Rapport projet vision numérique

Interface perceptuelle 2 : Contrôle par le geste de la main

Traval remis à :
Denis Laurendeau

Travail presenté par :
Mickael Chansavang
Laurence Delisle
Anh-My Buis
Long Tran

Cours GIF-7001 - Vision Numérique

18 décembre 2022

1 Description brève de la solution proposée

1.1 Description du projet

Le projet d'interface perceptuelle avec la main aura pour but de contrôler l'interface de l'application Youtube en utilisant seulement les signes de la main, c'est-à-dire que l'utilisateur pourra par exemple jouer une vidéo, l'arrêter, contrôler le niveau du volume et même faire avancer ou reculer la vidéo de 5 secondes. Un des défis du projet est de pouvoir faire la différence entre la diversité des signes pouvant être utilisés par l'utilisateur.

1.2 Correspondances actions, raccourcis claviers et gestes

Tout d'abord, il fallait choisir les différents signes dont l'utilisateur pourra se servir et de tous les raccourcis qui seront assignés à ceux-ci. Il fallait bien faire attention à utiliser des gestes bien distinctifs entre eux, sinon cela pourrait causer une confusion pour celui qui utilisera le programme. Voici ci-dessous un tableau représentant les correspondances entre les gestes et les raccourcis du clavier.

Action	**Raccourci clavier**	**Geste**
Play	Barre d'espace	Tous doigts levés collés ensemble
Pause	Barre d'espace	Poing fermé
Avancer 5 secondes	Flèche droite	Pouce vers gauche et poing fermé
Reculer de 5 secondes	Flèche gauche	Pouce vers droite et point fermé
Sous-titres	c	Tous doigts levés et écartés
Plein écran	f	Pouce, index et petit doigt levé
Volume plus, moins	Symbole de volume +, -	Index et pouce qui se touchent et le reste des doigts fermés
Arrêt du programme	-	Petit doigt levé seulement
Suivi du doigt	Souris	Index levé seulement
Clic gauche	Clic gauche de souris	Index et majeur levés
Clic droit	Clic droit de souris	Index, majeur et annulaire levés

Table 1: Correspondances actions, raccourcis claviers et gestes

Il est à noter que l'action Suivi du doigt est un mode dans lequel les actions Clic et Clic droit peuvent être réalisées. Par conséquent, les clics ne seront pas exécutés par le programme si le doigt n'est pas déjà suivi et simulé dans la souris.

1.3 La détection de la main

Pour la détection de la main il faut d'abord récupérer l'image de la caméra. Nous avons alors utilisé une webcam. L'image de la webcam sur laquelle nous allons travailler est donnée par la librarie OpenCV de python. Elle nous permet de lire une image sur la webcam puis

de la transformer en objet python où l'on va avoir plusieurs informations notamment le RGB de l'image de la webcam.

Une fois cet objet récupéré nous allons la transmettre à la solution MediaPipe Hand de la librairie MediaPipe de Google sur python. Elle nous permet, à partir d'une image, de détecter jusqu'à n mains (n est choisi arbitrairement) grâce à plusieurs modèles de machine learning qui travaillent ensemble. Sur chaque main détectée, il va placer 21 points de repères 3D (x,y,z) appelés "landmarks". On va alors utiliser ces landmarks afin de différencier les différents gestes qui nous intéressent. Ces coordonnées sont normalisées par rapport à la taille de l'écran, donc x, y et z sont compris entre 0 et 1. On a alors pour $x = y = 0$ le coin haut-gauche, et $x = y = 1$ le coin bas-droite de l'image de la caméra.

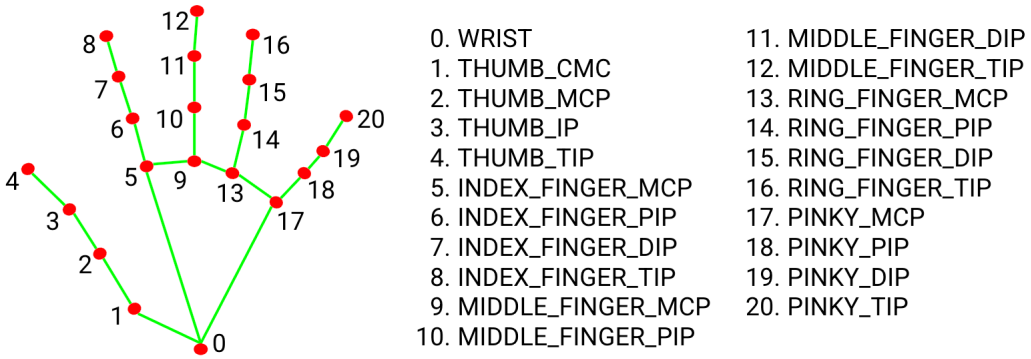

Figure 1: hand landmarks

Source: https://google.github.io/mediapipe/solutions/hands.html

1.4 Différentiation des gestes

Pour différencier les gestes, nous avons implémenté différentes fonctions pour détecter la quantité de doigts qui sont verticaux, la quantité de doigts horizontaux, la présence ouverte ou fermée du pouce et la distance entre deux doigts. Pour la vérification des doigts verticaux, on compare la coordonnée en y (la hauteur) du bout des doigts par rapport à la coordonnée en y de la jointure du milieu des doigts. D'un autre côté, la vérification des doigts horizontaux se fait en comparant la composante en x de ces mêmes repères. Dans le schéma de la main (Figure 1), cela représente les piliers suivants respectifs pour les différents doigts : index - (8, 6), majeur - (12, 10), annulaire - (16, 14), auriculaire - (20, 18) et pouce - (4,2). Il faut noter que la détection du pouce n'est faite qu'à l'aide de ses coordonnées en x dû au fait qu'il ne peut pas plier verticalement. Pour différencier les mouvements comme les doigts ouverts ou fermés, on utilise la notion de distance entre les bouts des doigts.

Une fois toutes ces informations recueillies, il suffit d'observer la présence ou l'absence des doigts détectés pour déterminer quelle forme de la main est observée par la caméra. Puis, un mapping est fait entre la forme de la main et l'action à faire dans la page youtube.

1.5 Du geste à l'action

Chaque forme de la main a été associée à une action selon la section 1.2. Par contre, certains cas particuliers ont été traités. Par exemple, les clic droit et clic gauche ne sont permis qu'en mode Suivi de souris. Par conséquent, ces actions sont accomplies seulement si le mode Suivi de souris est activé. Donc, nous avons utilisé la notion d'action précédente pour permettre à l'algorithme de suivre dans quelle action il se trouvait précédemment. Cette même notion a été utilisée pour détecter si l'action associée à la forme de la main qui a été détectée est une nouvelle action. Si ce n'est pas le cas, on ne veut pas refaire l'action, car cela provoquerait une répétition sans fin de la même action. Donc, chaque action est jouée seulement si c'est une nouvelle action ou après un certain temps appelé cooldown que l'on verra dans la prochaine section. Par exemple, cela a été très utile pour gérer la souris car la première fois que nous entrons dans le mode souris il faut replacer la souris au milieu de l'écran. On la replace qu'une seul fois, ensuite on veut seulement faire mouvoire la souris et non la replacer à chaque fois. On a donc absolument besoin de savoir si nous entrons en mode souris ou bien si l'on y est déjà.

1.6 Gestion de la stabilité des gestes détectés pour les actions

OpenCV utilisant des solutions de machine learning, il se peut que d'une frame à une autre la position des landmarks ne soient pas stables et nous retourne donc des positions erronées. Cela est problématique, car lors de la phase de différenciation des gestes, un geste peut être non reconnu ou pire un geste erroné peut être reconnu. On va alors garder en mémoire les k derniers gestes reconnus (k est choisi arbitrairement ici $k = 5$) et calculer le mode, c'est à dire le geste qui est le plus présent parmi ceux en mémoire. On utilisera alors ce geste reconnu pour le choix des actions.

Un autre problème à gérer était le spamming. Si nous ne mettons pas de solution en place alors à chaque image traitée, une action clavier/souris serait exécuté. On a alors mis un timer en place, une action peut être effectuée seulement toutes les s secondes (ici nous avons mis $s = 2$) excepté pour le tracking de la main pour la souris et le volume. Ce temps de repos avant qu'une nouvelle action soit engagée s'appelle cooldown.

2 Performances

Lien vers la vidéo : https://youtu.be/Q2ZEq7RuNRU

2.1 Réussites

Dans la vidéo, on voit que le système détecte bien la main avec tous ses repères, de plus, lorsque nous faisons un signe de la main le programme indique quel signe a été effectué sur la fenêtre de la caméra afin d'offrir une meilleure expérience utilisateur et de nous permettre d'évaluer la robustesse de notre système. Nous remarquons que le système réagit très bien aux différents gestes effectués dans la vidéo. Par exemple, la main sans écarter les doigts est bien reconnue comme l'action de lancer la vidéo. Le poing fermé est aussi bien détecté par le programme afin de stopper la vidéo. De plus, mettre le pouce à l'horizontale vers la gauche ou la droite avec le poing fermé fonctionne correctement afin de faire avancer ou reculer la vidéo de 5 secondes. Montrer la paume avec les doigts écartés permet parfaitement d'activer ou de désactiver les sous-titres. Le mode plein écran est aussi fonctionnel lorsque nous levons l'index, l'auriculaire et le pouce. Afin de contrôler le volume, coller son pouce et son index en ayant les autres doigts levés et lever sa main sur l'axe y fonctionne très bien. De ce qui est du suivi du doigt, lever seulement son index permet une navigation facile sur l'interface de Youtube, la simulation d'un clic droit en levant le majeur avec l'index levé et d'un clic gauche en levant l'index, le majeur et l'annulaire fonctionne aussi très bien.

2.2 Limitations

Afin de rendre la détection des gestes robustes, des critères uniques ont été utilisés pour détecter les gestes, tels que le nombre de doigts verticaux ou horizontaux. Par contre, certains gestes, dépendant de comment ils sont faits, peuvent se ressembler et porter à confusion à l'algorithme, comme le poing fermé seul et le poing fermé avec le pouce sorti. D'un autre côté, l'utilisation du mode statistique mentionné à la section 1.6 permet de palier à ce comportement, pour ne faire l'action que lorsque l'algorithme est certain. Ce n'est évidemment pas parfait et provoque certaines ambiguïtés lors de tests. De plus, le suivi du doigt n'est pas très précis et la simulation de clics peut être fastidieux à réaliser.

De plus nous sommes limité par la librarie utilisée MediaPipe et de la qualité de notre caméra. En effet, pour la détection des landmarks, lorsque l'on se trouve loin de la caméra (disons 3 mètres) les landmarks ont tendance à s'agglutiner sur un même point/pixel ce qui rend la tâche compliquée pour notre programme de différenciation des gestes.

Une autre limitation est simplement la physique. En effet, lors du suivi du doigt pour l'utilisation de la souris nous rotationnons autour de notre buste. Par conséquent, si l'on va d'un bout à un autre de l'écran la caméra ne voit pas notre main du même angle. Là où au départ le pouce est visible, à la fin du trajet il ne le sera plus, donc MediaPipe essaye de deviner où le pouce se situe caché derrière notre main ce qui se conclu souvent par des landmarks ambiguës. Le geste n'est alors plus reconnu.

3 Améliorations possibles

Une option que nous n'avons pas explorée est la détection de 2 mains. En effet, les gestes ont été conçus pour n'utiliser qu'une seule main, donc la deuxième ne cause pas erreur au programme, mais ne lui est pas non plus utile. Nous aurions pu utiliser des gestes qui demandent une combinaison des deux mains, rendant les gestes beaucoup plus variés.

La détection des gestes se fait de manière "hard coded". Par contre, nous aurions pu opter pour une approche d'apprentissage machine qui aurait appris sur un échantillon de gestes que nous avons définis. Ce genre d'algorithme serait plus robuste à des changements mineurs dans la position de la main et performerait probablement mieux.

De plus, seulement deux repères (celui du haut et du milieu de chaque doigt) ont été utilisés par doigt pour déterminer si le doigt est ouvert à la vertical, horizontal ou fermé. Pour améliorer la précsion de détection, il aurait été possible d'utiliser les 4 repères disponibles sur chaque doigt afin de s'assurer d'une bonne détection. Par contre, cela requière davantage de temps de traitement et aurait pu causer des lenteurs en temps réel. Aussi, les performances sont relativement bonnes avec seulement deux repères, donc il serait intéressant de voir si utiliser davantage de repères a une amélioration significative sur la précision de détection des formes de la main.

Afin de palier au problème de distance et des différents types de tailles de mains, nous pourrions trouver une échelle afin de normaliser les distances entre les doigts en fonction de la taille de la main et non de la normalisation par rapport à la résolution de la caméra. Une piste serait d'utiliser la taille des métacarpes comme échelle.

Nous n'avons pas beaucoup utilisé la coordonnée z de la main, car elle donnait des résultats assez aléatoires, cependant elle serait bien utile pour augmenter la robustesse de la détection de geste et ajouterait aussi plus de diversité dans les mouvements. L'idée serait alors de rajouter une deuxième caméra afin de pouvoir calculer la position de chaque landmarks en 3D de façon plus juste à partir des coordonnées x et y qui eux semblent beaucoup plus stables.

4 Conclusion

Pour conclure, il a été possible dans ce projet d'élaborer un système de détection de la main en temps réel relié à l'interface Youtube. Différents gestes de la main sont traduits en actions sur la page web, ce qui permet à un utilisateur de naviguer de à l'aide de sa main et non de sa souris ou son clavier directement. L'approche adoptée performe bien pour des gestes faiclement différenciables, mais pourrait être améliorée au niveau de la détection des landmarks de la main et de la robustesse à la position et la forme de celle-ci.

5 Références

ANALYTICS VIDHYA. *Building a hand tracking System using openCV*. 2021.
https://www.analyticsvidhya.com/blog/2021/07/building-a-hand-tracking-system-using-opencv/