

P24AB11 Mirroir Pepper

Note d'application :

Conversion des coordonnées 2D en 3D et
mappage des coordonnées



Développeur : Shengnian YE

Tuteur Académique et client : M. Sébastien Lengagne

Année-académique : 2024 - 2025

Introduction

Dans le cadre de notre projet de dernière année à Polytech Clermont, nous avons réalisé un projet intitulé "Miroir Pepper", dont le principal objectif était d'utiliser la technique de cinématique inverse pour contrôler en temps réel le bras robotique Braccio. Comme la **Figure1**, l'un des principaux défis du projet consistait à **convertir des coordonnées 2D en coordonnées 3D et à mapper ces coordonnées 3D avec précision dans le système de coordonnées de Unity**. Cette fonctionnalité était essentielle pour les calculs de cinématique inverse, influençant directement la réussite du projet.

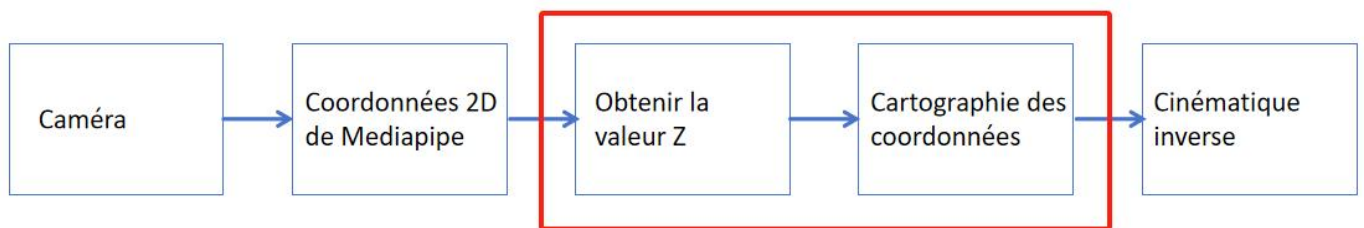


Figure 1 organigramme

1. Conversion des coordonnées 2D en coordonnées 3D

MediaPipe est un outil efficace et prêt à l'emploi pour la reconnaissance des postures humaines, capable de renvoyer les coordonnées tridimensionnelles (x, y, z) des points clés du corps humain. Parmi celles-ci, la coordonnée z représente la profondeur du point clé par rapport à la caméra. Cependant, au cours des expérimentations, il a été constaté que la valeur z retournée par MediaPipe n'est pas une profondeur réelle, mais une valeur normalisée, et donc non directement mesurée. Afin d'améliorer la précision de la conversion de la profondeur, j'ai adopté une méthode basée sur des mesures réelles et un ajustement des données pour convertir la valeur z retournée en une distance réelle.

Comme illustré dans la **Figure 2**, j'ai effectué les ajustements suivants pour atteindre l'objectif :

1. J'ai uniquement récupéré les coordonnées bidimensionnelles (x, y) fournies par MediaPipe.
2. Les coordonnées (x, y) ont été dénormalisées pour les convertir en coordonnées de pixels réels.
3. J'ai construit une expression de régression polynomiale, basée sur la distance en pixels entre les points 5 et 17, afin de calculer la profondeur réelle (z) du poignet.

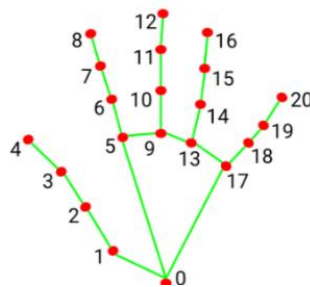


Figure 2 Modèle de main Mediapipe

Lorsque la résolution de l'image est de **(640, 480)**, les coordonnées (x, y) obtenues depuis MediaPipe sont normalisées. Pour les dénormaliser, nous les multiplions par la résolution correspondante **(640, 480)** afin de récupérer les coordonnées de pixels réelles.

À travers plusieurs expériences de mesure, nous avons établi une base de données, comme illustré dans la **Figure 3**, où x représente la distance en pixels entre les points clés 5 et 17, et y correspond à la distance réelle entre le poignet et la caméra (mesurée précisément à l'aide d'une règle).

Étant donné que la taille des mains varie d'une personne à l'autre, la valeur de x peut fluctuer légèrement. C'est précisément pour cette raison que nous avons adopté une régression

polynomiale, qui permet une certaine marge d'erreur tout en améliorant les capacités prédictives des données.

```
x = [300, 245, 200, 170, 145, 130, 112, 103, 93, 87, 80, 75, 70, 67, 62, 59, 57]
y = [20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100]
```

Figure 3 Ensemble de données

Afin d'améliorer la précision des données, nous avons collecté autant de mesures que possible et utilisé ces données pour effectuer une régression polynomiale. Comme illustré dans la **Figure 4**, nous avons tracé une courbe de régression à l'aide de Python. L'objectif n'était pas que tous les points se trouvent exactement sur la courbe, mais que le maximum de points mesurés s'en approche. Grâce à cette méthode, nous avons obtenu une fonction polynomiale idéale, capable de calculer avec précision la distance réelle entre le poignet et la caméra en entrant la distance en pixels entre les points 5 et 17.

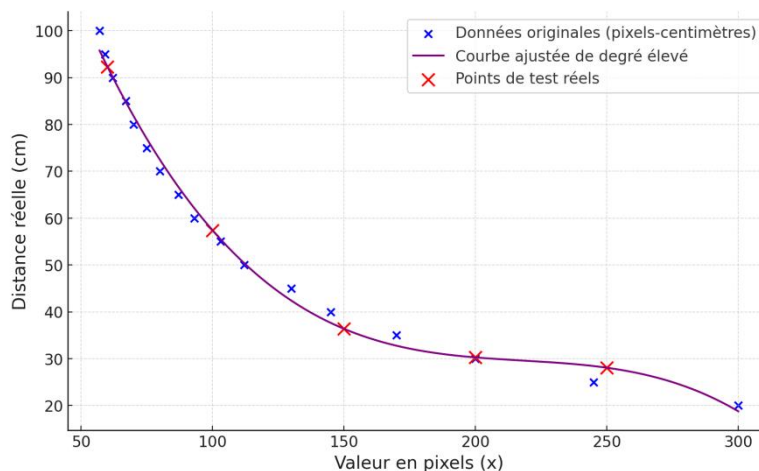


Figure 4 Courbe d'ajustement polynomiale

Au final, nous pouvons obtenir en sortie les coordonnées en pixels des points 0 à 20 ainsi que la valeur actuelle de **Z** (c'est-à-dire la distance réelle entre le poignet et la caméra).

Comme la **Figure 5**, par exemple, les résultats affichés dans l'image sont les suivants :

- Les coordonnées en pixels du point 0 sont **(212, 277)**.
- Les coordonnées en pixels du point 1 sont **(280, 254)**.
- Les coordonnées en pixels du point 2 sont **(333, 198)**.
- La valeur actuelle de **Z** est **47.532 cm**.

Ainsi, nous pouvons constater que les résultats en sortie sont tous dénormalisés. De cette manière, nous avons réussi à convertir des coordonnées 2D en coordonnées 3D tout en obtenant la valeur de **Z** (information de profondeur). Cela améliore non seulement la capacité de représentation

spatiale des coordonnées, mais fournit également des données de profondeur plus précises pour les opérations ultérieures.

```
les donnees: b'[(0, 212, 277), (1, 280, 254), (2, 333, 198),  
10, 256, 15), (11, 268, -28), (12, 282, -61), (13, 201, 86),  
, -26), 47.53290786920999]'
```

Figure 5 Quelques captures d'écran des résultats des tests

2. Cartographie des coordonnées

Dans le traitement d'image et l'interaction avec des environnements virtuels, le mappage des coordonnées est une étape clé. Ce projet utilise la détection de la main pour obtenir des coordonnées 2D dans une image, puis les combine avec des informations de profondeur pour calculer des coordonnées 3D. Afin de reproduire avec précision les mouvements de la main dans le monde virtuel Unity, il est nécessaire de mapper les coordonnées de l'image vers le système de coordonnées de Unity.

Le système de coordonnées de l'image a une résolution de **640 × 480 pixels**, comme illustré dans la **Figure 6**. Son origine est située dans le coin supérieur gauche, avec l'axe **X** s'étendant vers la droite et l'axe **Y** vers le bas. La valeur de l'axe **Z** est calculée à l'aide d'une régression polynomiale (voir la première section précédente). Dans le système de coordonnées de la scène Unity, l'origine est située au centre de la scène, avec l'axe **X** s'étendant vers la droite, l'axe **Y** vers le haut et l'axe **Z** vers l'avant (Unity utilise une convention en main droite).

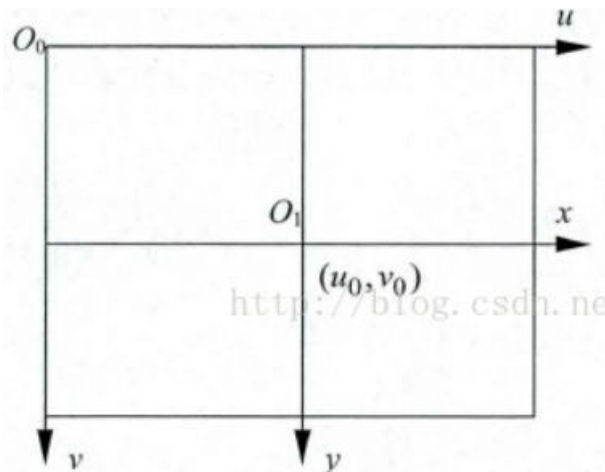


Figure 6 Diagramme de transformation des coordonnées

Nous avons constaté que les directions des systèmes de coordonnées de l'image et de Unity diffèrent. Par conséquent, nous devons d'abord unifier les directions de ces deux systèmes de coordonnées. Ensuite, à travers plusieurs expérimentations, nous avons mesuré l'étendue des coordonnées en pixels sur l'axe **X** lorsque le poignet se déplace jusqu'aux bords gauche et droit de l'image vidéo. De la même manière, nous avons également mesuré les plages de mouvement du poignet dans les directions verticale (**Y**) et de profondeur (**Z**). Finalement, nous avons établi la correspondance entre les plages des coordonnées détectées par l'image et celles utilisées dans le système de coordonnées de Unity, comme suit :

Pour l'Axe X (coordonnées horizontales) :

La plage des coordonnées détectées dans l'image est [96, 666], qui est mappée à la plage [-0.19, 0.19] dans Unity.

Pour l'Axe Y (coordonnées verticales) :

La plage des coordonnées détectées dans l'image est [209, 0] (avec inversion de direction), qui est mappée à la plage [-0.26, 0.02] dans Unity.

Pour l'Axe Z (coordonnées de profondeur) :

La plage des coordonnées détectées dans l'image est [0.025, 0.075], qui est mappée à la plage [-0.18, 0.18] dans Unity.

La valeur de profondeur **Z** est calculée à l'aide d'une régression polynomiale et est liée à la distance en pixels entre les points 5 et 17 dans l'image.

Si les coordonnées dépassent ces plages, cela signifie que la caméra ne peut pas détecter la position du poignet. Nous avons maintenant clarifié les plages de mouvement des axes **X**, **Y** et **Z** dans le système de coordonnées de l'image ainsi que leurs plages correspondantes dans le système de coordonnées de Unity, comme illustré dans la **Formule 6**. Par conséquent, nous pouvons établir une relation linéaire pour mapper les coordonnées de l'image au système de coordonnées de Unity.

- **Value** : La valeur d'entrée des coordonnées d'origine (coordonnées **X**, **Y**, **Z** détectées dans l'image).
- **OldMin**, **OldMax** : Les valeurs minimale et maximale des coordonnées dans le système de l'image.
- **NewMin**, **NewMax** : Les valeurs minimale et maximale des coordonnées dans le système de Unity.

$$\text{MappedValue} = \text{NewMin} + \frac{(\text{Value} - \text{OldMin}) \times (\text{NewMax} - \text{NewMin})}{\text{OldMax} - \text{OldMin}}$$

Grâce à ce mappage des coordonnées, lorsque nous obtenons les coordonnées du poignet dans le système de coordonnées de l'image, nous pouvons les convertir dans le système de coordonnées de Unity, permettant ainsi d'effectuer des calculs de cinématique inverse.

Conclusion et Perspectives

Ce défi technique nous a permis d'améliorer considérablement nos compétences dans les domaines du traitement d'image, du mappage des coordonnées et des calculs de cinématique inverse. Il nous a également permis de comprendre l'importance de la pensée innovante et de la persévérance dans la résolution de problèmes techniques complexes. Dans ce projet, le principal défi consistait à réaliser la conversion des coordonnées 2D en coordonnées 3D et à les mapper avec précision dans le système de coordonnées Unity. Bien que la réalisation de cette fonctionnalité ait présenté de nombreux obstacles techniques, nous avons finalement proposé une solution efficace grâce à de nombreuses expérimentations et tentatives.

À l'avenir, nous pensons qu'il sera possible d'optimiser davantage la précision du mappage des coordonnées et des calculs de profondeur, par exemple en intégrant des algorithmes d'apprentissage automatique plus avancés pour améliorer l'efficacité de la prédiction de la profondeur et de la conversion des coordonnées. Ces améliorations offriront un potentiel d'application encore plus prometteur.

Annexes

Les données de Figure 3

Valeur en pixels (X)	Distance réelle (cm)
300	20
245	25
200	30
170	35
145	40
130	45
112	50
103	55
93	60
87	65
80	70
75	75
70	80
67	85
62	90
59	95
57	100