12/02/2021

Projet Informatique – 1ère année
PRO 3600

Curseur Oculaire

Livrable 1

Adam DAHAN
Alexandre CHAUSSARD
Simon CHEREL
Tom SALEMBIEN
Christophe TROALEN

Tuteur : Patrick HORAIN

Table des matières

1. Introduction

L'oculométrie est le terme généralement employé pour parler de suivi oculaire.
Il s'agit d'un ensemble de techniques permettant d'enregistrer et d'analyser les mouvements des yeux dans une vidéo. Pour des questions de précision, on emploie souvent des caméras infra-rouges lors de la phase de détection de la pupille car elle est particulièrement noire et délimitée sur l'image.

Dès lors, les techniques d'oculométrie permettent de calculer la direction du regard d'un individu face à une caméra. Une fois la détection de l'œil effectuée, l'objectif est d'étudier différentes caractéristiques du regard (direction, inquiétude, …) pour en tirer des informations sur le sujet d'étude.
On retrouve notamment l'analyse de ces données obtenues dans le cadre de la psychologie (autisme et communication involontaire), la médecine (pour les patients atteints de locked-in syndrom ou ALS), le marketing, les recherches sur le système visuel (communication entre individus) ou encore les interactions homme-machine comme c'est notre cas.

En effet, nous nous intéressons ici au contrôle du pointeur de la souris par le regard. Cette technologie alors relativement récente et novatrice a toutefois déjà connu bon nombre de logiciels et dispositif permettant sa bonne réalisation. On notera particulièrement le premier oculomètre par Edmund Huey en 1908 dans le cadre de la lecture, jusqu'au premiers contrôleurs de curseur de Tobii Technology, leader dans le domaine, autour des années 2008 (*source : Wikipédia*).
Par ailleurs, dans le cadre la crise sanitaire du COVID-19, la technologie de suivi oculaire pourrait connaitre un certain essor dans la mesure où elle permet un respect des gestes barrières en minimisant les contacts avec les machines publiques.

2. Cahier des charges

2.1 Description de la demande :

I. Les objectifs

Nous cherchons à développer un logiciel capable de reconnaître le visage d'un individu, reconnaitre les yeux du sujet, cibler la pupille, puis coordonner les mouvements de la pupille avec ceux du curseur en plaçant ce dernier à l'endroit où le sujet regarde l'écran.
On notera également le besoin d'une interface légère présentant 2 boutons permettant pour l'un d'échelonner le pointeur par rapport au regard, pour l'autre de quitter l'application.

En raison de la qualité des bibliothèques présentes dans le domaine de l'analyse d'image et la reconnaissance sémantique de forme, nous avons choisi de développer notre projet sous Python en utilisant la bibliothèque OpenCV (opencv.org).
Pour ce qui est des boutons on pourra utiliser la bibliothèque Tkinter implémentée sur Python.

Dès lors, la réalisation du projet passera par plusieurs étapes :

- Intégrer la caméra dans Python (potentiellement la caméra infra-rouge selon la disponibilité de notre Kinect 2)
- Détecter le visage du sujet face à la caméra en utilisant le FaceDetect d'OpenCV
- Donner la position des yeux dans l'image
- Cibler un œil sous forme d'imagette pour limiter les traitements d'image
- Détecter la pupille dans l'imagette par des méthodes algorithmiques (préférée dans le cadre du PRO3600) ou des méthodes d'apprentissages (déjà établies dans des bibliothèques comme OpenCV)
- Effectuer un étalonnage du regard à l'aide de points sur l'écran. Le nombre de points permet de définir une approximation polynomiale. On partira pour l'instant sur une approximation d'ordre 4.
- Estimer la position du curseur sur l'écran et appliquer la position au curseur à l'aide d'une bibliothèque adaptée (win32api & win32con ou encore PyAutoGUI)
- Afficher deux boutons d'interfaces effectuant pour l'un « quitter l'application » et l'autre lançant l'échelonnage.

II. Produits du projet

Dans ce projet, nous nous appuierons sur différentes bibliothèques :

- OpenCV : bibliothèque de détection du visage et des yeux et de traitement d'image
- Numpy : bibliothèque de calcul scientifique de Python utilisée par openCV
- PyAutoGUI : bibliothèque assurant le contrôle du curseur (pyautogui.readthedocs.io)
- Tkinter : bibliothèque graphique de Python (docs.python.org)

Nous aurons également besoin de matériel physique :

- Webcam ou Kinect 2 (pour l'infra-rouge, présente sur le campus et possédée par l'équipe)
- Si Kinect 2, un adaptateur USB pour l'utilisation sur ordinateur

III. Les fonctions du produit

Les fonctionnalités qui seront délivrées par le programme Python sont :

En ce qui concerne l'image et son traitement :

- Récupérer une image depuis la caméra lisible sous OpenCV (infra-rouge ou webcam)
- Traiter une image pour détecter le visage et les yeux et renvoyer ces deux informations sous format traitable par OpenCV
- Récupérer une imagette de l'œil droit présent sur une image d'un visage
- Traiter l'image par des techniques algorithmiques (méthode du gradient, seuils colorimétriques, méthode de Canny, méthode de Hough)
- Appliquer des effets visuels sur une image pour montrer où la détection d'un objet reconnu (œil, visage) a été faite
- Afficher une image
- Supprimer les fenêtres ouvertes par OpenCV

En ce qui concerne le curseur :

- Donner la priorité de mouvement à la souris
- Afficher l'étalonnage du regard par des points sur lesquels cliquer pour commencer à utiliser le mode oculaire
- Trouver la position du curseur sur l'écran par rapport à la position du regard
- Déplacer le curseur à une position
- Terminer l'utilisation du mode contrôle oculaire

En ce qui concerne l'interface :

- Afficher 2 boutons distincts
- Quitter l'application au clic sur le bouton « Quitter l'application »
- Lancer l'échelonnage au clic sur le bouton « Echelonner »

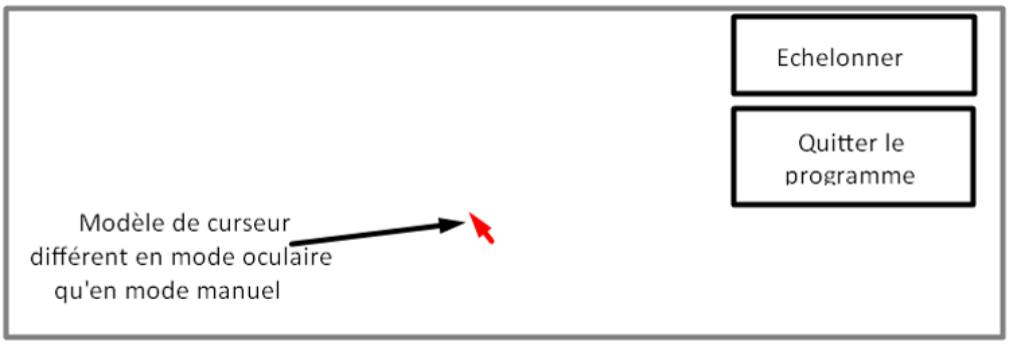
IV. Critères d'acceptabilité et de réception

Le bon fonctionnement de la technologie de suivi oculaire impose que le programme soit d'abord performant (utilisation en temps réel).

Il doit aussi être robuste : pouvoir fonctionner en cas de perte de contact visuel et proposer éventuellement un nouvel échelonnage si la précision devient faible.

Par ailleurs, le programme doit permettre la prise de contrôle manuel avec la souris (priorité à la souris pour le contrôle du curseur).

Le programme en fonctionnement devra s'appuyer sur la maquette suivante :



2.2 Contraintes

I. Contraintes de coûts

La caméra infra-rouge Kinect 2 a été achetée avant le projet par un membre de l'équipe.
Les bibliothèques étant toutes libres, il n'y a aucun frais pour ce projet.

II. Contraintes de délais

Le livrable final du projet est attendu pour le 18 mai, ce qui en fait notre date butoir.
Toutefois, le projet est attendu avant sur 2 livrables le 12 Février (structure) ainsi que le 16 mars (prototype) sur des objectifs précis.

III. Contraintes techniques

Comme on peut l'imaginer, la détection de la pupille dans une image est un challenge technique.

Aussi, il est important de bien faire attention à la précision de notre caméra qualifiée ici par sa résolution. En effet, une image de résolution trop faible peut entrainer une forte incertitude sur la détection de la pupille. De fait il y a un fort risque de faire des déplacements de curseurs erronés à cause d'une image de qualité médiocre.
Pour répondre à ce premier problème, nous proposons d'utiliser des caméras de bonne résolution (720p, 1080p) ou encore la Kinect 2 à haute résolution et sous infra-rouge, permettant ainsi une détection bien meilleure de la pupille en raison de son absorption importante dans l'infrarouge.
Toutefois cela pose une autre contrainte technique à résoudre : interfacer la Kinect 2 sur Python.
N'ayant pas encore la Kinect 2 en main, l'idée est pour l'instant en attente.
Sinon, les méthodes algorithmiques peuvent permettre d'améliorer de façon signifiante la qualité de traitement par analyse des formes (notamment les cercles dans notre cas), ce qui peut limiter le problème de résolution de la caméra.

Par ailleurs, il est faux de considérer qu'un visage plein face à la caméra est équivalent à un visage de biais sur la caméra. Ainsi pour limiter les erreurs de calculs liés à la profondeur de champ, nous nous plaçons dans des conditions idéales : il faut être bien en face de la caméra, avec un bon éclairage pour que le visage soit visible. Sinon il faudrait effectuer un étalonnage 3D, mais cela nous a été déconseillé par notre client M. Horain dans le cadre du PRO3600.

V. Clauses juridiques, etc.

Les bibliothèques étant en licence libre d'utilisation, le cadre juridique est exclu de notre projet.
On notera toutefois l'importance du droit d'image, aussi nous n'enregistrons aucune image captée par la caméra en dehors du traitement.

2.3 Déroulement du projet – Planification des Phases et Ressources

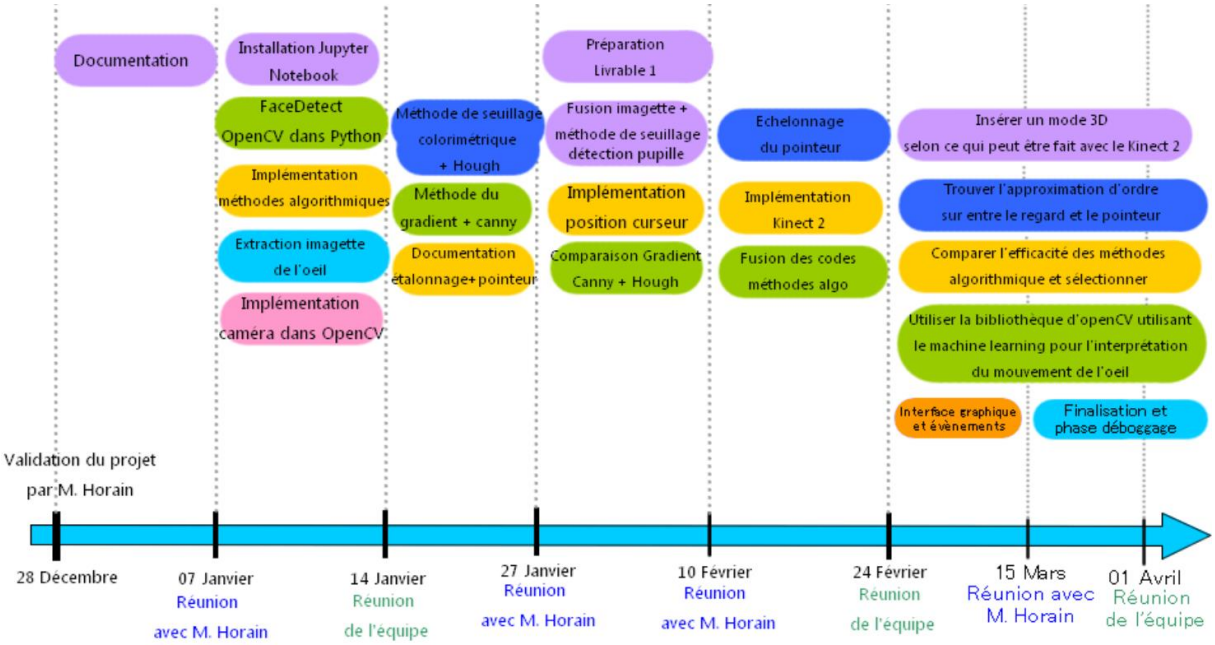


Diagramme Gantt du projet

Ce diagramme présente les tâches effectuées jusqu'à présent chronologiquement par l'équipe et celles à venir. On notera que **les tâches violettes sont des tâches de groupe** et les tâches des autres couleurs sont des tâches individuelles ou en binôme.

2.4 Authentification – Date & signature du chef de projet et du maître d'ouvrage

Simon Chérel

Adam Dahan

Alexandre Chaussard

Tom Salembien

Christophe Troalen

2.5 Annexes – Lister et joindre au cahier des charges les éventuels documents que le client peut mettre à disposition

Fichier à disposition : « cours transformation de Hough »

3. Développement

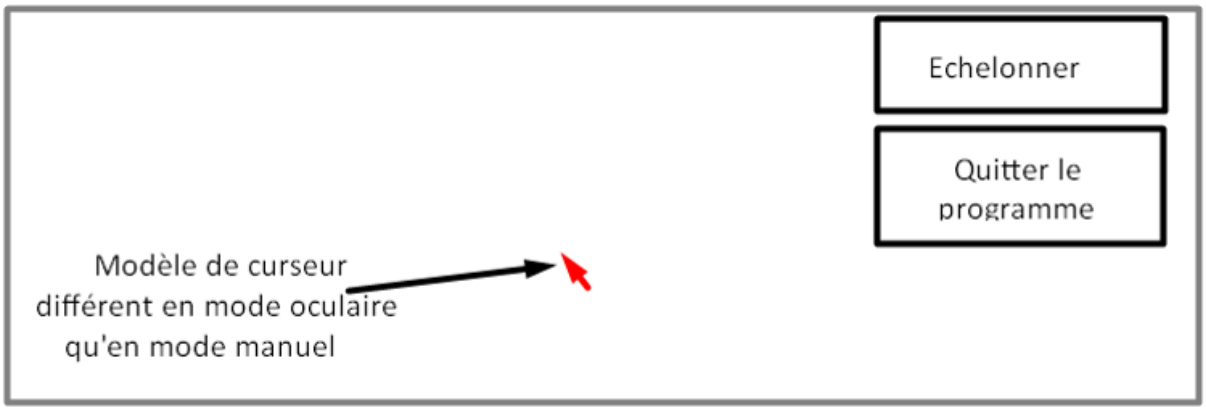
3.1 Analyse du problème et spécification fonctionnelle

Après lancement du programme, il est proposé à l'utilisateur une phase de calibrage.
Cette étape va permettre d'établir la relation mathématique entre la translation de la pupille et la position du curseur sur l'écran.
Pour ce faire, il sera imposé à l'utilisateur de regarder des points précis de l'écran comme par exemple les coins, sur lesquels il devra cliquer afin d'enregistrer le vecteur déplacement de sa pupille dans le programme.
Ainsi, le logiciel disposera des données nécessaires pour permettre le déplacement du curseur sur l'écran selon le regard définit par la pupille.

Cependant, si cette étape rencontre une quelconque difficulté, c'est tout le calibrage qui peut être faussé. Par exemple, un mauvais dimensionnement de l'écran peut entraîner l'existence de zones inaccessibles sur l'écran par le curseur oculaire ou encore un "dépassement" des limites de l'écran par celui-ci. De ce fait, il est envisagé de laisser à l'utilisateur un contrôle du curseur avec la souris primant sur celui issu du projet. La présence de deux boutons de contrôle : "Échelonner" et "Quitter le programme" accessibles par la souris, assurera donc la possibilité de rééchelonner le programme ou de pouvoir quitter l'application.

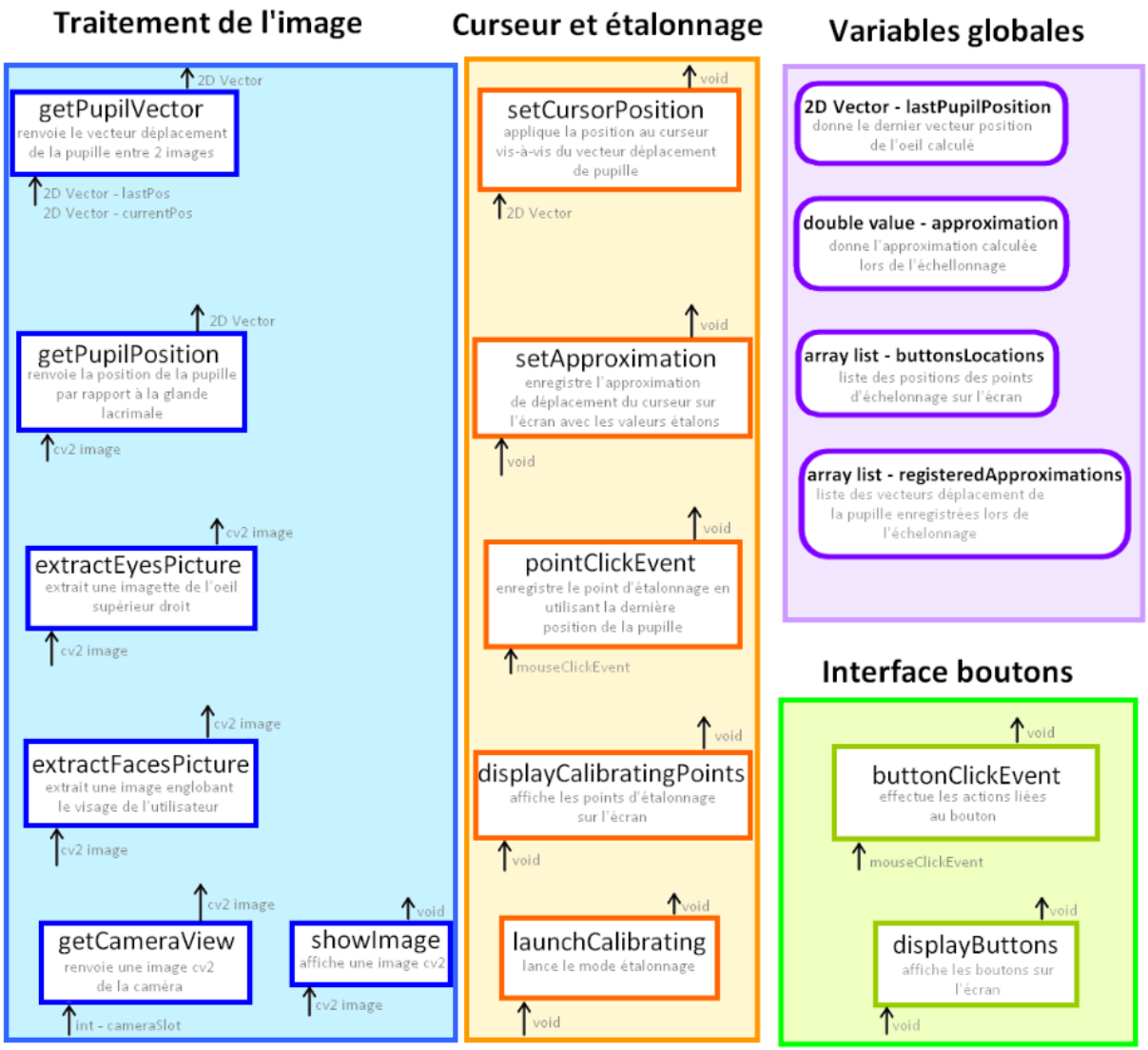
Afin d'effectuer le test de validation de notre programme, nous verrons si la phase d'étalonnage assurera bien un déplacement du curseur sur l'entièreté de l'écran avec précision. En effet en matière d'exigence du client, il n'est à tester que si le curseur est précis par rapport au point regardé et s'il peut atteindre tout point de l'écran. Le cas échéant, il faudra revoir la technique de fond d'étalonnage ou penser à utiliser une caméra à résolution plus élevée pour diminuer les incertitudes de calcul.

Enfin nous replaçons ici la maquette du produit imaginé :



3.2 Conception préliminaire

Le programme s'appuiera sur la description haut-niveau suivante :



Nous prévoyons les tests d'intégration suivants :

- getCameraView : vérifier que l'entier entrant correspond à un slot connecté (par défaut 0)
- showImage, extractFacesPicture, extractEyesPicture, getPupilPosition : vérifié que l'image d'entrée est non « None » et que l'affichage correspond à l'image souhaitée (on pourra s'appuyer sur le showImage après avoir vérifié son fonctionnement)
- getPupilVector : le vecteur sortant ne doit pas être nul et les vecteurs entrants initialisés. Pour s'assurer du bon fonctionnement, on pourra s'appuyer sur le rendu visuel image
- setCursorPosition : le déplacement du curseur peut être visualisé sur l'écran. Le vecteur entrant étant getPupilVector, nous n'avons pas besoin de vérifier son intégrité.
- Les autres fonctions sont des rendus visuels, l'affichage suffira donc à vérifier le fonctionnement.