INF573 project report

Adrien Lanne, Lilian Bichot

December 2021

Contents

1 Contextualisation

Le but de ce projet est de permettre à un utilisateur d'ordinateur (ou de toute interface comprenant des touches, par exemple une console de jeux) de remplacer son clavier (ou sa manette) par l'usage de son simple visage devant sa webcam. Il pourrait notamment permettre à des tétraplégiques d'utiliser un ordinateur sans commande vocale.

Il s'agit de détecter et d'interpréter des mouvements de tête capturés par webcam (les trois rotations dans chacune des deux directions, l'ouverture de la bouche), et de les convertir en commande clavier.

2 Problématisation

Pour ce faire, il nous fallait tout d'abord un outil de reconnaissance d'objet pour reconnaître en continu la présence d'un visage sur l'image capturée par la webcam. Ainsi, en détourant l'objet en question (le visage), nous pouvions déjà détecter une rotation selon l'axe normal à la caméra -notons-le x. En revanche, simplement détourer le visage ne nous permet a priori pas de détecter des rotations selon les axes y et z.

Pour rendre cela possible, il nous fallait un outil intelligent de reconnaissance tri-dimensionnelle d'objet, c'est-à-dire un outil qui interprète une vidéo en reconnaissant la présence à travers le temps d'un objet tri-dimensionnel dans la scène, et pouvant capturer sa structure, par exemple en plaçant dessus des points caractéristiques dont il connaîtrait les coordonnées tri-dimensionnelles à chaque instant.

Nous aurions pu essayer d'entraîner nous-mêmes une méthode *learning* créée de toutes pièces, mais il aurait fallu disposer d'un set de données considérable et d'une fonction d'évaluation adéquate, ce qui est particulièrement laborieux.

Nous avons donc choisi d'alternativement utiliser la fonctionnalité *face mesh* du célèbre module Mediapipe, qui place automatiquement des points caractéristiques sur le visage présent sur une vidéo et a accès en continu aux coordonnées 3D de tous ces points.

Comment extraire de ce nuage de points les informations qui nous intéressent, à savoir que la tête s'est tournée selon un certain axe ou que la bouche s'est ouverte ? Il nous faut extraire de l'information "coordonnées des points" les coordonnées de trois vecteurs caractéristiques, par exemple la verticale du visage, l'horizontale du visage i.e. l'axe reliant les yeux, et la normale principale au visage i.e. l'axe sortant entre les deux yeux. Comment extraire cette information ? Ici commence la valeur ajoutée.

3 Résolution

Nous avons eu l'idée d'effectuer une analyse par composante principale (PCA) sur le nuage de points tri-dimensionnel à chaque instant. En effet, le placement de points 3D sur un visage est presque planaire puisque le visage est globalement plat, et plus dispersé selon la verticale du visage que selon l'axe reliant ses yeux. Par conséquent, notre algorithme détecte en continu la direction de la verticale du visage, son horizontale et la normale au visage, et convertit les mouvements de ces vecteurs en commandes clavier.

Pour ce dernier point, nous avons utilisé la librairie Python *pyautogui*, qui permet de simuler la pression d'une touche ou combinaison de touches clavier ou le déplacement de la souris et son clic.

Ne restait alors plus qu'à allumer la webcam, lancer le script Python, changer de fenêtre, et laisser la magie opérer !

4 Détails

4.1 Capture du nuage de points : Mediapipe

La fonctionnalité *face mesh* de Mediapipe permet de se passer de capteur de profondeur, et se base sur une analyse de Procrustes. Elle est basée sur le co-fonctionnement en temps réel de deux réseaux de neurones : l'un fait de la reconnaissance d'objet sur la vidéo 2D, et l'autre cherche -par régression- à placer un ensemble de 468 points caractéristiques au plus proche des visages que le premier a trouvé. Comme le premier réseau de neurones permet de délimiter précisément l'emplacement du visage, l'algorithme complet peut consacrer la majeure partie de sa capacité à la précision du placement des points.

4.2 Extraction des vecteurs clé : PCA

Plutôt que d'utiliser un algorithme PCA d'une bibliothèque existante, nous avons préféré prendre ce projet comme un prétexte pour ré-implémenter nous-mêmes cet algorithme. De plus, cela nous a permis de voir plus clair dans la structure du code et d'implémenter dans le même parcours du nuage une autre fonctionnalité (le calcul de la dispersion, sur laquelle nous reviendrons), et donc de diminuer la complexité théorique.

Une fois les vecteurs clé obtenus, on en extrait les angles caractéristiques par des calculs trigonométriques d'angles d'Euler.

4.3 Conversion en commande clavier : *pyautogui*

Le coeur de notre implémentation se situe dans la classe *key_pressed*. La méthode *update* regarde si la tête est tournée dans une direction (auquel cas une touche doit être pressée), et compare avec une éventuelle touche déjà en cours de pression.

Nous avons implémenté à la fois :

- un mode discret (tourner la tête vers la gauche au-delà de l'angle limite appuie une fois sur la touche associée peu importe le temps passé au-delà de l'angle limite), plus utile pour écrire du texte ou utiliser des logiciels
- un mode continu (tant que ma tête est tournée au-delà de l'angle limite, cela appuie en continu sur la touche associée), plus utile pour certains jeux vidéo

Pour éviter des grésillements lorsque la tête est tournée presque pile à l'angle limite, nous avons choisi d'ajouter un phénomène d'hystérésis entre l'appui et le relâchement d'une touche.

4.4 Complément : l'ouverture de la bouche

Nous disposions alors par exemple de six commandes discrètes : non vers la gauche, non vers la droite, oui vers le haut, oui vers le bas, inclinaison de la tête à gauche, et inclinaison à droite.

Cependant, il nous manquait une commande à part, qui pourrait servir pour valider une décision ou mettre fin à une procédure. C'est en discutant avec les encadrants au sujet de l'emploi de notre implémentation pour jouer à un jeu vidéo de tir que nous avons évoqué l'emploi de l'ouverture de la bouche pour tirer. Par exemple, dans un tel jeu, les six rotations de la de l'utilisateur serviraient pour déplacer le personnage dans les quatre directions au sol (avancer, reculer, pas chassés à gauche, pas chassés à droite) et pour orienter son corps (se tourner vers la gauche ou vers la droite), et l'ouverture de la bouche pour effectuer un tir. Restait à inventer une façon de détecter ce signal.

Le problème est que la fonctionnalité *face mesh* de Mediapipe ne détecte pas réellement de bouche, mais ne fait que placer des points à des endroits caractéristiques. Nous avons donc contourné le problème en remarquant que l'ouverture de la bouche entraîne une augmentation de la dispersion moyenne des points du nuage.

Par conséquent, notre algorithme détecte lorsque la dispersion augmente ou diminue par rapport à une valeur de référence régulièrement mise à jour, et actionne la commande en conséquence. Comme Mediapipe détecte la position 3D des points, rapprocher ou éloigner sa tête de la caméra n'est pas censé augmenter ou diminuer la dispersion du nuage de points.

Notons que cette commande permet également d'utiliser des combinaisons de touche, puisqu'il est possible de tourner sa tête alors que sa bouche est encore ouverte.

4.5 *Tuning* des paramètres

La fixation des seuils de l'hystérésis à 5° et 10° d'angle s'avère assez satisfaisante pour les trois rotations. Le *tuning* des seuils de l'hystérésis de l'ouverture de la bouche s'avère particulièrement délicat, car la variation de dispersion associée à ce mouvement de tête est à peine plus grande que celles issues du bruit de Mediapipe. Nous sommes restés sur des seuils d'hystérésis de 102% et 104% par rapport à une valeur de référence régulièrement mise à jour, mais ce n'est pas encore parfait et parfois la commande se déclenche inopinément ou refuse de se déclencher quand elle devrait.

5 Résultats

Sur un ordinateur portable moyen, le nombre moyen de *fps* de l'exécution de l'algorithme est de 20. En effet, la lecture de chaque image prend en moyenne 0.013 seconde, son traitement par Mediapipe prend en moyenne 0.013 seconde également, et notre calcul de PCA et l'activation éventuelle de commande clavier prend en moyenne 0.021 seconde.

Dans un souci d'optimisation des performances, notons qu'il serait en théorie possible d'augmenter le nombre de *fps* jusqu'à la limite de capture de la caméra, en parallélisant le traitement par Mediapipe et le calcul de PCA et de commande clavier sur d'autres coeurs.

6 Prolongements

6.1 Nombre de commandes

La pression de sept touches est largement suffisante pour certains usages (par exemple des jeux vidéo simples), et très peu pratique pour d'autres. Par exemple, pour écrire un texte sur l'ordinateur, en supposant que 7x7=49 caractères suffisent, il faudrait pour chaque caractère utiliser une première commande pour choisir parmi sept paquets de sept caractères différents celui dans lequel se trouve le caractère qui nous intéresse, puis utiliser une deuxième fois la commande pour choisir le caractère parmi les sept, ce qui nous fait deux mouvements de tête par caractère.

6.2 *Joystick*

Notre méthode est adaptée à l'usage de jeux vidéos ou de logiciels comme Blender, Photoshop, Illustrator etc. Cependant, comme évoqué ci-dessus, elle s'avère inadéquate pour permettre la rédaction d'un texte.

Cela dit, si nous n'avons pas implémenté d'interface utilisateur développée en raison de l'éloignement avec le sujet du cours, il serait aisé d'afficher en transparence à l'écran un cercle contenant les différents caractères, et de choisir chaque caractère en orientant sa tête puis en ouvrant la bouche pour valider.

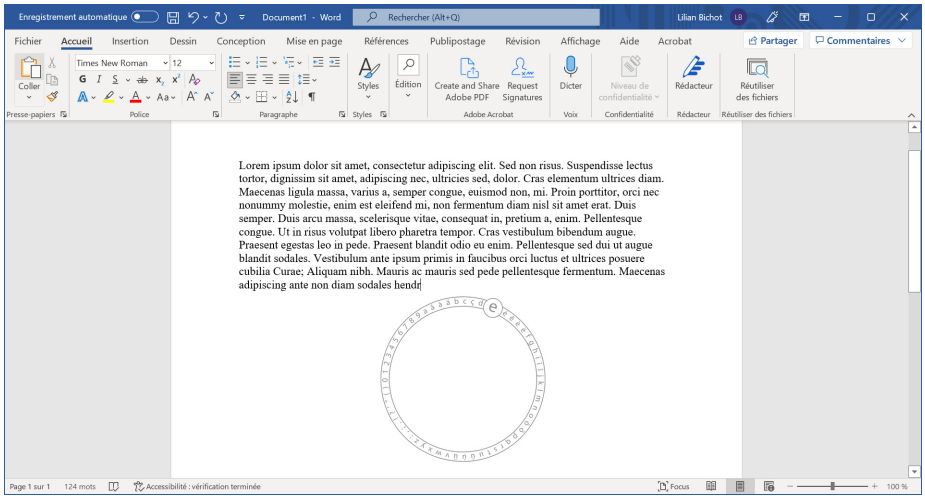

Figure 1: Ce à quoi l'interface pourrait ressembler

Figure 2: Simplement en bougeant la tête !

6.3 La souris

En fonctionnant sur le même principe que le *joystick* ci-dessus, il est également facile de commander sa souris plutôt que son clavier en bougeant la tête ! La direction de déplacement peut être donnée par la direction de la tête, la rapidité du déplacement de la souris par l'amplitude du mouvement de la tête, et -par exemple- le clic par l'ouverture de la bouche.

6.4 Translations

Il eut été facile de détecter une translation de la tête selon l'une des directions caractéristiques pour ajouter encore six commandes clavier supplémentaires, mais comme il est particulièrement désagréable d'avoir à déplacer sa chaise de bureau de gauche à droite, d'avant en arrière ou -pire encore- de haut en bas, nous avons choisi de ne pas implémenter cette fonctionnalité.

6.5 Clignements et mouvements d'yeux

Dans une optique de versatilité, un prolongement tout naturel serait d'utiliser la fonctionnalité *Iris* de Mediapipe pour détecter les clignements et mouvements des yeux. Par exemple, la direction du regard pourrait indiquer où déplacer la souris, et les clignements d'un oeil ou des deux pourraient constituer des commandes clé similaires à l'ouverture de la bouche.

Comme nous avions déjà utilisé le module *face mesh* et que l'étude promettait d'être similaire, nous avons choisi de ne pas creuser cette voie pour le moment.

7 Bibliographie

Site de Mediapipe : https://google.github.io/mediapipe/solutions/face_mesh