Enora DANIEL

Informatique et Electronique des Systèmes Embarqués (IESE)

Rapport de stage de 4e année

ADAPTATION D'UN DISPOSITIF OCULAIRE DESTINE A LA COMMUNICATION ALTERNATIVE ET AUGMENTÉE À UNE RASPBERRY PI

Tome Principal

ET

Annexe

2021-2022

Stage du 02/05/2022 au 28/07/2022

Table des matières

Table des figures

Glossaire et listes d'abréviations

GETALP : Groupe d'étude pour la traduction automatique et le traitement automatisé des langues et de la parole.

IESE : informatique et électronique des systèmes industriels

Oculomètre (ou eye tracker) : Dispositif oculaire qui est constitué de capteurs haute précision, associés à un écran ou intégrés à une paire de lunettes. Il mesure et enregistre les trajets oculaires d'une personne.

Raspberry Pi : Un nano-ordinateur monocarte à processeur ARM, depuis la sortie du premier modèle en 2013, 14 modèles ont été créés.

Caméra infrarouge : Caméra qui capte les ondes de chaleur (rayonnement infrarouge). Elle est particulièrement utile pour percevoir une image la nuit, ou pour fabriquer un eye tracker**.**

OpenCV : bibliothèque graphique libre, spécialisée dans le traitement d'images en temps réel.

LIG : Laboratoire d'informatique de Grenoble.

La communication alternative et augmentée (CAA) : elle recouvre tous les moyens humains et outils permettant à une personne rencontrant des difficultés dans la communication, de communiquer en remplaçant le langage oral s'il est absent (alternative) ou en améliorant une parole insuffisante (augmentée).

Architecture ARM : Architecture externe de type RISC (Reduced instruction set computer), ne supportant que des instructions simples et de taille fixe (4 octets pour le jeu d'instructions) et s'exécutant en un nombre constant de cycles.

Rétro-ingénierie (reverse engineering) : C'est l'activité qui consiste à étudier un objet pour en déterminer le fonctionnement interne, la méthode de fabrication.

Apprentissage supervisé : C'est une tâche d'apprentissage automatique consistant à apprendre une fonction de prédiction à partir d'exemples annotés.

Machine Learning : Le Machine Learning ou apprentissage automatique est un domaine scientifique, et plus particulièrement une sous-catégorie de l'intelligence artificielle. Elle consiste à laisser des algorithmes découvrir des » patterns «, à savoir des motifs récurrents, dans les ensembles de données. Ces données peuvent être des chiffres, des mots, des images, des statistiques… (*Références 11*)

Multithreading : Le multithreading résulte d'une interaction entre le matériel et le logiciel. Les programmes et les processus sont décomposés en threads individuels et traités par l'unité centrale

dans ces unités plus petites. Une distinction est faite entre le multithreading matériel et le multithreading logiciel. (Référence 15)

Remerciements

Je voudrais tout d'abord remercier mon tuteur de stage, Monsieur Didier Schwab, pour m'avoir accueilli au sein de l'équipe de GETALP et qui a contribué au bon déroulement de ce stage.

Je remercie également Jordan Arrigo et Vincent Arnone pour m'avoir consacré de leur temps quand j'ai rencontré des problèmes.

J'adresse mes remerciements à Madame Nathalie Guyader, ma tutrice à Polytech Grenoble, pour son suivi au cours de ce stage.

Enfin, mes remerciements vont aussi à l'ensemble des personnes que j'ai croisé lors de mon stage au sein du LIG et GETALP.

1 Introduction

Durant le semestre 8, les étudiants de la filière IESE à Polytech Grenoble doivent réaliser un stage dans un des domaines de leurs compétences (informatique industrielle, capteurs, traitement du signal, systèmes électroniques …). J'ai choisi d'effectuer mon stage au sein du laboratoire GETALP afin d'en apprendre plus sur le milieu de la recherche. Je n'avais jamais encore travaillé au sein d'un laboratoire et étais contente d'en apprendre plus. Ayant découvert les oculomètres durant mon projet d'école au semestre 8, j'étais très intéressée de continuer d'apprendre sur ces dispositifs. J'ai donc contacté plusieurs laboratoires pour y effectuer mon stage.

Le but de ce stage est de chercher des solutions qui permettent de faire fonctionner un dispositif oculaire destiné à la Communication Alternative et Augmentée sur une Raspberry Pi.

J'ai rejoint l'équipe de Getalp et ai travaillé sous la tutelle de Monsieur Didier Schwab, mon tuteur de stage, ainsi qu'avec Madame Nathalie Guyader, mon enseignante référente.

Tout d'abord nous verrons la présentation du laboratoire ainsi que l'équipe dans lesquels le stage a été effectué, ensuite il sera abordé tout l'enjeu du stage et le travail réalisé. Une conclusion viendra clore le rapport de stage.

2 Environnement de travail

2.1 GETALP et LIG

L'équipe GETALP (*figure 1*) est née en 2007 lors de la création du laboratoire informatique de Grenoble (LIG). Son chef est François PORTET. Avant 2007, GETALP était deux équipes, GETA (Groupe d'Étude pour la Traduction Automatique) et GEOD (Groupe d'Étude sur l'Oral et le Dialogue). Celles-ci ont fusionné pour donner naissance à l'actuelle équipe GETALP. (*Référence 1*)

Le LIG (*figure 1*) rassemble près de 450 chercheurs, enseignants-chercheurs, doctorants et personnels travaillant pour la recherche. Le directeur du LIG est Noël DE PALMA. Le LIG est présent sur trois sites : le campus, Minatec et Montbonnot. Le site de mon stage est le campus. (*Référence 2*)

Figure 1 : Logo GETALP et LIG

L'objectif du LIG est de s'appuyer sur la complémentarité et la qualité reconnue des 22 équipes de recherche qui le composent pour contribuer au développement des aspects fondamentaux de l'informatique et créer des synergies entre enjeux conceptuels, techniques et sociétaux. (*Référence 2*)

La recherche au LIG se décline en 5 Axes de recherche :

- Génie des Logiciels et des Systèmes d'Information
- Méthodes Formelles, Modèles et Langages
- Systèmes Intelligents pour les Données, les Connaissances et les Humains
- Systèmes Interactifs et Cognitifs
- Systèmes Répartis, Calcul Parallèle et Réseaux.

L'équipe GETALP est dans l'axe de recherche Systèmes intelligents pour les données, les connaissances et les humains. Elle se compose de 16 membres permanents (chercheurs, enseignants-chercheurs, maître de conférences) et de membres non permanents (stagiaires, doctorants, ingénieurs.

L'objectif de GETALP est d'aborder tous les aspects théoriques, méthodologiques et pratiques de la communication et du traitement de l'information multilingue (écrite ou orale). (*Référence 3*)

L'équipe du GETALP est actuellement organisée autour de six grands axes de recherche :

- Traduction automatique (TA) et automatisée (TAO)
- Traitement automatique des langues (TALN) et plates-formes associées
- Collecte et construction de ressources linguistiques
- Multilinguisme dans les systèmes d'information
- Reconnaissance automatique de la parole, des locuteurs, des sons et des dialectes
- Analyse sonore et interaction dans les environnements perceptifs.

2.2 Équipe de travail

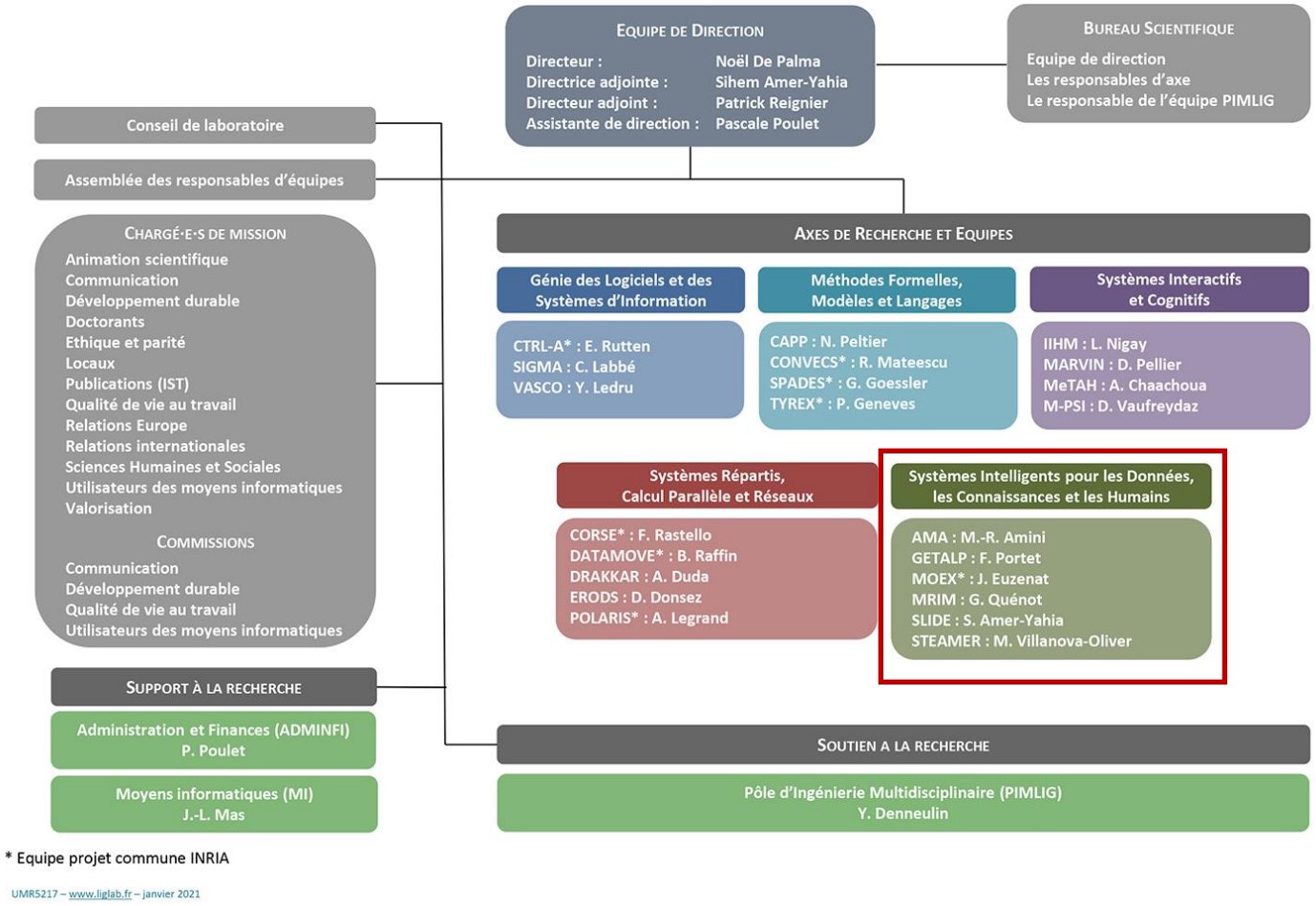

Figure 2 : Organigramme

L'équipe dans laquelle j'ai pu travailler est dans l'axe de recherche systèmes intelligents pour les données, les connaissances et les humains (*Figure 2*). Je suis dans le groupe GETALP avec à sa tête M. F. Portet. Mon équipe est composée de Didier Schwab, mon tuteur au laboratoire, de deux ingénieurs et également de plusieurs stagiaires. Notre équipe de travail contribue au développement de InterAACtion, un groupe de recherche et de formation sur la Communication Alternative et Augmentée de l'université Grenoble Alpes. Dans ce groupe de recherche, on peut retrouver plusieurs projets comme InterAACtionBox-AFSR qui un dispositif informatique intégré open source. Il s'agit d'une solution qui intègre un ordinateur tactile, un eye-tracker ainsi que des logiciels de CAA. Parmi les logiciels, on retrouve InterAACtionScene, Augcom, InterAACtionPlayer et GazePlay (voir 3.1).

3 Présentation du sujet de stage

3.1 Contexte du stage

L'utilisation des Communications Alternatives et Augmentées a commencé dans les années 1950 pour les patients qui ont perdu la capacité de parler après une intervention chirurgicale. Mais ce n'est qu'en 1980 que la **CAA** a commencé à émerger en tant que domaine à part entière, grâce aux progrès rapides de la technologie (ordinateurs, tablettes tactiles, systèmes de synthèse vocale, suivi oculaire, etc.) (*Référence 4*). Le **CAA** s'adresse également aux enfants présentant divers troubles de la parole et du langage, dont l'origine peut être la paralysie cérébrale, la déficience intellectuelle, l'autisme... Le but du **CAA** est de donner aux enfants une meilleure compréhension de leur environnement, en les rendant plus lisibles et prévisibles pour améliorer leur compréhension et leur fournir des moyens et des outils pour les aider à s'exprimer.

Le regard est considéré comme l'un des moyens les plus faciles à utiliser pour permettre à une personne de communiquer en cas de situation de handicap. Les eye tracking permettent le suivi oculaire de l'utilisateur, ils sont utiles dans la **CAA** car grâce à eux une personne qui n'a pas l'usage de ses membres peut ainsi interagir avec son environnement.

GazePlay (*figure 3*), **gazeplay.net**, est un logiciel open source et gratuit qui regroupe une soixantaine de mini-jeux jouables avec un eye tracker. Il rassemble 60 jeux à ce jour. Pour chacun des jeux, il existe de multiples variations (difficulté, de taille, etc). Ce logiciel est compatible avec tous les eye trackers capables de contrôler un curseur de souris. Parmi les eye trackers pouvant être utilisés, il y a le Tobii EyeX, Tobii4C et le Tobii 5 sous Windows et Eye Tribe Tracker sous Windows ou MacOs X.

Figure 3 : Logo GazePlay

Exemple d'un des jeux présent dans GazePlay (*Figure 4*), C'est le jeu du Labyrinthe. Il y a plusieurs version du jeu disponible, ici pour se déplacer, il faut fixer la caser, avec son

regard, où on veut aller. A la fin du jeu, on peut retrouver différentes informations comme le trajet du regard dans le jeu (*Figure 5*).

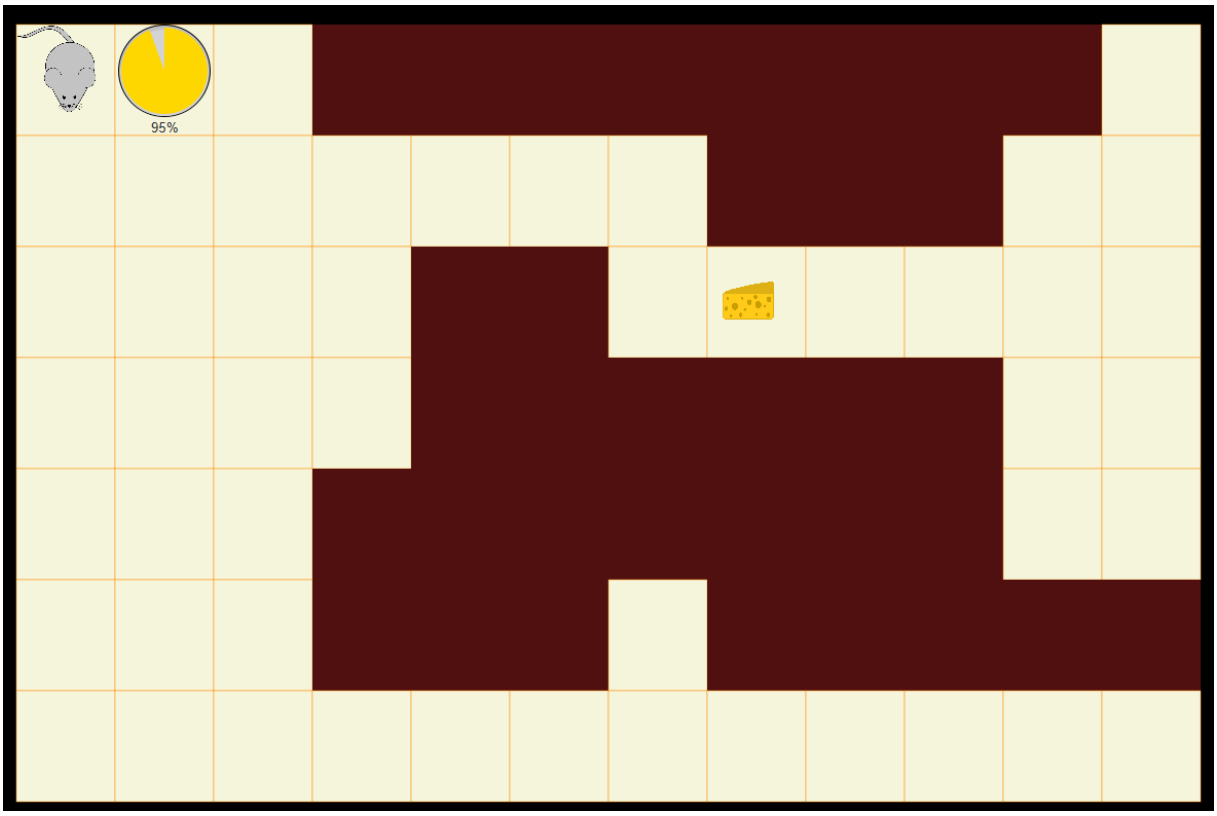

Figure 4 : Jeu Labyrinthe GazePlay

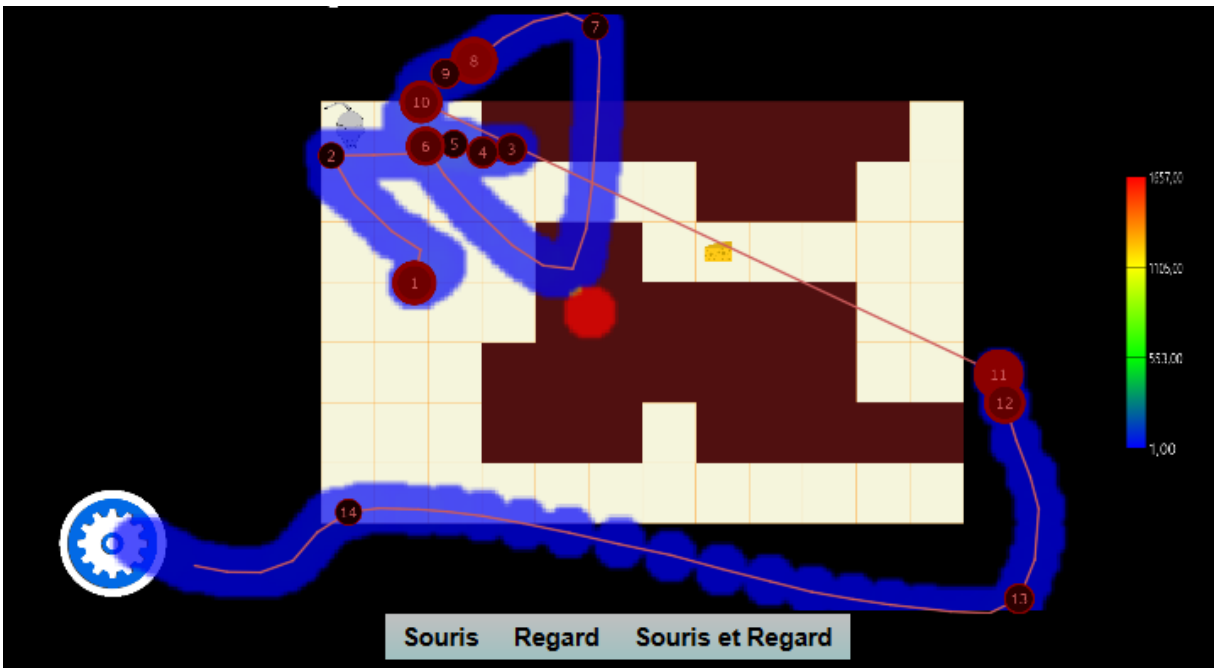

Figure 5 : Trajet du regard sur le jeu Labyrinthe

3.2 Problématique

Le logiciel GazePlay est disponible pour les versions de linux, MacOs et Windows. Pour rendre encore plus accessible GazePlay au plus grand nombre, utiliser une carte Raspberry Pi peut revenir moins cher qu'un ordinateur et ne prend pas beaucoup de place. Le but du stage est d'estimé à quel point on peut adapter un dispositif oculaire destiné à la communication alternative et augmentée à une Raspberry Pi.

3.3 Objectif du stage

L'objectif de ce stage est de découvrir ce qui est possible de réaliser sur la carte Raspberry PI et ses limites actuelles. Pour réaliser cet objectif, il y a plusieurs missions. La première est de regrouper les différents eye tracking / caméra déjà existant sur le marché pouvant être utiles au projet GazePlay. Une fois les différents appareils répertoriés, l'objectif est de rechercher si une solution existe pour faire fonctionner un appareil dédié sur la carte Raspberry Pi. Enfin la dernière mission est de tester un programme de suivi oculaire en Python sur la carte Raspberry.

3.4 Organisation du travail

Pour se tenir informé de l'évolution du travail et informer de notre présence, chaque jour un message était envoyé sur un discord, qui indique ce que nous allions faire pendant la journée puis à la fin de la journée à un autre message pour répertorier les différentes taches faites pendant la journée. Des réunions zoom ont été également faites, au moins une par semaine, pour faire le point sur le travail de chacun. Ma mission n'est pas directement en équipe donc je n'ai pas dû dépendre d'autres personnes pour avancer sur mes taches. Néanmoins si j'avais le moindre problème, je pouvais demander de l'aide à d'autres personnes que ce soit mon tuteur ou les deux ingénieurs qui travaillaient avec mon tuteur.

3.5 Outils à ma disposition

Durant mon stage, j'ai développé en python car c'est le langage de base qui est utilisé sur la carte Raspberry Pi. J'aurai à ma disposition une carte Raspberry Pi 4 (Figure 6), avec deux caméras qui ont été commandées. Une caméra normale (figure 7) et une caméra de vision nocturne (Figure 8). J'ai également à ma disposition deux eye trackers, le tobii 4c (Figure 9) et le Tobii 5 (Figure 10). J'ai également beaucoup utilisé la bibliothèque OpenCV, elle est spécialisée dans le traitement d'images, que ce soit pour de la photo ou de la vidéo. La

bibliothèque Dlib a été aussi utilisée pour faire du traitement d'image comme faire de la reconnaissance faciale.

Figure 6 : Raspberry Pi 4B

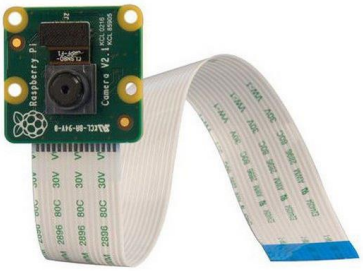

Figure 7 : Raspberry Pi Camera Module V2.1

Figure 8: Seamuing Raspberry Pi OV5647

Figure 9 : Tobii 4c

Figure 10 : Tobii 5

4 Les missions du stage

Pour réaliser mon objectif, adapter un dispositif oculaire destiné à la Communication Alternative et Augmentée à une Raspberry Pi, j'ai tout d'abord entrepris des recherches sur ce qu'était un oculomètre puis j'ai recensé les différents eye trackers qui étaient disponibles à ce jour. Ensuite j'ai cherché à adapter les eye trackers tobii (4c et 5) sur la carte Raspberry Pi. Et enfin j'ai effectué des recherches de programmes existant déjà d'eye tracking pour les tester sur la carte Raspberry Pi. Puis à partir des programmes, j'ai construit mon propre suivi oculaire pour qu'il puisse être utilisé avec le logiciel GazePlay dans le futur.

4.1 Les oculomètres (Eye tracker)

D'après Tobii Dynavox (référence 5), l'oculométrie est présentée comme une technologie utilisée pour voir où une personne regarde sur un écran d'ordinateur. Cela correspond à la poursuite du regard, ou interaction du regard. Elle peut également être utilisée pour contrôler un ordinateur avec les yeux au lieu d'utiliser un clavier et une souris traditionnels, ce qui permet aux personnes souffrant de problèmes physiques et cognitifs de vivre une vie plus riche et plus indépendante.

L'oculométrie est utilisée dans de nombreux domaines, comme en médecine pour repérer des dysfonctionnement du corps humain (cancer …) ou alors améliorer le quotidien de personnes pour communiquer (**CAA**). On retrouve l'oculométrie dans les jeux vidéo ou dans le marketing.

4.1.1 Types de dispositif

Pour réaliser le suivi oculaire, il existe plusieurs techniques qui correspondent à différents dispositifs. D'après le site *stringfixer.com* (*Référence 6*), il existe 3 techniques. La première consiste à être en contact avec la pupille au moyen d'une de contact spéciale pour obtenir le mouvement oculaire. La deuxième technique est réalisée avec une vidéo. On peut utiliser une lumière infrarouge, qui est réfléchie par l'œil et détectée par une caméra vidéo ou un autre capteur optique spécialement conçu. Les informations sont ensuite analysées pour extraire la rotation des yeux des changements de reflets. Les eye trackers vidéo utilisent généralement le reflet cornéen (*figure 11*). D'après *acces.ens-lyon.fr* (*Référence 7*), pour obtenir le reflet cornéen, cela consiste à envoyer au centre de la pupille une lumière infrarouge émise par une diode ou un ensemble de diodes. Le reflet infrarouge renvoyé par la cornée de l'œil est ensuite détecté et ce sont les variations d'intensité de ce reflet qui permettent, après calcul, de repérer le centre de la pupille et de connaître la position de l'œil. Les images utilisées pour cette technique proviennent de caméra sans le filtre infrarouge pour obtenir le reflet cornée.

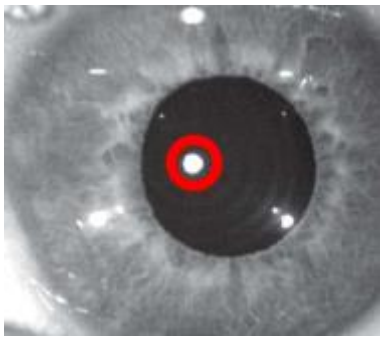

Figure 11 : Oeil avec reflet cornéen entouré en rouge

Une autre technique est de seulement récupérer les images de la vidéo prise, puis de les traiter avec un programme informatique comme Opencv pour en récupérer les informations souhaitées (*figure 12*). Les images peuvent provenir d'une caméra normale ou alors d'une caméra sans filtre infra-rouge (4.1.2 différence entre caméra et caméra IR).

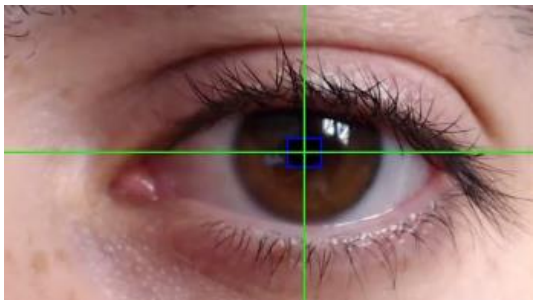

Figure 12 : traitement d'image pour en récupérer une information (suivi oculaire)

La dernière technique consiste à utiliser des potentiels électriques mesurés avec des électrodes placées autour des yeux *(Figure 13).* Les yeux produisent alors un champ de potentiel électrique permanent. Le signal qui sera capté des électrodes est appelé électro-oculogramme.

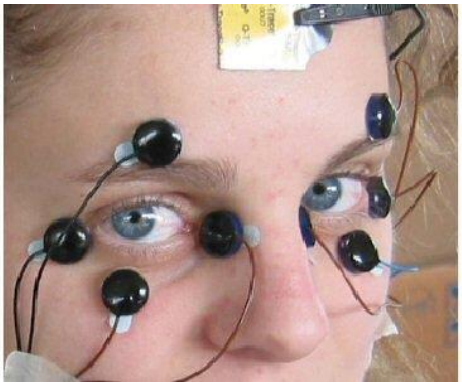

Figure 13: Oculomètre avec des électrodes

Après avoir récupérer les images de l'œil et convertit les mouvements oculaires en un flux de données, il est récupéré divers flux d'informations après traitement (position de la pupille, la durée de fixation, et le point, zone ou direction du regard).

4.1.2 Différence entre caméra et caméra IR

Comme expliqué dans le paragraphe précédemment, pour utiliser le reflet cornéen pour déterminer la position de la pupille, il faut une caméra sans filtre infrarouge. Mais l'utilisation de la caméra sans filtre infrarouge peut être utilisée également dans la détection de la pupille avec seulement une caméra et opencv par exemple. Dans un livrable *(Annexe 3*), ce document détaille les différentes technologies utilisées pour capturer une image puis les

différences entre les appareils utilisant l'infrarouge et enfin quel appareil est le plus efficace dans la détection de la pupille.

4.1 Étude des différents systèmes de capture du regard

Il existe de nombreux type d'oculomètre, pour utiliser un eye tracker avec GazePlay, il faut un dispositif qui laisse une liberté de mouvement à l'utilisateur car sinon cela risque de fortement gêner la personne. Les Head-Stabilizer (stabilisateur de tête) (*figure 14*) sont alors trop contraignants car ils bloquent la tête. Les dispositifs accrochés sur la tête, comme des lunettes (*figure 14*), risqueraient de gêner la vue des utilisateurs, ils sont alors enlevés des recherches.

Figure 14 : Head-Stabilized (gauche) et lunettes eye tracker (droite)

Un autre critère de recherche était le prix, en effet les dispositif sont à destination de famille. Moins l'oculomètre sera cher, plus GazePlay et les autres logiciels pourront être accessible au plus grand monde. Les deux documents regroupant mes recherches sont accessibles en annexe (Annexe 1 et 2). Ils regroupent un document Word qui présente en détail les eye trackers intéressants puis un tableau lui regroupant des eye trackers anciens et récents. Je me suis aidé du document *état de l'art des différents systèmes de pointages à l'œil (Référence 8)*. Ce document regroupe des systèmes de pointage disponibles en France et à l'étranger.

Deux caméras pour carte Raspberry Pi ont été également rajoutées au document car ce sont les caméras qui ont été choisies pour tester les programmes python trouvés/réalisés.

4.3 Étude des moyens pour adapter un appareil dédié à une Raspberry Pi

Les oculomètres disponibles sur le marché au meilleur prix sont le tobii 4c et 5. Ces deux produits sont déjà utilisés, intégrés sur le logiciel GazePlay. Le logiciel est installé sur de nombreuses plateformes comme linux, Mac os ou Windows. Pour faire fonctionner le logiciel un eye tracker est nécessaire. J'ai donc entrepris des recherches pour déterminer si les eye tracker tobii 4c et 5 peuvent fonctionner sur la carte Raspberry Pi 4B.

4.3.1 La carte Raspberry PI

La carte sur laquelle j'ai travaillé est une Raspberry PI 4B model 4 GB. Ses caractéristiques sont les suivantes *(Référence 9)* :

- CPU1: Broadcom BCM2711 (1.5 GHz, 4 cores)
- Processeur 1.5GHz quad-core 64-bit ARM Cortex-A72
- LPDDR1 4 Go

4.3.1 Tests

Il m'a été mis à ma disposition deux eye trackers, le tobii 4c et le tobii 5. Ces eye trackers ont besoin pour fonctionner d'un driver disponible sur le site de tobii gaming. Malheureusement ces drivers ne sont pas disponibles sous le format qui correspond à la carte Raspberry Pi, l'architecture ARM. D'après le forum de chat disponible sur https://developer.tobii.com/, le fabricant n'a pas pour objectif de mettre à disposition un driver qui pourrait convenir à l'architecture ARM. Leur décision vient du fait que peu de gens utilisent les processeurs ARM donc il serait moins intéressant de développer un driver pour seulement une petite portion d'utilisateur.

Un moyen de contourner le fait que les drivers ne soit pas disponible sous l'architecture ARM serait d'installer un émulateur sur la carte Raspberry PI. Les drivers seraient en effet installés mais la puissance disponible serait trop faible pour faire tourner un driver avec l'eye tracking et le code associé. Cette solution n'est pas viable.

Une dernière manière serait de faire du rétro-ingénierie, c'est le fait d'étudier un fonctionnement et pouvoir le comprendre jusque dans ses moindres détails de sorte à en faire ressortir tous les principaux mécanismes. En utilisant cette technique, nous pourrions comprendre comment fonctionne les tobii 4c/5 et contourner l'utilisation d'un driver. Le rétro-ingénierie a des limites de légalité dans certains cas précis. Ici il serait utilisé sur du matériel et non logiciel donc il n'y a pas de problèmes de légalité dans cette démarche. Malheureusement avec cette technique, mes compétences en termes de programmation n'étant pas assez développer, je n'ai pas pu commencer à fouiller dans les appareils.

Le but de mes recherches est de savoir jusqu'à quel point on peut utiliser la carte Raspberry Pi avec des programmes et dispositifs d'eye tracking. La carte Raspberry Pi devrait faire fonctionner également GazePlay, c'est donc en tentant de l'installer que nous nous sommes rendu compte que le logiciel n'était pas compatible avec l'architecture ARM. Les ingénieurs Jordan et Vincent ont alors remis à jour GazePlay, la nouvelle s'installe sur la carte mais malheureusement la carte n'est pas assez performante pour supporter le logiciel et un bug complet rend la carte non utilisable. Il faudra encore attendre que des nouvelles cartes Raspberry Pi beaucoup plus performante sortent pour que GazePlay fonctionne sans aucun problème. Cette information nous dit que la carte Raspberry Pi n'est pas encore une option dans le projet InterAACtion.

4.4 Adaptation à la Raspberry Pi de programmes (en python) de capture du regard par caméra

Pour tester les limites de la carte Raspberry PI avec des programmes de capture du regard, il faut trouver des programmes python d'eye tracking et les améliorer si besoin. Avec la carte Raspberry pi, j'ai deux caméras spécifiquement pour Raspberry pour venir capter et enregistrer le regard.

4.4.1 Recherche de programmes et fonctions python

Comme je l'ai expliqué en 4.1.1(Types de dispositif), il existe plusieurs façon pour récupérer la position de la pupille. J'ai choisi de prendre la technique qui consiste de ne pas utiliser le reflet cornéen et d'utiliser OpenCV car cette technique est plus facilement programmable.

Après de longues recherches, j'ai trouvé un programme assez complet d'eye tracking qui permet également d'être facilement modifiable si besoin. Ce programme (https://github.com/antoinelame/GazeTracking) est disponible sur GitHub en open source donc parfaitement utilisable dans ma démarche. Avant de pouvoir utiliser ce code, il faut en comprendre toutes ses fonctions et bibliothèques utilisées.

4.4.2 Tests des bibliothèques

Les principales bibliothèques python utilisés pour réaliser un eye tracker sur python sont Dlib et OpenCV. Pour pouvoir détecter la pupille d'une personne, il faut procéder à un traitement d'image. Ce traitement d'image peut être réalisé de nombreuses manières comme la segmentation. La segmentation permet d'isoler les différents objets présents dans une image.

Pour comprendre les bibliothèques Dlib et OpenCV, j'ai réalisé de nombreux programmes tests (~15). Pour visualiser ce que peut faire OpenCV et Dlib, j'ai choisi deux programmes. Le premier programme (Annexe 5), permet de détecter les visages des personnes présentes sur une image et d'en tirer différentes points qui correspondent à différentes parties du visages (Landmark). Ces points correspondent aux contours des yeux, des sourcils, de la bouche par exemple.

Pour réaliser le programme, on a besoin d'un détecteur (Dlib), Dlib contient déjà un model pre-entrainé, qui vient repérer le visage et d'un prédicteur (Dlib), associer à un fichier model prédicteur contenant des données d'images de visage qui ont été entrainé. On obtient ensuite (*figure 15*). La bibliothèque OpenCV est utilisé pour ouvrir l'image et la lire, elle a été aussi utilisée pour tracer le rectangle en vert et les points.

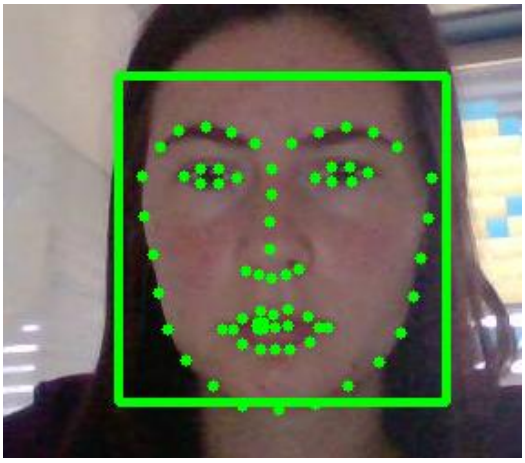

Figure 15 : Programme Python détectant visage et points du visage

Le deuxième utilise la même base que le premier programme mais va plus loin dans le traitement. Le programme utilise plusieurs fonction de Dlib et OpenCV (Annexe 4) pour réaliser des masques sur l'image et faire ressortir juste les yeux et les pupilles en rouge (*figure 16).*

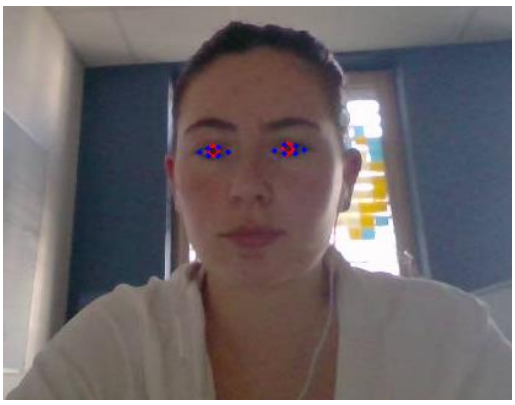

Figure 16 : Détection de la pupille

Grâce à ces programmes python, on comprend mieux le fonctionnement des bibliothèques Python Dlib et OpenCV.

4.4.3 Programme d'eye tracking

Les bases de OpenCV et Dlib établies ont peut décrypter ce que fait le programme d'eye tracking de antoinelame.

Programme initiale :

Le programme utilisé à 5 classes qui s'appellent entre elles (init (1 ligne), calibration (77 lignes), eye (119 lignes), gaze_tracking (133 lignes) et pupil (54 lignes)) et un programme principal exemple (44 lignes). Le fonctionnement du programme est décrit *figure 17* et son résultat *figure 18*.

Sur la figure 17, le fonctionnement du programme a été simplifié si plus de détails sont nécessaires, le code est disponible sur (https://github.com/antoinelame/GazeTracking).

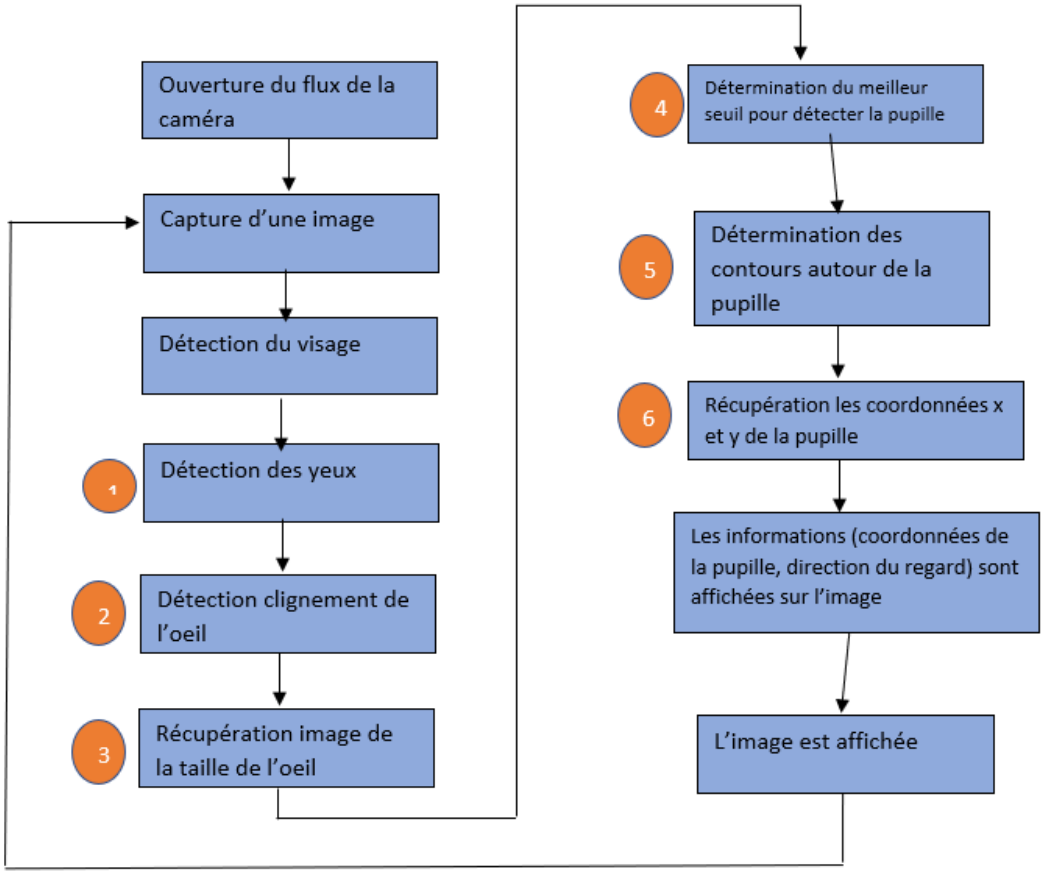

Figure 17 : Diagramme Code python eye tracking (les étapes 1-6 sont répétées pour l'œil droit et gauche)

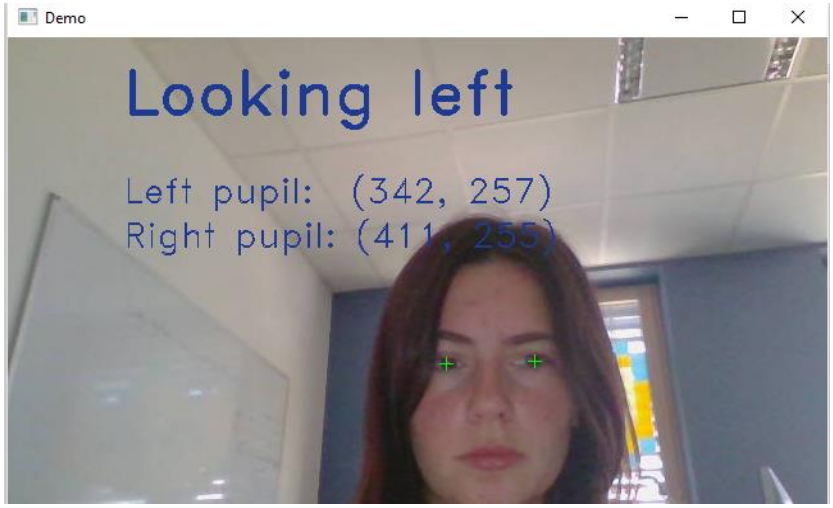

Figure 18 : Démonstration du programme

Programme modifié/finale :

Le programme récupéré est une base pour obtenir un eye tracking qui en plus de suivre le regard afficherait sa trajectoire à l'écran. Toutes les fonctions de ce programme ne seront pas utiles pour réaliser l'eye tracker. Il n'est pas nécessaire d'afficher les informations sur l'image ni de calculer si l'œil cligne. Nous voulons avoir le tracé du regard sur l'écran, donc il faut créer une fenêtre qui afficherait cela. Et une calibration entre ce que regarde l'utilisateur sur l'écran et les coordonnées des pupilles pour ensuite pouvoir afficher la trajectoire du regard. Ce programme est disponible sur mon GitHub.

Le programme étant très lourd est terme de calcul et de fonction est très lent sur la carte Raspberry Pi. Pour palier à ce problème, le multithreading est envisagé pour séparer les taches effectuées et ainsi accélérer le processus du programme. En Annexe 6, un programme python est affiché pour expliquer comment le multithreading est utilisé avec une caméra.

J'ai donc rajouté du multithreading à mon programme pour le rendre plus rapide. J'ai également simplifié mon programme pour qu'il affiche seulement la partie de l'écran que regarde l'utilisateur et non le suivi du regard. Dans ce programme, Il est donc récupéré la valeur de x et y des deux pupilles puis une calibration (l'utilisateur doit fixer un point (*figure 19*)) est effectuée qui sort un rapport pupille écran. Ce rapport est utilisé pour ensuite calculer la position du regard sur l'écran de l'ordinateur. On affiche alors la partie de l'écran qui est regardée, droite en rouge, gauche en bleu (*Figure 20*).

Figure 19 : Fenêtre de calibration

Figure 20 : Fenêtre affichant le côté regardé

Dans ce dernier programme final, on a 11 classes (Show_video 27 lignes, gaze_tracking 153 lignes, suivi_ecran 46 lignes, calibration_ecran 62 lignes, videostream 30 lignes, Point 20 lignes, Calibration 77 lignes, fenetre_ecran 68 lignes, eye 119 lignes, pupil 54 lignes) et un main (72 lignes). Il y a 6 classes de plus que le programme initiale. Le programme de antoinelame a été évolué et il ne reste qu'environ 50 % du programme initiale.

Tests de Performance :

J'ai réalisé des tests sur la carte Raspberry et mon ordinateur pour déterminer si mon programme finale est plus performant avec du multithreading. Des tests de comparaison entre les caméras de la Raspberry et la caméra du Pc sont également présents. Tous les tests

de performance sont détaillés en Annexe 7. Grâce à ses tests, on peut observer que même avec le multithreading, le programme est encore très lent. Pour obtenir un programme très fluide, il faudrait encore optimiser le code.

4.5 Limite

Grâce aux nombreuses recherches et tests que j'ai pu effectuer sur la carte Raspberry Pi, je peux lister les limites de la carte. La carte Raspberry Pi est très performante et permet de réaliser beaucoup de projets ludiques. La ligne directrice de mon stage était de savoir si avec la carte, il était possible de l'utiliser en tant qu'ordinateur supportant un œil tracking pour une potentielle utilisation avec également GazePlay. Le logiciel GazePlay n'est pas supporté par la carte donc une utilisation pour cet effet n'est pas envisageable. Les eye trackers disponible au plus bas prix ne sont également pas utilisable sur Raspberry Pi. Les programmes python d'eye tracking fonctionnent sur carte Raspberry Pi mais pas à une vitesse d'exécution qui serait utile et ne sont pas d'une grande précision.

Le programme finale développer sur ordinateur ne peut afficher que la partie de l'écran regardée (droite ou gauche). Il faudrait développer bien plus la précision du programme pour récupérer des coordonnées très précises et fiables pour ensuite développer un eye tracker capable de suivre correctement le regard. Le programme peut être utilisé dans le domaine de l'orthophonie, des jeux Bera sont présents sur GazePlay pour aider les orthophoniste dans leur travail. Ces jeux consistent à afficher des images à gauche ou droite de l'écran. Alors dans ce cas d'utilisation, GazePlay est utilisés comme une boite à outils pour faire des tests (ici en orthophonie).

Conclusion

Durant mon stage, j'ai approfondi mes connaissances sur le domaine des eye tracking, j'ai pu découvrir les différentes technologies qui étaient sur le marché mais également comment un eye tracker fonctionne. J'ai également pu tester une caméra Raspberry et ses caméras associées au travers de programmes python à destination d'un usage d'eye tracker. La ligne de trame de mon stage était de découvrir les limites de la carte Raspberry Pi. J'ai effectué de nombreux tests, et mes conclusions sur la carte Raspberry Pi sont que la carte n'est malheureusement pas assez performante mais également de logiciels ne fonctionnent pas sur les cartes Raspberry à cause de leur architecture ARM. J'ai développé un programme qui a pour but de suivre la pupille de l'œil afin créer un eye tracking.

Au travers de stage, j'ai développé plusieurs compétences. J'ai su développer mon autonomie dans le travail, résoudre les problèmes rencontrés par moi-même mais ne pas hésiter à demander de l'aide si un problème persiste. J'ai appris à mieux utiliser les bibliothèques python comme openCv, dlib, numpy.

Ce stage m'a permis de m'intéresser au traitement d'image d'une autre façon plus appliquée et également de m'orienter sur différents choix d'avenir professionnels. Le secteur médical m'intéresse tout particulièrement, étudier des dispositifs qui aident des personnes est une entrée dans ce domaine.

Annexes

Annexe 1 : Document Word regroupant des nouveaux systèmes de pointage

Le document est disponible dans un GitHub : https://github.com/enonono1/STAGE

Le livrable fait 20 pages, regroupant 12 solutions de systèmes de pointages allant de la caméra simple au logiciel eye tracking.

Annexe 2 : Tableau regroupant des anciens et nouveaux systèmes de pointage

Le document est disponible dans un GitHub : https://github.com/enonono1/STAGE

TYPE	entreprise	matériel ou logiciel	ressources
Camera System - CAM30NT	alea technologies	matériel	https://www.intelligaze.com/en/CAM30NT
MAGIC EYE	EYEMOBILE PLUS	matériel	http://www.magickey.ipg.pt/Default.aspx
TM5 mini10	eye tech	matériel	https://www.domodep.shop/fr/pc-et-mac/64-commande-oculaire-eyetech-tm5-mini10.html
VISION KEY	eyecan	matériel	http://www.eyecan.ca/
Eyegaze Edge (Version Tablette)	eyegaze	matériel	https://eyegaze.com/products/eyegaze-edge/
eyesee-research	eyesee-research	logiciel	https://eyesee-research.com/eye-tracking/
EYETALK EYE TRACKER	EyeTalk Systems	matériel	https://eyecomtec.com/fr/3136-EyeTalk-Eye-Tracker
VT3 MINI	EyeTech Digital Systems	matériel	https://eyetechds.com/eye-tracking-products/vt3-mini-eye-tracker/
Eye tech VT2	EyeTech Digital Systems	matériel	https://imotions.com/hardware/eyetech-vt2/
GazeSense	eyeware	logiciel	https://eyeware.tech/fr/gazesense/
Beam	eyeware	logiciel	https://beam.eyeware.tech/
eyezag (Webcam Eye Tracking)	Eyezag	logiciel	https://eyezag.com/eye-tracking/webcam/
GP3 EYETRACKER	gazepoint	matériel	https://www.gazept.com/product/gazepoint-gp3-eye-tracker/
Gazeflow	gazerecorder	logiciel	https://gazerecorder.com/gazeflow/
Hawkeye Access	hawkeye	logicel / application	https://www.usehawkeye.com/accessibility
SEETECH	HumanElektronik	matériel	https://humanelektronik.de/produktkatalog/kommunikation/augensteuerung/22/augensteuerung-seetech-only
irisbond hiru	irisbond	matériel	https://www.irisbond.com/tecnologia/hiru/
irisbond manu	irisbond	logiciel	https://www.irisbond.com/tecnologia/mamu-eye-tracker/
IRISBOND DUO	irisbond	matériel	https://www.irisbond.com/producto/irisbond-duo/
Tellus I5	jabbla	matériel	www.jabblasoft.com/files/manuals/tellusi5_fr_A4.pdf
Tellus I6 Avec Mind Express	jabbla	matériel	https://www.jabbla.com/fr/appareils/tellus-6-et-i6/
EAS monoculaire et EAS binoculaire	LC Technologies	matériel	https://eyecomtec.com/fr/3131-EAS-monoculaire-et-EAS-binoculaire
Eye Connected	Pertech Solutions	matériel	https://www.pertech-solutions.fr/nos-solutions/eye-connected/
FaceLAB 5	Seeing Machines	matériel	https://eyecomtec.com/3132-faceLAB
EyeDee	suricog	matériel	https://www.suricog.fr/fr/#eyebrain-fr
eye tribe	the eye tribe	logiciel	https://theeyetribe.com/dev.theeyetribe.com/dev.theeyetribe.com/general/index.html
Commande oculaire PCEye 5 - Tobii Dynavox	tobii	matériel	https://cenomy.fr/la-commande-oculaire-pc-eye-5-de-tobii-dynavox-pour-piloter-son-pc/
Tobii I-Series : Tobii I13/I16	tobii	matériel	https://fr.tobiidynavox.com/pages/i-series-ap
Eye Mobile mini	tobii	matériel	https://aides-techniques.handicap.fr/p-commande-eye-mobile-mini-256-8939.php
TOBII EYE TRACKER 4C	tobii	matériel	https://fr.shopping.rakuten.com/offer/buy/3395735446/tobii-eye-tracker-4c.html
TOBII EYE TRACKER 5	tobii	matériel	https://gaming.tobii.com/product/eye-tracker-5/
Tobii Dynavox TD Pilot	tobii	matériel	https://fr.tobiidynavox.com/products/td-pilot
Tobii EyeMobile 5	tobii	matériel	https://www.cimis.fr/poursuites-oculaires-et-logiciels-associes/300-tobii-eyemobile-5.html
EYEMOBILE PLUS	Tobii Dynavox	matériel	https://www.comptoirdessolutions.org/innovation/eye-mobile-plus/
Tobii Pro Fusion	Tobii Pro	matériel	https://www.tobiipro.com/fr/produits/fusion/
Tobii Pro Glasses 3	Tobii Pro	matériel	https://www.tobiipro.com/fr/produits/tobii-pro-glasses-3/
Tobii Pro Spectrum	Tobii Pro	matériel	https://www.tobiipro.com/fr/produits/tobii-pro-spectrum/
Tobii Pro Nano	Tobii Pro	matériel	https://www.tobiipro.com/fr/produits/tobii-pro-nano/
ITU GAZE TRACKER	université copenhague	logiciel	https://sourceforge.net/projects/gazetrackinglib/

Annexe 3 : document PDF, différences entre une caméra sans filtre infrarouge et une caméra normal pour le traitement d'image à destination d'un eye tracker

Le document est disponible dans un GitHub : https://github.com/enonono1/STAGE

Le livrable fait 9 pages, il explique comment une image est capturer avec une caméra normal et une caméra sans filtre infrarouge. Il expliquer également pourquoi il est intéressant d'utiliser une caméra sans filtre infrarouge pour l'oculométrie.

Annexe 4 : Code python, détectant les yeux et les pupilles

```
import cv2
import dlib
import numpy as np
```

```
cap = cv2.VideoCapture(0) #0 est le numéro de périphérique de la caméra

if not (cap.isOpened()):
    print("camera ne peut pas s'ouvrir")

detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')
gauche = [36, 37, 38, 39, 40, 41]
droit = [42, 43, 44, 45, 46, 47]


while True:
    #ret obtient un indicateur de succès d'image
    ret, frame = cap.read()

    # si la frame est lu correctement alors ret = True
    if not ret:
        print("fin du stream, frame non reçu")
        break

    thresh = frame.copy()
    kernel = np.ones((9, 9), np.uint8)

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    rects = detector(gray)

    for rect in rects:

        shape = predictor(gray, rect)

        # initialise (x,y) coordonées
        coords = np.zeros((68, 2), dtype="int")
        # boucle des 68 points du visages puis les convertis
        # vers un (x,y) coordonnées
        for i in range(0, 68):
            coords[i] = (shape.part(i).x, shape.part(i).y)

        shape = coords

        mask = np.zeros(frame.shape[:2], dtype=np.uint8)

        cote = gauche
        points = [shape[i] for i in cote]
        points = np.array(points, dtype=np.int32)
        mask = cv2.fillConvexPoly(mask, points, 255)

        cote = droit
        points = [shape[i] for i in cote]
        points = np.array(points, dtype=np.int32)
        mask = cv2.fillConvexPoly(mask, points, 255)

        mask = cv2.dilate(mask, kernel, 5)


        eyes = cv2.bitwise_and(frame, frame, mask=mask)
        mask = (eyes == [0, 0, 0]).all(axis=2)
        eyes[mask] = [255, 255, 255]
        mid = (shape[42][0] + shape[39][0]) // 2


        eyes_gray = cv2.cvtColor(eyes, cv2.COLOR_BGR2GRAY)
```

```
        _, thresh = cv2.threshold(eyes_gray, 94, 255, cv2.THRESH_BINARY)
        thresh = cv2.erode(thresh, None, iterations=2)
        thresh = cv2.dilate(thresh, None, iterations=4)
        thresh = cv2.medianBlur(thresh, 3)
        thresh = cv2.bitwise_not(thresh)

        #oeil gauche

        cnts, _ = cv2.findContours(thresh[:, 0:mid], cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_NONE)
        try:
            cnt = max(cnts, key = cv2.contourArea)
            M = cv2.moments(cnt)
            cx = int(M['m10']/M['m00'])
            cy = int(M['m01']/M['m00'])

            cv2.circle(frame, (cx, cy), 4, (0, 0, 255), 2)
        except:
            pass

        #oeil droit

        cnts, _ = cv2.findContours(thresh[:, mid:], cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_NONE)
        try:
            cnt = max(cnts, key = cv2.contourArea)
            M = cv2.moments(cnt)
            cx = int(M['m10']/M['m00'])
            cy = int(M['m01']/M['m00'])

            cx += mid

            cv2.circle(frame, (cx, cy), 4, (0, 0, 255), 2)
        except:
            pass


        for (x, y) in shape[36:48]:
            cv2.circle(frame, (x, y), 2, (255, 0, 0), -1)

        # montre image avec the face detections + facial landmarks
        cv2.imshow('eyes', frame)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

cap.release()
cv2.destroyAllWindows()
```

Annexe 5 : Programme python détectant visage et affiche des points sur le visage

```python
import cv2
import dlib

# Load the detector
detector = dlib.get_frontal_face_detector()

# Load the predictor
predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")


cap = cv2.VideoCapture(0) #0 est le numéro de périphérique de la caméra

if not (cap.isOpened()):
    print("camera ne peut pas s'ouvrir")


while True:
    #ret obtient un indicateur de succès d'image
    ret, frame = cap.read()

    # si la frame est lu correctement alors ret = True
    if not ret:
        print("fin du stream, frame non reçu")
        break
    key = cv2.waitKey(1) & 0xFF

    gray = cv2.cvtColor(frame, code=cv2.COLOR_BGR2GRAY)


    faces = detector(gray)

    #pour chaque visage on vient récupérer des coordonnées des points que l'on veut (coordonnées pixel de l'image)
    for face in faces:
        x1 = face.left()
        y1 = face.top()
        x2 = face.right()
        y2 = face.bottom()


        landmarks = predictor(image=gray, box=face)
        #le N 27 cordonnées du point entre les deux yeux.
        # x = landmarks.part(27).x
        #y = landmarks.part(27).y


        for n in range(0, 68):
            x = landmarks.part(n).x
            y = landmarks.part(n).y


            cv2.circle(frame, center=(x, y), radius=3, color=(0, 255, 0), thickness=-1)

        cv2.circle(frame, center=(x, y), radius=5, color=(0, 255, 0), thickness=-1)
        cv2.rectangle(frame, pt1=(x1, y1), pt2=(x2, y2), color=(0, 255, 0), thickness=4)

        cv2.imshow('frame', frame)
        if key == 27: #Quitter avec la touche ESC
            break


cv2.destroyAllWindows()
```

Annexe 6 : Programme d'affichage de vidéo venant d'une caméra en multithreading

Le multithreading est la capacité d'un processeur à exécuter plusieurs threads (taches) simultanément. Par exemple, pour afficher une vidéo avec du multithreading on utilise 3 threads différents.

Pour faire tourner le programme, on besoin de la classe VideoShow qui affiche le flux vidéo, la classe VideoStream qui récupère le récupère le flux vidéo et met l'image récupérer dans une variable (frame) et du programme principale qui appel les classes.

```python
from threading import Thread
import cv2
class VideoStream:

    def __init__(self, src=cv2.CAP_DSHOW):
       self.stream = cv2.VideoCapture(src)
       (self.grabbed, self.frame) = self.stream.read()
       self.stopped = False

    def start(self):

        Thread(target=self.get, args=()).start()
        return self

    def get(self):
        while not self.stopped:
            if not self.grabbed:
                self.stop()
            else:
                (self.grabbed, self.frame) = self.stream.read()

    def stop(self):
        self.stopped = True
```

```python
from threading import Thread
import cv2

class VideoShow:

    def __init__(self, frame=None):
        self.frame = frame
        self.stopped = False

    def start(self):
        Thread(target=self.show, args=()).start()
        return self

    def show(self):
        while not self.stopped:
                cv2.imshow("Frame", self.frame)
                key = cv2.waitKey(1) & 0xFF
                if key == ord("q"):
                    break

    def stop(self):
        self.stopped = True
        cv2.destroyAllWindows()
```

```python
from videostream import VideoStream
from Show_video import VideoShow
import time


# démare thread stream video , show video
# démare le compteur
print("Start")
vs = VideoStream().start()
vs2= VideoShow(vs.frame).start()

time.sleep(2.0)



while True :

    if vs.stopped or vs2.stopped:
        vs2.stop()
        vs.stop()
        break

    frame = vs.frame
    vs2.frame = frame
```

= > diagramme qui montre comment ça fonctionne

Annexe 7 : Tests de performance des programmes Python

Tests affichage vidéo :

Pour tester la performance, on va calculer combien de boucle (itération) est effectuée par seconde entre la capture de l'image et son affichage.

J'ai testé les programmes avec les différentes caméra pour déterminer si une différence existait entre chaque caméra. Pour chaque caméra, j'ai également testé plusieurs Fps (Frame per seconde), qui correspondent à la vitesse d'envoie des images de la caméra. Plus le Fps est élevé, plus la vidéo sera fluide.

Les caméras pour Raspberry Pi ont un FPS qui peut être changé alors que la webcam de l'ordinateur a un FPS a 30.

	Fps 5	Fps 10	Fps 20	Fps 25	Fps 30	MP (méga pixels)
Caméra Module V2.1 (figure 5)	5 itérations /seconde	10 itérations /seconde	7 itérations /seconde	8 itérations /seconde	8 itérations /seconde	8
Caméra infrarouge mode normal (figure 6)	5 itérations /seconde	9 itérations /seconde	7 itérations /seconde	8 itérations /seconde	8 itérations /seconde	5
Caméra infrarouge mode nuit (figure 6)	5 itérations /seconde	9 itérations /seconde	7 itérations /seconde	8 itérations /seconde	8 itérations /seconde	5
Webcam Ordinateur	/	/	/	/	29,58 itérations /seconde	2,1

Figure 21 : tableau comparant les caméras Raspberry suivant leur nombre d'itération par seconde - programme simple

On observe que la webcam de l'ordinateur est plus rapide que les autres caméra mais la webcam à un capteur de moins bonne résolution que les caméras donc l'image sera de moins qualité.

Tests programmes suivi oculaire :

J'ai également fait des tests de performance de mon programme sur l'ordinateur, ces tests de performances sont les mêmes que ceux réalisés pour la carte Raspberry PI. Les programmes sont sur mon GitHub dans mon dossier tests_performance.

Programme 1 : Programme ouvrant la caméra webcam et affichant le Stream vidéo.

Programme 2 : Programme qui ouvre la caméra webcam et récupère une frame dans un thread et affiche le Stream vidéo dans un autre thread.

Programme 3 : Programme de Antoine lame sans modification.

Programme 4 : Programme de Antoine lame mais en plaçant la récupération d'une frame dans un thread et affiche le Stream vidéo dans un autre thread.

Programme 5 : Programme d'eye tracking final mais en plaçant la récupération d'une frame dans un thread, affiche le Stream vidéo dans un autre thread et toute la partie traitement de l'image dans un autre thread.

	Temps écoulé (secondes)	Nombre d'itération par seconde
Programme 1	20	29,58
Programme 2	20	855001
Programme 3	20	9,90
Programme 4	20	13,22
Programme 5	20	2,15

Figure 22 : tableau répertoriant vitesse des programmes python sur ordinateur

J'ai également réalisé un test sur la carte Raspberry Pi du premier programme de Gaze tracking mais le résultat est très lent par rapport au résultat de l'ordinateur (*figure 21*). Il a donc été décidé de continuer sur le pc pour le reste des tests.

Avec caméra vision nocturne	Temps écoulé (secondes)	Nombre d'itération par seconde
Programme 1	30	1,91

Figure 23 : Tableau répertoriant la performance du programme gaze tracking sur la carte Raspberry PI

Programme 1 : Gaze tracking simple

Références

1- https://lig-getalp.imag.fr/fr/accueil/

2- https://www.liglab.fr/fr/presentation

3- https://lig-getalp.imag.fr/fr/accueil/#:~:text=Au%20niveau%20national%20et%20international,%2DRECITAL%2C%20etc

4- https://www.leneurogroupe.org/copie-de-template-fiche-methode-19

5- https://www.tobii.com/group/about/this-is-eye-tracking/

6- https://stringfixer.com/fr/Eye_tracker

7- http://acces.ens-lyon.fr/acces/thematiques/neurosciences/glossaire/reflet-corneen

8- http://www.handicap.org/wp-content/uploads/2020/09/Etat-de-lart-des-differents-systemes-de-pointages-a-loeil-2020.pdf

9- https://fr.manuals.plus/Raspberry-Pi/raspberry-pi-4-mod%C3%A8le-b-manuel#ixzz7Yoixlcut

10 - https://www.societe-informatique-de-france.fr/wp-content/uploads/2020/04/1024-numero-15_Article8.pdf

11 - https://datascientest.com/machine-learning-tout-savoir

12- https://ca.tobiidynavox.com/pages/what-is-eye-tracking?lang=fr

13- https://gazeplay.github.io/GazePlay/fr

14- https://interaactionbox.afsr.fr/

15 - https://www.ionos.fr/digitalguide/serveur/know-how/le-multithreading-explique/#:~:text=Le%20multithreading%20r%C3%A9sulte%20d'une,mat%C3%A9riel%20et%20le%20multithreading%20logiciel

DOS DU RAPPORT

Etudiant : DANIEL Enora Année d'étude dans la spécialité : 4

Laboratoire : LIG

Adresse complète : 700 Avenue Centrale - 38401 St Martin D'Hères

Téléphone : +33 4 57 42 14 00

Responsable administratif : Pascale POULET

Téléphone : /

Courriel : /

Tuteur de stage : Didier Schwab

Téléphone : 04 76 28 47 11

Courriel : didier.schwab@univ-grenoble-alpes.fr

Enseignant-référent : Nathalie Guyader

Téléphone : 04 76 57 43 73

Courriel : nathalie.guyader@univ-grenoble-alpes.fr

Titre : Adaptation d'un dispositif oculaire destiné à la Communication Alternative et Augmentée à une Raspberry Pi

Résumé : Les eye tracking sont en constante évolution, ils sont utilisés pour de nombreux domaines dont la communication alternative et augmentée. Par le regard, des personnes ne disposant pas de leur membre peuvent communiquer avec les entourages. Ce rapport décrit alors les 3 mois durant lesquels j'ai dû tester la carte Raspberry PI pour en découvrir ses limites pour une utilisations en oculométrie. Les différentes missions étaient d'effectué une recherche des différents type d'eye tracking disponible sur le marché puis adapter des dispositif existants sur la carte et enfin tester et développer des programmes python sur la carte Raspberry Pi avec des caméras.