Lacheze Enzo

**Tache 14800: Trouver un code permettant de reconnaitre**

 **les mouvements d’une main avec une caméra**

code1 :

Le code permet de réaliser un système de reconnaissance des gestes de la main en temps réel en utilisant le cadre MediaPipe et Tensorflow en OpenCV et Python.

Le code utilise opencv sur du python via le package openCV-python

**MediaPipe :**

MediaPipe est un cadre de solutions d'apprentissage automatique personnalisable développé par Google. Il s'agit d'un cadre open-source et multiplateforme, et il est très léger. MediaPipe est livré avec des solutions d'apprentissage automatique pré-entraînées telles que la détection des visages, l'estimation de la pose, la reconnaissance des mains, la détection des objets, etc.

**TensorFlow :**

TensorFlow est une bibliothèque open-source pour l'apprentissage automatique et l'apprentissage profond développée par l'équipe des cerveaux de Google. Elle peut être utilisée pour toute une série de tâches, mais elle est particulièrement axée sur les réseaux neuronaux profonds.

Version utilisée :

Python 3.8.8
OpenCV 4.5
MediaPipe 0.8.5
TensorFlow 2.5.0
Numpy 1.19.3

Etape 1 : Importation des packages nécessaires



Etape 2: Initialiser les modèles

Le module mp.solution.hands exécute l'algorithme de reconnaissance des mains. Nous créons donc l'objet et le stockons dans mpHands.

En utilisant la méthode mpHands.Hands, on configure le modèle. Le premier argument est max\_num\_hands, ce qui signifie le nombre maximum de mains qui seront détectées par le modèle dans une seule image. MediaPipe peut détecter plusieurs mains dans une seule image, mais nous ne détecterons qu'une seule main à la fois dans ce projet.

Mp.solutions.drawing\_utils va dessiner les points clés détectés pour nous afin que nous n'ayons pas à les dessiner manuellement.

Initialisation de TensorFlow :



En utilisant la fonction load\_model nous chargeons le modèle pré-entraîné de TensorFlow.

Le fichier Gesture.names contient le nom des classes de gestes. Nous ouvrons d'abord le fichier en utilisant la fonction open intégrée de Python, puis nous lisons le fichier.

Après cela, nous lisons le fichier en utilisant la fonction read().

Sortie :

['okay', 'peace', 'thumbs up', 'thumbs down', 'call me', 'stop', 'rock', 'live long', 'fist', 'smile']

Le modèle peut reconnaître 10 gestes différents.

Etape 3 : lire les images d’une camera



Nous créons un objet VideoCapture et passons un argument '0'. Il s'agit de l'ID de la caméra du système. Dans ce cas, nous avons une webcam connectée au système. Si vous avez plusieurs webcams, changez l'argument en fonction de l'ID de votre caméra. Sinon, laissez-le par défaut.

La fonction cap.read() lit chaque image de la webcam.

La fonction cv2.flip() retourne l'image.

La fonction cv2.imshow() affiche l'image dans une nouvelle fenêtre openCV.

La fonction cv2.waitKey() garde la fenêtre ouverte jusqu'à ce que la touche 'q' soit pressée.

Etape 4 : détection des points de la main



MediaPipe travaille avec des images RGB mais OpenCV lit les images au format BGR. Ainsi, en utilisant la fonction cv2.cvtCOLOR(), nous convertissons l'image au format RGB.

La fonction process prend une image RGB et retourne une classe de résultat.

Ensuite, nous vérifions si une main est détectée ou non, en utilisant la méthode result.multi\_hand\_landmarks.

Après cela, nous bouclons sur chaque détection et stockons les coordonnées dans une liste appelée points de repère.

Ici, la hauteur (y) et la largeur (x) de l'image sont multipliées par le résultat car le modèle renvoie un résultat normalisé. Cela signifie que chaque valeur du résultat est comprise entre 0 et 1.

Et finalement, en utilisant la fonction mpDraw.draw\_landmarks(), nous dessinons tous les points de repère dans l'image.

Etape 5 : Reconnaitre les gestes de main



La fonction model.predict() prend une liste de points de repère et renvoie un tableau contenant 10 classes de prédiction pour chaque point de repère.

La sortie ressemble à ceci-

[[2.0691623e-18 1.9585415e-27 9.9990010e-01 9.7559416e-05

1.6617223e-06 1.0814080e-18 1.1070732e-27 4.4744065e-16 6.6466129e-07 4.9615162e-21]]

Np.argmax() retourne l'indice de la valeur maximale dans la liste.

Après avoir obtenu l'index, nous pouvons simplement prendre le nom de la classe dans la liste classNames.

Ensuite, en utilisant la fonction cv2.putText, nous affichons le geste détecté dans le cadre.

Code 2 :

Ce projet utilise opencv2 sur python pour détecter les mains à travers la webcam. On utilise ensuite cette information pour créer un modèle de main dans Unity et faire interagir cette main avec l'environnement. J'ai l'intention d'utiliser ce projet pour la conception de jeux innovants utilisant la détection du mouvement de la main IRL.

Avantage :

-utilisation pour faire des jeux qui utilisent la détection de main comme gameplay

Inconvénients :

-peu d’explication du code

-pas très utile pour une utilisation autre que unity

Lien :

<https://github.com/tayyababbas2000/Hand-Recognition.git>

Code 3 :

Le code permet de suivre un point comme un doigt d’une main.

Avantage :

-Pratique pour nombreuses utilisation.

-Permet de suivre le mouvement d’un point

Inconvenient:

-Pas sûr que ce soit utile dans ton cas, le programme permet de dessiner et de suivre un point mais il ne marque pas tous les mouvements de la mains comme les autres programmes.

<https://github.com/akshaybahadur21/HandMovementTracking.git>

Code 4 :

Ce code est basé sur le même principe que le premier code à la différence qu’il n’utilise pas TensorFlow, il n’a donc pas de signes de main prédéfinis. Il est donc plus simple, moins lourd mais n’a pas la reconnaissance de gestes disponible. On a bien les différents points de la main qui s’affichent.



Avantage :

-Affichage des points de la main

-Code plus court et moins lourd que le premier code

Inconvénient :

-ne possède pas la reconnaissance de mouvements

Code 5 :

Le code est écrit en c++ avec openCV

Le code fonctionne ne suivant plusieurs étapes :

-détection de la main au premier plan

-effacement de l’arrière-plan ne gardant que la main

-modification du contraste, ne laissant que la main en blanc et tout le reste en noir.

-détermination des points de la main.

Lien : <https://github.com/PierfrancescoSoffritti/handy.git>

Code 6 :

Ce code c++ permet de traquer les déplacements de la main :

-Le code traite l’image puis créer un rectangle qui va contenir la main, le programme ensuite compare la position de la main par rapport au rectangle ce qui lui permet de déterminer le déplacement de la main.

Note : de base le programme ne détecte que les déplacements horizontaux, il faut décommenter les lignes de code a la fin du programme pour détecter les déplacements verticaux.

Lien : https://github.com/iitmcvg/Hand-Gesture-Based-Control.git