dans une image. Cette méthode s'est par la suite généralisée à toutes sortes d'objets. L'algorithme nécessite une base de connaissances composée des caractèristiques de l'objet recherché. Cette base de connaissances est utilisé dans un apprentissage supervisé, c'est à dire que l'algorithme a besoin de données représentant l'objet à détecter pour classifier les caractèristiques de celui-ci.

2. les objets métier s'occuppent d'une tâche précise

Cette algorithme est basé sur des caractèristiques pseudo-Haar qui crée des masques rectangulaires et adjacentes dans différente zone de l'image. Chaque masque calcule l'intensité des pixels qu'il contient, puis l'algorithme fait la différence entre les masque blanc et les masque noir. Cette méthode va permettre de détecter des contours ou des changements de texture.

FIGURE 2 – Exemple de caractèristiques pseudo-Haar utilisé pour l'algorithme Viola et Jones

Pour améliorer les perfomance de leur algorithme, Viola et Jones utilise la méthode Adaboost. Son principe est de séléctionner les caractèristiques les plus performante pour la détection de l'objet grâce à un calcule de probabilité utilisant l'entropie [3] des données.

2.4 Suivi des yeux

3. valeur mesurant l'incertitude d'une données

3 Recherche de solution

3.1 L'algorithme de Canny

3.1.1 Version de base

Le filtre de Canny a été créé en 1986 dans le but d'améliorer les résultats du filtre de Sobel. Le principe du filtre est d'utiliser deux seuils, un seuil haut et un seuil bas. L'algorithme commence par sélectionner les pixels supérieurs au seuil haut, puis recherche à partir de chaque pixel en dessous du seuil haut les pixels qui sont au-dessus du seuil bas dans son voisinage. Chaque voisin qui est entre les deux seuils appartient à un contour, on passe donc sa valeur à 255 pour le prendre en compte dans celui-ci. Ainsi on voit que cet algorithme prend en compte deux caractéristiques, l'intensité et la direction du gradient de l'image.

Nous utilisons cet algorithme dans les régions autour des yeux afin de délimiter le contour de chaque oeil. Le résultat permet de faire ressortir l'oeil, mais aussi les sourcils et certaines ombres présentent dans le creux de l'oeil. Ce filtre permet donc de détecter les contours des yeux, mais le résultat comporte soit beaucoup de bruit, soit trop peu de détails. Dans un premier temps, pour obtenir un seuil adaptatif, nous prenons la moyenne des pixels.

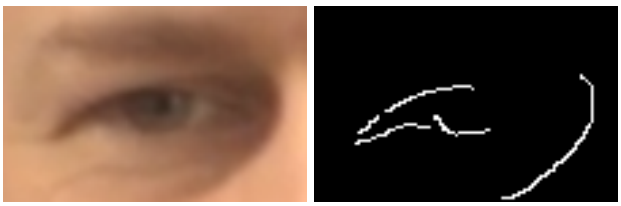

FIGURE 3 – Image de test et résultat de l'algorithme de Canny avec une moyenne des pixels

On voit que cette méthode, n'est pas optimale et supprime trop d'informations. De plus, l'algorithme prend beaucoup trop en compte les ombres présentes dans l'image. Nous utilisons donc une autre méthode permettant de mieux détecter les contours dans les zones que nous souhaitons.

3.1.2 Avec une moyenne de pixels sur des parties d'image

Plutôt, que de prendre la moyenne des pixels sur l'ensemble de l'image de la zone péri-oculaire, nous segmentons l'image en plusieurs zones dans lesquels nous calculons la moyenne et où nous effectuons l'algorithme de Canny.

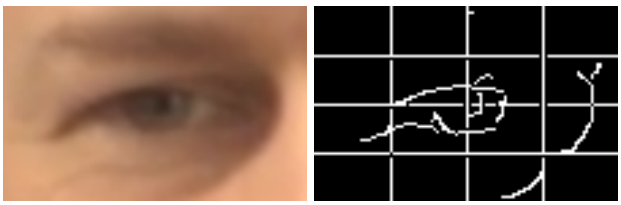

FIGURE 4 – Image de test et résultat de l'algorithme de Canny avec division de l'image en 16

Avec cette méthode, nous obtenons une image bruité avec les bords de chaque rectangle

utilisés pour le calcul précédent, car l'algorithme de Canny ne prend pas en compte les bords de l'image qu'il traite. Avec ce bruit nous ne pouvons clairement pas effectuer une recherche de l'oeil dans l'image. Il nous faut donc trouver une solution pour supprimer les traits présents dans l'image sans supprimer les contours présents dans l'image résultante.

3.2 Ouverture des sous-images

La dilatation d'une image est une opération morphologique [4] qui utilise un élément structurant afin d'effectuer une convolution de l'image. Si l'un des voisins du pixel traité est blanc alors celui-ci sera blanc, sinon il sera noir. Cette opération a, en général, pour effet d'élargir une forme et de combler les trous qui peuvent être présent dans cette forme. Nous avons également l'opération inverse qui est l'érosion, qui va diminuer la surface de la forme. Lorsqu'une dilatation est effectuée sur une image en niveau de gris, le pixel traité prend la valeur maximum parmi ses voisins, et le minimum pour la dilatation.

L'ouverture est la combinaison des deux opérations morphologiques vues précédemment, en commençant par l'érosion puis la dilatation. Cela a pour effet de fusionner des formes si elles sont proches (dans ce cas il y a des chances que ce soit la même forme) et de diminuer la largeur de la forme pour s'approcher de la forme réel.

Afin de supprimer le bruit présent dans l'image de l'oeil après utilisation du filtre de Canny, nous effectuons une ouverture de chaque sous image.

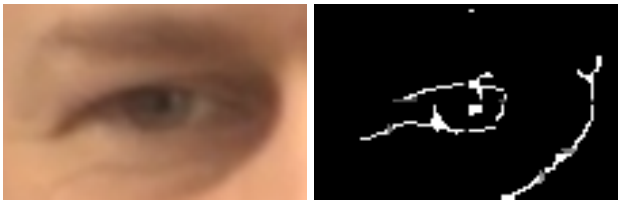

FIGURE 5 – Image de test et résultat de l'algorithme de Canny avec dilatation puis érosion des sous-images

Grâce à ce procédé, nous obtenons une image sans la grille que nous avions précédemment. Cependant, le bruit du aux ombres et aux rides ne permet pas encore d'effectuer un algorithme de détection de forme pour la localisation de l'oeil. Afin de supprimer les bruits du aux ombres présentent dans l'image, nous décidons de travailler sur d'autre modèle colorimétrique comme le HSV [5] dont la composante « saturation » peut nous permettre de ne plus prendre en compte les changements importants de luminance.

4. filtre non linéaire
5. Hue Saturation Value

3.3 Les modèles Colorimètrique

3.3.1 Le modèle HSV

Le modèle colorimètrique HSV est composé de la teinte [6], la saturation [7] et la valeur [8]. La teinte correspond à la forme la plus pur des couleurs. La saturation est l'intensité de la couleur, plus cette valeur est faible et plus la couleur semble fade. Et enfin la valeur correspond à la luminosité de la couleur, plus cette valeur est petite et plus la couleur est sombre.

Nous avons essayé de décomposer l'image dans les trois canaux de HSV sur notre image de test. Nous pensons pouvoir utiliser la composante saturation afin de de ne plus avoir le problème des ombres qui doivent être représenté dans la composante valeur.

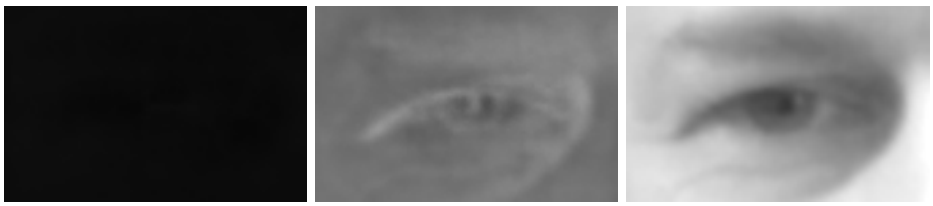

FIGURE 6 – Décomposition du modèle HSV

Nous pouvons voir que la saturation est initulisable tel quel, car les valeurs sont beaucoup trop basse et trop proche.

3.3.2 Le modèle YCbCr

Nous avons testé un second modèle colorimètrique suite à l'étude des travaux de Evangelos Skodras et Nikolaos Fakotakis[?]. En effet, leur étude montre que les différentes couleurs de peau de l'Homme ont à peu près la même valeur dans les deux composantes du modèle YCbCr.

On peut voir sur le résultat de la décomposition de l'image que le canal représentant la première chominance met en évidence la pupille de l'oeil.

FIGURE 7 – Décomposition du modèle YCbCr

6. Hue
7. Saturation
8. Value

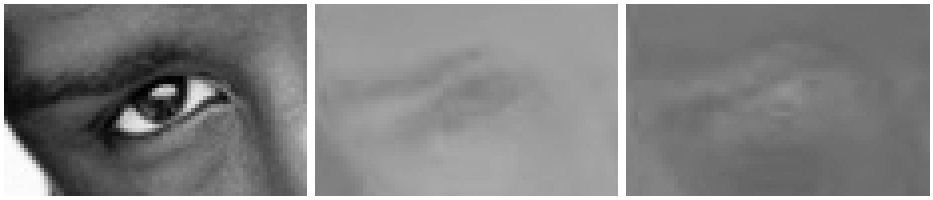

FIGURE 8 – Décomposition du modèle YCbCr avec une couleur de peau plus foncée

Pour obtenir la valeur des différentes composantes de cet espace colorimètrique à partir d'une image RGB il faut effectuer les calculs suivant :

$$Y = 0.299R + 0.587G + 0.114B$$

$$Cb = -0.1687R - 0.3313G + 0.5B + 128$$

$$Cr = 0.5R - 0.4187G - 0.0813B + 128$$

couleur de peau	rouge	verte	bleu	luminance(Y)	chrominance 1(Cb)	chrominance 2(Cr)
blanc	234	171	130	185	97	163
bronzé	175	112	95	129	109	161
asiatique	202	143	111	157	102	160

FIGURE 9 – Tableau des valeurs des couleurs de peau dans les espaces RGB et YCbCr

Nous pouvons voir avec ce tableau que la valeurs des deux chrominances sont très proche quelque soit la couleur de la peau. Pour la suite des traitements, nous décidons de garder la chrominance 1 qui semble varier un peu moins avec le changement de luminosité. En prenant le canal de la première chrominance et en trouvant le bon seuil pour la binarisation, nous obtenons une image comportant la forme de l'oeil au bon endroit. Nous pouvons alors calculer le barycentre de la forme obtenu afin de récupérer le centre de l'oeil.

FIGURE 10 – Résulat après traitement sur la chominance 1

3.4 Détection des contours avec des blobs

Maintenant que nous obtenons une image binarisé comportant la forme de l'oeil, nous pouvons déterminer ou est positionné le centre de l'oeil. Pour cela nous avons choisi l'utilisation de blob afin de résoudre un second problème que nous avons rencontré. Lorsque nous effectuons les

traitements précédent sur la vidéo, nous avons parfois des éléments qui perturbe la détection de contour comme des montures de lunette ou les cheveux de la personne filmé. Ces éléments peuvent faire partie des formes que nous récupérons lors de la binarisation. Les blobs sont des objets qui vont détecter des formes dans une image, mais il est possible d'ajouter des caractèristiques à ces blobs afin de récupérer seulement les formes qui possède ces caractèristiques.

Les blobs sont des éléments formé à partir d'une courbe 3D des fréquences d'une image.
Pour obtenir les blobs correspondant au yeux nous avons utilisé les paramètres :
— la convexité : les yeux étant censé être une forme ovale, il ne devrait pas y'avoir de segment, entre deux points, qui dépasse de la forme.
— l'aire de la forme : ce qui permet de ne pas prendre en compte les formes plus petites.
— la circularité de la forme : ce qui permet de ne pas prendre en compte les formes longue comme les cheveux sur le côté du visage.

3.5 L'algorithme de Gabor

4 Implémentation de la solution

4.1 Solutions utilisé dans l'application

4.2 Comparaison des résultats obtenu avec les anciens résultats

4.3 Améliorations envisageables

5 Conclusion

Annexes

Références