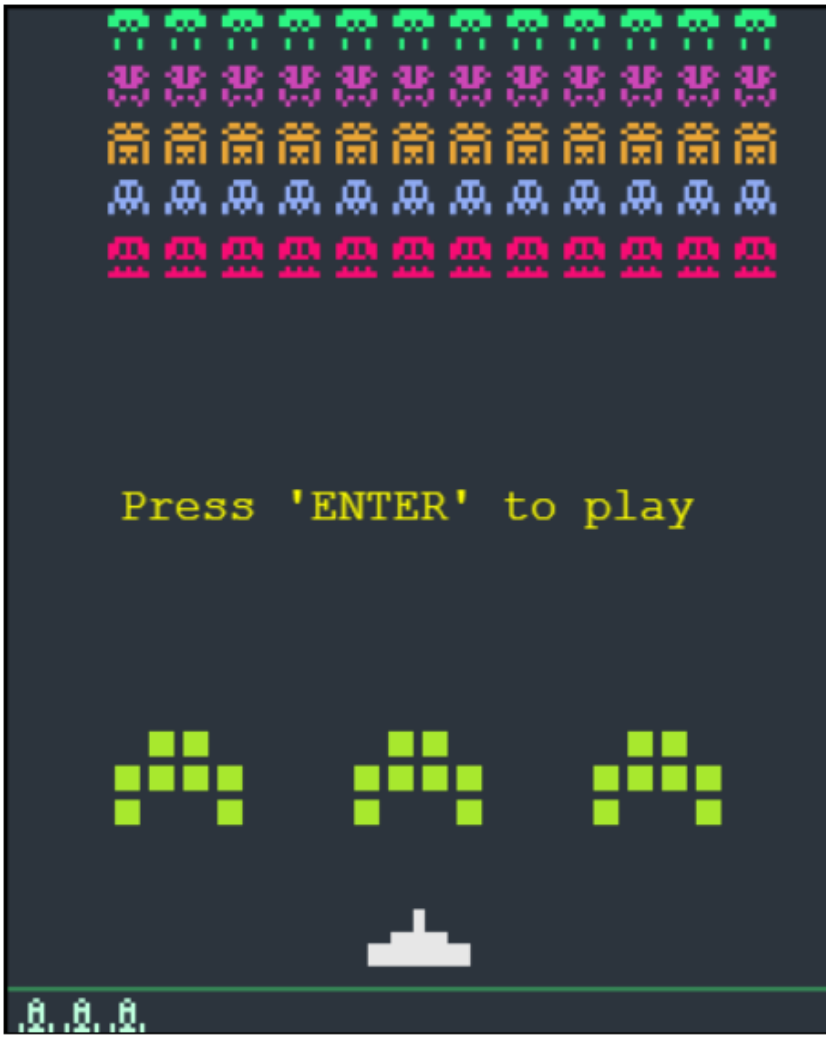
Computer Vision Project

Sacha Fela et Ève Bousquet

April 2025



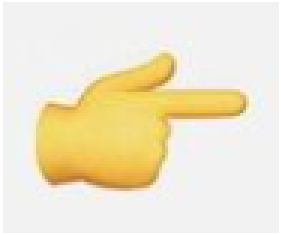
Introduction

L'objectif de ce projet est de faire fonctionner le jeu *Space Invaders* sans recourir à des commandes clavier. Pour cela, des mouvements effectués devant la caméra seront interprétés et associés à des commandes du jeu. Nous présentons deux approches pour réaliser ce projet. La première repose sur l'utilisation de la librairie *MediaPipe*, tandis que la seconde exploite une base de données provenant de Kaggle contenant des images de l'alphabet en langue des signes.

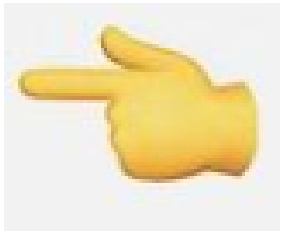
1 Première approche

1.1 Méthodologie

Le jeu *Space Invaders* dispose uniquement de trois commandes : aller à droite, aller à gauche et tirer.
Les mouvements effectués devant la caméra sont plus précisément des gestes de la main. Nous en avons donc choisi trois pour les associer chacun à une commande respective : un pouce vers la droite pour aller à droite, fig.(a) (c'est un index et non un pouce sur l'image mais il faut que ce soit un pouce !), le pouce vers la gauche pour aller à gauche, fig(b) (même remarque), ainsi qu'un rond formé par la main pour tirer (le signe ok de plongée), fig(c).



(a) Aller à droite



(b) Aller à gauche

(c) Tirer

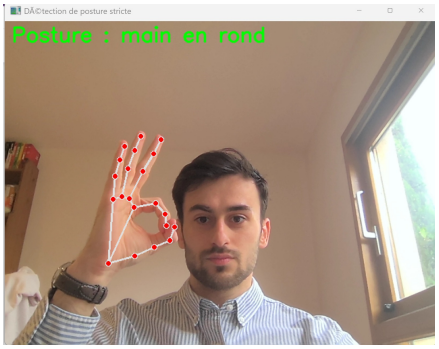
Figure 1: Les trois gestes de la mains

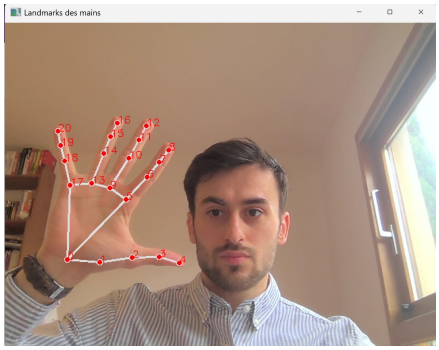
1.2 Modélisation

1.2.1 MediaPipe

Pour cette première approche, nous avons utilisé la bibliothèque open source *MediaPipe* de Google. Cette solution repose sur un pipeline capable de classifier les gestes en détectant plusieurs points de repère sur la main. L'entrée du système est une image capturée, qui subit ensuite un prétraitement : elle est convertie et redimensionnée si nécessaire, afin de faciliter l'analyse des pixels d'interêts.
La détection de la main commence par l'identification de sa position à l'aide d'une boîte englobante, générée par un réseau de neurones convolutifs (CNN). Une fois la main localisée, le système procède à l'estimation de sa pose. Pour cela, un modèle de suivi est utilisé, capable de prédire des points clés correspondant aux articulations de la main (phalanges, poignet, etc.). Cette étape est illustrée dans les figures ci-dessous : 21 points clés sont générés et constituent les caractéristiques d'entrée du modèle de classification.

Par ailleurs, MediaPipe propose des modèles préentraînés, ce qui nous a permis de l'intégrer et de l'utiliser rapidement dans notre système.





1.2.2 Le code

Le code fonctionne de la manière suivante :
Tout d'abord, *MediaPipe* est initialisé pour détecter la main dans l'image capturée par la webcam. Ensuite, des distances entre certains points clés de la main sont calculées pour déterminer si tous les doigts sont repliés, à l'exception du pouce.
Si c'est le cas, la direction du pouce (gauche ou droite) est analysée afin d'interpréter un mouvement. Un autre geste est reconnu lorsque le pouce et l'index sont proches l'un de l'autre, ce qui déclenche une commande de "tir".
En parallèle, le script établit une connexion WebSocket avec un jeu (par exemple, Space Invaders). Une fois la commande "enter" saisie par l'utilisateur, la partie démarre.
La caméra est ensuite activée avec "cv2.VideoCapture(0)", et chaque image est

analysée en temps réel. Si un geste correspondant à une commande est reconnu, celle-ci est envoyée au jeu via "await websocket.send(command)".
Il est important de noter que *Mediapipe* n'est pas compatible avec les versions ulterieurs de python 3.9. Il faut donc envisager de se créer un environnement virtuel afin de pouvoir telecharger tous les packages néscessaires. Un récapitulatif du setup nescessaire est disponible sur le repo sur le fichier *environment.yml*.

2 Seconde approche

2.1 Méthodologie

Le second modèle est basé sur une architecture type *YOLO*. Contrairement à *mediapipe*, qui présentait l'avantage d'être un modèle déjà au point, ici il faut entraîner notre modèle sur un jeu de données adéquat. Afin d'avoir un jeu de donnée de taille et qualité satisfaisante on choisi d'utiliser un dataset préexistant disponible librement sur *Kaggle*, ici. Le data set est consitué de l'alphabet de la langue des signes américaine. Nous utilisons alors la lettre A pour aller à droite, B pour aller à gauche et C pour tirer.

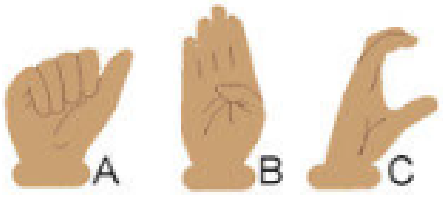


Figure 3: Lettres A, B et C de la langue des signes

Pour éviter les faux positifs/négatifs, on crée une quatrième catégorie regroupant toutes les postures qui ne peuvent pas être reconnus. Pour alimenter cette catégorie on conserve les dataset associées au lettres *D,E,F*, un dataset *Nothing* (simplement des arrieres plan vides), et deux autres dataset regroupant des postures non conventionnelles.

2.2 Structure du modèle

Le modèle utilisé est YOLOv8, une architecture de détection d'objets développée par Ultralytics (dont il faut la bibliothèque pour faire tourner l'algorithme).YOLOv8 est une version moderne de la famille des modèles YOLO, conçue pour repérer et reconnaître des objets dans une image en une seule étape rapide. Contrairement aux versions précédentes, il n'utilise pas de "boîtes prédéfinies" (ancres), mais apprend directement à prédire où se trouvent les objets. L'image est analysée à plusieurs niveaux de détail (analyse multi-échelle), ce qui permet de détecter aussi bien des petits objets que des grands. Enfin, à partir de cette analyse,

le modèle prédit à la fois l'emplacement, la catégorie, et la certitude de chaque objet détecté. Le modèle a été entraîné pendant 50 epochs, avec une taille de batch dépendant de la mémoire GPU disponible. Le fichier *args.yaml* décrit les chemins du dataset et les noms des classes. En sortie, le modèle best.pt permet de détecter les gestes en temps réel à partir de la webcam, et les prédictions sont traduites en action sur le jeu grâce à WebSocket. Les résultats de l'entraînement sont disponibles sur le repository dans le dossier *mon_dossier*.

Résultats et conclusion

Après test, il est évident que le modèle base sur *mediapipe* est supérieur en tout point de vue. Son approché systématique d'isoler et décomposer les articulations de la main le rend extrêmement robuste et adapté quel que soit la situation (différence d'éclairage, de morphologie, d'image de fond). De plus son implémentation est facile car toute la partie entraînement à déjà été fait. Il a même été possible de gagner une partie du jeu avec notre code.
De l'autre coté, le second modèle présente plusieurs défauts : il faut mettre en place l'entraînement ce qui est coûteux en temps et en puissance de calcul, et il est très sensible au dataset utilisé. Dans notre cas nous avons préféré utiliser un dataset générique qui avait le bon goût d'être de grande taille. Il en résulte que notre modèle ne reconnaît pas les mains si quelqu'un se trouve sur l'image : il faut que la caméra capture uniquement l'avant bras et la main, en étant le plus centré possible. Cela est du à la structure du dataset, dont les images présentent uniquement des photos d'avant bras+main. Il sera potentiellement sensible au variations de luminosité, et sera moins résilient de manière générale. Cela met en évidence l'importance d'avoir un jeu de données qui met en jeu la variable d'intérêt dans un maximum de situations différentes.


You Lost!
Press 'ENTER' to play