



RAPPORT ÉCRIT

Projet HELP

Hope to Emulate the Life of Paralyzed
people

Rédigé par :

Semestre 3	Semestre 4
Hussain Al Othman	Hussain Al Othman
Katleen Blanchet	Katleen Blanchet
Titouan Boulmier	Titouan Boulmier
Laure Dupasquier	Pierre Jacquot
Pierre Jacquot	
Marie-Alice Schweitzer	

Sous la direction de :
Ali Mansour et Olivier Reynet

22/05/2015

Remerciements

Nous tenons à remercier notre encadrant, M. Ali MANSOUR, pour ses conseils dans la réalisation physique du projet et sa disponibilité.

Nous témoignons également nos remerciements à M. Olivier REYNET pour ses précisions sur l'ingénierie système et pour son accompagnement tout au long du projet.

Nous remercions aussi les responsables de l'U.V. 3.4 pour leur présentation des techniques de gestion de projet et la mise en place des ateliers techniques.

Enfin nous tenons à exprimer notre reconnaissance à l'ensemble du personnel de la médiathèque pour leur disponibilité et leurs éclaircissements sur la recherche de documents.

Sommaire

Remerciements	2
Introduction au projet	5
1 État de l'art	6
1.1 Présentation des technologies existantes	6
1.1.1 Systèmes avec contact	6
1.1.2 Systèmes sans contact	9
1.2 La détection de pupilles	11
1.2.1 L'œil	11
1.2.2 Caractéristiques d'une image d'un œil acquise par caméra traditionnelle	12
1.2.3 Les différentes problématiques de la détection par caméra traditionnelle	14
1.2.4 Détection par éclairage infrarouge	15
1.2.5 Traitement d'images	16
1.3 Méthode de détection de la direction du regard	27
1.4 Solutions techniques et méthodes envisagées	28
1.4.1 Au préalable, suite à l'état de l'art	28
1.4.2 Méthodes choisies lors de la réalisation	29
2 Dossier fonctionnel	30
2.1 Ingénierie des exigences	30
2.1.1 Approche Top-Down	30
2.1.2 Approche Bottom-Up	31
2.1.3 Fonctions principales du système	32
2.2 Spécification fonctionnelle 3 axes	33
2.2.1 Raffinement FAST	33
2.2.2 Spécification des données	34
2.2.3 Spécification des comportements	34
2.3 Architecture fonctionnelle	35
3 Implémentation	37
3.1 Architecture physique et interfaces	37
3.1.1 Interface d'acquisition	38
3.1.2 Interface logicielle	38
3.1.3 Interface graphique	38

3.2 WBS	38
4 Organisation	40
4.1 Méthodes de travail	40
4.2 Outils pour les échanges	40
4.3 Répartition des tâches dans le temps	41
5 Journal du projet	45
5.1 Choix et justifications	45
5.1.1 Émetteurs infrarouges	45
5.1.2 Caméras	45
5.2 Résultats et analyses	46
Conclusion	47
Bibliographie	49

Introduction

Contexte

Dans le cadre de l’U.V. 3.4, notre équipe a choisi de développer le projet HELP (Hope to Emulate the Life of Paralyzed people) proposé par Mr Mansour. Ce projet a pour objectif la réalisation d’un système permettant de remplacer la souris d’ordinateur grâce aux mouvements des yeux. Cette étude semble très intéressante car elle demande une analyse et une compréhension de systèmes complexes d’eye tracking (oculométrie) déjà existants afin de mettre au point une version simplifiée et moins onéreuse. De plus, elle réunit différents aspects du travail d’ingénieur en informatique tels que le traitement de l’image, l’algorithmique, le travail en équipe,... Enfin, ce projet peut éventuellement mener à deux finalités différentes : d’abord, l’aide aux personnes tétraplégiques, qui, grâce à ce système, pourraient être moins dépendantes et retrouver un peu de liberté. Ensuite, ce projet pourrait aussi être utilisé afin d'aider les scientifiques à mettre au point un robot travaillant en zones hostiles qui puisse être contrôlé facilement grâce à la détection des mouvements de la tête et des yeux de l’opérateur.

Expression initiale du besoin

Le but premier de ce projet était le développement d’un système permettant à une personne tétraplégique d’utiliser un ordinateur et surfer sur internet. Cependant, face à l’ampleur de la tâche, et suite à un entretien avec nos encadrants, nous avons décidé de commencer par développer un système permettant à une personne ordinaire de contrôler un ordinateur. Ainsi, le dispositif développé doit permettre à un utilisateur d’exécuter différentes applications sans avoir besoin de toucher une souris ou un clavier. L’usager doit pouvoir effectuer les opérations usuelles en bougeant et clignant des yeux.

Partie 1

État de l'art

1.1 Présentation des technologies existantes

1.1.1 Systèmes avec contact

Le suivi de l'œil (aussi appelé « eye tracking ») a de nombreuses applications, tels que le contrôle d'un système. Une pluralité de dispositifs a déjà été développée dans ce domaine et nous allons tâcher dans cette partie de vous présenter les procédés majeurs. Deux grandes catégories de systèmes 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 d'être plus agréable est facile à utiliser. Le deuxième, avec contact, est monté sur l'usager (comme des lunettes par exemple). Il offre ainsi une plus grande mobilité. Dans notre cas, les deux méthodes sont à envisager.

1.1.1.1 Les Tobii Glasses

Les Tobii Glasses [13] sont des lunettes capables de filmer et d'enregistrer le mouvement des yeux. Elles peuvent ainsi savoir en temps réel ce que l'utilisateur fixe et voit. Ces lunettes sont équipées de (cf. figure 1.1) :

- caméras qui filment directement l'œil
- une caméra qui filme ce que voit l'utilisateur
- plusieurs illuminateurs permettant d'éclairer l'œil
- un capteur infrarouge

Le capteur infrarouge permet au système de connaître la position de la tête de l'utilisateur dans l'espace à l'aide d'émetteurs infrarouges balisant la zone qu'il regarde (cf. figure 1.2).

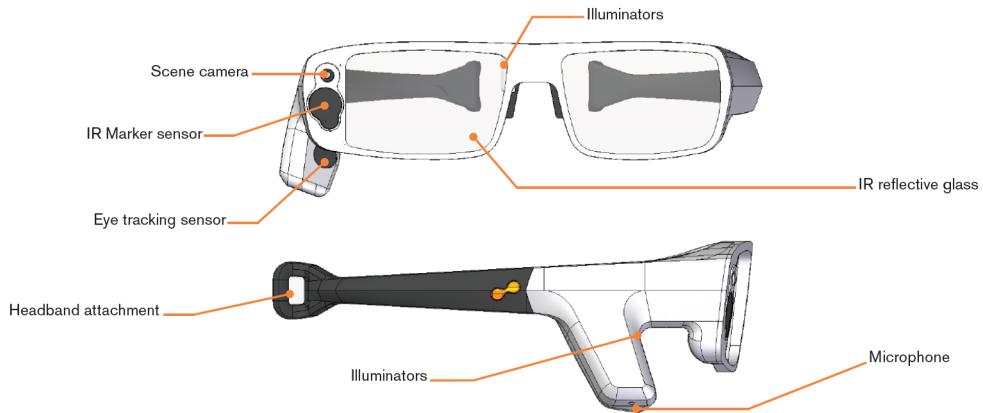


FIGURE 1.1 – Tobii Glasses

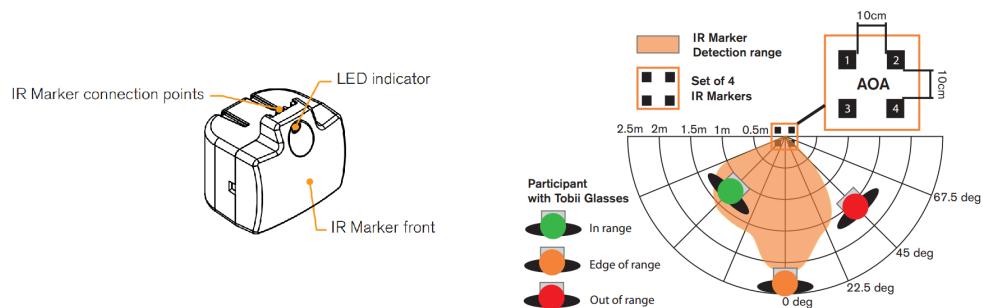


FIGURE 1.2 – Emetteur

Toutes les données sont enregistrées dans un boîtier relié aux lunettes (cf. figure 1.3). Elles peuvent ensuite être récupérées, traitées et analysées sur un ordinateur. La calibration est aussi effectuée via le boîtier.

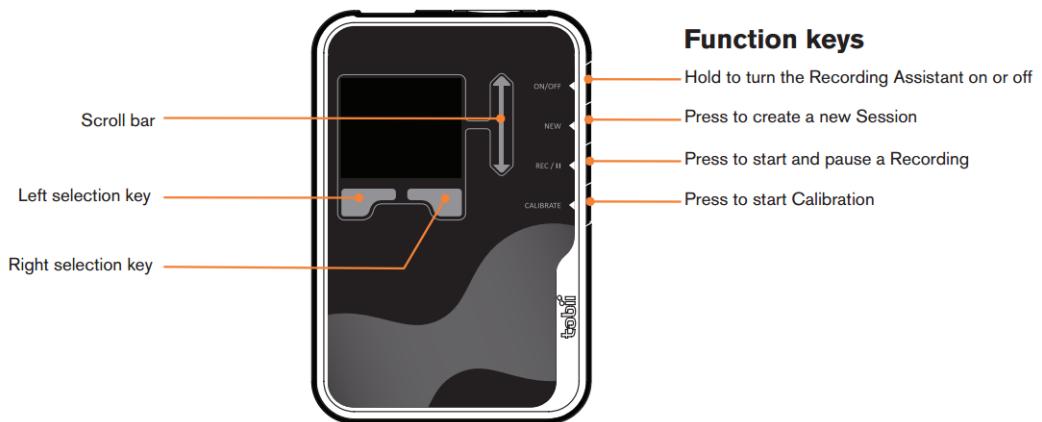


FIGURE 1.3 – Boîtier

D'un point de vue technique ces lunettes utilisent le Pupil Centre Corneal Reflection (PCCR, cf. figure 1.4). 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.

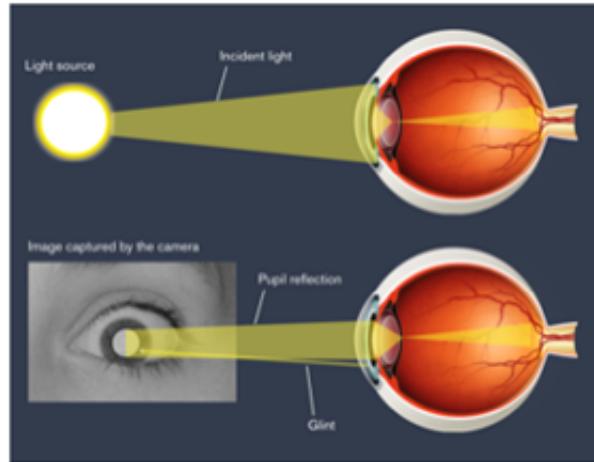


FIGURE 1.4 – Pupil Centre Corneal Reflection PCCR

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

1.1.1.2 PUPIL



FIGURE 1.5 – PUPIL

PUPIL [5] 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 permet d'enregistrer les mouvements de l'œil, et donc de retrouver par la suite la position de la pupille. La seconde filme ce qu'est censé voir l'utilisateur et permet de connaître la direction du regard de l'utilisateur en utilisant les données de la

première caméra. Deux notions sont donc à distinguer : la position de la pupille et la direction du regard. La première nous aide à déduire la seconde. L'avantage de cette méthode est l'usage d'une caméra classique afin de détecter la direction du regard. Il n'est ici pas obligatoire d'employer 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 qui filme le monde alentour possède quant à elle une grande résolution (1920×1080 , la Logitech HD 1080p Webcams est recommandée). PU-PIL peut être acheté pour la somme de 380 €.

1.1.2 Systèmes sans contact

1.1.2.1 Tobii X2-30&60



FIGURE 1.6 – Tobii X2-30&60

Autre produit créé par Tobii, le X2 [12] rentre dans les systèmes d'oculométrie sans contact. Ne mesurant que 184 mm, il se branche directement en USB. Il est très facile d'utilisation et s'adapte à de nombreux supports (ordinateurs portables, tablettes, télévisions...). Cet outil ne permet pas de contrôler le système sur lequel il est branché mais d'enregistrer ce que l'utilisateur regarde afin 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.

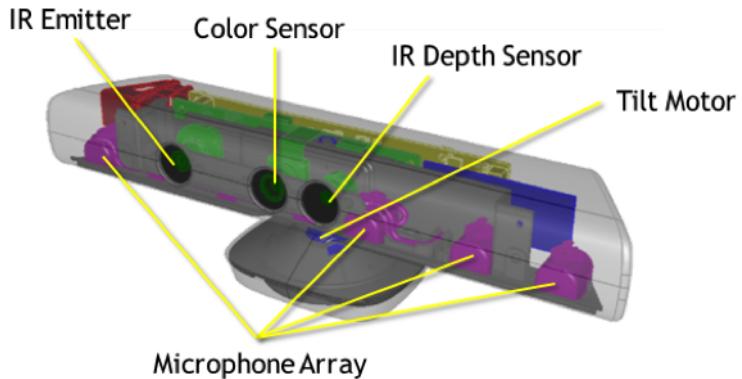


FIGURE 1.7 – EyeCharm

1.1.2.2 Eye Charm

Créé par une société allemande (4tiito), EyeCharm [15] 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.

1.1.2.3 Robust Eye and Pupil Detection Method for Gaze Tracking [4]

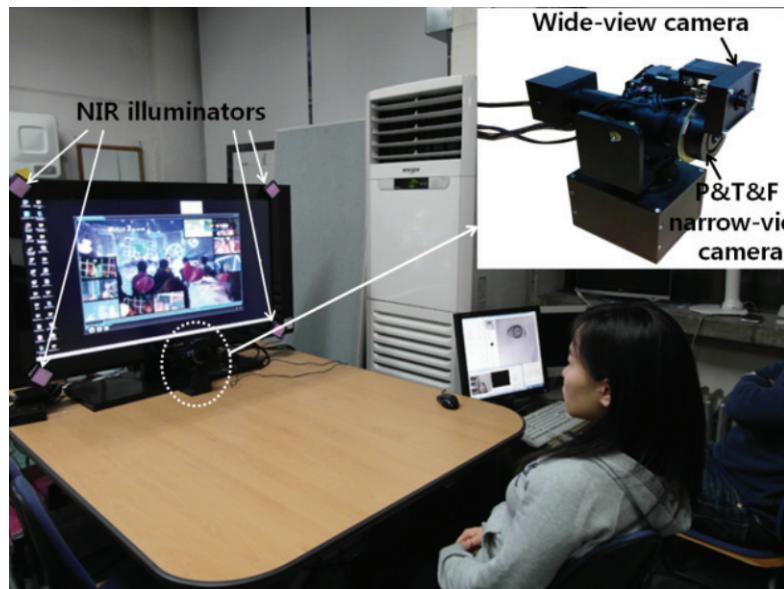


FIGURE 1.8 – Robust Eye and Pupil Detection Method for Gaze Tracking

Ce système est intéressant car il n'est pas directement embarqué sur l'utilisateur. La détection des yeux se fait à l'aide de deux caméras. Ces dernières sont motorisées et peuvent donc tourner sur elle-même ou s'orienter vers le haut ou le bas. Une caméra dite wide-view filme l'utilisateur dans son intégralité. Dans les faits cette caméra permet de repérer son visage, 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). Elle va pouvoir zoomer sur l'œil et ainsi obtenir un gros plan, tandis que la première ne sert qu'à repérer l'usager. Une fois l'œil dans le champ de vision de la caméra, la direction de celui-ci est calculée à l'aide du centre de la pupille et des réflexions spéculaires créées sur l'œil 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 méthode emploie 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.

1.2 La détection de pupilles

1.2.1 L'œil

Il est important de commencer par un rappel des caractéristiques biologiques de l'œil. Le lecteur acquerra certaines notions de bases qui lui permettront de

mieux appréhender les différentes problématiques de la détection de pupilles. La figure 1.9 nous présente les différentes parties de l'œil.

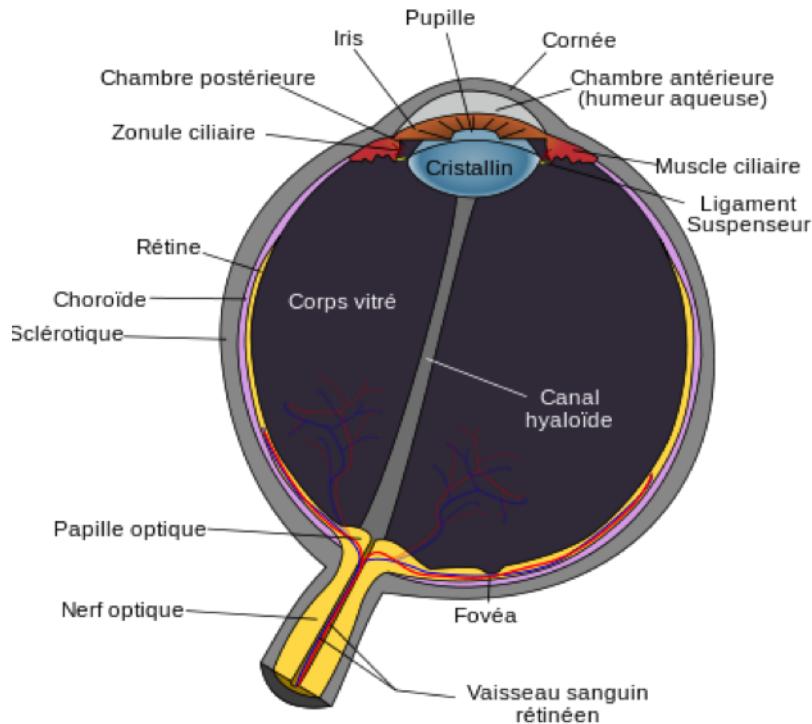


FIGURE 1.9 – Schéma anatomique de l'œil humain

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 1.10 présente ces 3 parties.

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

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

Il existe deux types de caméras pour acquérir l'image d'un œil : la caméra traditionnelle et la caméra infrarouge. Alors que la caméra traditionnelle permet

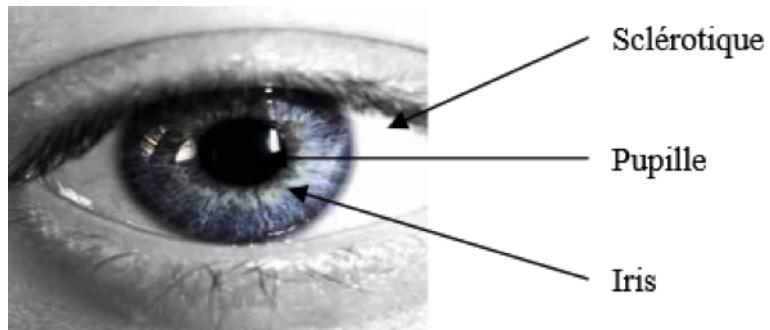


FIGURE 1.10 – Les différentes parties externes de l’œil humain

de capturer le spectre de couleurs visibles par l’œil humain, la caméra infrarouge permet de capter les ondes infrarouges de la lumière naturelle (voir figure 1.11). 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. La caméra infrarouge permet d’éviter la majorité des problèmes de réflexion rencontrés par une caméra traditionnelle. Nous pouvons voir figure 1.12(a) et figure 1.12(b) les différents types d’image.

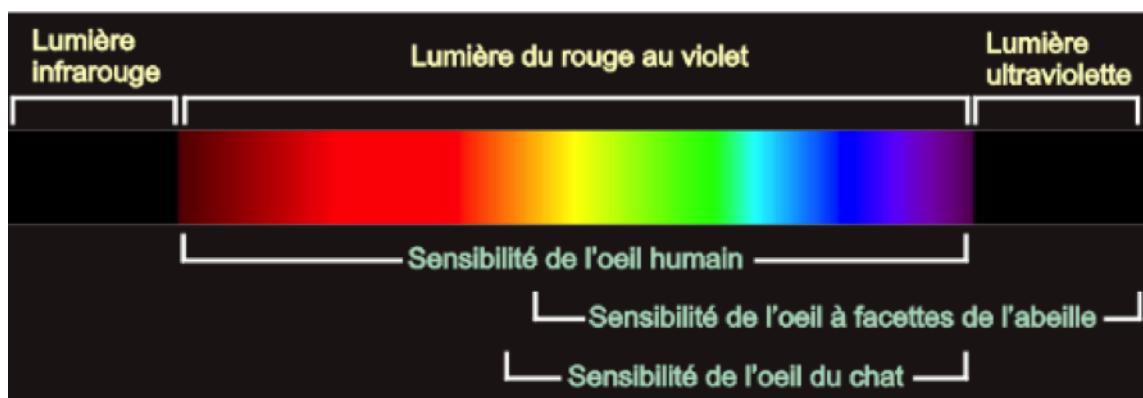


FIGURE 1.11 – Contenu spectral de la lumière

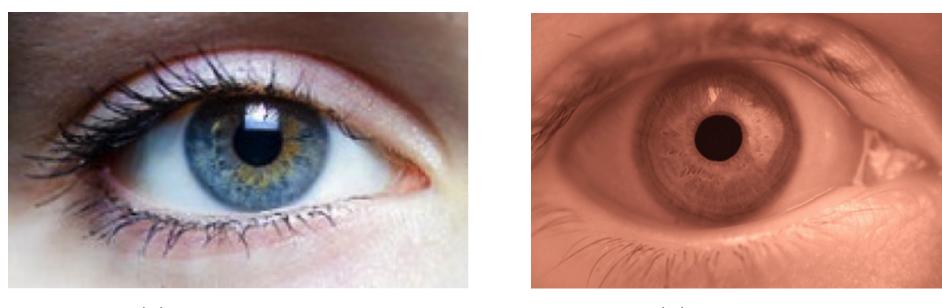


FIGURE 1.12 – Capture d’œil à l’aide d’une caméra

1.2.3 Les différentes problématiques de la détection par caméra traditionnelle

1.2.3.1 Ombres

La première difficulté rencontrée lors de la détection de pupilles par caméra traditionnelle est la présence de l'ombre des cils (voir figure 1.13). 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.

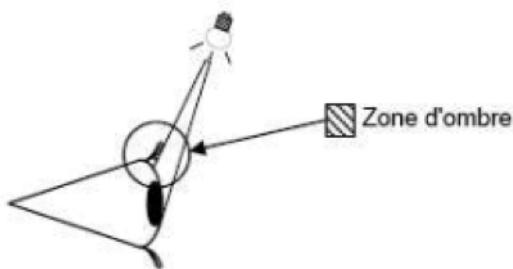


FIGURE 1.13 – Formation d'ombre dans une image d'un œil

1.2.3.2 Reflets

La surface de l'œil étant lisse et recouverte par la cornée, structure transparente recouvrant et protégeant l'iris (figure 1.9), les reflets émanant de l'œil sont considérables. L'intensité des reflets sont fonction de l'intensité lumineuse dégagée par la scène. Plus la scène sera lumineuse, plus elle sera reflétée dans l'œil.

Cependant, ces reflets sont gênants pour le traitement de l'image, et plus particulièrement pour la reconnaissance de la pupille. Par exemple, la figure 1.12(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.

Une attention particulière devra être portée aux éclairages de la pièce afin d'éviter les réflexions lumineuses ambiantes dues aux fenêtres, lampes et néons.. Cependant, il est important de noter qu'une réduction des sources lumineuses n'est pas une solution car l'image perdrait en clarté, 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 morphologiques¹ en dilatant puis érodant la zone atteinte par la réflexion, soit en considérant qu'un reflet possède

1. 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.

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'images). Le seuillage de l'image originale de la pupille et l'image arborant les zones de réflexions seront 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.

1.2.3.3 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.

1.2.4 Détection par éclairage infrarouge

Si la caméra infrarouge permet d'éviter la majorité des problèmes de réflexion rencontrés par une caméra traditionnelle, sa mise en place est néanmoins plus complexe (utilisation d'émetteurs infrarouges afin d'éclairer l'oeil, utilisation de caméras infrarouges, ...) et plus cher. En effet, l'utilisation de caméras infrarouges multiplierait le coût du système par deux ou trois.

La détection de la pupille par illumination de l'œil avec des éclairages infrarouges se décline en deux grandes méthodes : la méthode dite de la pupille lumineuse (bright pupil) et celle de la pupille noire (dark pupil). Suivant la position de l'éclairage infrarouge la pupille aura des propriétés optiques différentes. Si l'illumination infrarouge est coaxiale avec l'axe optique de la caméra alors la pupille apparaîtra totalement blanche (bright pupil). Au contraire si la source lumineuse est décalée (toujours par rapport à l'axe optique), alors la pupille apparaîtra noire (dark pupil) et l'on pourra voir distinctement les reflets de la source lumineuse infrarouge sur la cornée du sujet.

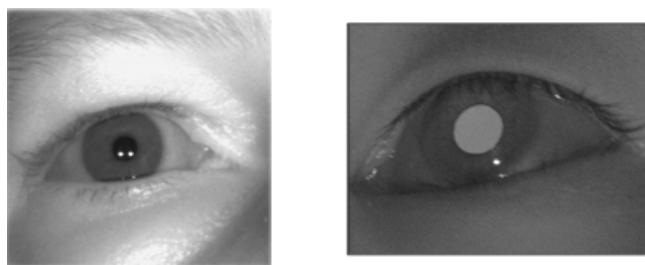


FIGURE 1.14 – Effet black et bright pupil

1.2.5 Traitement d'images

1.2.5.1 Segmentation

La segmentation (aussi appelée Clustering) [3] est une étape de base du traitement d'images 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...). En d'autres mots, cette opération rassemble 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 définie. Concernant les méthodes de segmentation non supervisée, nous nous intéresserons à deux méthodes : le Fuzzy C-means et le k-means.

Description des algorithmes

Les principales étapes d'un algorithme de classification 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.

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.

La classification floue des objets est décrite par une matrice floue à n lignes et c colonnes dans laquelle n est le nombre d'objets de données et c est le nombre de grappes. μ_{ij} est l'élément de la i-ème ligne et la colonne j. μ , représente le degré de fonction d'appartenance de l'i-ème objet avec le pôle j. Les étapes de l'algorithme FCM sont présentées figure 1.15 :

Algorithm 1 Pseudo code de Fuzzy C-Means

```
1: for  $t = 1, 2, \dots$  do
2:   Step1 Calcule des centres de clusters  $c_i^{(t)} = \frac{\sum_{j=1}^N (\mu_{ij}^{(t-1)})^m x_j}{\sum_{j=1}^N (\mu_{ij}^{(t-1)})^m}$ 
3:   Step2 Calcule des distances  $D_{ijA}^2$  avec :

$$D_{ijA}^2 = (x_j - c_i)^T A (x_j - c_i), \quad 1 \leq i \leq n_c, 1 \leq j \leq N.$$

4:   Step3 Mise à jour de la matrice des partitions floues :

$$\mu_{ij}^{(t-1)} = \frac{1}{\sum_{k=1}^{n_c} (D_{ikA}/D_{kjA})^{2/(m-1)}}$$

5: end for
6: return  $\|U^{(t)} - U^{(t-1)}\| < \epsilon$ 
```

FIGURE 1.15 – Pseudo Code de Fuzzy C-Means

Le FCM est un algorithme très puissant, mais son inconvénient principal réside dans l'initialisation des centres.

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. Il renvoie une partition des données, dans laquelle les objets à l'intérieur de chaque cluster sont aussi proches que possible 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 qu'ils 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 apparait 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 Optical Character Recognition OCR, 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 convient plus aux images à faible résolution.

1.2.5.2 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 complexes et meilleures mais plus longues à réaliser (au niveau du temps de calcul). Pour notre projet, nous avons besoin de faire du traitement d'images 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, développée par Jérôme Schmaltz de l'Ecole de Technologie Supérieure de Montréal, repose sur le principe décrit figure 1.16.

Suppression de surplus de contours, atténuation du bruit

Tout d'abord, il est nécessaire de réduire le bruit occasionné par la caméra, le bruit engendré par le transfert des données et leur compression. Différents filtres [1] peuvent être utilisés, même si leur application altérera les contours de l'image.

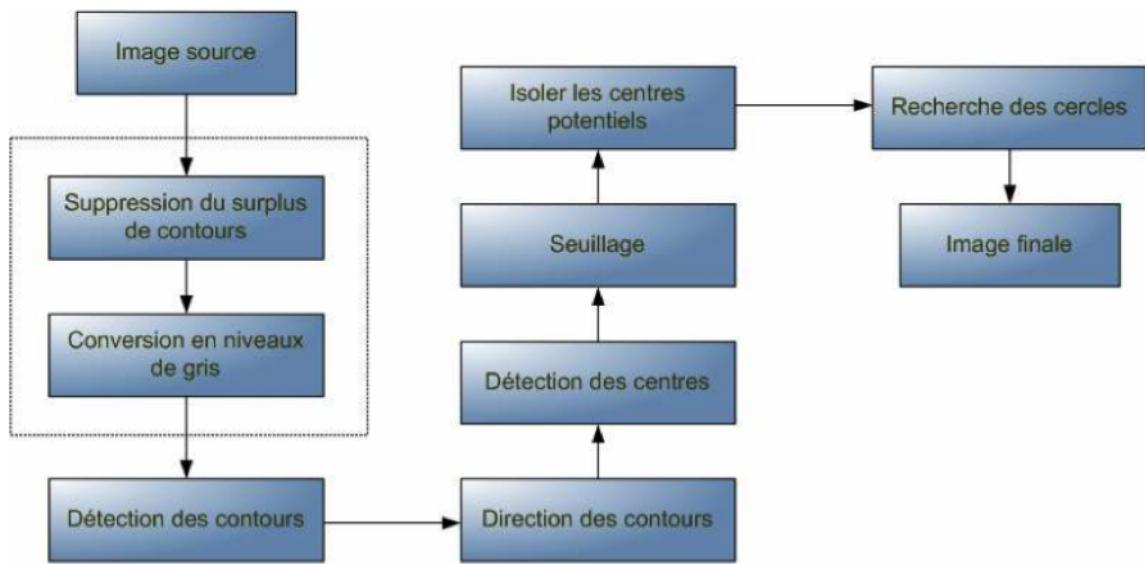


FIGURE 1.16 – Différentes étapes de traitement de l'image afin de détecter une pupille dans une image.

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 soit remplacé par la moyenne pondérée de ses voisins. Si on applique un filtre moyenneur 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 moyenneur 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 permet 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 égale 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 moyenieur, mais la moyenne est pondérée : les pixels près du centre ont un poids plus important 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.

1.2.5.3 Conversion en niveau de gris

La seconde étape du prétraitement consiste à convertir l'image RGB en niveau de gris. Cette conversion est nécessaire puisque les différents algorithmes 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, il devrait plutôt être de la forme : $Gris = 0.299R + 0.587G + 0.114B$. Nous pouvons voir figure 1.17 le rendu de cette conversion.

1.2.5.4 Détection des contours

Filtre de Canny

Il existe différentes méthodes de détection des contours. Des filtres peuvent être utilisés (voir paragraphe 1.2.5.2). Le choix du filtre doit prendre en compte

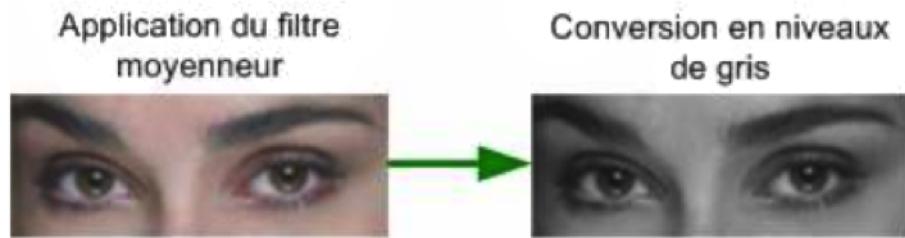


FIGURE 1.17 – Conversion en niveau de gris

qu'une réduction trop importante du bruit masque certains contours, mais qu'une réduction trop faible peut laisser apparaître trop de contours non significatifs. Nous nous intéresserons ici au meilleur filtre pour la détection de contours : le filtre de Canny. Il est considéré comme tel car il offre un très bon compromis entre la réduction du bruit et la localisation des contours. 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 (1.1) et (1.2).

$$g = \sqrt{g_x^2 + g_y^2} \quad (1.1)$$

$$\theta = \arctan \frac{g_y}{g_x} \quad (1.2)$$

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 1.18 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 1.18 – 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 des gradients maximaux engendrés par le bruit). De plus, l'application d'un seuil trop élevé engendre une fragmentation des chaînes de pixels qui forment des contours significatifs de l'image. Ainsi, le seuil par hystérisis offre une bonne solution à ce problème.

La figure 1.19 illustre l'idée de détection des contours grâce au filtre de Canny.

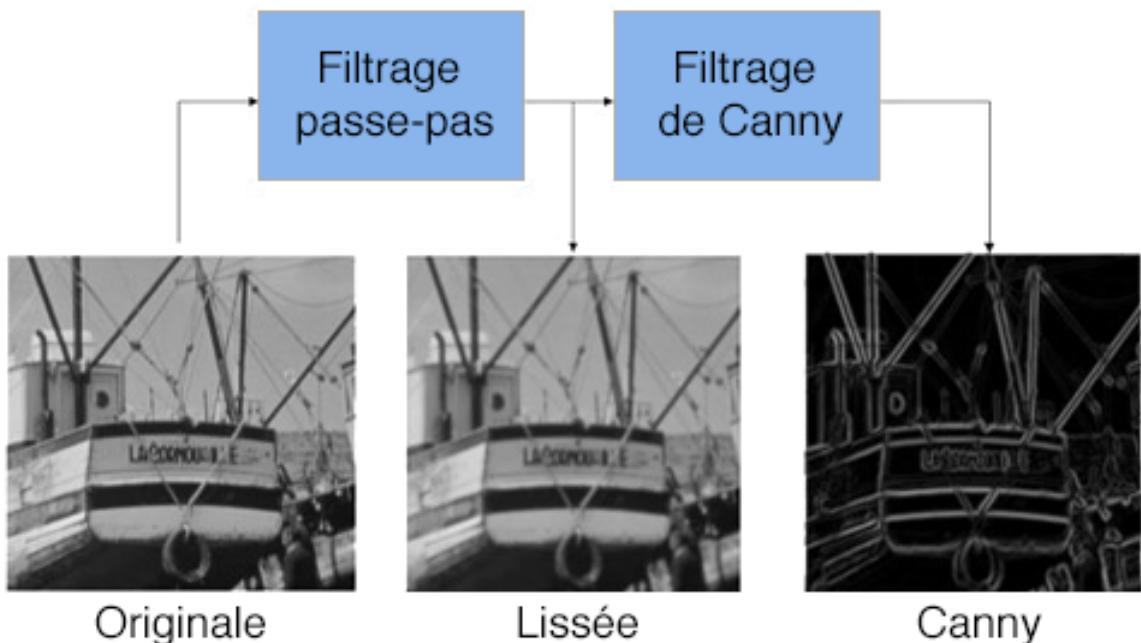


FIGURE 1.19 – Détection des contours grâce au filtre de Canny

Filtre de Sobel

L'opérateur calcule le gradient de l'intensité de chaque pixel. Celui-ci 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 : $\Theta = \text{atan}2(G_y, G_x)$, et créer une matrice des directions.

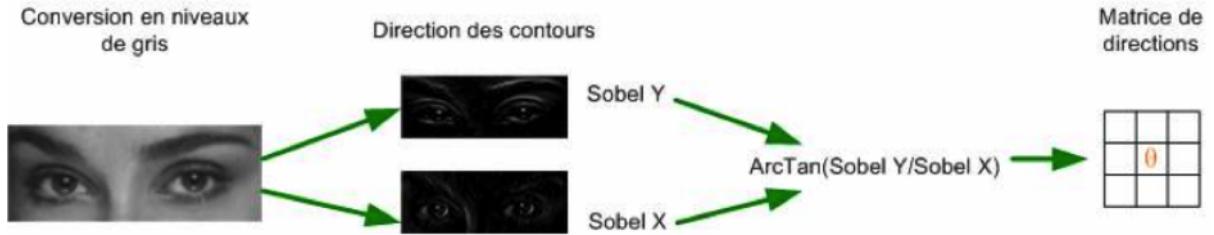


FIGURE 1.20 – Déterminer les directions des gradients à partir de l’application du filtre de Sobel

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

1.2.5.5 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 1.21.

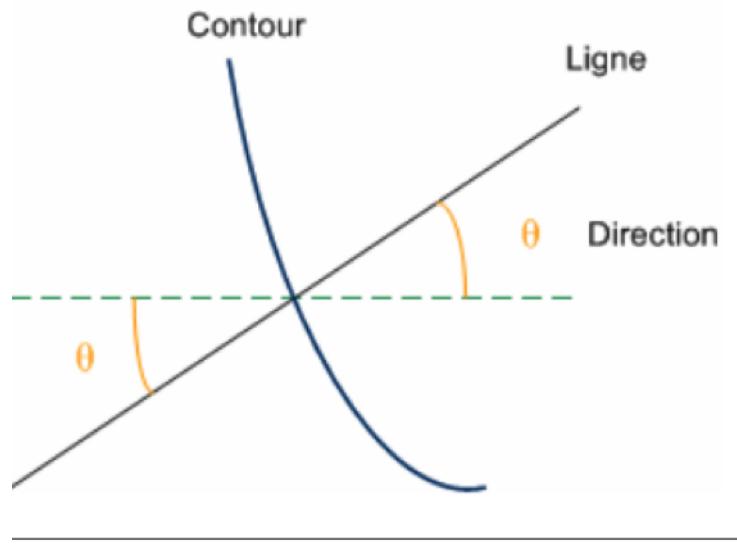


FIGURE 1.21 – Traçage d’une droite en fonction d’un contour

Tracer une droite dans un accumulateur revient à incrémenter chaque cellule d’une matrice aux endroits où elle passe (voir figure 1.22).

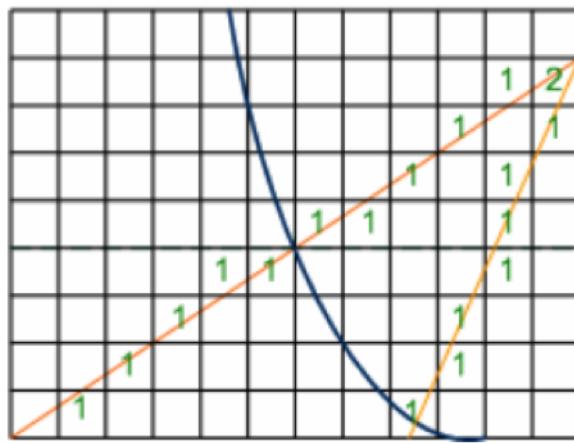


FIGURE 1.22 – Traçage d’une droite dans l’accumulateur

Ainsi, suivant ce principe, les valeurs maximales de la matrice indiqueront les différents centres potentiels. La figure 1.23 synthétise la méthode énoncée.

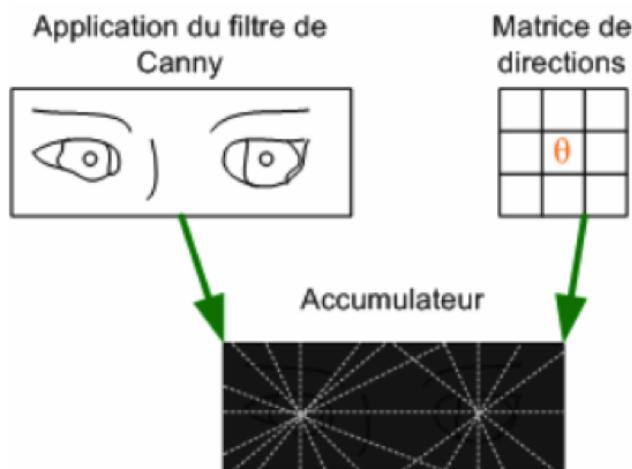


FIGURE 1.23 – Traçage des lignes dans l’accumulateur en fonction de l’image de Canny et de la matrice de directions

1.2.5.6 Seuillage

Le seuillage permet de désencombrer la matrice accumulateur des informations superflues. 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 supérieure à celui-ci (les autres pixels seront mis à zéro) car ils représenteront les différents centres potentiels.

Pour ce faire, nous pouvons par exemple utiliser un seuil du type :
 $\text{valeur pixel actuel} > 0.5 \cdot \max(\text{valeur pixel image})$, 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 1.24 le résultat du seuillage.

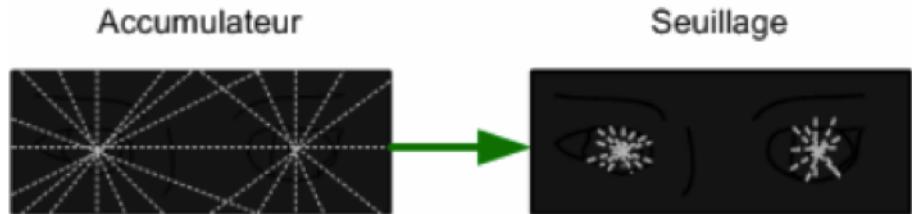


FIGURE 1.24 – Application d'un seuil dans l'accumulateur

1.2.5.7 Isoler les centres potentiels

L'application d'une convolution d'un chapeau mexicain (voir formule (1.3) et figure 1.25) sur l'accumulateur permet d'isoler les points les plus au centre. Un chapeau mexicain est la combinaison d'un filtre Gaussien (voir paragraphe 1.2.5.2) avec un filtre Laplacien.

$$\nabla^2 G_\sigma(r) = \frac{-1}{2\pi\sigma^4} \left(2 - \frac{r^2}{\sigma^2}\right) e^{-\frac{r^2}{2\sigma^2}} \quad (1.3)$$

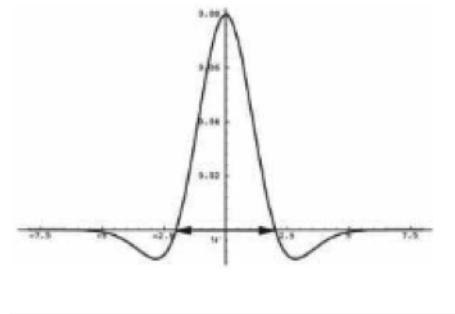


FIGURE 1.25 – Représentation graphique d'un Laplacien de Gaussien

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 1.26 présente le résultat de cet isolement des centres.

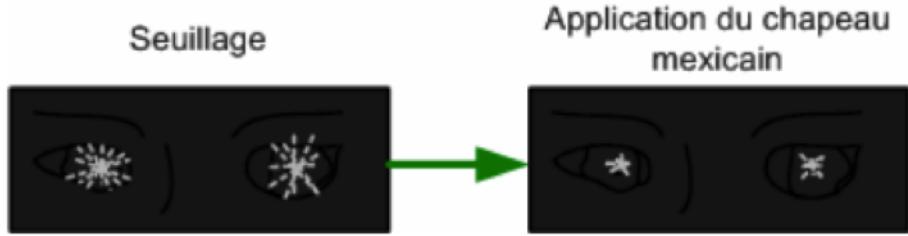


FIGURE 1.26 – Application d'un chapeau mexicain dans l'accumulateur

1.2.5.8 Recherche des cercles

La recherche des cercles s'appuie sur la création d'accumulateurs et la transformée de Hough. En effet, pour un 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 contour 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 1.27, 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.

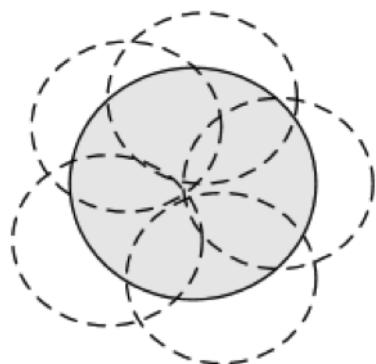


FIGURE 1.27 – Traçage des cercles avec l'image des contours

4. Ensuite, on peut diviser la zone d'intérêt de l'image en deux parties : partie droite et partie gauche du visage. 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 restants 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 1.28 résume cette démarche.

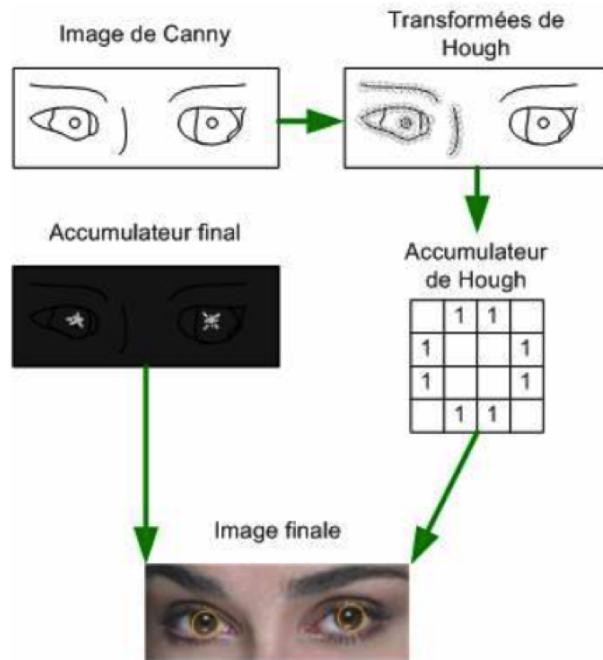


FIGURE 1.28 – Les différentes opérations permettant de localiser les pupilles

1.3 Méthode de détection de la direction du regard

La direction du regard d'une personne peut être obtenue à l'aide de deux paramètres : L'orientation du visage et l'orientation oculaire. La détection du visage permet de connaître l'orientation globale de l'utilisateur et de vérifier que ses deux yeux sont bien en face de la caméra. Suivant l'orientation de la tête de l'utilisateur, une approximation de la direction du regard peut aussi être réalisée. La détection effectuée à l'aide de l'orientation oculaire approche la direction du regard à l'aide de propriétés géométriques et physiques de l'iris et de la pupille. Le problème principal réside dans le fait que cette étude est une étude locale, c'est-à-dire que la caméra qui filme l'œil de l'utilisateur doit avoir un fort niveau de zoom, obligeant le sujet à rester immobile afin de ne pas sortir du champ de la caméra.

Les reflets détectés dans l'œil du sujet par la détection de pupille avec infrarouge (voir paragraphe 1.2.4) vont permettre de déterminer la direction du

regard de l'utilisateur. En supposant la tête de l'utilisateur immobile, et en prenant comme référence le reflet formé par la lumière infrarouge sur la cornée, on peut déterminer la direction du regard en se servant du vecteur créé entre le reflet et le centre de la pupille (ou l'iris) (voir figure ci-dessous). Cependant cette méthode demande certains prérequis et notamment une calibration et ce avec chaque nouvel utilisateur. Pour effectuer cette calibration, il est demandé au sujet de fixer quatre points (minimum), situés sur l'écran, afin d'obtenir des valeurs étalons (souvent les coins de l'écran), qui permettront de déduire la direction du regard par la suite.

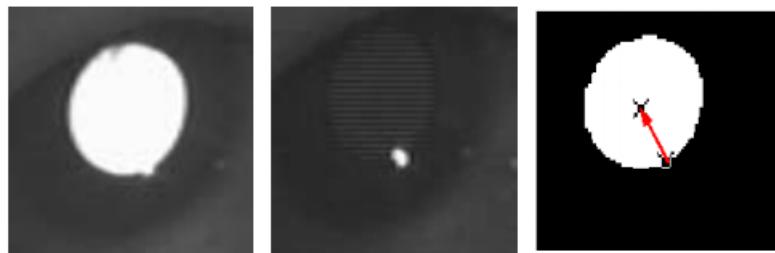


FIGURE 1.29 – Tracé du vecteur entre la réflexion infrarouge dans la cornée et la pupille

Cette méthode présente tout de même deux défauts majeurs : l'impossibilité pour l'utilisateur de bouger la tête ou le corps, sous risque de devoir recommencer la calibration, et l'obligation de re-calibrer le système à chaque changement d'utilisateur.

1.4 Solutions techniques et méthodes envisagées

1.4.1 Au préalable, suite à l'état de l'art

Suite à l'état de l'art, la méthode d'eye tracking sans contact a été retenue (pas de système embarqué). Elle est similaire à celle de la partie 1.1.2.3 et semble plus facilement réalisable qu'un système utilisant des lunettes avec caméras embarquées. Ainsi, dans un premier temps, cette technique sera développée afin d'obtenir un système qui fonctionne correctement. Puis, le système pourra être perfectionné et, s'il reste du temps sur les délais fixés, la méthode utilisant des lunettes sera approfondie.

Pour l'approche sans contact, deux caméras seront utilisées. La première, la caméra dite "grand angle", permettra de filmer la scène dans son ensemble et de localiser le visage ainsi que certains de ses éléments morphologiques tels que les oreilles et la bouche. Elle transmettra ensuite les coordonnées spatiales du visage à la seconde caméra "petit angle". Cette dernière zoomera sur le visage (grâce aux coordonnées fournies) et effectuera un tracking des pupilles.

La détection de la pupille s'effectuera à l'aide de la méthode de la bright et de la black pupille. Les différentes étapes vues dans l'état de l'art seront respectées et des tests seront menés afin de déterminer la meilleure solution pour réaliser chaque étape.

Le calcul de la direction du regard s'appuiera sur la méthode vue dans la partie [1.3](#). Des LEDs infrarouges, placées devant l'utilisateur, illumineront sa cornée, créant une réflexion spéculaire exploitable pour estimer la direction du regard. La caméra "petit angle" devra ainsi être munie d'un filtre infrarouge. Ce filtre pourra être réalisé à l'aide d'une pellicule photo placée devant l'objectif de la caméra.

1.4.2 Méthodes choisies lors de la réalisation

Devant notre effectif réduit lors de la deuxième phase du projet, nous avons décidé de nous concentrer sur l'utilisation d'une seule caméra combinant "grand angle" et "petit angle". En effet, cela nous a permis d'éliminer tout problème de calibrage entre les deux caméras. De plus, nous avons constaté que la résolution de notre webcam était suffisamment élevée pour la détection du centre des pupilles. Ainsi, puisque l'infrarouge, qui permet certes une détection plus précise, est inconfortable et moins robuste à la lumière du jour, nous avons décidé d'abandonner cette technologie dans un premier temps.

Partie 2

Dossier fonctionnel

2.1 Ingénierie des exigences

2.1.1 Approche Top-Down

Pour notre approche Top-Down, nous sommes revenus à la demande initiale du projet : remplacer la souris d'un ordinateur par le mouvement oculaire de l'utilisateur. La fonction principale du système est donc apparue clairement : permettre à l'utilisateur d'interagir avec une interface via ses yeux.

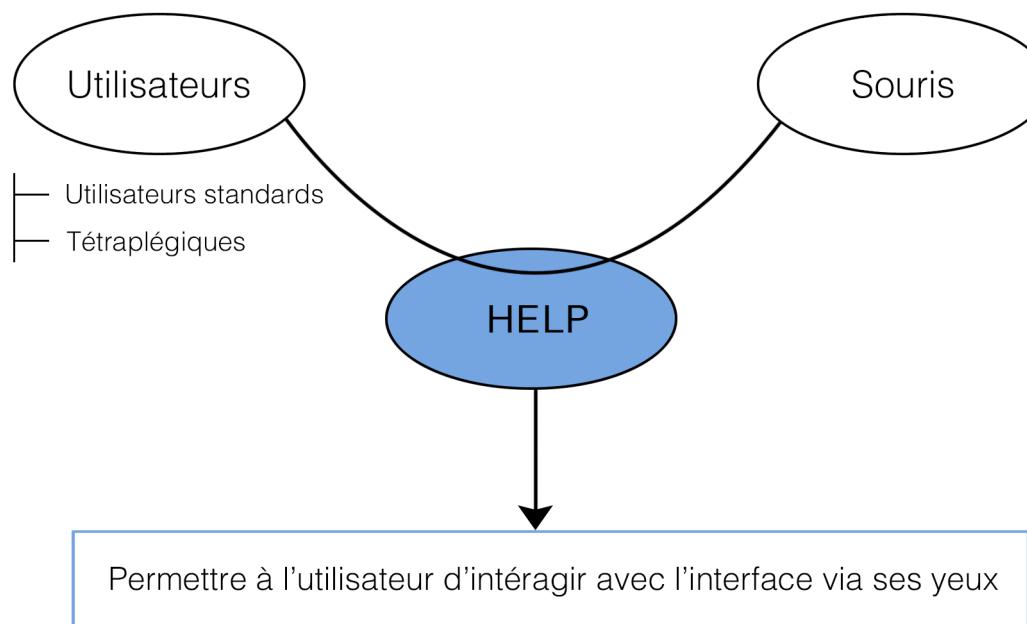
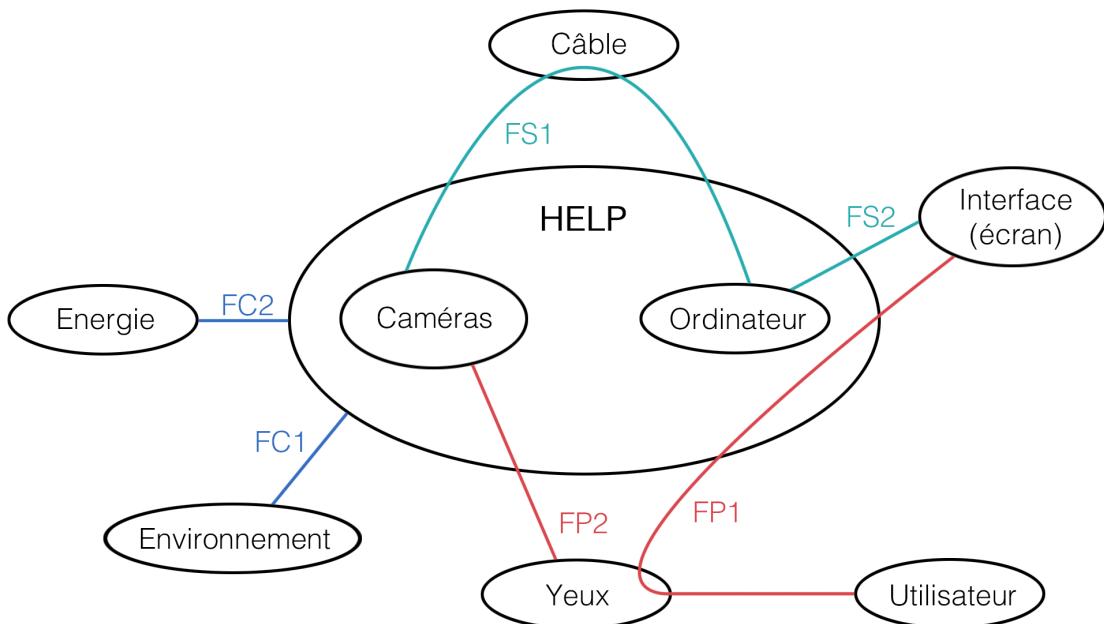


FIGURE 2.1 – Bête à cornes



FONCTIONS PRINCIPALES

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

FP2 : Repérer et filmer les yeux de l'utilisateur

FONCTIONS CONTRAINTES

FC1 : Résister à l'environnement

FC2 : S'adapter à l'alimentation

FONCTIONS DE SERVICE

FS1 : Transmettre des informations à l'ordinateur

FS2 : Proposer une IHM adaptée

FIGURE 2.2 – Diagramme pieuvre

2.1.2 Approche Bottom-Up

Nous avons défini les différentes exigences auxquelles le système devra répondre, indépendamment de nos choix de réalisation technique. Nous les avons alors rassemblées par groupement logique, ce qui nous permettra de définir nos fonctions principales. Les lignes en ■ sont uniquement valables pour la méthode embarquant un système sur l'utilisateur. Elles sont donc à ignorer dans un premier temps puisque nous avons choisi une approche différente.

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 3h	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	Extraire les repères anatomiques
1.3.1.1	Service	DéTECTER le visage de l'utilisateur	5	5	
1.3.1.2	Service	DéTECTER les yeux de l'utilisateur	5	5	
1.3.1.3	Service	DéTECTER les pupilles de l'utilisateur	5	5	
1.3.1.4	Service	DéTECTER le mouvement des pupilles de l'utilisateur	5	5	
1.3.1.5	Service	DéTECTER le clignement des yeux de l'utilisateur	4	3	
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	Interpréter les données pour déduire l'action à exécuter
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 2.3 – Cahier des exigences

2.1.3 Fonctions principales du système

Dans le cadre de ce projet, nous cherchons donc à remplacer la souris d'un ordinateur par un système qui suit le mouvement des yeux de l'utilisateur. Pour ce faire, nous avons mis en évidence des groupements logiques d'exigence. Tout d'abord, le système doit acquérir les données nécessaires à la détection du mouvement de l'œil. Ces données devront alors être traitées par l'ordinateur. Ce dernier doit alors interpréter ces données pour en déduire l'action à exécuter. De ces nouvelles informations, l'ordinateur doit pouvoir effectuer l'action que l'utilisateur veut effectuer sur l'IHM, la mettre en place, et montrer que ces modifications ont été exécutées.

- FP1 : Acquérir les données nécessaire à la détection du mouvement de l'œil
- FP2 : Transférer les données enregistrées
- FP3 : Extraire les repères anatomiques
- FP4 : Interpréter les données pour déduire l'action à exécuter
- FP5 : Agir sur l'IHM

2.2 Spécification fonctionnelle 3 axes

2.2.1 Raffinement FAST

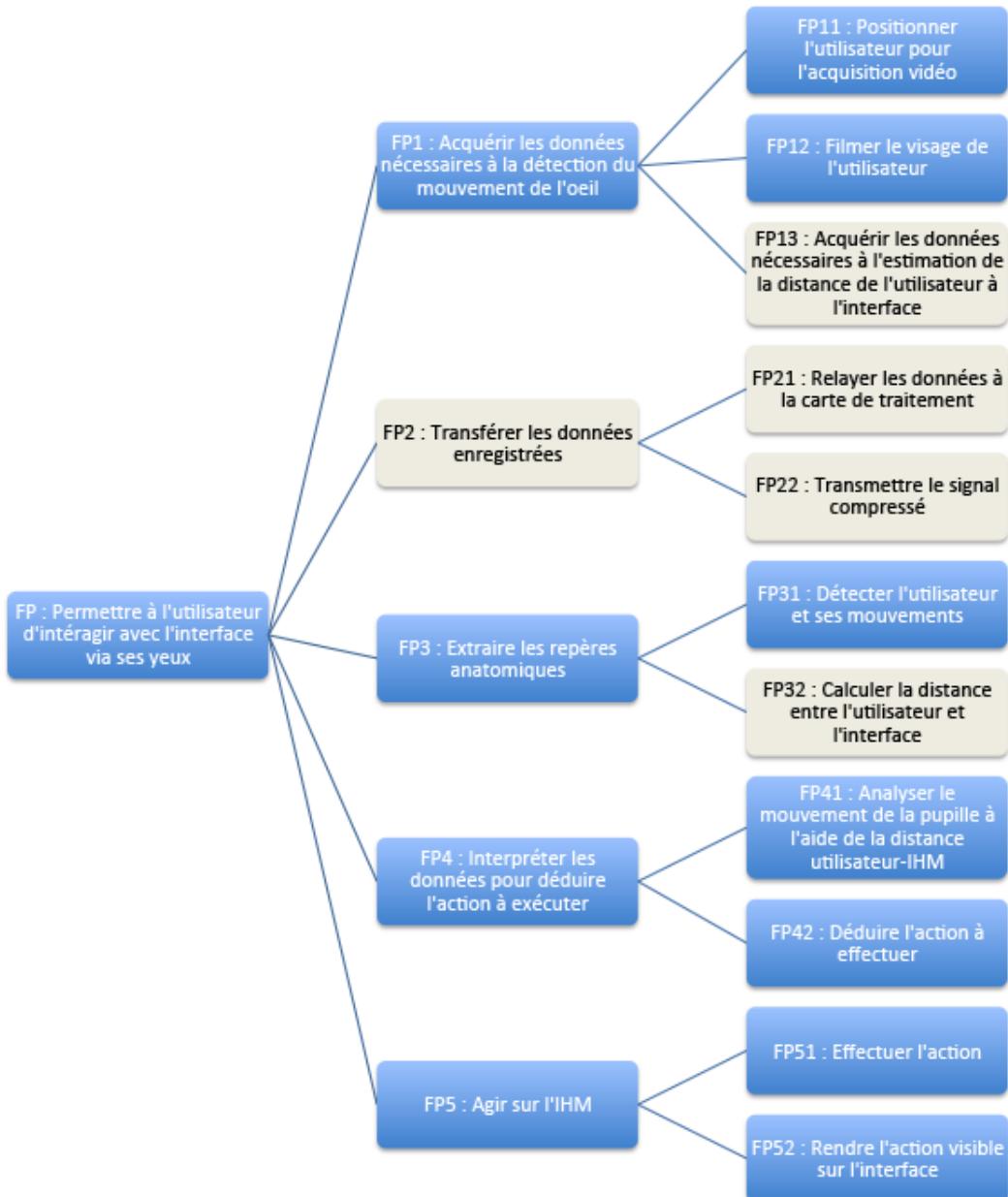


FIGURE 2.4 – FAST raffiné

Le cahier des exigences précédent nous a permis de définir les fonctions principales du système. Nous avons alors cherché à les raffiner par la méthode FAST (cf. figure 2.4) pour obtenir des fonctions plus précises auxquelles nous pourront

apporter des solutions techniques propres à chacune. Nous pouvons ainsi entrevoir l'architecture fonctionnelle.

2.2.2 Spécification des données

La spécification du flux de données va permettre de comprendre les interactions et les échanges entre les différentes parties de notre système. Spécifier ainsi les données nous a permis de mieux comprendre les interactions entre chaque sous-partie du système. Il est ainsi clair que les données que nous allons principalement traiter et transférer seront des flux vidéos et qu'il sera donc important d'optimiser au mieux la rapidité de ces échanges. Au travers de la figure 2.5, nous pouvons aussi voir l'ordre avec lequel les sous-systèmes vont rentrer en jeu. Ainsi la Camera Narrow View sera dépendante du flux envoyé par la Camera Wide View. Un parallélisme de traitement des données est tout de même envisageable sur l'ordinateur pour gérer le flux vidéo de deux caméras.

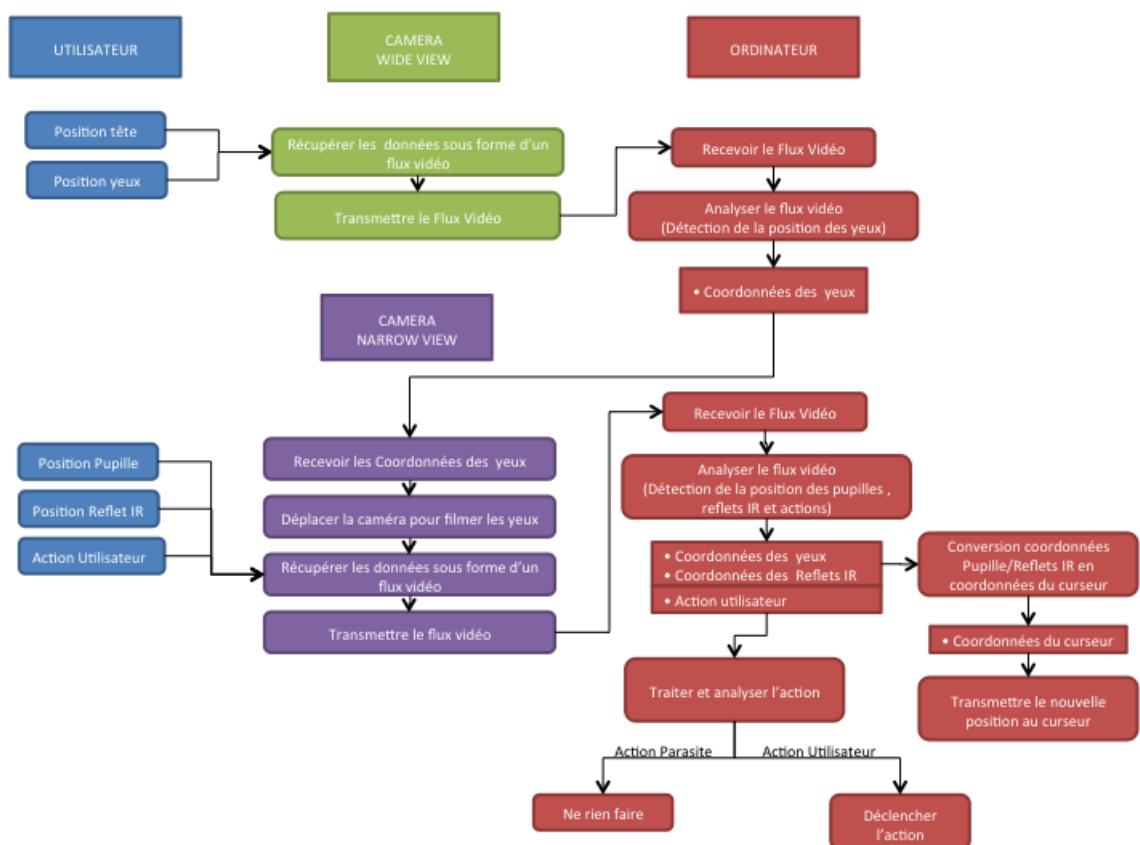


FIGURE 2.5 – Flux de données

2.2.3 Spécification des comportements

Le diagramme de séquence de la figure 2.6 illustre le comportement global du système.

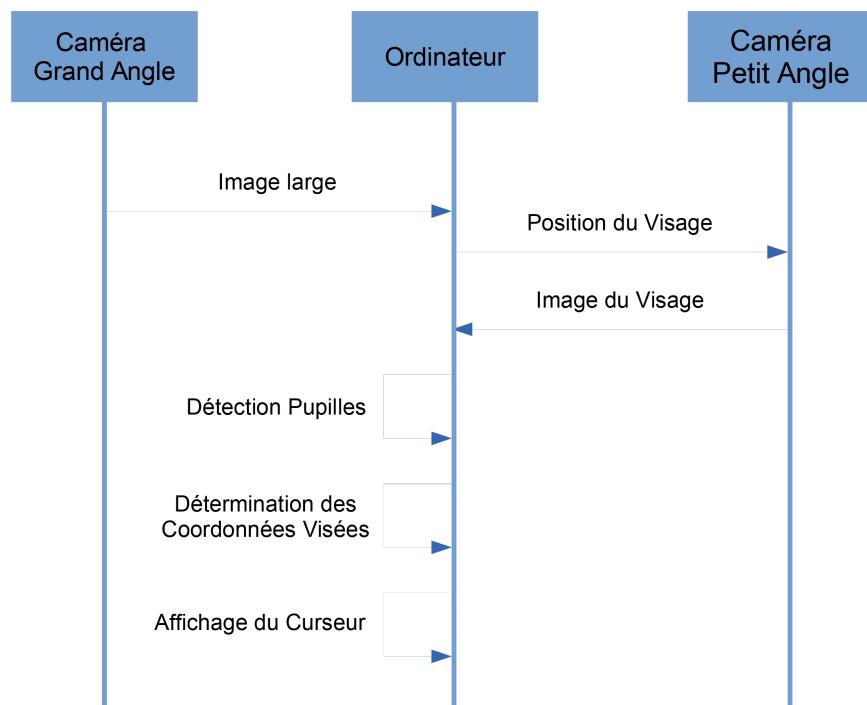


FIGURE 2.6 – Diagramme des spécifications du comportement global

2.3 Architecture fonctionnelle

Tout le travail réalisé au préalable sur la spécification fonctionnelle 3 axes permet finalement de proposer une architecture fonctionnelle pour notre système (cf. figure 2.7).

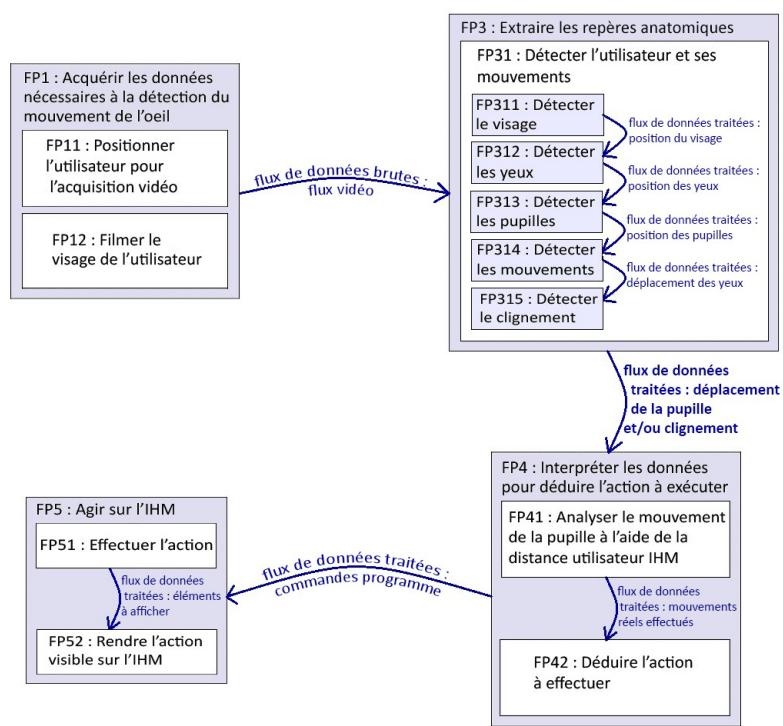


FIGURE 2.7 – Architecture Fonctionnelle

Partie 3

Implémentation

3.1 Architecture physique et interfaces

Grâce à l'architecture physique (cf figure 3.1), nous avons pu déduire que notre projet se décomposait en trois sous-systèmes : les interfaces d'acquisition, logicielle et graphique.

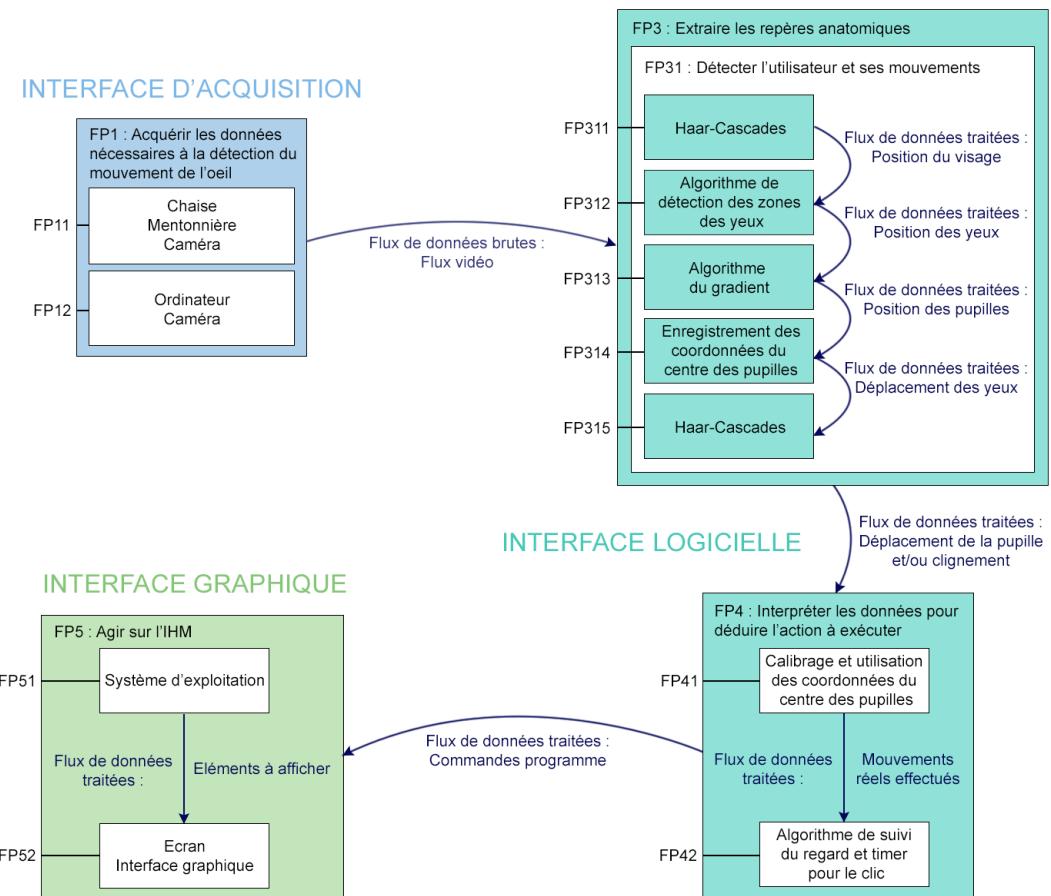


FIGURE 3.1 – Architecture Physique

3.1.1 Interface d'acquisition

Cette interface est constituée principalement de la caméra et de la mentonnière. Cette dernière est utilisée pour éviter les mouvements de la tête, non traités par le programme. L'ordinateur occupe également une place secondaire, puisqu'il permet de vérifier que l'utilisateur est convenablement installé.

3.1.2 Interface logicielle

L'interface logicielle s'appuie sur le programme contenant des algorithmes développés en C++ avec OpenCV. Ils ont pour but de traiter les images transmises par la webcam afin de détecter le visage, puis les yeux et enfin les pupilles et les clignements pour déduire l'action que l'utilisateur souhaite effectuer. Que ce soit pour le clic ou le mouvement du curseur de la souris, une interaction avec l'OS est nécessaire. Le programme doit donc être adapté suivant le système d'exploitation utilisé.

3.1.3 Interface graphique

Enfin, le rôle de l'interface graphique est de proposer à l'utilisateur un choix de boutons qui lancent des applications. Cette dernière est développée en C++ à l'aide de Qt Creator et ne peut être utilisée que sur Windows car il faudrait changer toutes les commandes d'ouverture d'applications pour exécuter le programme sur un autre système d'exploitation.

3.2 WBS

La structure de découpage du projet découle de l'architecture physique. La figure 3.2 illustre le WBS ainsi établit.

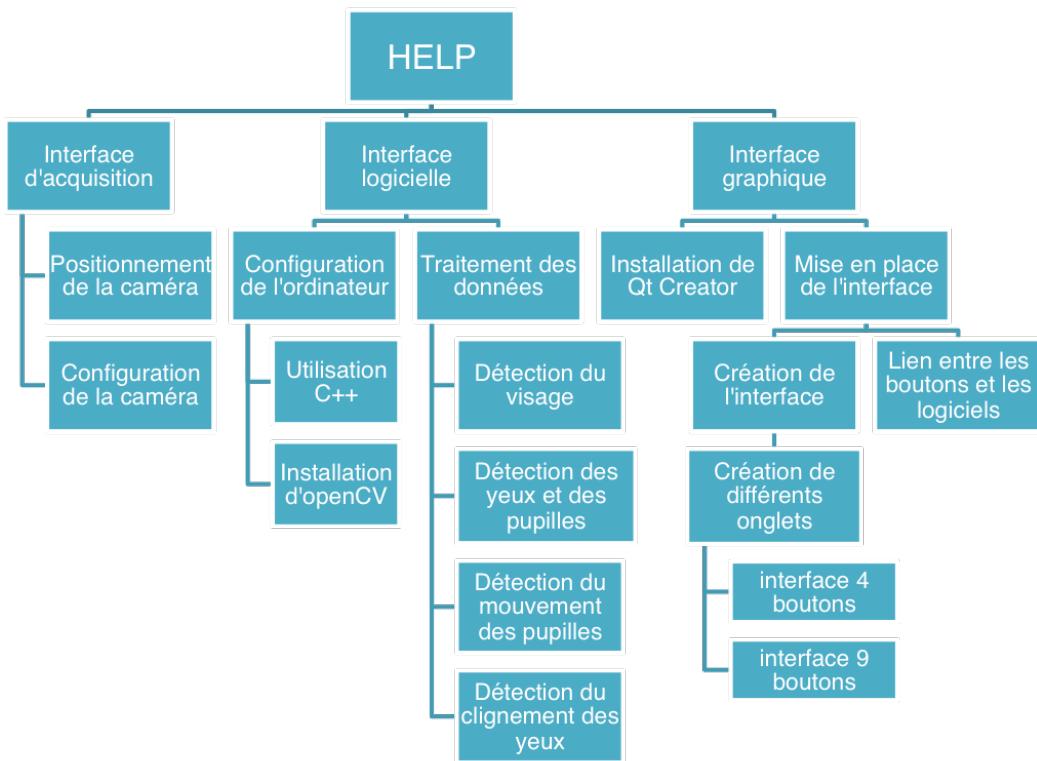


FIGURE 3.2 – WBS

Partie 4

Organisation

4.1 Méthodes de travail

Pour ce deuxième semestre, notre équipe s'est vue diminuée de deux membres : Marie-Alice Schweitzer et Laure Dupasquier. Nous avons donc dû totalement remanier notre système de fonctionnement. Notre groupe s'est séparé en deux parties distinctes. D'un côté Katleen Blanchet, Titouan Boulmier et Pierre Jacquot ont travaillé sur la réalisation concrète du projet décrit dans le premier rapport d'avancement. Cela consistait à la création d'un système composé d'une webcam relié à un ordinateur, permettant d'enregistrer la direction du regard de l'utilisateur et de déplacer le curseur sur l'écran en fonction de cette direction. (Les tâches spécifiques données à chacun seront décrites par la suite).

Et de l'autre Hussain Al Othman a été chargé de réaliser des recherches sur un second système. Ce second système est directement embarqué sous forme de lunettes sur l'utilisateur. Cette partie vient compléter notre rapport, en donnant une alternative à notre prototype, mais n'est pas réalisée et reste purement théorique. Elle permet d'offrir une alternative à notre projet, si celui-ci est poursuivi par la suite.

Pour ce qui est des réunions, elles étaient conduites tous les mardi matin, afin de se répartir les tâches et aussi de connaître l'avancement de chacun dans son travail. Le rôle de secrétaire, autrefois occupé par Laure Dupasquier a été effectué par Pierre Jacquot afin de garder une trace de chaque réunion et de planifier au mieux nos actions futures.

Katleen Blanchet et Titouan Boulmier ont continué de s'occuper de la rédaction et de la mise en page du rapport ainsi que de la mise à jour de la page GitHub. Les livrables réalisés au cours de ce semestre ont été réalisés en majorité par Pierre Jacquot et relus par le reste de l'équipe.

4.2 Outils pour les échanges

Concernant ce semestre les échanges ont été beaucoup plus simples à effectuer. Étant trois à travailler sur la partie principale du projet, la répartition du travail a été rapide et efficace. Le déroulement des séances se faisait le matin à notre

arrivée, et nous nous adaptions au fur et à mesure de la journée au travail qu'il restait à faire. La majorité des échanges s'est fait via e-mail.

En dehors du projet, pour les questions de logistique (horaires de travail, lieux), nous restions en contact par mail.

4.3 Répartition des tâches dans le temps

Afin de visualiser au mieux les tâches à effectuer dans le temps nous avons réalisé un diagramme de Gantt (voir figure 4.6).

Semaine	Jour	Travail à effectuer durant la séance	Travail réalisé	Tâches à réaliser lors de la prochaine séance
37	lundi 08		Introduction de l'UV et de l'IS	
	jeudi 11		Cours analyse fonctionnelle	
38	lundi 15		Cours fonctions système	
	jeudi 18		Rencontre avec les encadrants pour la présentation des projets	
39	lundi 22		Choix sujet et constitution des groupes + prise de contact avec les membres du groupe (méthodes de travail, rythme...)	
	jeudi 25	Se répartir les tâches pour bien préparer l'interview	Préparation de l'interview avec Mr Mansour	Prévoir le matériel pour l'interview
40	lundi 29	Interview + restitution	Interview	Poster l'interview sur moodle
	jeudi 02	Exigences	Travail en commun pour lister les fonctions, exigences du système	Continuer l'expression des besoins
41	lundi 06	Mise en place de la répartition des activités	Début état de l'art	
	jeudi 09		Etat de l'art	
42	lundi 13		Atelier	
	jeudi 16		Pas cours : journée internationale	
43	lundi 20		Atelier	
	jeudi 23		Etat de l'art	
45	lundi 03		Atelier	
	jeudi 06		Restitution de chaque atelier (explication Github, simulink) à tous les membres du groupe	Faire un point état de l'art
44	vacances	Continuer l'état de l'art chacun de son coté		
46	lundi 10	Pas cours		
	jeudi 13	Point état de l'art	Redistribution état de l'art + raffinement	Continuer le passage de la spécification à l'architecture fonctionnelle
47	lundi 17		Etat de l'art	Commencer la rédaction du rapport
	jeudi 20		Etat de l'art	
48	lundi 24	Répartition des parties du rapport d'avancement	Rapport avancement + état de l'art	
	jeudi 27		Rapport avancement + état de l'art	
49	lundi 1		Rapport avancement	
	jeudi 4		Rapport avancement + choix du matériel	
	vendredi 5		Rapport avancement + commande du matériel	
50	lundi 8		Début des tests de simulations et de codage	
	jeudi 11		Codage et simulation	
51	lundi 15		Codage et simulation	
	jeudi 18	Préparation et organisation de notre présentation projet pour les portes ouvertes		
2	lundi 5		Finalisation du rapport avant de le poster	
	jeudi 8		Atelier II	Poster le rapport avant le dimanche 23
3	lundi 12		Atelier II + évaluation P2P	
	jeudi 15		Atelier II	

FIGURE 4.1 – Carnet de bord semestre 1

5	mardi 27		codage (sur l'existant)	Rédaction livrable sur la réunion de lancement
6	mardi 03		Modification du rapport, codage, livrable	Risque infrarouge
7	mardi 10	Travail sur l'architecture physique/Analyse de codes/Algorithmes existantes	Architecture physique possible établie/Algorithme trouvé	Démarrage Analyse Fonctionnel
8	mardi 17		Travail sur des algorithmes dans différents langages (Matlab, simpleCV, open CV)	Rédaction de l'Analyse Fonctionnel
9	mardi 24		Rédaction de l'Analyse Fonctionnelle/Compréhension de l'algorithme en opencv	
10	mardi 3		Rencontre avec Ali Mansour/Travail sur le code de détection de pupille (travail sur une image plus grosse et suppression de la partie détectant la tête)/ Recherche d'une caméra Narrow View/ TD Gestion de groupe	
11	mardi 10	Trouver une caméra Narrow View	Calcul de la résolution minimale pour la caméra Narrow View/Rencontre avec Mr Reynet afin de commander une caméra	Loi de commande pour servomoteur à déterminer
13	mardi 24	Rédaction livrable 2ème compte-rendu avec l'encadrant	Rencontre avec Ali Mansour, Détection de la direction du regard/Détection du clignement des yeux/Interface Qt	
14	mardi 31		Détection de la direction du regard/Détection du clignement des yeux/Interface Qt	
15	mardi 07		Auto-évaluation de groupe/Détection de la direction du regard/Interface Qt	Avancer les deux parties pendant les vacances
17	mardi 21	Travail sur l'interface Qt /Test de la nouvelle caméra	Avancement de l'interface Qt/Amélioration de la détection avec la nouvelle caméra	
18	mardi 28	Rédaction livrable 3ème compte-rendu avec l'encadrant	Rencontre avec Ali Mansour et M Reynet concernant le projet et la rédaction de la validation fonctionnelle	Finalisation de la validation fonctionnelle/Réalisation Tests Unitaires
19	mardi 05	Centrage de l'image filmée par la caméra/Tests Unitaires / Validation Fonctionnelle	Rédaction de la validation fonctionnelle/Finalisation de l'interface graphique/ Réalisation et Rédaction de tests unitaires	Réalisation de la validation fonctionnelle
20	mardi 12	Avancement du rapport/ Réalisation de tests fonctionnels	Détection de la direction du regard/Interface Qt	Finalisation du rapport/ Création de l'eportfolio

FIGURE 4.2 – Carnet de bord semestre 2

Ce diagramme nous a permis d'anticiper au mieux les échéances, et de commencer à réfléchir aux livrables en avance. En plus de la réunion de planification, nous faisions un point sur l'avancement de chacun tous les matins.

Il est vrai que notre faible nombre a rendu l'organisation compliquée, car l'un de nous devait régulièrement interrompre ses avancées sur la partie technique, qui était pourtant cruciale car non débutée (ou peu) au premier semestre, afin de rédiger des livrables. L'équipe fonctionnait donc à 2/3 de ses possibilités sur la partie technique la plupart du temps au cours du mardi.

Etapes	Date de début	Réalisé (en jours)	Restant à faire (en jours)	Date de fin
Cours IS	08/09/14	10	0	18/09/14
Choix sujet	22/09/14	1	0	22/09/14
Interview encadrant	25/09/14	11	0	05/10/14
Atelier technique I	06/10/14	16	0	03/11/14
Etat de l'art	22/09/14	40	0	27/11/14
Analyse fonctionnelle	03/11/14	56	0	11/01/15
Rapport avancement	06/10/14	82	1	15/01/15
Raffinement	03/11/14	56	0	11/01/15
Achat matériel	05/12/14	4	0	09/12/14
codage et simulation	08/12/14	7	1	15/12/14
Atelier technique II	08/01/15	7	1	15/01/15

FIGURE 4.3 – Diagramme de Gantt semestre 1

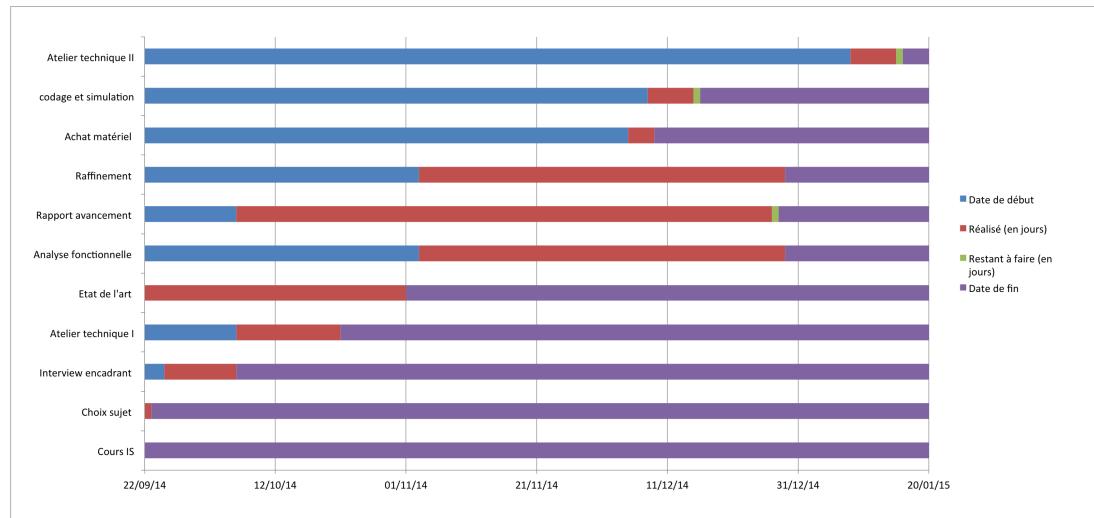


FIGURE 4.4 – Diagramme de Gantt version graphique semestre 1

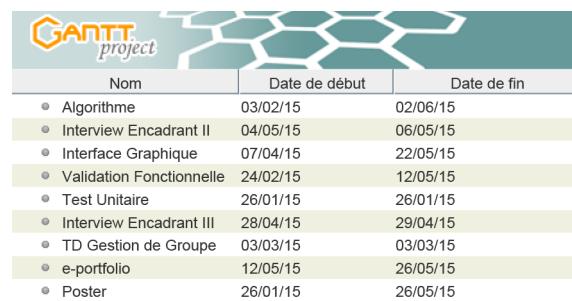


FIGURE 4.5 – Diagramme de Gantt semestrelle 2

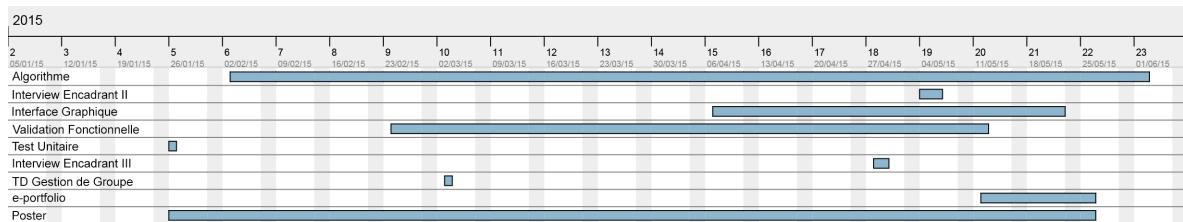


FIGURE 4.6 – Diagramme de Gantt version graphique semestrelle 2

Partie 5

Journal du projet

5.1 Choix et justifications

5.1.1 Émetteurs infrarouges

Concernant les émetteurs infrarouges, des LEDs pourront être utilisées. En effet, la distance écran/utilisateur ne sera pas très grande, et de simple LEDs seront suffisantes pour éclairer la pupille de l'utilisateur. Nous pourrons par exemple nous servir de l'Émetteur infrarouge (IR) Kingbright L-934SF4BT (880 nm 50 ° 3 mm) ou alors de l'Émetteur infrarouge (IR) Harvatek HT-159IRAJ 940 nm 20 ° 1206 CMS coûtant respectivement 0.90€ et 0.35€ pièce. Le choix se fera en fonction des connectiques nécessaires (la première LED offre une sortie radiale avec des connecteurs UY2, alors que la seconde se présente sous forme de puce).

5.1.2 Caméras

Si de nombreux types de caméras sont possibles, nous avons retenu la webcam. En effet, la résolution des webcams HD actuelles est largement suffisante pour une détection de pupilles. De plus, elles sont facilement utilisables via des ordinateurs, c'est-à-dire simple d'installation et il est aisément de récupérer les flux vidéos. Leur connexion USB facilite leur emploi. Nous pouvons également noter que les webcams sont relativement peu coûteuses.

5.1.2.1 Caméras «grand-angle»

Pour le suivi du visage, la Webcam LOGILINK USB avec LED a été retenue. Elle est compatible avec Windows, Mac et Ubuntu et ne posera donc pas de problème de compatibilité. De plus elle possède un système lui laissant la possibilité de pivoter sur 360° permettant ainsi le meilleur suivi du visage possible. Cependant, d'autres webcams présentent ces avantages et le détail qui nous a décidés dans ce choix est la présence de LEDs (voir figure 5.1). En effet, trois LEDs sont présentes de chaque côté de l'objectif et pourraient éventuellement être remplacées par des émetteurs infrarouges.



FIGURE 5.1 – Présence de LEDs sur la webcam

Le placement des LEDs sur la webcam pourrait nous éviter de devoir mettre en place un système pour pouvoir brancher ces LEDs.

5.1.2.2 Caméras «petit-angle»

Ensuite, concernant la caméra petit-angle, la Webcam Trust Widescenn Full HD 1080p (voir figure 5.2) a été choisie. D'abord parce que le tracking des pupilles demande une grande résolution et que cette webcam est adaptée à la résolution Full HD 1080p (1920x1080), mais aussi parce qu'elle intègre un éclairage LED permettant, comme précédemment, de placer nos émetteurs infrarouges.



FIGURE 5.2 – Webcam Trust Widescenn Full HD 1080p

Émetteur infrarouge

5.2 Résultats et analyses

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

Conclusion

Ce projet de l'U.V. 3.4 nous permet de comprendre les étapes qui mènent à l'aboutissement d'un projet. En effet, afin de se faire une idée de ce qui est réalisable, nous avons tout d'abord procédé à un état de l'art. Cette étude préalable nous a permis de comprendre que de nombreuses technologies existent déjà afin d'effectuer de l'eye tracking. En comparant les possibilités offertes avec nos attentes, nous avons décidé que la technologie la plus pertinente serait la suivante :

- Filmer l'utilisateur à l'aide d'une première caméra. Celle-ci repère la position du visage dans l'ensemble de l'image, et transmet la position à une deuxième caméra.

- Filmer l'utilisateur à l'aide d'une première caméra. Celle-ci repère la position du visage dans l'ensemble de l'image, et transmet la position à une deuxième caméra.

- Zoomer sur le visage à l'aide de la deuxième caméra qui est infra-rouge. Pour cela nous installerons un filtre sur une caméra normale afin de diminuer les coûts. Celle-ci pourra alors détecter la pupille de l'utilisateur et déterminer de manière précise ses mouvements.

A partir de cette décision nous avons pu mettre en place un dossier fonctionnel. Dans un premier temps, l'approche Top-Down nous a permis d'identifier les exigences de notre système. Nous avons défini une fonction principale, deux fonctions de service et deux fonctions de contraintes. Ces exigences ont ensuite été caractérisées par une approche Bottom-Up. Nous les avons regroupées par fonctions principales, et raffinées en FAST. Nous sommes restés le plus général possible dans les intitulés des fonctions pour qu'elles soient adaptées même si l'on était amenés à changer notre système. Enfin nous avons spécifié les données et proposé une architecture fonctionnelle. Cependant, ces définitions pourront être amenées à changer.

A partir de cette décision nous avons pu mettre en place un dossier fonctionnel. Dans un premier temps, l'approche Top-Down nous a permis d'identifier les exigences de notre système. Nous avons défini une fonction principale, deux fonctions de service et deux fonctions de contraintes. Ces exigences ont ensuite été caractérisées par une approche Bottom-Up. Nous les avons regroupées par fonctions principales, et raffinées en FAST. Nous sommes restés le plus général possible dans les intitulés des fonctions pour qu'elles soient adaptables si l'on était amenés à changer notre système. Enfin nous avons spécifié les données et proposé une architecture fonctionnelle. Cependant, ces définitions pourront être amenées à changer.

En effet, concernant la partie Ingénierie des Exigences, nous avons pris conscience que la mise en place de l'architecture fonctionnelle n'est pas un exercice aisément tenu compte de l'objectif de notre projet. Celui-ci offre une grande liberté de réalisation avec un grand choix de méthodes, ce qui ne nous permet pas à l'heure

actuelle de connaître précisément l'architecture de notre futur système. Notre projet demande une avancée dans la conception physique (début d'algorithme) afin de pouvoir caractériser correctement l'architecture fonctionnelle. Tant que le choix de notre système n'est pas validé, le dossier fonctionnel pourra évoluer. Nous avons donc décidé de nous consacrer désormais à l'obtention d'un eye tracking performant. En effet, à ce stade du projet, si nous parvenons à suivre les yeux d'un utilisateur, ce flux vidéo reste trop saccadé. L'étape suivante sera de fluidifier ce résultat afin d'effectuer une détection de la pupille et de valider notre système.

Références bibliographiques

- [1] Maitine BERGOUNIOUX : Quelques méthodes de filtrage en traitement d'image. 2010.
- [2] Chul Woo CHO, Ji Woo LEE, Kwang Yong SHIN, Eui Chul LEE, Kang Ryoung PARK, Heekyung LEE et Jihun CHA : Gaze detection by wearable eye-tracking and nir led-based head-tracking device based on svr. *ETRI Journal*, 34(4):542–552, 2012.
- [3] El Khayati Brahim D'HONDT FRÉDÉRIC : Etude de méthodes de clustering pour la segmentation d'images en couleurs. *Faculté Polytechnique de Mons, 5ème Electricité, Certificat Applicatifs Multimédia*, 2004.
- [4] Su Yeong GWON, Chul Woo CHO, Hyeyon Chang LEE, Won Oh LEE et Kang Ryoung PARK : Robust eye and pupil detection method for gaze tracking. *Int J Adv Robotic Sy*, 10(98), 2013.
- [5] Pupil LABS : Pupil v0.3.6 - marker tracking. <http://www.pupil-labs.com/blog/2013/12/036-release.html>, 2013. Accédé le 15 janvier 2015.
- [6] Hyeyon Chang LEE, Won Oh LEE, Chul Woo CHO, Su Yeong GWON, Kang Ryoung PARK, Heekyung LEE et Jihun CHA : Remote gaze tracking system on a large display. *Sensors*, 13(10):13439–13463, 2013.
- [7] 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.
- [8] Angèle MASSONNEAU et Nicolas BIARD : Etat de l'art des différents systèmes de pointages à l'oeil. *Dossier de la Plate-Forme Nouvelles Technologies, Assistance Publique Hôpitaux de Paris*, 2013.
- [9] Benjamin RAYNAL et Venceslas BIRI : Détection de pupille par combinaison des critères morphologiques et colorimétriques.
- [10] Jérôme SCHMALTZ : Détection des contours de la pupille à l'aide des transformées de hough.
- [11] Tobii TECHNOLOGY : An introduction to eye tracking and tobii eye trackers. <http://www.tobii.com/en/eye-tracking-research/global/library/white-papers/tobii-eye-tracking-white-paper/>, 2010. Accédé le 15 janvier 2015.
- [12] Tobii TECHNOLOGY : Tobiix2-30eyetracker. *User Manual*, 2014.
- [13] Tobii® TECHNOLOGY : Tobii glasses eye tracker. *User Manual*, 2011.

- [14] ZAFERSAVAS : Trackeye : Real-time tracking of human eyes using a webcam. <http://www.codeproject.com/Articles/26897/TrackEye-Real-Time-Tracking-Of-Human-Eyes-Using-a>, 2008. Accédé le 15 janvier 2015.
- [15] Marc ZAFFAGNI : Eyecharm : contrôler son pc du regard avec un capteur kinect. <http://www.futura-sciences.com/magazines/high-tech/infos/actu/d/informatique-eyecharm-controler-son-pc-regard-capteur-kinect-45381/>, 2013. Accédé le 15 janvier 2015.