



RAPPORT D'AVANCEMENT

Projet HELP

Hope to Emulate the Life of Paralyzed
people

Rédigé par :
Hussain Al Othman
Katleen Blanchet
Titouan Boulmier
Laure Dupasquier
Pierre Jacquot
Marie-Alice Schweitzer

Sous la direction de :
Ali Mansour
Olivier Reynet

 © 2014 Olivier Reynet

Licensed under the Creative Commons Attribution-ShareAlike 4.0 International Public License.

Première impression, juillet 2014

Sommaire

Remerciements	i
I Introduction au projet	1
1 Formulation initiale du projet	3
1.1 Contexte	3
1.2 Expression initiale du besoin	3
2 État de l'art	5
2.1 Présentation des technologies existantes	5
2.1.1 Systèmes avec contact	5
2.1.2 Systèmes sans contact	8
2.2 La détection de pupilles	10
2.2.1 L'œil	10
2.2.2 Caractéristiques d'une image d'un œil acquise par caméra traditionnelle	12
2.2.3 Les différentes problématiques de la détection par caméra traditionnelle	13
2.2.4 Traitement d'image	14
II Dossier fonctionnel	27
3 Ingénierie des exigences	29
3.1 Approche Top-Down	29
3.2 Approche Bottom-Up	31
3.3 Fonctions principales du système	31
4 Spécification fonctionnelle 3 axes	33
4.1 Raffinement FAST	33
4.2 Spécification des données	33
4.3 Spécification des comportements	33
5 Architecture fonctionnelle	35

III	Organisation	37
6	Méthodes de travail	39
7	Outils pour les échanges	41
8	Répartition des tâches dans le temps	43
IV	Journal du projet	45
9	Choix et justifications	47
10	Résultats et analyses	49
11	Conclusion	51
V	Annexes	53
A	Première annexe	55
B	Deuxième annexe	57
C	Troisième annexe	59
	Bibliographie	61
	Index	63
	Glossaire	63

Remerciements

La gratitude est non seulement la plus grande des vertus, mais c'est également la mère de tous les autres.

Emil Cioran

Je tiens à remercier tous les contributeurs de L^AT_EX qui nous permettent aujourd'hui de produire des documents de qualité professionnelle sans avoir à se préoccuper de son apparence : des livres, des articles, des mémentos dans presque toutes les langues, mais aussi de la musique et des dessins. Ce logiciel ne connaît pas de limites.

Première partie

Introduction au projet

Chapitre 1

Formulation initiale du projet

1.1 Contexte

1.2 Expression initiale du besoin

Chapitre 2

État de l'art

2.1 Présentation des technologies existantes

2.1.1 Systèmes avec contact

Le suivi de l'œil (aussi appelé « eye tracking ») a de nombreuse application, de l'étude de l'attention, au contrôle de signe physiologique en passant par le contrôle d'un système (étude qui nous intéresse ici). Une pluralité de systèmes ont déjà été développés dans ce domaine et nous allons tâcher dans cette partie de vous présenter les systèmes majeurs. Deux grandes catégories de système existent : avec ou sans contact. Un système est dit sans contact lorsque celui-ci n'est pas attaché ou relié à l'utilisateur. Il a l'avantage de d'être plus agréable est facile à utiliser. Le deuxième, avec contact est monté sur l'utilisateur (comme des lunettes par exemple). Il offre ainsi une plus grande mobilité. Dans notre cas, l'utilisateur ce trouvant en face de son ordinateur, les méthodes avec et sans contact sont toutes deux à envisager.

2.1.1.1 Les Tobii Glasses

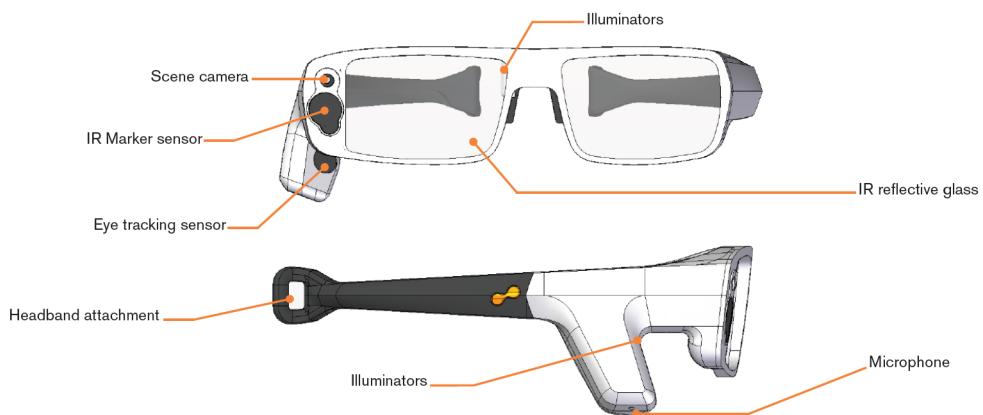


FIGURE 2.1 – Tobii Glasses

Les Tobbi Glasses sont des lunettes capables de filmer et d'enregistrer le mou-

vement des yeux. Elles peuvent ainsi savoir en temps réel ce que l'utilisateur fixe et voit. Ces lunettes sont équipées de caméras filmant directement l'œil, d'une caméra filmant ce que voit l'utilisateur, d'illuminateur permettant d'éclairer l'œil et enfin d'un capteur infrarouge (cf. figure 2.1). Ce dernier permet au système de connaître la position de la tête et de l'utilisateur dans l'espace, à l'aide de d'émetteur infrarouge balisant la zone que regarde l'utilisateur. Ces émetteurs communiquent donc avec les lunettes, permettant de localiser spatialement l'utilisateur et de calculer la position de sa tête (cf. figure 2.2).

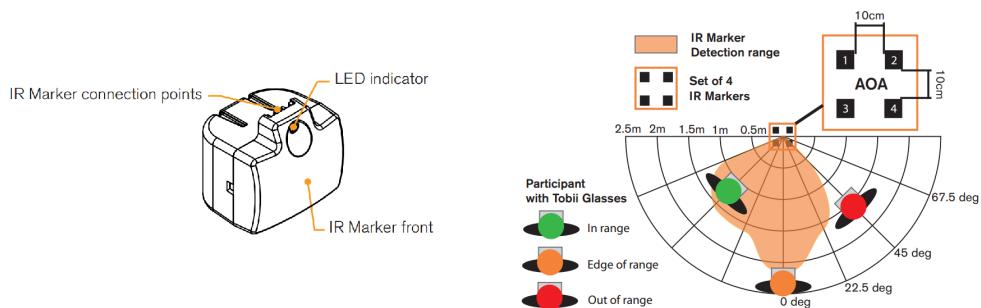


FIGURE 2.2 – Emetteur

Toutes les données sont enregistrées dans un boîtier relié aux lunettes (cf. figure 2.3). Elles peuvent ensuite être récupérées, traitées et analysées sur un ordinateur. La calibration est aussi effectuée via le boîtier.

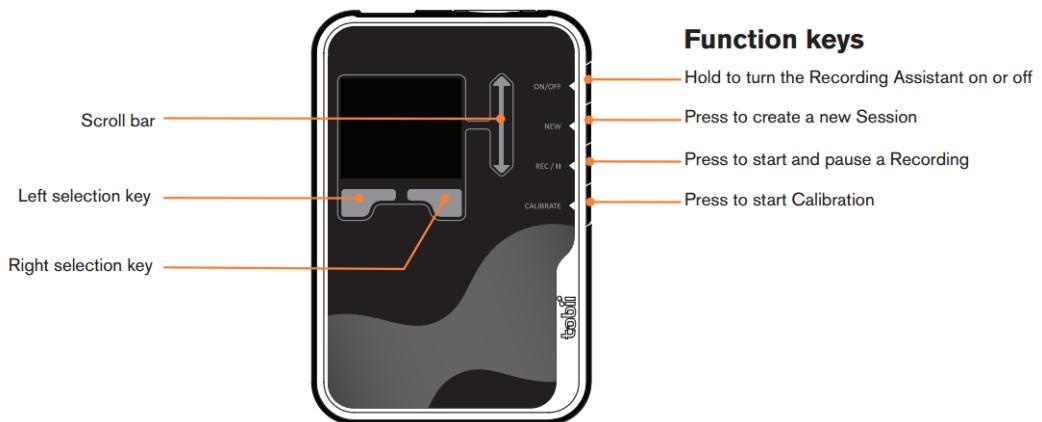


FIGURE 2.3 – Boîtier

D'un point de vue technique ces lunettes utilisent le Pupil Centre Corneal Reflection (PCCR, cf. figure 2.4). Cette méthode consiste à illuminer l'œil, créant ainsi un reflet sur la pupille et la cornée. Une caméra récupère ensuite une image de cette réflexion. Le vecteur formé par l'angle entre la cornée et le reflet lumineux sur la pupille est ensuite calculé. La direction correspond alors à la direction du regard de l'utilisateur. Malheureusement l'algorithme de calcul n'est pas donné.

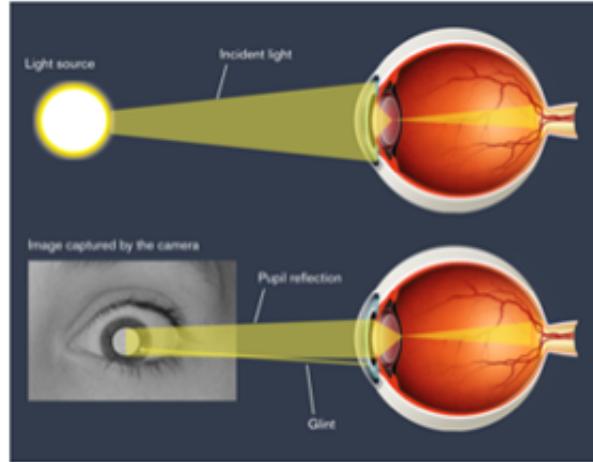


FIGURE 2.4 – Pupil Centre Corneal Reflection PCCR

Le prix n'est pas communiqué sur le site internet, mais il semblerait qu'il soit d'environ 10 000 €.

2.1.1.2 PUPIL



FIGURE 2.5 – PUPIL

PUPIL est un projet développé par trois étudiants du MIT. Tout comme les Tobbi Glasses il peut capter et enregistrer les mouvements effectués par l'œil de l'utilisateur.

Le principe de fonctionnement est basé sur deux caméras, la première permettant d'enregistrer les mouvements de l'œil, et donc de retrouver par la suite la position de la pupille et la seconde filmant ce qu'est sensé voir l'utilisateur et permettant en utilisant les données de la première caméra de connaître la direction du regard de l'utilisateur. Deux notions sont donc à distinguer : la position de la pupille et la direction du regard. La première nous aidant à déduire la seconde. L'avantage de cette méthode est l'utilisation de caméra classique afin de détecter la direction du regard. Il n'est ici pas obligatoire d'utiliser la réflexion d'une lumière infra-rouge dans l'œil de l'utilisateur.

D'un point de vue matériel, les deux caméras utilisées sont très différentes. Celle enregistrant le mouvement des yeux est une caméra basse résolution (640*480 à 30 fps) avec un filtre infrarouge (la Microsoft LifeCam HD-6000 est recommandée). La caméra filmant le monde alentour est possède quant à elle une grande résolution (1920*1080, la Logitech HD 1080p Webcams est recommandée). PUPIL peut être acheté pour la somme de 380 €.

2.1.2 Systèmes sans contact

2.1.2.1 Eye Charm

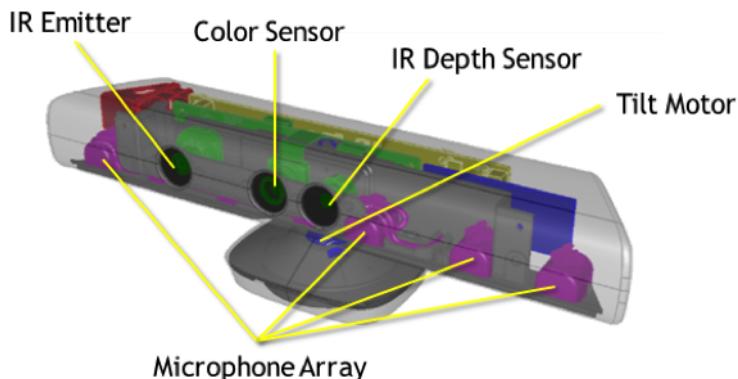


FIGURE 2.6 – EyeCharm

Créé par une société allemande (4tiito), EyeCharm est un adaptateur qui se clipse sur la Kinect et exploite sa caméra infrarouge pour suivre le mouvement des yeux. Un logiciel compatible avec Windows 7 & 8 (NUIA) a été développé pour contrôler les principales fonctions d'un ordinateur grâce à ce système, comme le navigateur internet, les jeux vidéo et d'autres applications. L'utilisation d'EyeCharm ne nécessite aucun changement dans le code source des applications. L'algorithme de suivi traite des images. La puissance de calcul nécessaire à son utilisation est ainsi assez importante (exemple : « son algorithme consomme 5 % de la puissance d'un processeur Intel Core i5-3470 cadencé à 3,2 GHz »). Il est recommandé de détenir « un pc équipé d'un processeur AMD ou Intel multicœur et avec au moins 2 Go de mémoire vive ». Le rôle d'EyeCharm est de projeter une lumière infrarouge sur le visage de l'utilisateur. Celle-ci est captée par la caméra de la Kinect (pour Xbox ou Windows, connectée en USB 2.0), ce qui lui permet de suivre le mouvement des yeux. Il est conseillé de se tenir à 75 cm de l'écran pour obtenir de bons résultats.

Le logiciel compte plusieurs extensions pour étendre les possibilités de contrôle par les yeux à :

- plusieurs navigateurs internet (Chrome, Internet Explorer, Firefox)
- Adobe Photoshop
- la suite Office
- les jeux World of Warcraft et Minecraft

Certaines fonctionnalités, comme le zoom ou le retour à la page précédente peuvent nécessiter une action supplémentaire, non réalisée par les yeux. Pour cela, il est possible d'appuyer sur une touche du clavier ou d'utiliser une commande vocale, prise en charge par la Kinect. Un kit de développement a été prévu pour que les utilisateurs puissent développer eux-mêmes des applications à l'aide de Qt Creator, Visual Studio, et les langages C, C++, C# et Java.

2.1.2.2 Robust Eye and Pupil Detection Method for Gaze Tracking

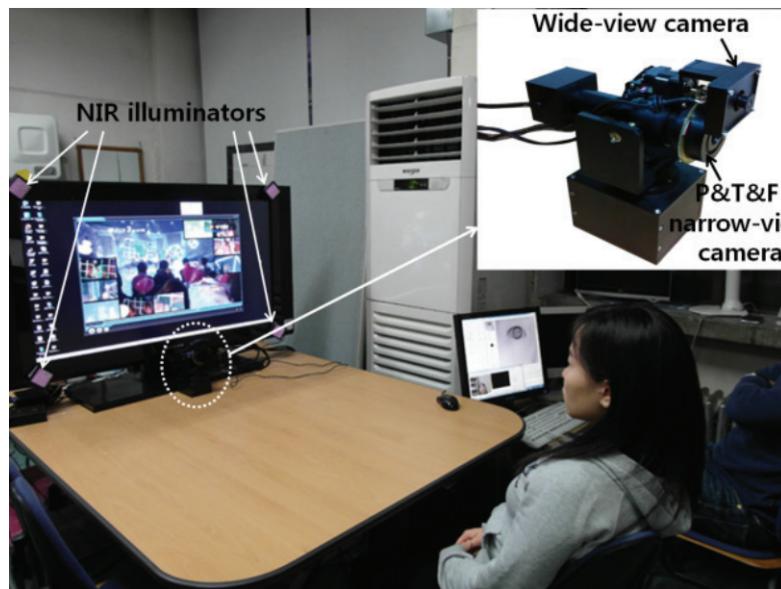


FIGURE 2.7 – Robust Eye and Pupil Detection Method for Gaze Tracking

Ce système est intéressant car contrairement aux précédents il n'est pas directement embarqué sur l'utilisateur. La détection des yeux ce fait à l'aide de deux caméras. Ces deux caméras sont motorisées et peuvent donc tourner sur elle-même ou s'orienter vers le haut ou le bas. Une caméra dite wide-view film l'utilisateur dans son intégralité. Dans les faits cette caméra permet de repérer le visage de l'utilisateur, puis la position de son œil grâce à deux algorithmes de reconnaissance faciale, l'Adaboost et le CAMShift. Une fois l'œil détecté, sa position est transmise à la seconde caméra dite narrow-view (d'une résolution de 1600x1200 réduite à du 240x320 pour améliorer le temps de calcul), qui va ainsi pouvoir bouger pour avoir l'œil de l'utilisateur dans son champ. Cette caméra permet d'obtenir un gros plan de l'œil contrairement à la première qui ne sert qu'à repérer l'utilisateur. Une fois l'œil dans le champ de vision de la caméra, la direction de celui-ci est calculé à l'aide du centre de la pupille et des réflexions spéculaires créées sur l'œil de l'utilisateur par quatre « near-infrared illuminators », eux même placés aux quatre coins de l'écran que regarde l'utilisateur. Ici encore l'algorithme de calcul reste flou.

D'un point de vue logiciel, cette étude a utilisé du C++ et la librairie OpenCV. Tous les calculs sont effectués directement sur un ordinateur classique équipé d'un

processeur Intel Core 2 Quad 2.3 GHz et de 4 GB. N'étant qu'un sujet de thèse, ce montage n'est pas en vente.

2.1.2.3 Tobii X2-30&60



FIGURE 2.8 – Tobii X2-30&60

Autre produit créé par Tobii, le X2 rentre dans les systèmes d'oculométrie sans contacts. Ne mesurant que 184 mm, et ce branchant directement en USB il est très facile d'utilisation et adaptable à de nombreux support (Ordinateurs portables, tablettes, télévisions etc.) Cet outil ne permet pas de contrôler le système sur lequel il est branché mais d'enregistrer ce que l'utilisateur regarde afin de d'obtenir des statistiques. La détection de la direction du regard se fait aussi à l'aide de la réflexion causée par des LEDs infrarouges sur la cornée du sujet. Combiné avec la localisation de la pupille, le système peut en déduire la direction du regard de l'utilisateur. Encore une fois l'algorithme n'est malheureusement jamais décrit. Cependant la précision de l'appareil permet une précision d'environ 0.34° sur la direction du regard du sujet.

2.2 La détection de pupilles

2.2.1 L'œil

Il est important de commencer par un rappel des caractéristiques biologiques de l'œil afin de permettre au lecteur d'avoir certaines notions de bases qui lui permettront de mieux appréhender les différentes problématiques de la détection de pupilles. La figure 2.9 nous présente les différentes parties de l'œil.

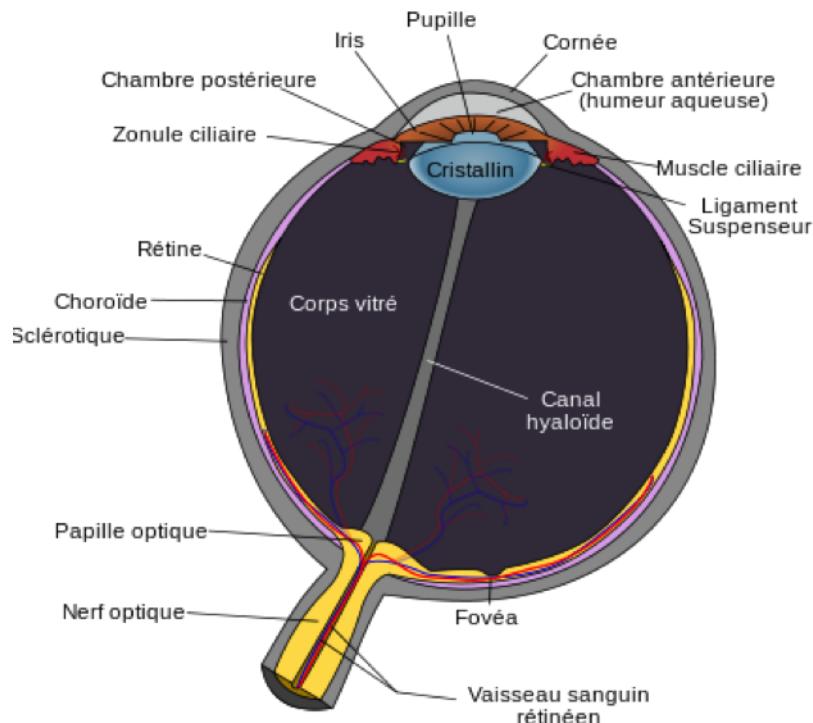


FIGURE 2.9 – Schéma anatomique de l'œil humain

Dans le cadre de la détection de pupilles, nous nous intéresserons plus particulièrement à la partie visible de l'œil. Cette partie externe est composée de 3 zones :

- La partie centrale : la pupille. C'est un orifice noir permettant de laisser passer la lumière.
- La bande colorée entourant la pupille : l'iris. Il permet de plus ou moins dilater la pupille.
- La zone blanche qui recouvre le reste de la partie visible : la sclérotique.

La figure 2.10 présente ces 3 parties.

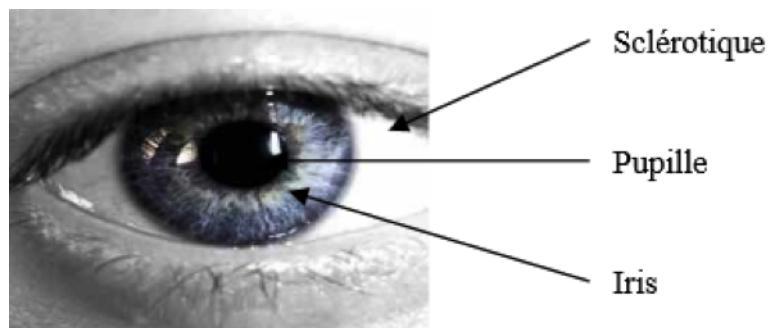


FIGURE 2.10 – Les différentes parties externes de l'œil humain

Nous allons maintenant nous intéresser à la capture d'une image d'un œil, à son traitement puis à la détection de la pupille.

2.2.2 Caractéristiques d'une image d'un œil acquise par caméra traditionnelle

Il existe deux types de caméras pour acquérir une image d'un œil : la caméra traditionnelle et la caméra infrarouge. Alors que la caméra traditionnelle permet de capturer le spectre de couleurs visible par l'œil humain, la caméra infrarouge, de son côté, permet de capter les ondes infrarouges de la lumière naturelle. (Voir figure 2.11). Cependant, la lumière naturelle ne comprend que très peu de composantes infrarouges. Ainsi, il est souvent nécessaire de soumettre l'objet d'étude (l'œil pour notre projet) à une source de lumière infrarouge supplémentaire afin d'améliorer la qualité et la luminosité de l'image acquise. Néanmoins, la caméra infrarouge permet d'éviter la majorité des problèmes de réflexion rencontrés par une caméra traditionnelle. (Nous pouvons voir figure 2.12(a) et figure 2.12(b) les différentes types d'image).

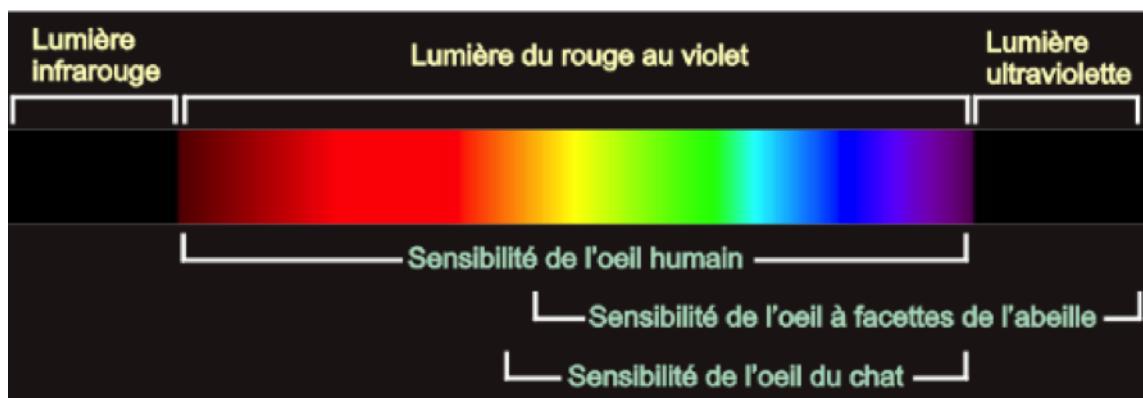


FIGURE 2.11 – Contenu spectral de la lumière

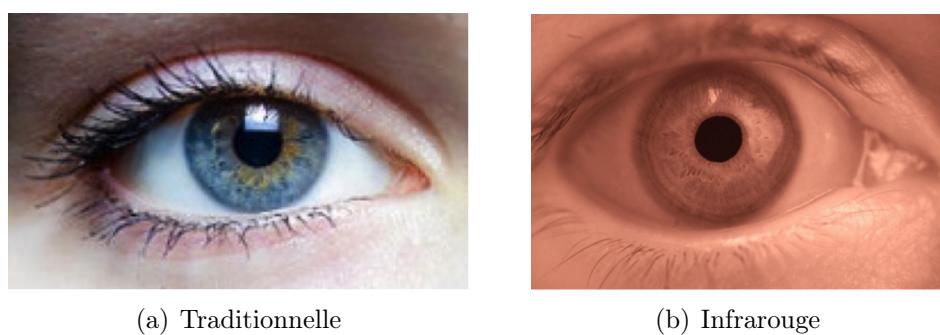


FIGURE 2.12 – Capture d'œil à l'aide d'une caméra

2.2.3 Les différentes problématiques de la détection par caméra traditionnelle

2.2.3.1 Ombres

La première difficulté rencontrée lors de la détection de pupille par caméra traditionnelle est la présence d'ombre dû aux cils (voir figure 2.13). Cet ombrage est dû à un angle trop grand entre la source de lumière et l'axe de la pupille. Afin de réduire ce phénomène au maximum, une attention particulière doit être portée au positionnement et à l'axe de la source de lumière.

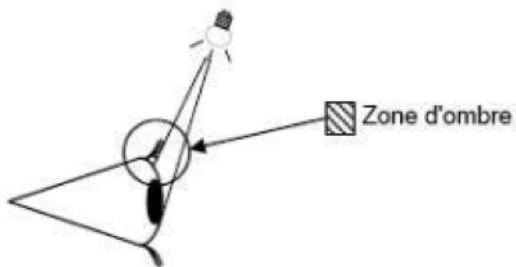


FIGURE 2.13 – Formation d'ombre dans une image d'un œil

2.2.3.2 Reflets

La surface de l'œil étant lisse et recouverte par la cornée, structure transparente recouvrant et protégeant l'iris (figure 2.9), les reflets émanant de l'œil sont considérables. L'intensité des reflets sont fonctions de l'intensité lumineuse dégagée par la scène. Plus la scène sera lumineuse, plus elle sera reflétée dans l'œil.

Le problème est que ces reflets sont gênants pour le traitement de l'image, et plus particulièrement pour la reconnaissance de la pupille. Par exemple, la figure 2.12(a) présente un œil avec un reflet recouvrant une partie de la pupille et une partie de l'iris. Ainsi, la pupille n'est plus tout à fait ronde et cela peut poser problème lors de sa détection.

Parmi les différents types de réflexions lumineuses, nous pouvons trouver les réflexions lumineuses ambiantes dues aux fenêtres, lampes et néons. Afin de réduire ce phénomène, une attention particulière devra être portée aux éclairages de la pièce. Cependant, il est important de noter qu'une réduction des sources lumineuses ambiantes n'est pas une solution car l'image perdrait en luminosité, en qualité, et son traitement n'en serait que plus difficile. La solution réside plus dans l'harmonisation de l'éclairage afin de ne pas créer de zone de réflexion.

Plusieurs approches sont possibles pour la résolution de ce problème de réflexion. Soit par l'application d'opérateurs morphologiques (La morphologie mathématique est une théorie et technique mathématique et informatique d'analyse. Son développement est inspiré des problèmes de traitement d'images, domaine qui constitue son principal champ d'application. Elle fournit en particulier des

outils de filtrage, segmentation, quantification et modélisation d'images.) en dilatant puis érodant la zone atteinte par la réflexion, soit en considérant qu'un reflet possède des caractéristiques contraires à celles de la pupille. Une source lumineuse étant très claire, on peut définir des bornes de niveau de gris à partir de l'histogramme de l'image et effectuer une segmentation (opérations de traitement d'image). Le seuillage originale de l'image originale représentant la pupille ainsi que l'image arborant les zones de réflexions seront par la suite combinés afin de former une image où les trous provoqués par les sources lumineuses seront remplis.

Il faut aussi noter que le port de verres correcteurs ou de lentilles accentue le phénomène de reflets.

2.2.3.3 Pupille

Il faut noter que si l'axe de la source de lumière correspond à celui des pupilles, il en résultera un effet d'yeux rouges qui peut être gênant pour le traitement de l'image et surtout la détection des pupilles. De plus, la taille des pupilles varie en fonction de l'intensité lumineuse présente. Il faudra donc veiller à ne pas imposer une source lumineuse trop importante afin d'avoir une taille de pupille suffisamment grande pour une bonne détection.

2.2.4 Traitement d'image

2.2.4.1 Segmentation

La segmentation (aussi appelée Clustering) est une étape de base du traitement d'image qui a pour but de séparer différentes zones homogènes d'une image en groupes (clusters) dont les membres ont en commun diverses propriétés (intensité, couleur, texture, etc.). En d'autres mots, cette opération a pour but de rassembler des pixels entre eux suivant des critères prédéfinis. Les pixels sont ainsi regroupés en régions, qui constituent un pavage ou une partition de l'image. Il peut s'agir par exemple de séparer les objets du fond. On peut regrouper les différentes méthodes de segmentation en deux catégories : la segmentation non supervisée, qui vise à séparer automatiquement l'image en différents clusters naturels (sans aucune connaissance préalable des classes) ; et la segmentation supervisée, qui s'opère à partir de la connaissance de chacune des classes définies par un approche probabiliste. Concernant les méthodes de segmentation non supervisée, nous nous intéresserons à deux méthodes : le Fuzzy C-means et le k-means.

Description des algorithmes

Fuzzy C-Means (FCM) est un algorithme de classification non-supervisée floue. Il introduit la notion d'ensemble flou dans la définition des classes : chaque point (pixel) dans l'ensemble des données appartient à chaque cluster avec un certain degré d'appartenance, et tous les clusters sont caractérisés par leur centre de gravité. Ainsi, il permet d'obtenir une partition floue de l'image en donnant à chaque pixel un degré d'appartenance compris entre 0 et 1 à un cluster donné. Le cluster auquel est associé un pixel est celui dont le degré d'appartenance est le plus élevé.

Les principales étapes de l'algorithme Fuzzy C-means sont :

1. La fixation arbitraire d'une matrice des classes.
2. Le calcul des centroïdes des classes.
3. Le réajustement de la matrice d'appartenance suivant la position des centroïdes.
4. Calcul du critère de minimisation et retour à l'étape 2 s'il y a non convergence de critère.

L'algorithme k-means est l'algorithme de Clustering le plus connu et le plus utilisé (simple à mettre en œuvre). Il permet de partitionner les pixels d'une image en K clusters. L'algorithme k-means ne crée qu'un seul niveau de clusters. L'algorithme renvoie une partition des données, dans laquelle les objets à l'intérieur de chaque cluster sont aussi proches que possibles les uns des autres et aussi loin que possible des objets des autres clusters. Chaque cluster de la partition est défini par ses objets et son centroïde. Le k-means est un algorithme itératif qui minimise la somme des distances entre chaque objet et le centroïde de son cluster. La position initiale des centroïdes conditionne le résultat final, de sorte que les centroïdes doivent être initialement placés le plus loin possible les uns des autres de façon à optimiser l'algorithme. K-means réitère ses opérations jusqu'à ce que la somme ne puisse plus diminuer. Le résultat est un ensemble de clusters compacts et clairement séparés, à condition d'avoir choisi la bonne valeur K du nombre de clusters. Les principales étapes de l'algorithme k-means sont :

1. Choix aléatoire de la position initiale des K clusters.
2. (Ré-)Affecter les objets à un cluster suivant un critère de minimisation des distances (généralement selon une mesure de distance euclidienne).
3. Une fois tous les objets placés, recalculer les K centroïdes.
4. Réitérer les étapes 2 et 3 jusqu'à ce que plus aucune réaffectation ne soit faite.

Statistics Toolbox implémentée sous MatLab 7 contient la fonction kmeans.m, facile à prendre en main et bien documentée et Fuzzy Logic Toolbox contient fcm.m.

Comparaison

Il apparaît que ces deux algorithmes sont assez efficaces pour des images couleurs et permettent une bonne segmentation. Cependant, lors de la présence de défauts (reflets, ombres, ...), la segmentation paraît correcte (elle ne permet pas une bonne détection des caractères par OCR Optical Character Recognition, ce qui ne nous intéresse pas). Enfin, il apparaît que ces algorithmes ne sont pas adaptés à des images contenant un grand nombre d'objets (au niveau du temps de calcul). La méthode FCM semble plus efficace pour la haute résolution, et au contraire, k-means est plus adapté aux images à faible résolution.

2.2.4.2 Traitement de l'image pour la détection de pupilles

La détection des pupilles est composée de plusieurs étapes et peut être codée à l'aide de plusieurs méthodes différentes en fonction de la complexité choisie. Certaines méthodes sont plus complexe et meilleures mais plus longue à réaliser (au niveau du temps de calcul). Pour notre projet, nous avons besoin de faire du traitement d'image en temps réel donc nous devrons faire attention à la complexité de notre algorithme. Les différentes étapes de la détection de la pupille sont :

1. Filtrage du bruit (filtre médian, ...)
2. Prétraitement (égalisation histogramme + seuillage bas pour supprimer le reflet)
3. Extraction de contour (filtre variance, ...)
4. Recherche des cercles/ellipses (Hough)
5. Discrimination du cercle de la pupille (couleur, position, surface, ...)

Une première méthode de détection de la pupille, développé par Jérôme Schmaltz de l'Ecole de Technologie Supérieure de Montréal, repose sur le principe décrit figure 2.14.

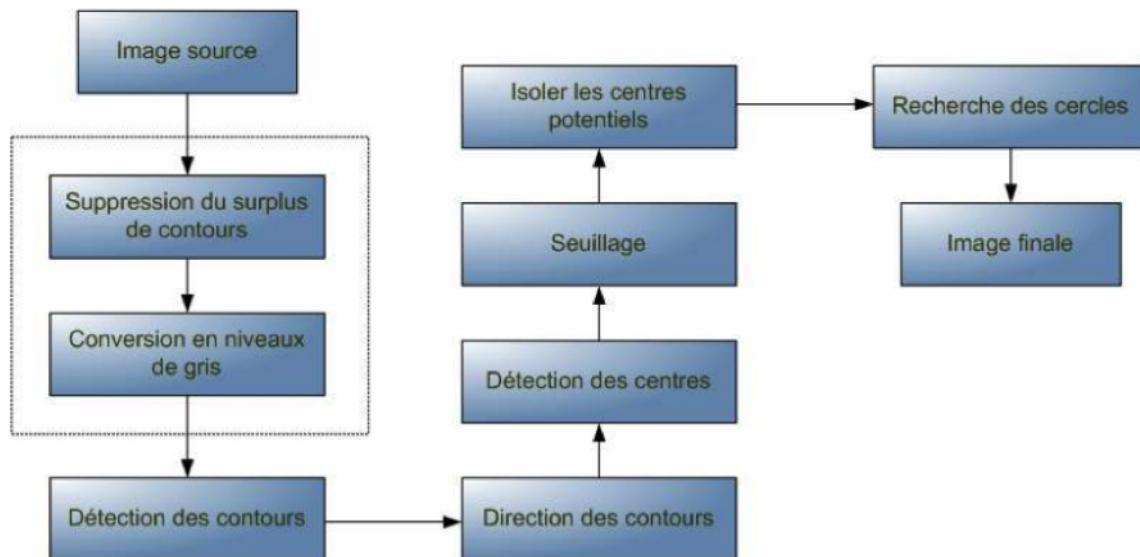


FIGURE 2.14 – Différentes étapes de traitement de l'image afin de détecter une pupille dans une image.

Suppression de surplus de contours, atténuation du bruit

Tout d'abord, il est nécessaire de réduire le bruit occasionné par la caméra elle-même ainsi que le bruit engendré par le transfert des données ainsi que leur compression. Différents filtres peuvent être utilisés pour réduire ce, même si son application altérera les contours de l'image.

Filtre Moyenneur

Ce filtre lisseur part du principe que la valeur d'un pixel est relativement similaire à son voisinage. Il fait donc en sorte que chaque pixel est remplacé par la moyenne pondérée de ses voisins. Si on applique un filtre moyenieur de taille $\lambda=3$, cela signifie qu'on additionne la valeur de tous les pixels du voisinage du pixel traité. On obtient ainsi la matrice de convolution suivante :

$$h = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

h s'appelle le masque de convolution. La somme des coefficients du masque valant 1, le lissage préservera toute zone de l'image où le niveau de gris est constant. Ce filtre peut engendrer des phénomènes de fausses couleurs, contrairement au filtre médian car son efficacité est moindre lorsque les objets présents dans l'image sont de faible dimension par rapport aux dégradations. Ce filtre est isotrope.

Une amélioration du filtre moyenieur consiste à jouer sur les valeurs des coefficients du masque :

$$h = \frac{1}{10} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Coupe Médiane

Une coupe médiane d'atténuer le bruit d'une image. Le principe est de prendre dans le voisinage la valeur la moins extrême. Pour cela, on crée une liste des valeurs du voisinage, puis on trie cette liste et on prend la valeur qui se trouve au milieu de la liste. Cette valeur "médiane" est la plus éloignée des deux extrêmes.

Diffusion

Le principe est d'atténuer les différences d'intensité entre le pixel central et ses voisins. Pour chaque voisin, on calcule la différence d'intensité avec le pixel central. Plus la différence est faible, plus elle est propagée vers le pixel central. Cela permet d'uniformiser les zones d'intensité proche et de conserver les forts contrastes (et donc les contours).

Filtre Gaussien

Le filtre Gaussien est un filtre isotrope spécial avec des propriétés mathématiques bien précises. σ caractérise l'écart type soit la largeur du filtre : autrement dit la largeur du filtre en partant du point central est égal à 3σ arrondi à l'entier supérieur. En deux dimensions, le filtre de Gauss est le produit de deux fonctions gaussiennes, une pour chaque direction :

$$g(x, y) = \frac{1}{2\pi\sigma^2} \cdot e^{-\frac{x^2+y^2}{2\sigma^2}}$$

L'effet de ce filtre sur l'image est assez similaire au filtre moyenneur, mais la moyenne est pondérée : les pixels près du centre ont un poids plus important que ceux que les autres. En général, un filtre Gaussien avec $\sigma < 1$ est utilisé pour réduire le bruit, et si $\sigma > 1$ c'est dans le but de flouter volontairement l'image. Il faut noter que plus σ est grand, plus la cloche Gaussienne est large et plus le flou appliqué à l'image sera marqué.

2.2.4.3 Conversion en niveau de gris

La seconde étape du prétraitement consiste à convertir l'image RGB en niveau de gris. Cette conversion est nécessaire à l'utilisation des différents algorithmes qui seront ensuite utilisés car ils se basent sur des images en niveau de gris. On pourrait d'abord penser que le niveau de gris se calcule comme la somme des 3 pixels divisée par 3 : $Gris = \frac{R+G+B}{3}$

Cependant, d'après les recommandations de la commission internationale de l'éclairage devrait plutôt être de la forme : $Gris = 0.299R + 0.587G + 0.114B$. Nous pouvons voir figure 2.15 le rendu de cette conversion.

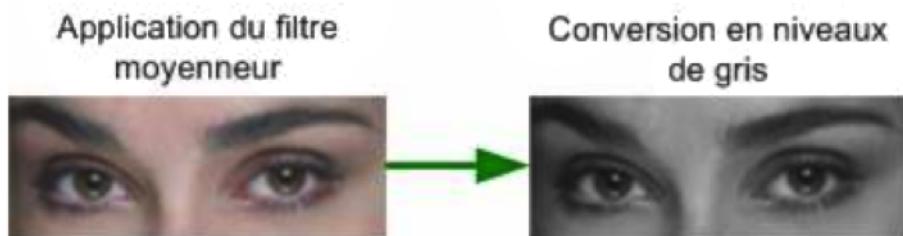


FIGURE 2.15 – Conversion en niveau de gris

2.2.4.4 Détection des contours

Filtre de Canny

Il existe différentes méthodes de détection des contours. Des filtres peuvent être utilisés (voir paragraphe 2.2.4.2). Nous nous intéresserons ici au meilleur filtre pour la détection de contours : le filtre de Canny. Il est considéré comme le meilleur car il offre un très bon compromis entre la réduction du bruit et la localisation des contours. D'ailleurs, il faut savoir qu'une réduction trop importante du bruit masque certains contours mais qu'une réduction trop faible peut laisser apparaître trop de contours non significatifs. Ensuite, les différentes étapes du filtre de Canny sont :

- Réduction du bruit grâce à la convolution d'un filtre passe-bas Gaussien, ce qui permet de tamiser les contours.
- Eclaircissement des contours grâce à l'utilisation des valeurs des amplitudes des gradients ainsi que leurs directions. Pour cela, nous utilisons les formules (2.1) et (2.2).

$$g = \sqrt{g_x^2 + g_y^2} \quad (2.1)$$

$$\theta = \arctan \frac{g_y}{g_x} \quad (2.2)$$

Nous pouvons voir que l'amplitude est proportionnelle à g_x et g_y , ce qui montre qu'elle mesure la force d'un contour indépendamment de sa direction.

- Suppression des non maximaux, cela permet de réduire la largeur des arrêtes détectées à un pixel. La figure 2.16 décrit un algorithme permettant de supprimer ces non maximaux.

```

Soit  $g_s$  une image de mêmes dimensions que l'image source  $g$ .
Pour toutes les coordonnées des pixels  $x$  et  $y$ ,
    Approximer  $\theta(x, y)$  par 0, 45, 90 ou 135 degrés.
    Si  $g(x, y) < g$  dans un voisinage de direction approximée ou que
         $g(x, y) < g$  dans un voisinage de direction  $\theta + 180$  alors
             $g_s(x, y) = 0$ 
    Sinon
         $g_s(x, y) = g(x, y)$ 
Fin si
Fin pour

```

FIGURE 2.16 – Algorithme permettant de supprimer des non maximaux

- Localisation : elle repose sur l'identification des contours significatifs à partir des amplitudes des gradients précédemment calculés.

La méthode triviale consiste à appliquer un seuillage aux gradients calculés. Cependant, l'application d'un seuil trop faible implique la prise en compte de tous les contours, y compris les contours non significatifs (prise en compte les gradients maximaux engendrés par le bruit). De plus, l'application d'un seuil trop élevé, quant à elle, engendre une fragmentation des chaînes de pixels qui forment des contours significatifs de l'image. Ainsi, le seuil par hystérésis offre une bonne solution à ce problème.

La figure 2.17 illustre l'idée de détection des contours grâce au filtre de Canny.

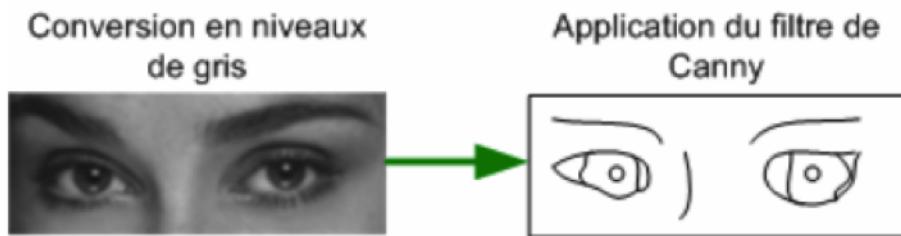


FIGURE 2.17 – Détection des contours grâce au filtre de Canny

Filtre de Sobel

L'opérateur calcule le gradient de l'intensité de chaque pixel. Ceci indique la direction de la plus forte variation du clair au sombre, ainsi que le taux de changement dans cette direction. On connaît alors les points de changement soudain de luminosité, correspondant probablement à des bords, ainsi que l'orientation de ces bords. L'opérateur utilise des matrices de convolution. La matrice (généralement de taille 3×3) subit une convolution avec l'image pour calculer des approximations des dérivées horizontale et verticale. Soit A l'image source, G_x et G_y deux images qui en chaque point contiennent des approximations respectivement de la dérivée horizontale et verticale de chaque point. Ces images sont calculées comme suit :

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * A \text{ et } G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A$$

En chaque point, les approximations des gradients horizontaux et verticaux peuvent être combinées comme suit pour obtenir une approximation de la norme du gradient : $G = \sqrt{G_x^2 + G_y^2}$

On peut également calculer la direction du gradient comme suit : $\Theta = \text{atan2}(G_y, G_x)$, et créer une matrice des directions.

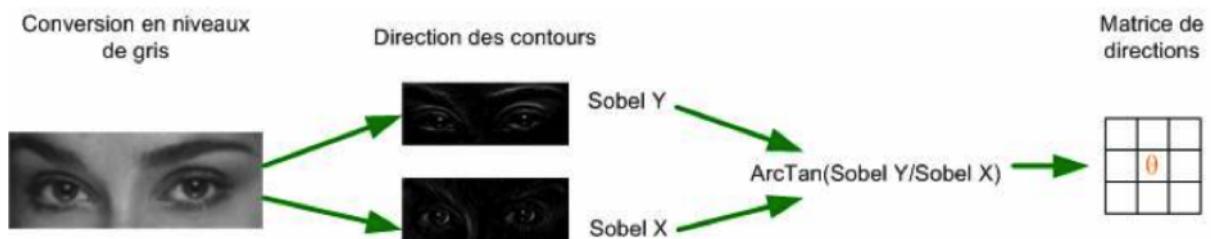


FIGURE 2.18 – Déterminer les directions des gradients à partir de l'application du filtre de Sobel

Enfin, d'autres filtres tels que le filtre de Kirsch, le filtre MDIF, filtre de Prewitt ou encore celui de Roberts, permettent aussi la détection de contours, mais nous ne les détaillerons pas ici.

2.2.4.5 Détection des centres

Le principe de la détection des centres se base sur les deux étapes précédentes. Pour chaque point des contours de l'image de Canny, on trace une droite en fonction de la direction calculée par le biais de l'application des filtres de Sobel. Ainsi, par l'application de la formule de la droite $y = m \cdot x + b$, on trace une droite dans l'accumulateur en fonction des points de contours de l'image de Canny tels qu'exposés dans la figure 2.19.

Tracer une droite dans un accumulateur revient à incrémenter chaque cellule d'une matrice de un aux endroits où elle passe (voir figure 2.20).

Ainsi, suivant ce principe, les valeurs maximales de la matrice indiqueront les différents centres potentiels. La figure 2.21 synthétise la méthode énoncée.

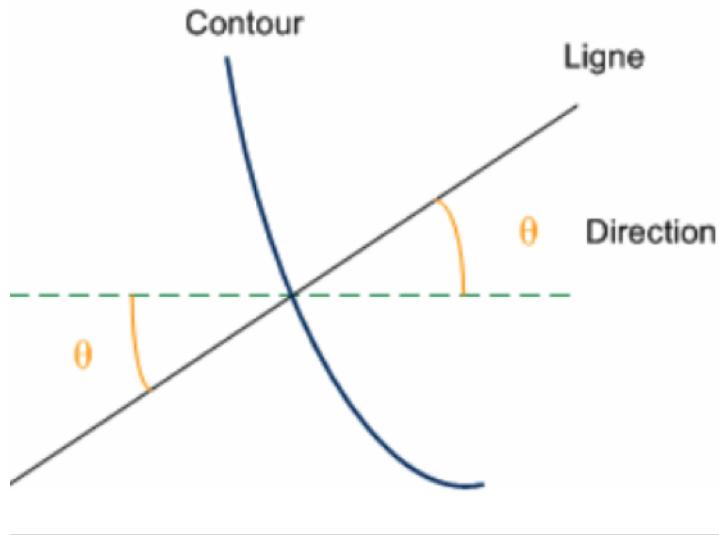


FIGURE 2.19 – Traçage d'une droite en fonction d'un contour

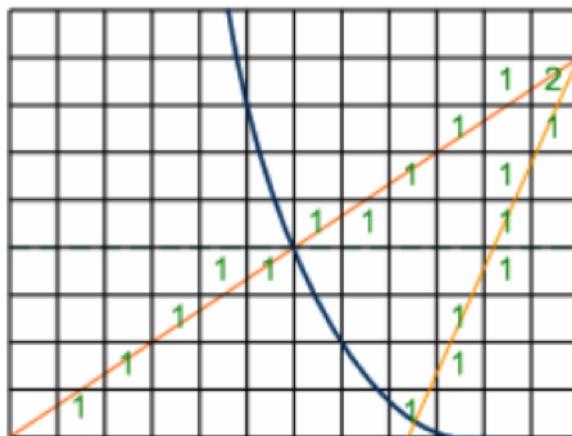


FIGURE 2.20 – Traçage d'une droite dans l'accumulateur

2.2.4.6 Seuillage

Le seuillage permet de désencombrer la matrice accumulateur d'informations superflues. En effet, nous recherchons les centres potentiels et ainsi, nous n'avons pas besoin de toutes les valeurs de l'accumulateur. En effet, il suffit d'appliquer un seuil et de garder les pixels ayant une valeur supérieur à ce seuil (les autres pixels seront mis à zéro) car se seront ceux qui représenteront les différents centres potentiels.

Pour ce faire, nous pouvons par exemple utiliser un seuil du type :

valeur pixel actuel $> 0.5 \cdot \max(\text{valeur pixel image})$, qui permet de garder les pixels ayant au moins la moitié de la valeur maximale d'une cellule de l'accumulateur.

Nous pouvons voir figure 2.22 le résultat du seuillage.

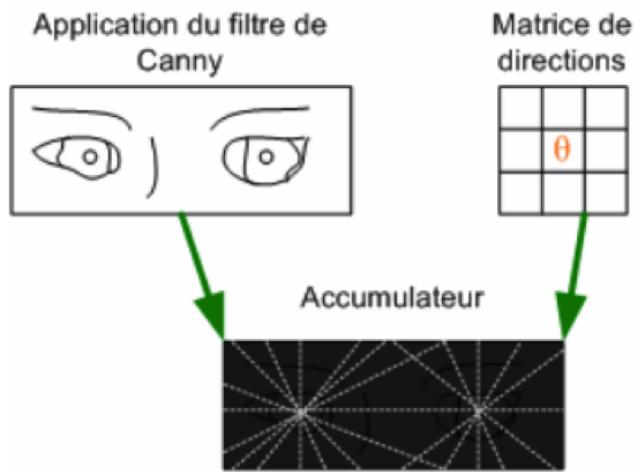


FIGURE 2.21 – Traçage des lignes dans l’accumulateur en fonction de l’image de Canny et de la matrice de directions

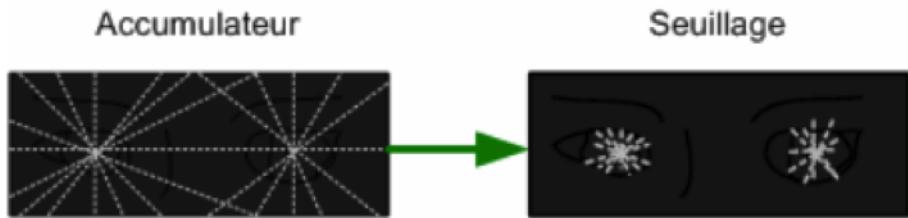


FIGURE 2.22 – Application d’un seuil dans l’accumulateur

2.2.4.7 Isoler les centres potentiels

L’application d’une convolution d’un chapeau mexicain (voir formule (2.3) et figure 2.23) sur l’accumulateur permet d’isoler les points les plus au centre. Un chapeau mexicain est la combinaison d’un filtre Gaussien (voir paragraphe 2.2.4.2) avec un filtre Laplacien.

$$\nabla^2 G_\sigma(r) = \frac{-1}{2\pi\sigma^4} \left(2 - \frac{r^2}{\sigma^2}\right) e^{-\frac{r^2}{2\sigma^2}} \quad (2.3)$$

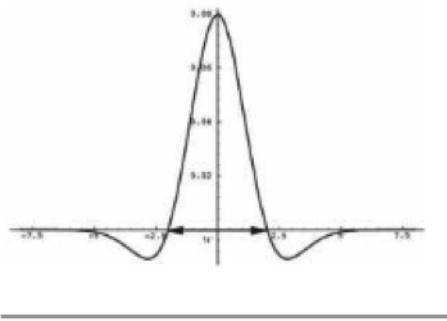


FIGURE 2.23 – Représentation graphique d'un Laplacien de Gaussien

Ainsi, l'application d'un chapeau mexicain sur l'accumulateur permet de conserver les points les plus à même d'être les centres de cercles potentiels en décrémentant les valeurs des cellules entourant les centres. La figure 2.24 présente le résultat de cet isolement des centres.

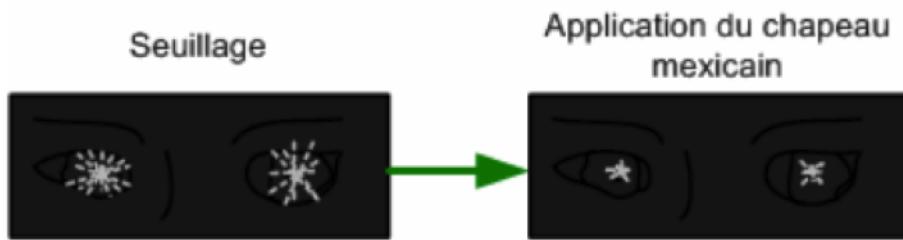


FIGURE 2.24 – Application d'un chapeau mexicain dans l'accumulateur

2.2.4.8 Recherche des cercles

La recherche des cercles s'appuie sur la création d'accumulateurs et la transformée de Hough. En effet, pour une cercle de rayon inconnu, nous utilisons des accumulateurs en 3 dimensions : les deux premières servent à ranger les coordonnées x et y du centre du cercle et la troisième permet de ranger le rayon r du cercle.

1. On initialise les accumulateurs
2. A partir de l'image des contours de Canny, on trace littéralement des cercles dans un des accumulateurs en faisant coïncider les centres des cercles avec les coordonnées du pixel de contours pour un rayon donné. La méthode de Bresenham pourra être employée afin de tracer plus efficacement les cercles dans l'accumulateur de Hough.
3. Comme on peut le voir figure 2.25, plusieurs cercles sont tracés (dans un accumulateur de rayon r) en suivant les contours de l'image de Canny. Nous pouvons voir que les cellules contenant les valeurs maximales coïncident avec des centres de cercles potentiels.

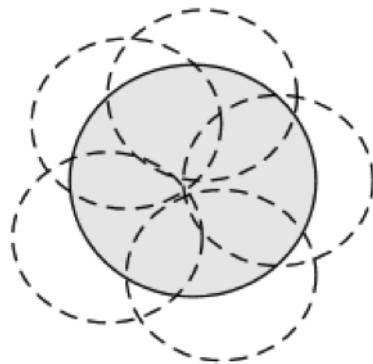


FIGURE 2.25 – Traçage des cercles avec l'image des contours

4. Ensuite, on peut diviser la zone d'intérêt de l'image en deux parties : partie droite et partie gauche du visage qui aborde les yeux. Pour chacune de ces parties, on applique un seuil aux accumulateurs d'Hough permettant de conserver les centres les coordonnées et rayons de cercles potentiels et on vérifie qu'ils font bien partie de l'accumulateur des cercles potentiels de l'étape de l'isolement des centres potentiels.
5. Pour finir, une moyenne des positions et des rayons des cercles restant permet une identification des cercles entourant l'iris et la pupille (à gauche et à droite). Il ne reste plus qu'à conserver le cercle ayant le rayon le plus petit, correspondant à la pupille.

La figure 2.26 résume cette démarche.

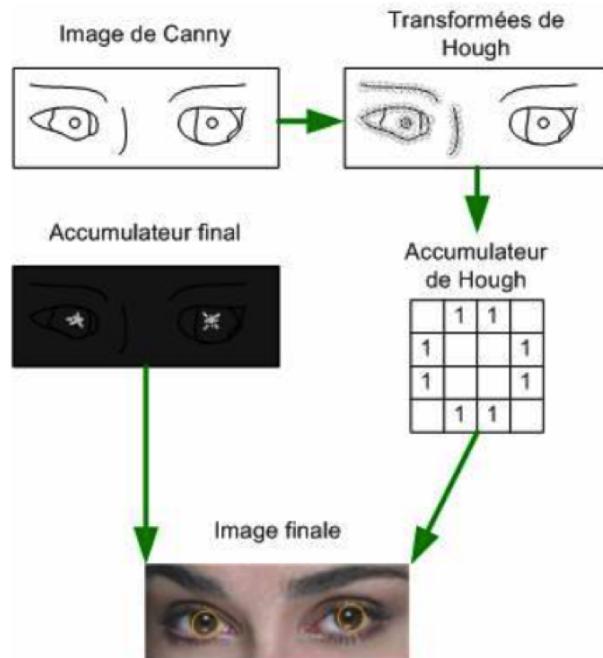


FIGURE 2.26 – Les différentes opérations permettant de localiser les pupilles

Quelles sont les solutions connues (algorithmes, matériels, logiciels, comportements, stratégies) ?

Que pourrait-on (ré)utiliser pour le projet ?

Deuxième partie

Dossier fonctionnel

Chapitre 3

Ingénierie des exigences

3.1 Approche Top-Down

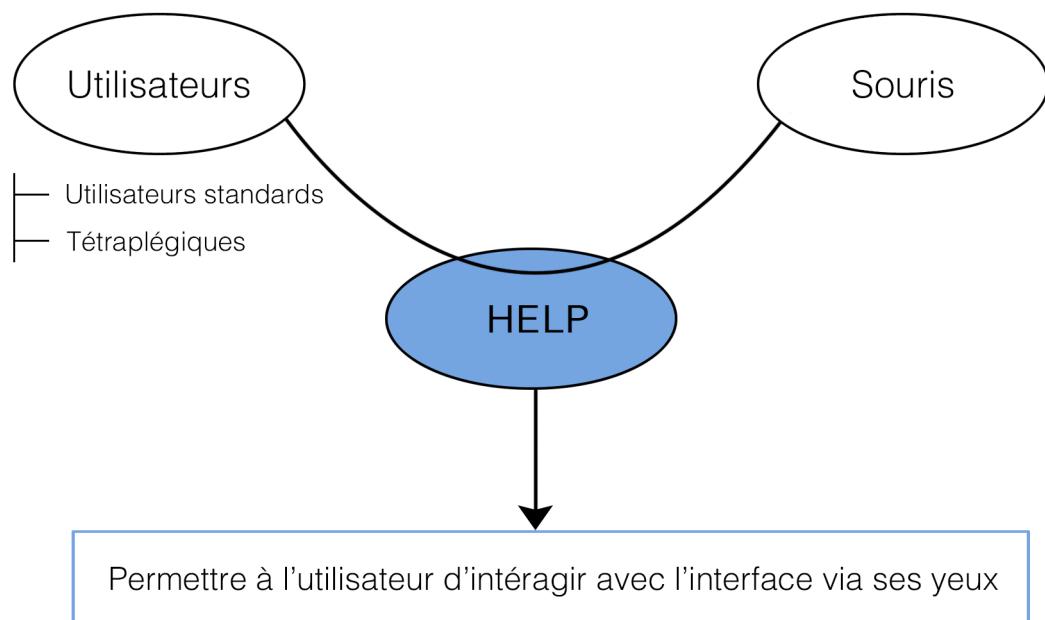
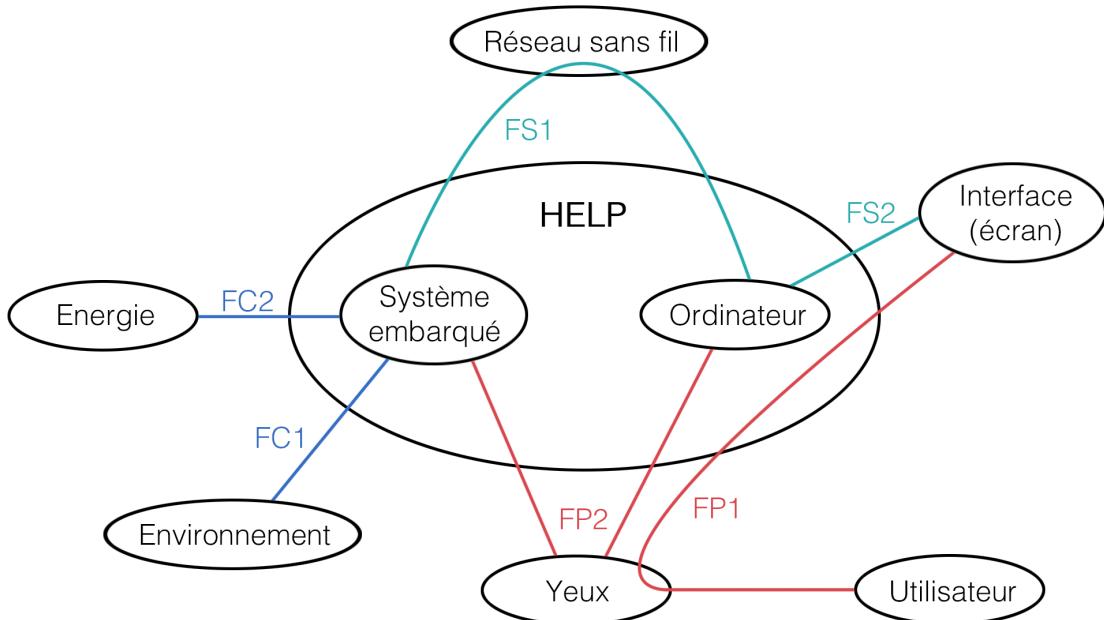


FIGURE 3.1 – Bête à cornes



FONCTIONS PRINCIPALES

FP1 : Permettre à l'utilisateur d'intéragir avec l'interface via ses yeux

FP2 : Déetecter le mouvement des yeux

FONCTIONS CONTRAINTEES

FC1 : Résister à l'environnement

FC2 : Être autonome en énergie (ou) S'adapter à l'alimentation

FONCTIONS DE SERVICE

FS1 : Transmettre des informations à l'ordinateur via un réseau sans fil

FS2 : Proposer une IHM adaptée

FIGURE 3.2 – Diagramme pieuvre

3.2 Approche Bottom-Up

Id	Type	Expression	Criticité	Flexibilité	Expression de la fonction
1.1.1	Service	Positionner l'utilisateur pour l'acquisition vidéo	5	5	Acquérir les données nécessaires à la détection du mouvement de l'oeil
1.1.2	Service	Filmer le visage de l'utilisateur	5	4	
1.1.3	Service	Acquérir les données nécessaires à l'estimation de la distance de l'utilisateur à l'interface	5	5	
1.1.4	Contrainte	Etre utilisable à une distance de 50cm à 3m	5	3	
1.1.5	Contrainte	Avoir une autonomie de Xh.	4	3	
1.2.1	Service	Relayer les données à la carte de traitement	5	5	Transférer les données enregistrées
1.2.2	Service	Transmettre le signal compressé	5	5	
1.2.3	Contrainte	Effectuer la transmission sans fil	3	3	
1.3.1	Service	Déetecter l'utilisateur et ses mouvements :	5	5	Traiter les données enregistrées
1.3.1.1	Service	Déetecter le visage de l'utilisateur	5	5	
1.3.1.2	Service	Déetecter les yeux de l'utilisateur	5	5	
1.3.1.3	Service	Déetecter les pupilles de l'utilisateur	5	5	
1.3.1.4	Service	Déetecter le mouvement des pupilles de l'utilisateur	5	5	
1.3.1.5	Service	Déetecter le clignement des yeux de l'utilisateur	4	3	Interpréter les données pour déduire l'action à exécuter
1.3.2	Service	Calculer la distance entre l'utilisateur et l'interface	5	5	
1.4.1	Service	Analyser le mouvement de la pupille à l'aide de la distance utilisateur-IHM	5	5	
1.4.2	Service	Déduire l'action à effectuer	5	5	
1.5.1	Service	Afficher le curseur à l'endroit regardé par l'utilisateur	5	4	Agir sur l'IHM
1.5.2	Contrainte	Actualiser l'affichage en moins de 10ms	5	2	
1.5.3	Service	Permettre le clic gauche / la sélection	5	3	
1.5.4	Service	Permettre à l'utilisateur de désactiver (mettre en pause) le système de détection	3	2	

FIGURE 3.3 – Cahier des exigences

3.3 Fonctions principales du système

Chapitre 4

Spécification fonctionnelle 3 axes

4.1 Raffinement FAST

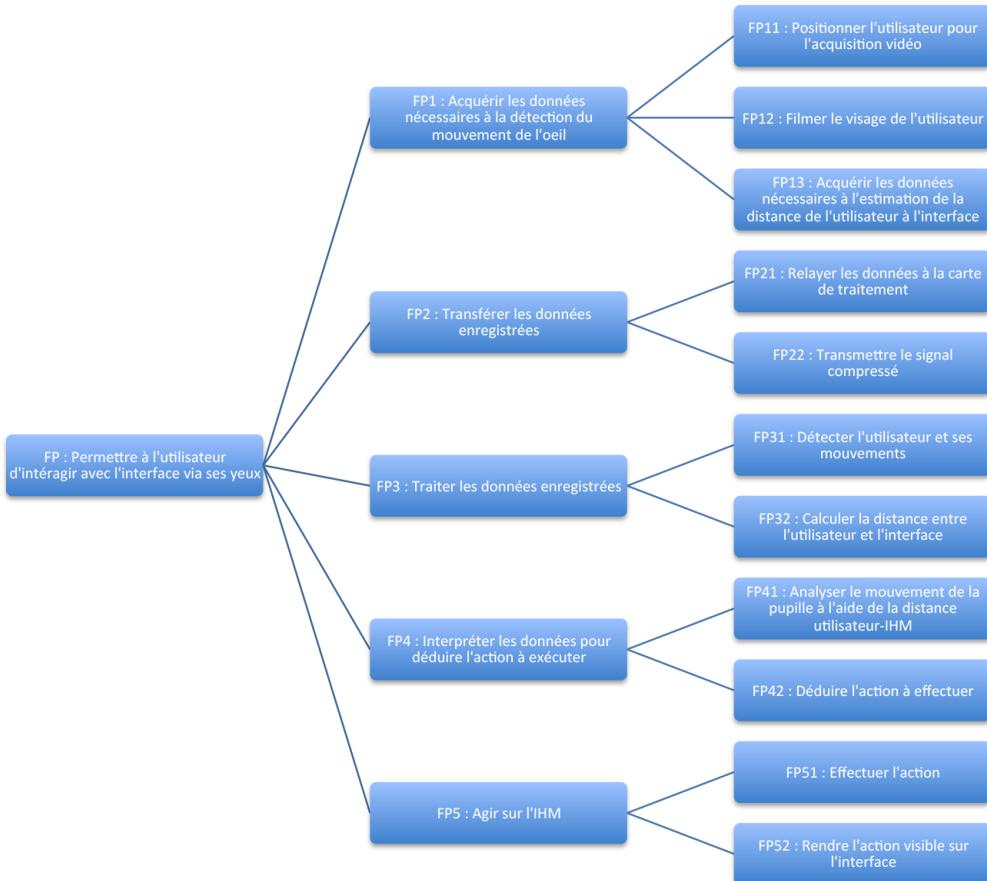


FIGURE 4.1 – FAST raffiné

4.2 Spécification des données

4.3 Spécification des comportements

Chapitre 5

Architecture fonctionnelle

Troisième partie

Organisation

Chapitre 6

Méthodes de travail

Méthodes de travail
Organisation temporelle, spatiale, humaine
interactions des membres de l'équipe
projet interactions avec les encadrants
interactions avec les tiers

Chapitre 7

Outils pour les échanges

Quels sont les outils qui nous permettent de travailler ensemble ?

Chapitre 8

Répartition des tâches dans le temps

WBS et diagramme de Gantt

Quatrième partie

Journal du projet

Chapitre 9

Choix et justifications

détails techniques et justification du choix des architectures cheminement du projet, évolution

Chapitre 10

Résultats et analyses

analyse des tests et des performances analyse des échecs, des décalages et des retards Que reste-t-il à faire ? Comment ?

Chapitre 11

Conclusion

Cinquième partie

Annexes

Annexe A

Première annexe

Annexe B

Deuxième annexe

Annexe C

Troisième annexe

Références bibliographiques

- [1] AFIS : Accueil - notre métier : L'ingénierie système. <http://www.afis.fr/nm-is/default.aspx>, 2010. Accédé le 30 août 2014.
- [2] Denis BITOUZÉ et Jean-Côme CHARPENTIER : *LATEX, l'essentiel*. Pearson Education France, 2010.
- [3] Bernard DESGRAUPES : *LATEX : apprentissage, guide et référence*. Vuibert informatique, 2003.
- [4] Leslie LAMPORT : *LATEX—A Document*, volume 410. pub-AW, 1985.
- [5] Noël-Arnaud MAGUIS : *Rédigez des documents de qualité avec LaTeX*. Livre du Zéro, 2010.
- [6] Noël-Arnaud MAGUIS : Rédigez des documents de qualité avec latex. <http://fr.openclassrooms.com/informatique/cours/redigez-des-documents-de-qualite-avec-latex>, 2013. Accédé le 30 août 2014.
- [7] Fabrizio SEBASTIANI : Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47, 2002.

