

Rapport du projet GE5A

Miroir Pepper

P24AB04

Développeurs	:	Shengnian Ye Fatima Elkhadiri
Tuteur Académique et client	:	M. Sébastien Lengagne
Tuteur Industriel	:	M. François Kersulec
Professeure de communication	:	Mme. Myriam Doghmi
Responsable du projet GE5A	:	M. Jacques Laffont

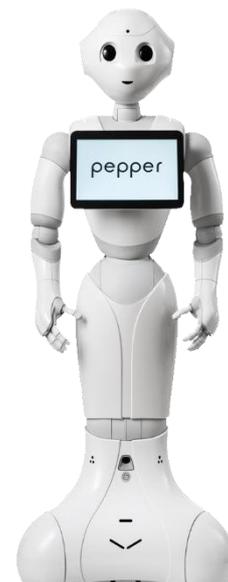
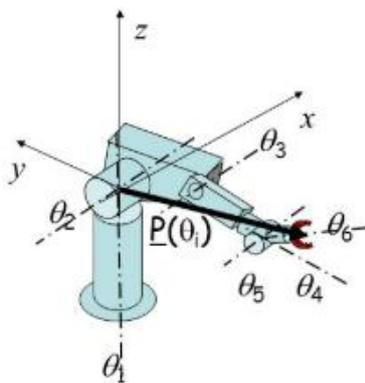


Table des figures

Figure 1 - Logo Fete des sciences . Source : https://www.fetedelascience.fr/	2
Figure 2 - Illustration du projet Miroir Pepper. Source : Auteurs	2
Figure 3 - Logo mediapipe : Source : https://www.youtube.com/c/MediaPipe	5
Figure 4 - Logo ROS - Source : Wikipédia	5
Figure 5 - Logo Unity . Source : Wikipédia	6
Figure 6 - Articulations Pepper. Source : http://doc.aldebaran.com/2-5/family/pepper_technical/joints_pep.html	8
Figure 7 – Exemple de l’output de Mediapipe. Source : Auteur	10
Figure 8 – Fonctionnement général du système (Robot Pepper) . Source : Auteurs	11
Figure 9- Nœuds Ros du système Miroir Pepper. Source : Auteurs	12
Figure 10 - Braccio. Source : https://docs.rs-online.com/e477/0900766b814da22f.pdf	13
Figure 11 - Moteurs de Braccio Source : https://docs.rs-online.com/e477/0900766b814da22f.pdf ..	13
Figure 12 - Fonctionnement général du système miroir Braccio. Source : Auteurs.....	14
Figure 13 - Flux du calcul des angles à transmettre à Braccio . Source : Auteurs	14
Figure 14 - Paramètres de DH. Source : https://www.hackster.io/rpatterson/arduino-tinkerkit-braccio-robot-arm-kinematics-1a8303	15
Figure 15 - Description de l'effecteur terminal – Source : https://www.hackster.io/rpatterson/arduino-tinkerkit-braccio-robot-arm-kinematics-1a8303	16
Figure 16 - Illustration du système Miroir braccio. Source : https://github.com/ohlr/braccio_arduino_ros_rviz	16
Figure 17 - Braccio dans Unity . Source : Auteurs	17
Figure 18 - Imitation du mouvement par Pepper. Source : Auteurs	19
Figure 19 - Imitation de prise d'objet par Pepper. Source : Auteurs.....	19
Figure 20 - Robot pepper (joints) . Source : http://doc.aldebaran.com/2-5/family/pepper_technical/joints_pep.html	20
Figure 21 - partie bras du corps humain . source : https://fr.wikipedia.org/wiki/Anatomie_humaine	20
Figure 22- Les limites des angles de la tete de pepper. Source : http://doc.aldebaran.com/2-5/family/pepper_technical/joints_pep.html	21
Figure 23 - Limites des angles du bras gauche . Source : http://doc.aldebaran.com/2-5/family/pepper_technical/joints_pep.html	21
Figure 24 - Délai entre mouvements de l’humain et le mouvement de Pepper . Source : Auteurs	22
Figure 25 - robot Braccio . Source : https://docs.rs-online.com/e477/0900766b814da22f.pdf	23
Figure 26 - Limitations des angles de Braccio . Source : https://docs.rs-online.com/e477/0900766b814da22f.pdf	23
Figure 27 - Simulation de Braccio sur Unity. Source : Auteurs	24
Figure 28 - Comparaison des angles mesurés manuellement avec celle calculés via Mediapipe	25
Figure 29 - Décalage entre mouvement de la main humaine et Braccio. Source : Auteurs	26

Table des Abréviations

Abréviation	Signification
DoF	Degrés de Liberté
IP	Protocole Internet
MGD	Modèle Géométrique Direct (Cinématique Directe)
MGI	Modèle Géométrique Inverse (Cinématique Inverse)
NAOqi	Système d'exploitation des robots SoftBank
PC	Ordinateur Personnel
PWM	Modulation de Largeur d'Impulsion
ROS	Robot Operating System
RViz	ROS Visualization Tool
SDK	Software Development Kit
WBS	Work Breakdown Structure

Glossaire

- **Arduino** : Plateforme open source de prototypage électronique permettant de contrôler des composants tels que des moteurs ou des capteurs.
- **Braccio** : Bras robotique développé par Arduino, conçu pour des usages éducatifs et expérimentaux, doté de 6 degrés de liberté.
- **Cahier des charges** : Document décrivant les besoins, les fonctionnalités attendues et les contraintes d'un projet.
- **Cinématique directe et inverse** : Branche de la robotique qui concerne les calculs permettant de trouver la position de l'effecteur terminal (cinématique directe) ou les angles nécessaires pour atteindre une position donnée (cinématique inverse).
- **DoF (Degrés de Liberté)** : Nombre de mouvements indépendants qu'un système mécanique ou un robot peut réaliser.
- **Filtre de Kalman** : Algorithme utilisé pour estimer des variables d'état à partir de données bruitées, appliqué ici pour lisser les angles calculés.
- **Landmarks** : Points clés détectés sur le corps humain, utilisés pour modéliser les articulations ou les segments dans la vision par ordinateur.
- **Latence** : Délai entre l'entrée d'une commande et son exécution par le robot, qui peut affecter la fluidité des mouvements.
- **MediaPipe** : Bibliothèque open source développée par Google pour le traitement et l'analyse en temps réel des flux multimédias, notamment pour la détection de poses et la reconnaissance de gestes.
- **MGD (Modèle Géométrique Direct)** : Calcul des coordonnées de l'extrémité d'un robot en fonction des angles actuels de ses articulations.
- **MGI (Modèle Géométrique Inverse)** : Ensemble de calculs permettant de déterminer les angles des articulations d'un robot à partir de la position souhaitée de son extrémité (effecteur terminal).

- **NAOqi** : Système d'exploitation dédié aux robots SoftBank (comme Pepper), qui permet de les programmer et de les contrôler.
- **Pepper** : Robot humanoïde conçu par SoftBank Robotics, utilisé principalement pour les interactions homme-machine et les applications éducatives ou d'accueil.
- **Protocole de communication** : Ensemble de règles permettant l'échange de données entre différents systèmes (ex. : USB, Ethernet, ROS topics).
- **PWM (Modulation de Largeur d'Impulsion)** : Méthode de modulation utilisée pour contrôler les moteurs et les composants électroniques.
- **ROS (Robot Operating System)** : Framework open source permettant le développement de logiciels pour robots. Il facilite la communication entre différents composants robotiques via des nœuds et des topics.
- **RViz** : Outil graphique de ROS utilisé pour visualiser les données robotiques en 3D.
- **SDK (Software Development Kit)** : Ensemble d'outils logiciels permettant de développer des applications spécifiques pour une plateforme donnée.
- **Unity** : Moteur de jeu et plateforme de simulation permettant de créer des environnements virtuels pour tester et visualiser les algorithmes robotiques.
- **Vision par ordinateur** : Domaine de l'intelligence artificielle qui permet aux ordinateurs de comprendre et d'interpréter des images ou des vidéos pour en extraire des informations.

Résumé

L'objectif de ce projet est d'utiliser la technologie de vision par ordinateur et les principes de la cinématique pour réaliser une imitation en temps réel des mouvements humains par le robot Pepper et le bras robotique Braccio, et pour compléter les tâches de saisie d'objets. Le projet est divisé en deux parties : la première partie utilise MediaPipe pour extraire les coordonnées 3D du mouvement humain, calcule les angles des articulations et transmet les données d'angle au robot Pepper via le nœud ROS pour contrôler le robot Pepper afin d'imiter le mouvement humain dans en temps réel, et réalisez enfin que Pepper saisit l'objet cible. Fonction. La deuxième partie étudie la méthode de contrôle basée sur Mediapipe et la cinématique directe et inverse. En prenant les coordonnées du poignet humain comme référence, le jumelage des données est réalisé via Unity et combiné avec ROS pour contrôler le bras robotique Braccio afin de compléter sa fonction de saisie précise de la cible objet. Ce projet fournit une solution efficace pour l'imitation et la manipulation en temps réel de robots basée sur la vision et la cinématique.

Mots clés : Cinématique directe et inverse, MediaPipe, ROS, vision par ordinateur, reconnaissance de posture, tâches de préhension

Abstract

The goal of this project is to use computer vision technology and kinematics principles to realize real-time imitation of human movements by Pepper robot and Braccio robotic arm, and to complete the tasks of grasping objects. The project is divided into two parts: the first part uses MediaPipe to extract 3D coordinates of human movement, calculates joint angles, and transmits the angle data to Pepper robot through ROS node to control Pepper robot to imitate human movement in real time, and finally realize Pepper grasping the target object. Function. The second part studies the control method based on Mediapipe and forward and inverse kinematics. Taking the coordinates of human wrist as a reference, data matching is realized through Unity and combined with ROS to control Braccio robotic arm to complete its function of accurately grasping the target object. This project provides an effective solution for real-time imitation and manipulation of robots based on vision and kinematics.

Keywords : Forward and inverse kinematics, MediaPipe, ROS, computer vision, posture recognition, grasping tasks

Remerciements

Nous souhaitons exprimer notre profonde gratitude et notre respect à toutes les personnes qui nous ont soutenus et aidés dans ce laboratoire tout au long de ce projet de cinq mois. Nous avons bénéficié d'une attention constante et d'une aide précieuse de la part des professeurs, des étudiants de ce laboratoire, ainsi que de collaborateurs externes. Leur soutien nous a permis de progresser rapidement et de surmonter les défis rencontrés durant ce projet. Nous souhaitons profiter de cette occasion pour exprimer notre sincère reconnaissance.

Tout d'abord, nous sommes très reconnaissants à **M. Sébastien Lengagne**, en tant que client et tuteur, pour ses précieux conseils et son accompagnement tout au long du projet. Il nous a guidés face aux difficultés rencontrées et a constamment proposé diverses solutions. Il a également été d'une aide essentielle pour résoudre les problèmes techniques rencontrés avec le bras, travaillant avec nous pour en identifier les causes.

Nous tenons également à remercier **M. Laffont Jacques**, notre tuteur école, pour ses conseils avisés et son soutien constant. Son aide a été d'une valeur inestimable pour notre progression et pour l'élaboration de notre parcours professionnel.

Nous sommes également très reconnaissants à **M. Kersulec François**, qui nous a enseigné les bonnes pratiques pour rédiger un rapport de projet dans une optique professionnelle, et à **Mme Doghmi Myriam**, dont les conseils et l'aide dans la rédaction du rapport ont été d'une grande utilité.

Enfin, nous tenons à exprimer notre reconnaissance aux étudiants sous-traitants de GE4A, et notamment à **Ikram Soundouss Fissal** et **Johan Petit**, dont l'aide et le soutien ont considérablement contribué à la réussite de ce projet.

En conclusion, nous remercions chaleureusement toutes les personnes qui nous ont aidés tout au long de ce projet. Leur apport nous a permis d'approfondir notre expertise et de progresser dans notre carrière professionnelle.

Sommaire

Résumé	
Abstract	
I. Introduction.....	1
II. Contexte	2
III. Problématique du sujet.....	2
IV. Gestion du projet.....	2
V. Travail réalisé.....	5
A. Outils utilisés	5
B. Robot Pepper.....	7
C. Robot Braccio	13
VI. Résultats.....	18
A. Etat d'avancement.....	18
B. Analyse des résultats.....	19
C. Propositions d'amélioration	26
VII. Compétences acquises	28
VIII. Conclusion générale.....	29
IX. Bibliographie et webographie	
X. Annexes.....	
A. WBS.....	
B. Cahier de charges	
C. Gantt.....	
D. Etat d'avancement.....	
E. Etapes de calcul géométrique.....	
F. MGD et MGI pour Braccio.....	

I. Introduction

Ces dernières années, avec le développement rapide de l'intelligence artificielle et de la robotique, l'utilisation des robots de service dans le domaine de l'interaction homme-machine s'est considérablement élargie. Cependant, permettre à un robot d'imiter les mouvements humains reste un défi technique important. L'objectif principal de ce projet est de développer un système basé sur **MediaPipe** et le contrôle des mouvements des robots, permettant à ces derniers de reproduire les mouvements humains et d'exécuter des tâches simples de saisie d'objets.

Le principal problème de ce projet réside dans la manière d'utiliser **MediaPipe** et la cinématique directe et inverse au sein d'un environnement ROS pour permettre à **Pepper** et au bras robotique **Braccio** d'imiter en temps réel les mouvements humains et de saisir des objets simples. Ce projet pose également de nombreux défis, notamment la manière de mapper avec précision les informations de mouvement humain (telles que les variations d'angles des épaules, des coudes et des poignets) sur le système de mouvement des robots Pepper et Braccio, tout en surmontant les limitations des plages de mouvement articulaires et en garantissant la gestion des données en temps réel.

Ce projet s'appuie sur le soutien technique de **Polytech Clermont-Ferrand**. Pour répondre aux problématiques mentionnées ci-dessus, nous avons structuré notre travail en deux parties :

- Dans la première, nous utilisons **la reconnaissance d'image avec MediaPipe** pour obtenir les coordonnées tridimensionnelles des mouvements humains, calculer les angles articulaires, puis transmettre ces données via des nœuds ROS afin de permettre à **Pepper** de reproduire en temps réel les mouvements humains.
- Dans la deuxième partie, nous appliquons **la reconnaissance d'image avec Mediapipe** et des algorithmes de géométrie directe et inverse pour planifier et optimiser les trajectoires du bras robotique **Braccio**, lui permettant ainsi de saisir avec précision les objets situés devant lui.

Ce Rapport est structuré comme suit :

D'abord, il introduit le projet en expliquant son contexte global et son importance, avant d'approfondir les enjeux liés au sujet et d'exposer la problématique centrale, notamment les défis techniques liés à la commande et au contrôle des robots.

Ensuite, il aborde la gestion du projet, illustrée par un diagramme de Gantt et une structure WBS (Work Breakdown Structure), permettant une organisation claire et efficace des tâches.

Le rapport décrit ensuite le travail réalisé, en commençant par les outils utilisés pour le développement du projet. Les sections suivantes détaillent les aspects techniques et expérimentaux liés au robot Pepper et au robot Braccio, mettant en lumière les méthodologies employées, notamment l'utilisation de MediaPipe et des algorithmes de cinématique directe et inverse.

Par la suite, une analyse approfondie des résultats obtenus est proposée, accompagnée de pistes d'amélioration pour surmonter les limitations identifiées, telles que les imprécisions dans les mouvements des robots.

Enfin, le rapport met en avant les compétences et connaissances acquises au cours de ce projet, qu'elles soient techniques, méthodologiques ou liées à la gestion de projet, et conclut en récapitulant les principaux apports de cette étude tout en ouvrant la voie à des perspectives futures.

II. Contexte

Le projet **Miroir Pepper** a été initié dans le cadre d'une collaboration avec **M. Sébastien Lengagne**, un des professeurs du génie électrique, chercheur et client principal du projet. Il s'inscrit dans une double ambition :

1. **Offrir une vitrine technologique** pour des événements tels que les portes ouvertes, la Fête de la Science, ou d'autres manifestations publiques.
2. **Créer une base de travail évolutive** pour poursuivre les recherches en perception et interaction robotique, conformément aux objectifs définis par notre client **M. Sébastien Lengagne**.

**fête de
la Science**

Figure 1 - Logo Fete des sciences .

Source :

<https://www.fetedelascience.fr/>

III. Problématique du sujet

Le projet **Miroir Pepper** vise à permettre au robot humanoïde Pepper d'imiter en temps réel les mouvements humains capturés par MediaPipe. Les principaux défis consistent à convertir les données tridimensionnelles des articulations humaines en commandes robotiques adaptées aux contraintes mécaniques et aux plages de mouvements de Pepper. Cela nécessite l'utilisation de la cinématique directe et inverse pour assurer une correspondance précise entre les mouvements humains et ceux du robot, tout en garantissant une gestion fluide des données en temps réel via ROS.



Figure 2 - Illustration du projet Miroir Pepper. Source : Auteurs

En parallèle, le bras robotique **Braccio** doit saisir des objets simples en utilisant les coordonnées obtenues par MediaPipe. Les problématiques incluent la planification des trajectoires à l'aide d'algorithmes de cinématique, l'adaptation aux limitations mécaniques du bras, et la synchronisation entre la perception visuelle et les actions du robot. L'objectif est d'assurer une manipulation précise et efficace, malgré les contraintes liées aux variations d'objets et aux collisions potentielles.

IV. Gestion du projet

a) Organisation et chronologie avec Gantt

Le diagramme de Gantt présenté illustre le planning détaillé du projet, qui se divise en deux parties principales : **Mirror Pepper** et **Mirror Braccio**. Ces deux volets sont organisés autour de jalons clés, permettant une planification et un suivi rigoureux. Chaque partie aborde des objectifs spécifiques, de la recherche initiale à la soutenance finale. Voici les jalons principaux organisés par volet :

Partie 1 : Mirror Pepper

1. Recherche initiale et configuration (13/09/2024 - 07/10/2024)

- Étape préparatoire pour recueillir les informations nécessaires et configurer l'environnement de travail.

2. Développement de l'algorithme de reconnaissance d'images (08/10/2024 - 16/11/2024)

- **08/10/2024 - 14/10/2024** : Test de l'algorithme OpenPose.
- **15/10/2024 - 25/10/2024** : Intégration de MediaPipe pour la reconnaissance des poses.
- **26/10/2024 - 14/11/2024** : Calcul des angles articulaires et optimisation des données.
- **Jalon clé : Validation initiale de l'algorithme (16/11/2024).**

3. Validation et communication ROS (15/12/2024 - 10/01/2025)

- Développement du code ROS et intégration avec Pepper.
- **Jalon clé : Démonstration finale pour Mirror Pepper (10/01/2025).**

Partie 2 : Mirror Braccio

1. Développement de l'algorithme pour la reconnaissance des bras et doigts (07/11/2024 - 13/01/2025)

- **07/11/2024 - 13/12/2024** : Reconnaissance des mouvements des bras et doigts.
- **14/12/2024 - 13/01/2025** : Application de la géométrie inverse pour synchroniser les mouvements.
- **Jalon clé : Validation de l'algorithme bras/doigts (13/01/2025).**

2. Communication ROS + Arduino (15/12/2024 - 20/01/2025)

- Développement et débogage du système
- **Jalon clé : Finalisation et revue du système (17/01/2025).**

3. Soutenance finale (20/01/2025)

- Présentation des résultats pour les deux volets du projet.

Le GANTT entier se trouve dans l'« Annexe C ».

b) Organisation des tâches

Le projet Ge5A est structuré autour de trois volets principaux : **Mirror Pepper**, **Mirror Braccio**, et **Documentation**. Ces volets sont interconnectés pour réaliser un système global permettant la reconnaissance et la reproduction des gestes humains par des robots, tout en assurant une documentation claire et pratique pour la reproduction et l'utilisation du système. Voici les grandes étapes et jalons de chaque volet :

1. Miroir Pepper

Ce volet se concentre sur la reconnaissance des gestes et leur transmission au robot Pepper pour une commande en temps réel. Les jalons clés incluent :

- **État de l'art et apprentissage ROS** : Compréhension des concepts clés pour le contrôle robotique.
- **Adaptation des outils de reconnaissance des gestes au système ROS** : Intégration de technologies comme MediaPipe ou OpenPose.
- **Algorithme pour la reconnaissance des gestes** : Développement d'algorithmes permettant une détection précise et en temps réel.
 - **Amélioration du filtrage de l'image.**
 - **Calcul et amélioration des angles articulaires en temps réel.**
- **Commande et test du robot Pepper** : Validation du système avec des tests pratiques.

2. Miroir Braccio

Ce volet cible la commande du bras robotisé Braccio pour la reproduction des gestes manuels et des mouvements des doigts. Les jalons importants comprennent :

- **État de l'art et apprentissage** : Exploration des concepts liés au contrôle précis des bras robotisés.
- **Algorithme pour la reconnaissance des gestes des mains et doigts** : Conception d'un système capable de détecter et de reproduire les gestes.
 - **Calcul de la distance optimale pour la reconnaissance d'images.**
- **Simulation de Braccio avec Unity** : Simuler les mouvements dans un environnement virtuel.
- **Commande et test du robot Braccio** : Validation pratique et ajustement des paramètres.

3. Documentation

La documentation accompagne le projet pour assurer sa reproductibilité et faciliter son utilisation. Les étapes incluent :

- **Rédaction du rapport** : Synthèse des résultats et des démarches suivies.
- **Tutoriel et mode d'emploi** : Documentation détaillée des outils utilisés, comme ROS, et guide d'installation.

Un WBS est illustré dans l'annexe A

V. Travail réalisé

A. Outils utilisés

1. Mediapipe

Mediapipe est une bibliothèque open source développée par **Google**, qui fournit des solutions prêtes à l'emploi pour le traitement et l'analyse en temps réel des flux multimédias, notamment des vidéos et des images. Elle est largement utilisée pour des applications impliquant la vision par ordinateur, comme le suivi de poses humaines, la reconnaissance de gestes ou la segmentation d'images.



MediaPipe

Figure 3 - Logo mediapipe : Source : <https://www.youtube.com/c/MediaPipe>

Pourquoi Utiliser Mediapipe ?

- **Facilité de Développement** : Vous n'avez pas besoin d'entraîner vos propres modèles de machine learning.
- **Efficacité** : Les algorithmes sont optimisés pour fonctionner rapidement sur les appareils à faible puissance.
- **Polyvalence** : Utilisable pour diverses applications comme la robotique, la vision augmentée, et les interfaces gestuelles.

2. ROS (Robot Operating System)

Le Robot Operating System (ROS) est un framework open source conçu pour le développement de logiciels de robotique. Il fournit un ensemble d'outils, de bibliothèques et d'API permettant aux développeurs de concevoir, programmer et contrôler des robots de manière efficace et modulaire.



Figure 4 - Logo ROS - Source : Wikipédia

Caractéristiques Principales

Architecture Modulaire et Distribuée

ROS repose sur une architecture distribuée, où les différentes fonctionnalités du système robotique sont divisées en **nœuds**. Ces nœuds communiquent entre eux via un système de messagerie basé sur des **topics** (communication asynchrone), des **services** (communication synchrone) et des **actions** (tâches longues avec suivi d'état). Cette approche permet une grande flexibilité dans la conception des systèmes complexes.

Multiplateforme

ROS est principalement compatible avec les distributions Linux, en particulier **Ubuntu**, mais il prend également en charge Windows et macOS. Il peut être utilisé sur des systèmes embarqués, rendant son application possible dans des environnements à ressources limitées.

Écosystème Étendu

ROS bénéficie d'une large communauté de développeurs et d'utilisateurs qui contribuent à un

écosystème riche en packages et en outils préconçus. Ces derniers couvrent des domaines variés tels que la navigation autonome, la vision par ordinateur, la manipulation robotique et la planification des mouvements.

Simulation et Visualisation

ROS intègre des outils comme **Gazebo** (simulateur physique) et **RViz** (visualisation des données en 3D), permettant aux utilisateurs de tester et de valider leurs algorithmes avant leur mise en œuvre sur un robot physique.

Composants Clés

- **Nœuds**
Les nœuds sont des programmes ou processus indépendants responsables de tâches spécifiques, comme la lecture de capteurs ou le contrôle des moteurs.
- **Topics**
Les messages échangés entre nœuds sont publiés sur des topics. Par exemple, un nœud peut publier les données d'un capteur sur un topic auquel d'autres nœuds s'abonnent pour les traiter.
- **Services**
Les services permettent des interactions bidirectionnelles entre les nœuds, où une requête envoie une action spécifique et reçoit une réponse.
- **Actions**
Les actions sont utilisées pour des tâches complexes et longues (par exemple, déplacer un bras robotique) tout en offrant un retour d'état pendant l'exécution.

Versions et Évolutions

ROS 1

La première génération de ROS, largement utilisée, offre une architecture robuste mais dépendante d'un maître central (ROS Master).

- Exemples de distributions : ROS Kinetic, Melodic, Noetic.

ROS 2

Une version améliorée avec des fonctionnalités avancées, telles que le support natif en temps réel, la communication sécurisée et une meilleure compatibilité avec les systèmes embarqués.

- Exemples de distributions : ROS Foxy, Galactic, Humble.

Dans ce projet c'est la version ROS1, et surtout la distribution Melodic a été utilisé .

3. Unity

Unity est un moteur de jeu (game engine) doté d'outils puissants pour développer des environnements en 3D ou 2D. Bien qu'il soit principalement utilisé pour le développement de jeux vidéo, Unity est



Figure 5 - Logo Unity . Source : Wikipédia

également largement adopté dans des secteurs comme la robotique, la réalité augmentée/virtuelle, la formation, et la recherche.

Unity Simulator en Robotique et Technologie

Dans les projets technologiques et robotiques, un **Unity Simulator** est utilisé pour :

1. **Tester des Algorithmes :**
 - Les algorithmes de navigation autonome, de vision par ordinateur ou de contrôle des bras robotiques peuvent être testés dans des environnements simulés avant leur mise en œuvre physique.
2. **Créer des Environnements Réalistes :**
 - Unity permet de générer des environnements très détaillés et réalistes (par exemple, des terrains, des bâtiments, des obstacles) qui simulent des scénarios réels.
3. **Intégration avec des Frameworks comme ROS :**
 - Unity peut être intégré à **ROS (Robot Operating System)** pour tester les systèmes robotiques. Par exemple, les capteurs simulés dans Unity (caméras, LIDAR, etc.) peuvent envoyer des données à ROS, où les algorithmes robotiques les traitent.
4. **Simulation Avancée :**
 - Unity offre des outils pour simuler les trajectoires, les mouvements et l'interaction des robots avec leur environnement en 3D.

B. Robot Pepper

1. Présentation du Robot

Le **robot Pepper**, conçu par **SoftBank Robotics**, est un robot humanoïde interactif conçu principalement pour des applications dans les domaines de l'accueil, de l'éducation, des services, et de l'interaction homme-machine. Introduit en 2014, Pepper est reconnu pour sa capacité à détecter et interpréter les émotions humaines, ce qui le rend particulièrement adapté pour des interactions sociales.

Caractéristiques Principales

1. Design Physique

- **Dimensions :**
 - Hauteur : 1,20 m
 - Poids : 28 kg
- **Mobilité :**
 - Base avec 3 roues omnidirectionnelles permettant des déplacements fluides dans toutes les directions.
- **Articulations :**
 - 17 moteurs assurent la mobilité de la tête, des bras et de la base.

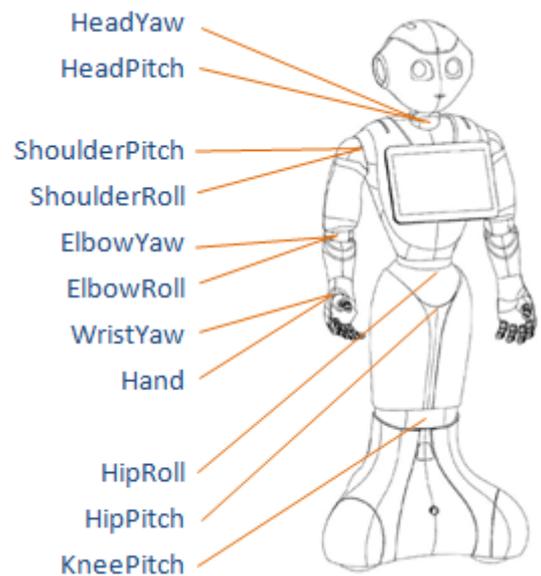


Figure 6 - Articulations Pepper. Source : http://doc.aldebaran.com/2-5/family/pepper_technical/joints_pep.html

2. Capacités d'Interaction

- **Reconnaissance des émotions :**
 - Capacité à analyser le ton de la voix, les expressions faciales et le langage corporel pour adapter son comportement.
- **Écran tactile :**
 - Une tablette de 10,1 pouces située sur la poitrine permet une interaction visuelle supplémentaire.

3. Capteurs

- **Capteurs visuels :**
 - Caméras HD (avant et arrière) pour la reconnaissance faciale et la détection d'objets.
 - Capteur 3D pour la perception de la profondeur et l'analyse de l'environnement.
- **Capteurs tactiles :**
 - Situés sur la tête, les mains et la base pour détecter les contacts humains.
- **Capteurs de mouvement :**
 - Gyroscope et accéléromètre pour la stabilité et le positionnement.
- **Capteurs sonores :**
 - 4 microphones directionnels pour localiser et analyser les sources sonores.

4. Connectivité

- **Wi-Fi et Ethernet :**
 - Permettent au robot de se connecter à des réseaux pour accéder à des bases de données, des applications cloud, ou interagir avec d'autres dispositifs.

5. Logiciel

- **Système d'exploitation :**
 - Basé sur **NAOqi OS**, un framework spécialement conçu pour les robots SoftBank.
- **Programmabilité :**

- Programmation en Python ou en C++ via le SDK NAOqi.
- Compatibilité avec ROS (Robot Operating System) pour des applications plus complexes.
- **IA et apprentissage :**
 - Intégration possible avec des solutions d'intelligence artificielle pour des tâches spécifiques.

6. Applications

- **Interactions sociales :**
 - Accueillir, guider, et divertir les visiteurs dans des lieux publics ou des entreprises.
- **Éducation :**
 - Utilisé comme outil pédagogique pour enseigner des concepts en robotique, en programmation, ou en intelligence artificielle.
- **Analyse des données :**
 - Collecte et analyse des données comportementales des utilisateurs pour des études ou des optimisations de service.

7. Personnalisation

- **Langues :**
 - Supporte plusieurs langues, ce qui le rend utilisable dans divers contextes culturels.
- **Applications :**
 - Des applications spécifiques peuvent être développées ou téléchargées pour étendre ses fonctionnalités.

2. Objectifs réalisés

Pour cette partie, un système intégré sous **ROS** (Robot Operating System) a été conçu afin de répondre aux objectifs suivants :

1. Reconnaissance des mouvements des articulations humaines avec MediaPipe

- La bibliothèque **MediaPipe** a été utilisée pour détecter et reconnaître en temps réel les articulations clés du corps humain (épaules, coudes, poignets, etc.)
- Ces données sont obtenues à partir des flux vidéo capturés par une caméra, traitées pour extraire les coordonnées 3D des points clés des articulations.



Figure 7 – Exemple de l'output de Mediapipe. Source : Auteur

Commentaire : Cette image présente une sortie générée par le programme utilisant MediaPipe pour détecter et analyser les mouvements humains en temps réel. Les angles articulaires, tels que l'inclinaison de l'épaule, la rotation du poignet et l'ouverture de la main, sont calculés et affichés directement sur l'image.

Les points clés des articulations et des mains sont détectés sous forme de landmarks rouges, reliés par des lignes formant un squelette simplifié. Ces données sont utilisées pour modéliser les mouvements de l'utilisateur et peuvent être transmises au robot Pepper pour permettre une imitation en temps réel.

2. Mapping des mouvements humains par rapport à Pepper

- Les coordonnées détectées des articulations humaines ont été converties dans un référentiel compatible avec la cinématique du robot **Pepper**.
- Un **mapping géométrique** a été conçu pour adapter les positions humaines aux degrés de liberté disponibles sur Pepper tout en respectant ses limitations mécaniques.

3. Calcul des angles pour la transmission à Pepper

- Les coordonnées des articulations humaines ont été utilisées pour calculer les **angles inter-articulations** nécessaires à la reproduction des mouvements.
- Les angles obtenus ont été optimisés pour correspondre à la cinématique de Pepper, garantissant une imitation fidèle et naturelle des mouvements humains.

4. Contrôle du robot Pepper

- Les commandes générées à partir des angles calculés ont été transmises à Pepper via les interfaces de contrôle ROS.

- Un module de contrôle temps réel a été développé pour assurer que Pepper réagisse rapidement et précisément aux variations des mouvements humains.

5. Tests et garantie d'une faible latence

- Des tests ont été réalisés pour évaluer la **latence** entre les mouvements humains et leur reproduction par Pepper.
- Les optimisations ont permis de minimiser cette latence, garantissant une synchronisation fluide et une expérience utilisateur réaliste.

3. Fonctionnement du système

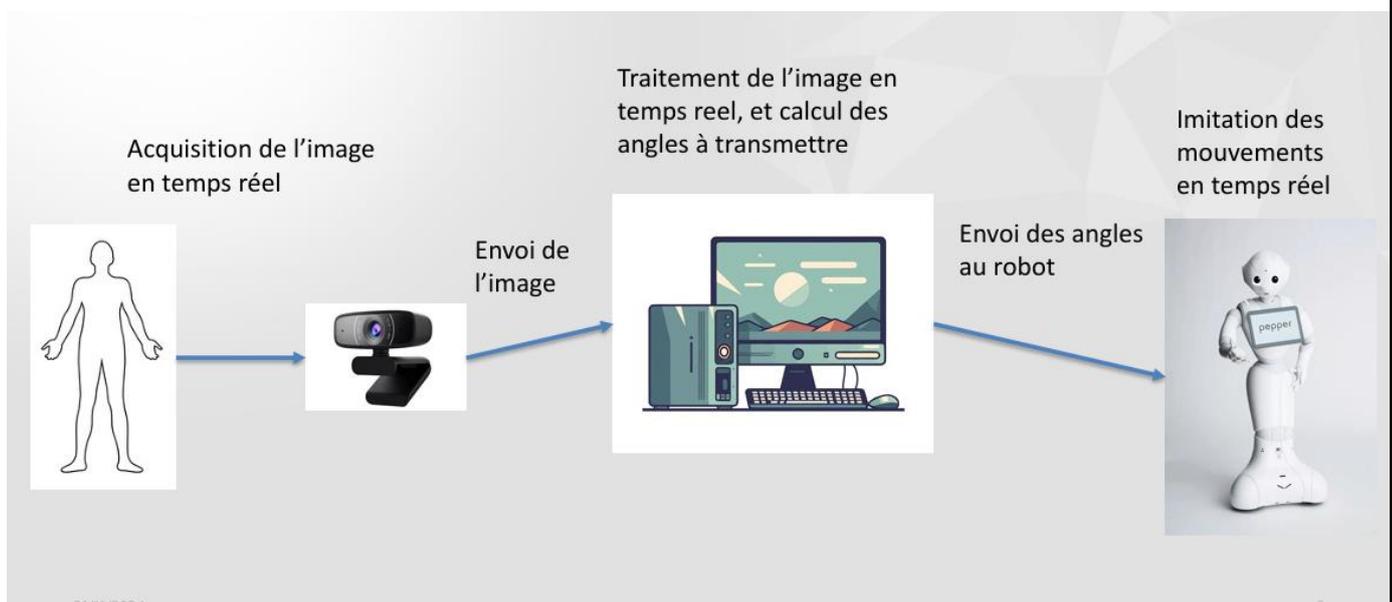


Figure 8 – Fonctionnement général du système (Robot Pepper) . Source : Auteurs

Cette figure illustre un processus technique de capture et de traitement d'images en temps réel, aboutissant à l'imitation des mouvements humains par un robot, en plusieurs étapes :

- 1. Acquisition de l'image en temps réel :**
 - Un système capte les mouvements d'un humain en temps réel. Cela pourrait impliquer l'utilisation d'un capteur ou d'une caméra capable de détecter la position et les articulations du corps humain.
- 2. Envoi de l'image :**
 - L'image capturée est transmise à un système de traitement, ici représenté par une caméra connectée à un ordinateur ou un système central.
- 3. Traitement de l'image et calcul des angles :**
 - Un ordinateur effectue le traitement de l'image en temps réel. Ce processus inclut :
 - La détection des articulations et des segments corporels à partir de l'image.
 - Le calcul des angles nécessaires pour représenter les mouvements détectés.
 - L'extraction des données pertinentes pour les transmettre au robot.
- 4. Envoi des angles au robot :**

- Les angles calculés sont transmis au robot via un protocole de communication adapté (par exemple, réseau ou liaison directe). Ces données permettent au robot de comprendre comment effectuer les mouvements correspondants.

5. Imitation des mouvements en temps réel :

- Le robot utilise les angles reçus pour imiter les mouvements humains détectés. Cette phase inclut le contrôle moteur précis pour reproduire les actions captées.

a) Calcul des angles à transmettre à Pepper

Dans ce projet, les angles articulaires sont déterminés à partir des vecteurs définis entre les points clés des articulations détectés par **MediaPipe**. Les calculs reposent sur des principes mathématiques, notamment l'utilisation du produit scalaire, du produit vectoriel et des projections dans des plans spécifiques.

Les détails de calcul sont mentionnés dans l'annexe D.

b) Programme et architecture du système Miroir Pepper

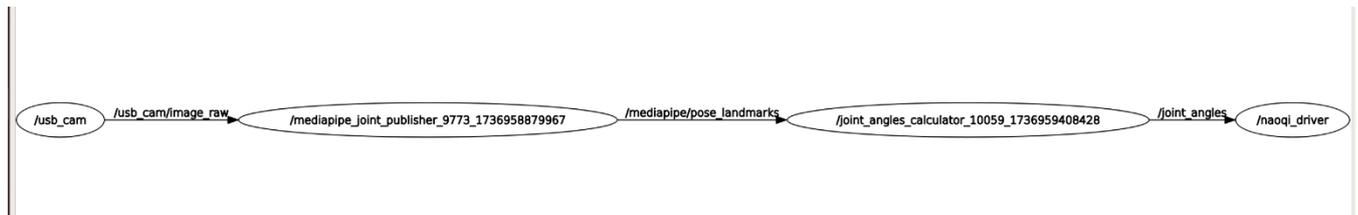


Figure 9- Nœuds Ros du système Miroir Pepper. Source : Auteurs

Ce diagramme représente le flux de données ou la chaîne de traitement entre différents nœuds ROS (Robot Operating System) utilisés pour contrôler le robot Pepper, via une caméra connectée. Voici une explication des étapes principales :

1. **/usb_cam** :
Ce nœud est responsable de la capture des images en direct à partir d'une caméra USB. Les images capturées sont publiées sur le topic /usb_cam/image_raw.
2. **mediapipe_joint_publisher** :
Ce nœud utilise MediaPipe pour analyser les images fournies par la caméra. Il détecte les landmarks (points clés) des articulations humaines et publie ces informations sous forme de coordonnées tridimensionnelles. Ce traitement est effectué sur le topic mediapipe_joint_publisher_<ID>.
3. **mediapipe/pose_landmarks** :
Ce topic reçoit les landmarks (points clés) générés par MediaPipe. Il sert d'interface pour transmettre les données de pose détectées (par exemple, les positions des articulations humaines).
4. **joint_angles_calculator** :
Ce nœud calcule les angles articulaires nécessaires à partir des landmarks reçus. Ces angles représentent les mouvements à imiter par le robot (par exemple, les angles des épaules ou des coudes). Les données calculées sont publiées sur le topic joint_angles.

5. /naoqi_driver :

Ce nœud communique directement avec le robot Pepper via le driver NAOqi. Il reçoit les angles articulaires calculés et les transmet au robot pour qu'il effectue les mouvements correspondants.

C. Robot Braccio

1. Présentation du robot

Braccio est un bras robotique développé par **Arduino** et destiné principalement à des usages éducatifs et expérimentaux. Il s'agit d'un robot de bureau programmable, conçu pour initier les utilisateurs à la robotique, à l'électronique et à la programmation. Braccio est une plateforme flexible, idéale pour des projets d'apprentissage, de prototypage ou d'expérimentations simples en robotique.



Figure 10 - Braccio. Source : <https://docs.rs-online.com/e477/0900766b814da22f.pdf>

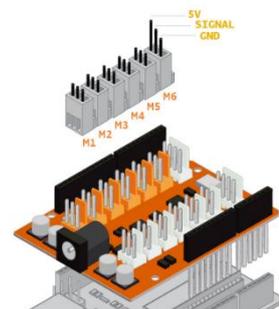
Caractéristiques Techniques

1. Structure Physique

- **Matériaux :**
 - Structure en plastique léger mais robuste, conçue pour être facilement assemblée et manipulée.
- **Dimensions :**
 - Compact et adapté à une utilisation sur un bureau ou dans des environnements éducatifs.
- **Degrés de Liberté (DoF) :**
 - **6 degrés de liberté**, incluant la base rotative, les articulations du bras et l'effecteur terminal.

MOTORS ASSEMBLY

MOTOR "1" BASE
MOTOR "2" SHOULDER
MOTOR "3" ELBOW
MOTOR "4" VERTICAL WRIST
MOTOR "5" ROTATORY WRIST
MOTOR "6" GRIPPER



2. Mouvements et Articulations

- **Servomoteurs inclus :**
 - Utilisation de servomoteurs pour contrôler chaque articulation.
 - Rotation précise et fluide permettant une bonne maniabilité.
- **Effecteur terminal :**
 - Pinces mécaniques à l'extrémité pour saisir ou manipuler de petits objets.

Figure 11 - Moteurs de Braccio Source : <https://docs.rs-online.com/e477/0900766b814da22f.pdf>

3. Électronique et Programmation

- **Carte contrôleur :**
 - Compatible avec les cartes **Arduino Uno** ou similaires.
 - Contrôle des servomoteurs et des capteurs via des broches d'E/S de la carte Arduino.
- **Programmation :**
 - Programmable avec l'environnement Arduino IDE.

- Supporte le langage C++ et des bibliothèques spécifiques pour simplifier le développement.

4. Alimentation

- **Source d'alimentation :**

- Fonctionne avec une alimentation externe pour fournir suffisamment de puissance aux servomoteurs.

2. Travail réalisé

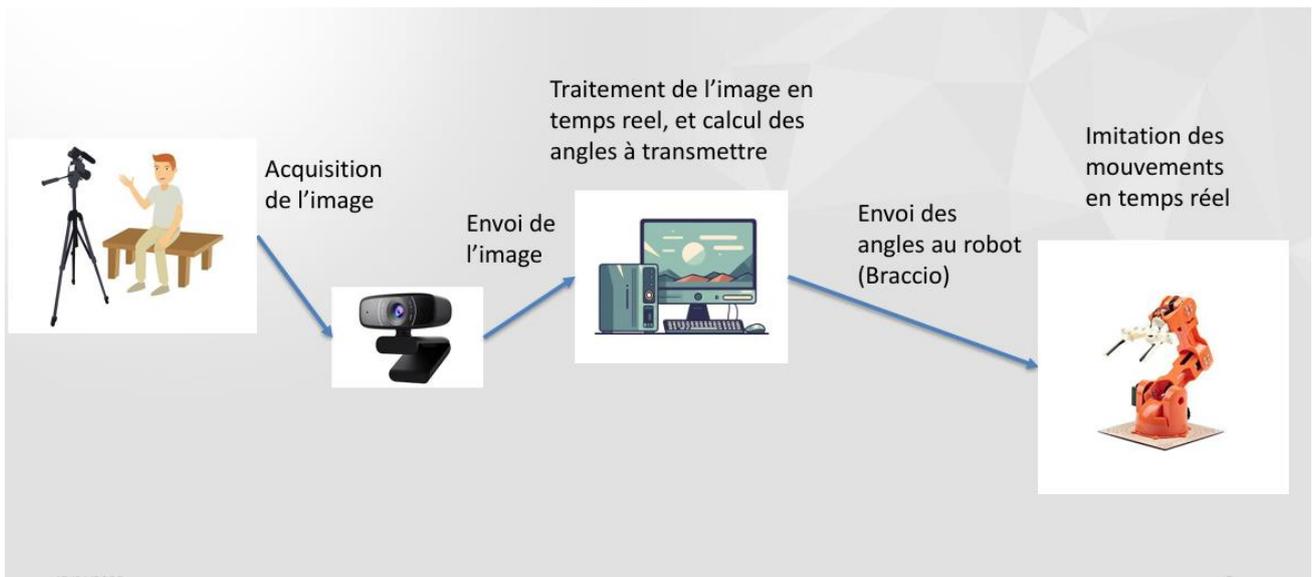


Figure 12 - Fonctionnement général du système miroir Braccio. Source : Auteurs

Le système « **Miroir Braccio** » repose sur les mêmes bibliothèques issues de **MediaPipe Solutions** que celles utilisées pour le Miroir Pepper. Cependant, l'objectif ici est centré sur le suivi et l'analyse des mouvements du poignet de la main, plutôt que sur l'ensemble des articulations. Ce choix est motivé par la nécessité de diriger l'effecteur terminal du robot Braccio de manière précise à l'aide des mouvements du poignet.

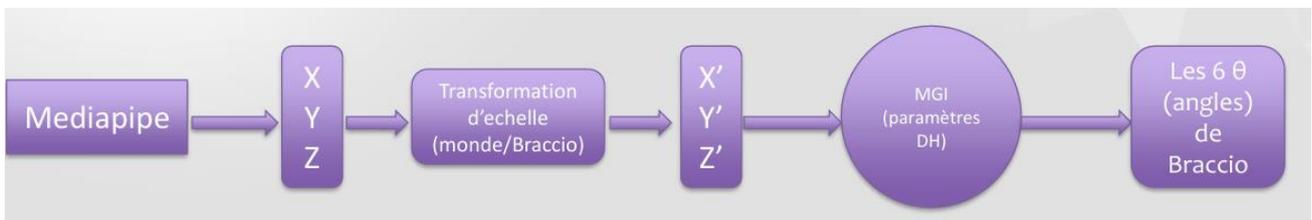


Figure 13 - Flux du calcul des angles à transmettre à Braccio. Source : Auteurs

L'idée principale est de capter les coordonnées tridimensionnelles (x,y,z) du poignet de la main humaine. Ces coordonnées servent ensuite à calculer la **cinématique inverse (MGI)**, qui permet de déterminer les angles nécessaires pour chaque articulation du robot afin de positionner son effecteur

terminal au bon endroit. Ce calcul repose sur des matrices générées à partir des paramètres spécifiques du robot Braccio, **fournis par le constructeur** (par exemple, les longueurs des segments du bras et les limitations des articulations). Une fois les angles calculés, ils sont transmis au robot pour qu'il ajuste sa configuration en conséquence.

Ce processus permet de faire en sorte que le robot Braccio suive les mouvements du poignet de manière fluide et naturelle. Par exemple, en bougeant simplement son poignet, l'utilisateur peut déplacer l'effecteur terminal pour atteindre un objet, en tenant compte des contraintes mécaniques du robot et des limites des articulations. Ce mécanisme met en œuvre des concepts de robotique avancés, tels que la planification des trajectoires, la gestion des collisions potentielles et la correspondance entre l'espace de travail humain et celui du robot. Cela permet au système de répondre efficacement à des besoins tels que la manipulation précise d'objets ou des tâches d'assistance collaborative.

a) *Calcul géométrique*

1. Cinématique Inverse (MGI)

La cinématique inverse consiste à résoudre les équations qui relient la matrice de transformation T et la position cible (x,y,z) afin de déterminer les valeurs des angles $\theta_1, \theta_2, \dots, \theta_5$.

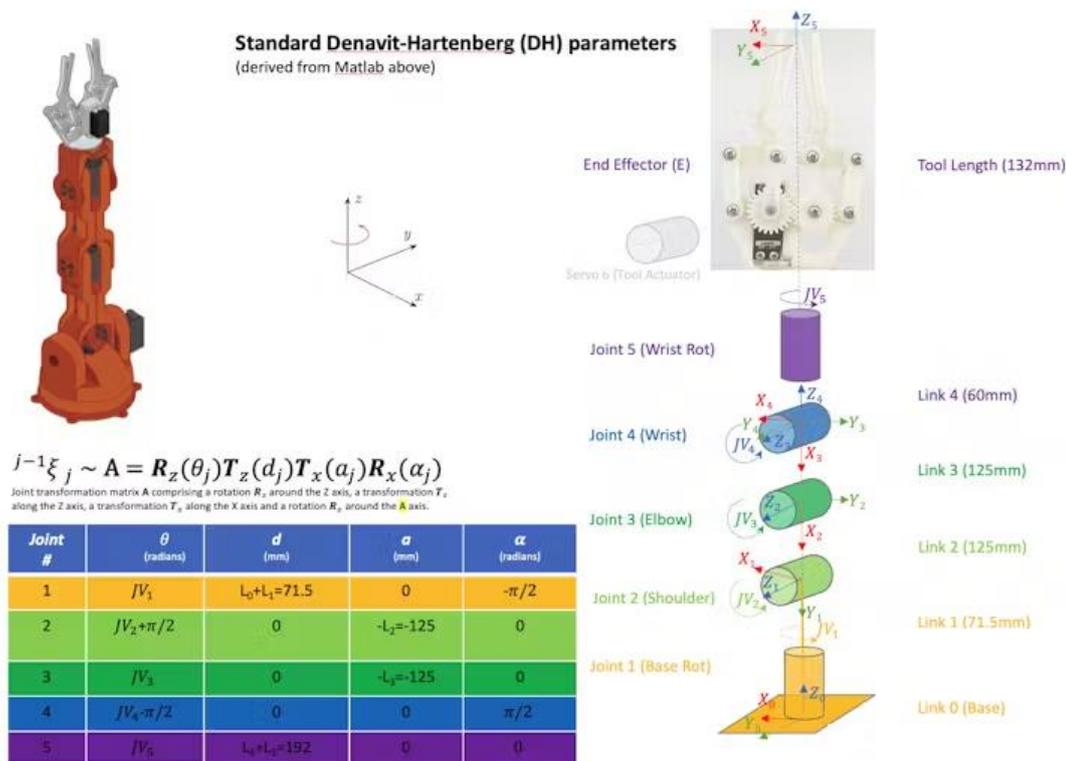


Figure 14 - Paramètres de DH. Source : <https://www.hackster.io/rpatterson/arduino-tinkerkit-braccio-robot-arm-kinematics-1a8303>

Les paramètres Denavit-Hartenberg (DH) sont utilisés pour modéliser la cinématique des robots articulés en définissant les relations géométriques entre les différents segments et articulations du

robot, permettant ainsi de calculer précisément les positions et orientations de l'effecteur terminal dans l'espace en fonction des angles articulaires.

Structure Générale

- **Entrées :**
 - 0T : Matrice de transformation homogène (4x4) représentant la position et l'orientation de l'effecteur terminal dans le repère global.
 - $ol1, ol2, ol3, ol4$: Longueurs des segments du robot.

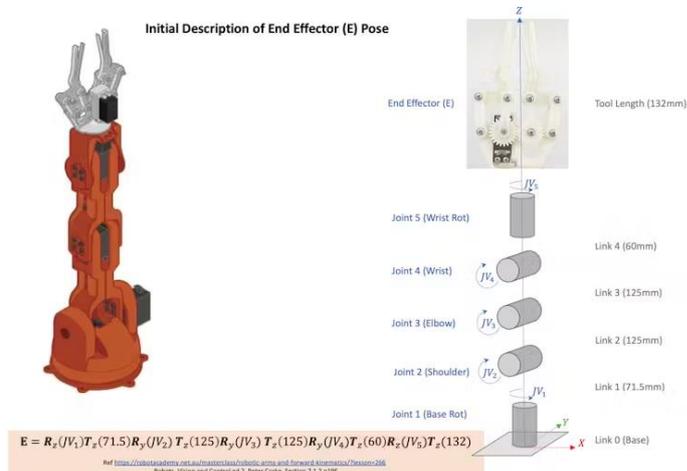


Figure 15 - Description de l'effecteur terminal – Source : <https://www.hackster.io/rpatterson/arduino-tinkerkit-braccio-robot-arm-kinematics-1a8303>

- **Étapes Principales :**
 - Extraction des éléments de la matrice 0T pour obtenir la position (px,py,pz) et l'orientation (rijr).
 - Calcul des angles des articulations ($\theta1, \theta2, \theta3, \dots$) à partir de relations trigonométriques et géométriques.

- **Sorties**

La fonction renvoie une liste des solutions possibles pour les angles des articulations $\theta1, \theta2, \theta3, \theta4, \theta5$. Chaque solution correspond à une configuration valide du robot.

Les détails de calcul sont mentionnés dans l'annexe F

b) Programme et architecture du système Miroir Braccio



Figure 16 - Illustration du système Miroir braccio. Source : https://github.com/ohlr/braccio_arduino_ros_rviz

Ce schéma illustre l'architecture fonctionnelle du contrôle du bras robotique **Braccio**. Le robot est équipé de **6 servomoteurs** permettant de contrôler les différentes articulations du bras. Ces

servomoteurs reçoivent des signaux de **modulation de largeur d'impulsion (PWM)** générés par une carte **Arduino UNO**, qui agit comme le contrôleur principal. L'ordinateur (PC) est connecté à l'Arduino via une communication **USB série** et sert à transmettre des instructions ou des commandes spécifiques. L'Arduino interprète ces commandes reçues du PC et envoie les signaux PWM correspondants aux servomoteurs, permettant ainsi de piloter précisément les mouvements du bras robotique selon les besoins de l'utilisateur. Ce système établit une chaîne de communication claire et efficace entre le PC, l'Arduino et le robot Braccio.

Notre programme est construit essentiellement de ces fichiers :

- **hands_filter.py** : fichier pour filtrer les coordonnées du poignet de la main.
- **hands.py** : fichier pour reconnaître les coordonnées x et y du poignet de la main, et le calcul de FPS (frames per second) pour optimiser la reconnaissance.
- **GetZCalculate.py** : obtenir la coordonnées Z , qui représente aussi la distance entre (détails en annexe F)
- **DOF5fk.cs et DOF5ik.cs** : scripts pour le MGI et le MGD. (en langage C#) ils sont conçus pour communiquer avec l'outil Unity.

➤ Simulation avec Unity

Unity a été utilisé pour visualiser et tester le bras robotique **Braccio**. Les angles des articulations, calculés par un nœud ROS, ont été transmis à Unity afin de permettre une simulation des mouvements en temps réel dans un environnement 3D.

Le processus peut être décrit en trois étapes :

1. Les angles nécessaires aux articulations du Braccio sont calculés par un nœud ROS à partir des données d'entrée (comme des coordonnées ou des consignes).
2. Ces données sont ensuite transmises à Unity via un protocole de communication (comme ROS Bridge).
3. Dans Unity, les angles reçus sont appliqués au modèle 3D du Braccio, ce qui permet de visualiser ses mouvements de manière réaliste.

Grâce à cette méthode, la précision des calculs effectués dans ROS a pu être vérifiée visuellement avant leur utilisation sur le robot réel. De plus, des ajustements ont pu être réalisés pour corriger d'éventuels problèmes et optimiser le flux de données entre ROS et Unity.

Ainsi, une meilleure compréhension du comportement du robot a été obtenue, et les mouvements simulés ont pu être validés avant leur implémentation finale.

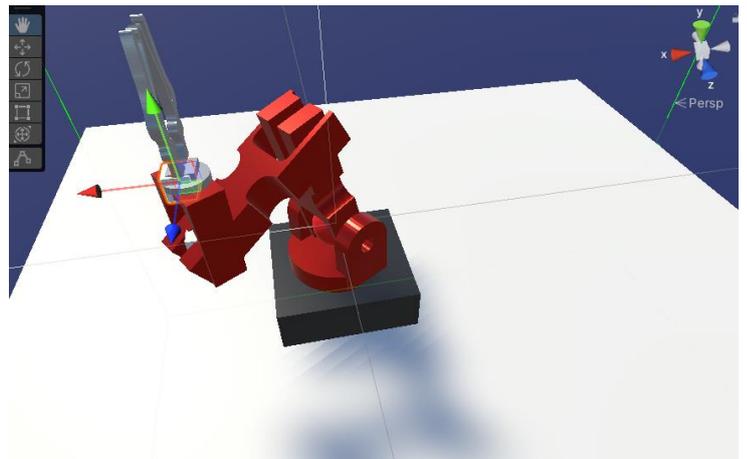


Figure 17 - Braccio dans Unity . Source : Auteurs

VI. Résultats

Dans cette section, sont présentés les résultats obtenus dans le cadre du projet "**Miroir Pepper**", qui vise à permettre aux robots Pepper et Braccio d'imiter les mouvements humains en temps réel. Ce projet s'appuie sur des techniques avancées de reconnaissance des articulations basées sur la vision artificielle, en utilisant **MediaPipe** pour identifier les positions des articulations et calculer les angles géométriques entre elles.

Pour le robot Pepper, les mouvements humains détectés ont été traduits en commandes de joints grâce aux outils de contrôle fournis par **ROS**. De même, pour le bras robotique Braccio, les angles calculés ont été appliqués en utilisant des commandes Arduino, permettant une imitation fluide et précise des mouvements humains.

Les résultats obtenus démontrent la capacité des deux robots à reproduire une large gamme de mouvements, mettant en évidence l'efficacité des méthodes employées et les limites à prendre en compte pour des améliorations futures.

A. Etat d'avancement

Etat d'avancement : 92.3 %

Le calcul est mentionné en annexe D.

La partie **Miroir Pepper** est entièrement finalisée, incluant à la fois la reconnaissance des mouvements humains et leur imitation par le robot.

En ce qui concerne la partie **Miroir Braccio**, elle a été réalisée, mais reste pour l'instant limitée à une simulation effectuée via Unity. Il reste à tester notre programme directement sur le robot physique Braccio.

De plus, les calculs actuellement utilisés pour commander la version virtuelle de Braccio nécessitent encore des ajustements afin de pouvoir être appliqués directement sur le robot réel de manière efficace et précise.

B. Analyse des résultats

1. Robot Pepper

a) Résultats

Le système conçu permet désormais à Pepper d'imiter les mouvements suivants :

- ✓ **Bras (gauche et droit)**
- ✓ **Tête**
- ✓ **Main (ouverture fermeture de la main gauche / droite)**

Comme illustré dans la figure suivante, l'humain, debout face à la caméra, lève les deux bras, un mouvement immédiatement reproduit par Pepper :



Le programme est exécuté dans l'environnement ROS au sein du OS UBUNTU

Output du programme de reconnaissance

Caméra (Webcam) Logitech C170

Figure 18 - Imitation du mouvement par Pepper. Source : Auteurs

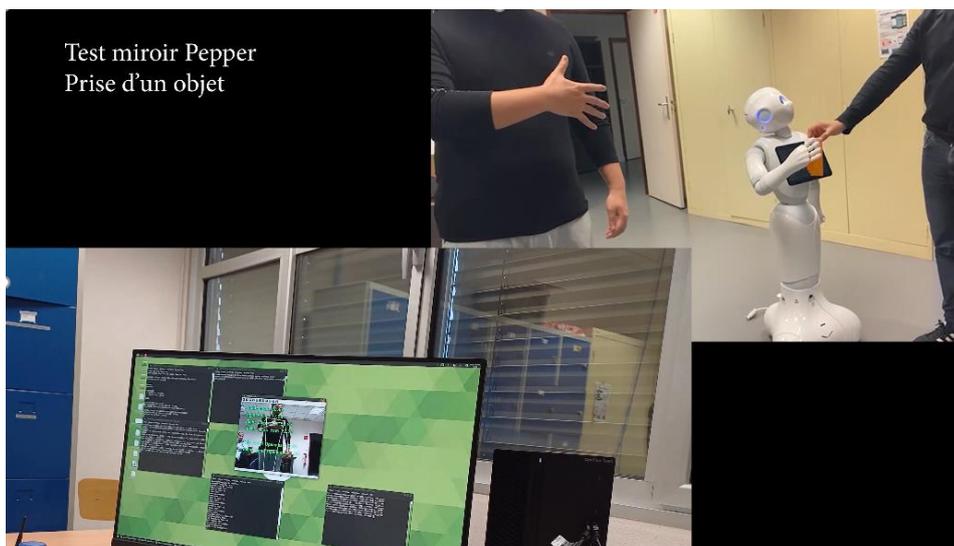


Figure 19 - Imitation de prise d'objet par Pepper. Source : Auteurs

Pepper a été testé pour reproduire les mouvements humains détectés à l'aide de MediaPipe. L'analyse de ses performances a permis de mettre en évidence plusieurs points clés, regroupés ci-dessous.

a. Précision des mouvements

La reproduction des mouvements humains par Pepper a montré une fidélité raisonnable, en particulier pour les mouvements simples et linéaires.

Cependant, plusieurs limitations ont été identifiées :

- **Écarts d'angles** : Les angles calculés via la reconnaissance par MediaPipe ne correspondaient pas toujours parfaitement aux angles appliqués par Pepper, en raison de ses degrés de liberté limités et des restrictions mécaniques.

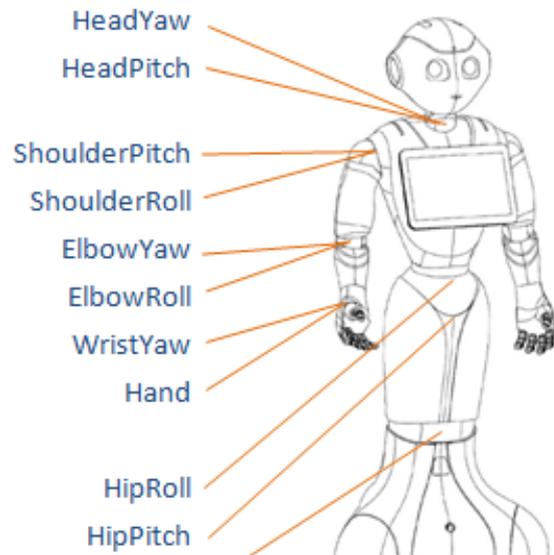


Figure 20 - Robot pepper (joints) . Source : http://doc.aldebaran.com/2-5/family/pepper_technical/joints_pep.html

Par exemple le mouvement du coude a été altéré à cause de la confusion du bras avec l'avant-bras

En effet, le calcul de l'angle repose sur le produit scalaire entre les vecteurs définis par :

$$\vec{v}_1 = \text{Coude} - \text{Épaule} \quad \text{et} \quad \vec{v}_2 = \text{Poignet} - \text{Coude}$$

Où le vecteur v_1 représente le bras et v_2 représente le vecteur de l'avant-bras.

Lorsqu'il y a une confusion :

Les vecteurs sont déformés, car leurs extrémités (Coude, Poignet) ne sont pas bien définies. Cela modifie la valeur du produit scalaire et, par conséquent, l'angle calculé.

- **Postures complexes** : Certaines postures humaines, impliquant des rotations complexes ou une synchronisation entre plusieurs articulations, n'ont pas pu être reproduites avec précision.

b. Limites mécaniques de Pepper

Pepper a un nombre limité de degrés de liberté (DoF) :

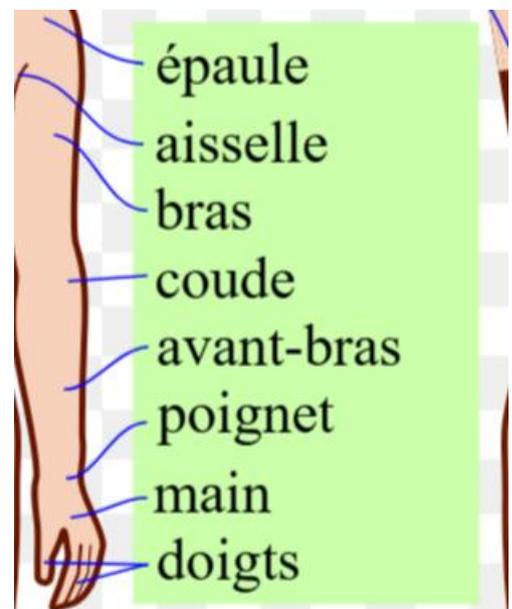
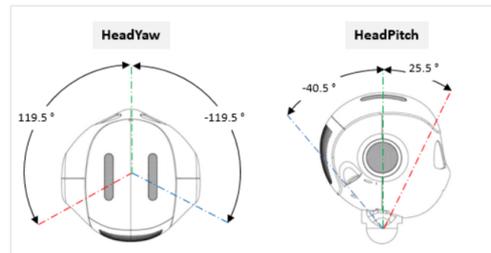


Figure 21 - partie bras du corps humain . source : https://fr.wikipedia.org/wiki/Anatomie_humaine

- Les articulations de ses bras, de son torse et de sa tête ne peuvent pas se déplacer dans toutes les directions possibles.
- Par exemple, une rotation combinée du buste et de l'épaule humaine peut dépasser les capacités de mouvement des moteurs de Pepper, ce qui conduit à une posture partiellement exécutée ou approximative.



Due to collision with Robot casing and tablet, the head yaw and pitch are limited.

HeadYaw (°)	HeadPitch - (°)	HeadPitch + (°)
-119.5	-35.0	13.5
-91.4	-35.1	13.5
-61.6	-35.2	20.9
-33.33	-40.5	25.5
33.33	-40.5	25.5
61.6	-35.2	20.9
91.4	-35.1	13.5
119.5	-35.0	13.5

Figure 22- Les limites des angles de la tete de pepper. Source : http://doc.aldebaran.com/2-5/family/pepper_technical/joints_pep.html

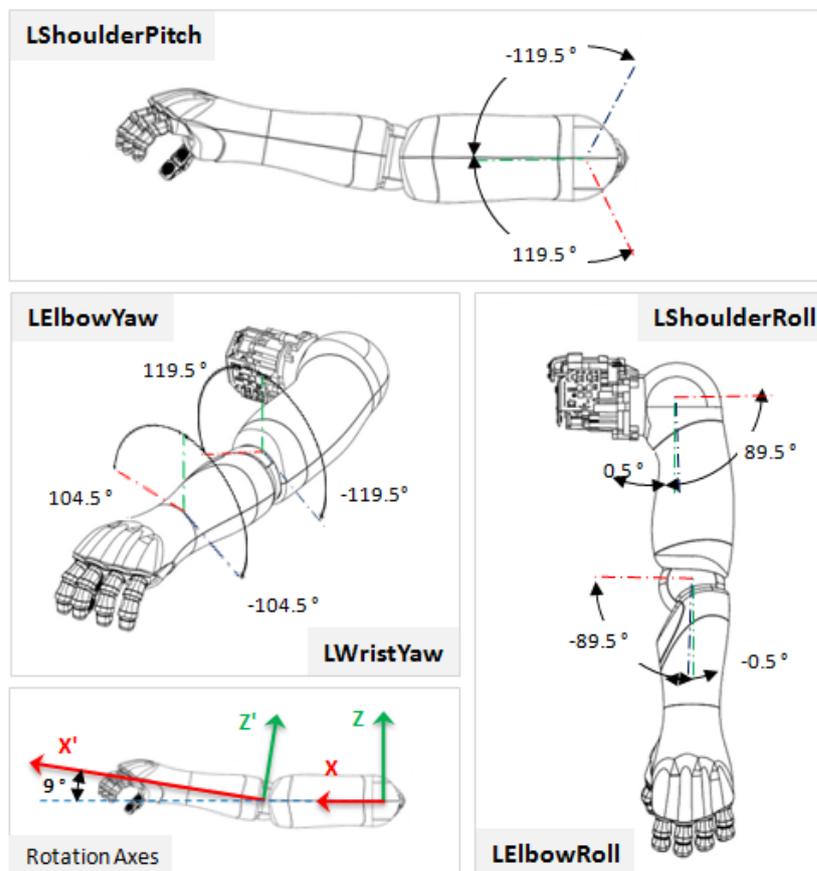


Figure 23 - Limites des angles du bras gauche . Source : http://doc.aldebaran.com/2-5/family/pepper_technical/joints_pep.html

c. Temps de réponse et fluidité

Le système a démontré une latence perceptible entre la détection des articulations par MediaPipe et l'exécution des mouvements par Pepper. Cela a influencé la fluidité des mouvements, particulièrement lors de transitions rapides entre différentes postures.

- **Sources de latence** : La latence était principalement due à la communication entre MediaPipe, ROS, et le système de contrôle de Pepper.
- **Effet sur la fluidité** : Les mouvements étaient parfois saccadés, notamment lors de changements brusques d'orientation ou de vitesse.

d. Robustesse face aux variations environnementales

Les tests ont révélé que la performance de Pepper était sensible aux variations environnementales, notamment :

- **Conditions d'éclairage** : Un éclairage insuffisant a réduit la précision de la détection des articulations par MediaPipe, ce qui a conduit à des mouvements imprécis.
- **Couleurs des vêtements portés par l'humain devant la caméra** : les vêtements en noir ont posé un problème quant à la détection des mouvements et la reconnaissance des articulations.
- **Angle de la caméra** : Des positions de caméra non optimales ont également affecté la qualité des mouvements imités, notamment lorsque certaines articulations étaient partiellement masquées.

b) Validation du cahier de charges

Comme est observé dans ce tableau la **contrainte d'un décalage < 1 seconde** a été validée :

id	Mesure (en s)	Moyenne(en s)
1	0.55	0.61
2	0.6	
3	0.38	
4	0.75	
5	0.8	

Figure 24 - Délai entre mouvements de l'humain et le mouvement de Pepper . Source : Auteurs

2. Robot Braccio

a) Résultats

Le système développé permet à la **version virtuelle** de Braccio (sous Unity) d'ajuster l'ensemble de ses articulations en calculant les angles requis pour atteindre une position spécifique dans son espace. Cette position est définie en fonction de la localisation du poignet de la main humaine.

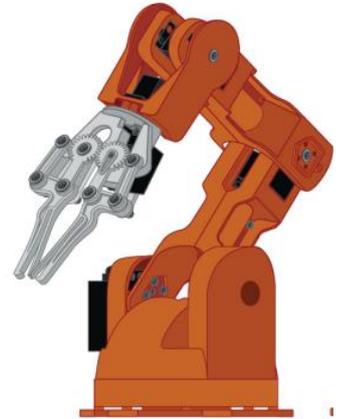


Figure 25 - robot Braccio .
Source : <https://docs.rs-online.com/e477/0900766b814da22f.pdf>

a. Précision des mouvements

- **Utilisation de la MGI pour le positionnement** : Grâce à la MGI, la version virtuelle de Braccio a pu atteindre les positions cibles correspondant aux coordonnées détectées du poignet humain. Les résultats montrent que les mouvements reproduits sont généralement précis.

M1 = base degrees
M2 = shoulder degrees
M3 = elbow degrees
M4 = vertical wrist degrees
M5 = rotatory wrist degrees
M6 = gripper degrees

Step Delay: a milliseconds delay between the movement of each servo. Allowed values: from 10 to 30 msec.

M1 allowed values from 0° to 180°

M2 allowed values from 15° to 165°

M3 allowed values from 0° to 180°

M4 allowed values from 0° to 180°

M5 allowed values from 0° to 180°

M6 allowed values from 10° to 73°. (10°: the gripper is open, 73°: the gripper is closed).

Figure 26 - Limitations des angles de Braccio . Source : : <https://docs.rs-online.com/e477/0900766b814da22f.pdf>

- **Limites mécaniques du bras** : La structure mécanique de Braccio impose certaines contraintes, notamment en termes de portée et d'angles de rotation des articulations. Ces limitations ont parfois restreint sa capacité à atteindre des positions très éloignées ou complexes, particulièrement pour des mouvements impliquant des angles extrêmes.

b. Temps de réponse et fluidité

- **Latence des calculs** : Le temps nécessaire pour convertir les coordonnées du poignet humain en angles d'articulations via la MGI, puis pour transmettre ces commandes à Braccio via Arduino, a été mesuré pendant les tests (presque 900 ms en moyenne). Une légère latence a été observée, mais elle reste suffisamment faible pour permettre des mouvements en temps quasi réel.

- **Transitions fluides** : Les mouvements de la version virtuelle de Braccio (sous Unity) se sont révélés globalement fluides lors de changements progressifs de position. Cependant, des saccades ont été notées dans des cas où les positions cibles impliquaient des changements rapides ou importants dans les angles.

c. Précision des commandes Arduino

- **Conversion des angles en signaux** : Les commandes générées pour Braccio à partir des calculs de MGI ont été correctement traduites en signaux Arduino. Néanmoins, certaines imprécisions ont été détectées dans les mouvements des moteurs, liées à des tolérances mécaniques et des limites dans la précision des servomoteurs utilisés.

Résultats sur l'outil Unity :

Importance de la Coordonnée Z dans la Simulation

L'utilisation des coordonnées x,y,z du poignet, détectées via une caméra, est essentielle pour contrôler le bras robotique. Cependant, la coordonnée z, qui correspond à la distance entre la caméra et le poignet, joue un rôle particulièrement important.

Cette distance doit rester inférieure à la **portée maximale du Braccio**, qui est d'environ **80 cm** lorsqu'il est entièrement déployé. Dépasser cette distance rendrait impossible pour le robot d'atteindre la position du poignet, ce qui compromettrait la pertinence des calculs de **Modèle Géométrique Inverse (MGI)**. De plus :

- Une distance excessive pourrait réduire la précision des coordonnées détectées en raison des limitations de la caméra.
- Les angles calculés pour le contrôle du robot deviendraient incohérents, rendant les mouvements irréalisables.

Pour garantir la fiabilité de la simulation, un contrôle a été mis en place dans Unity pour surveiller la coordonnée z.

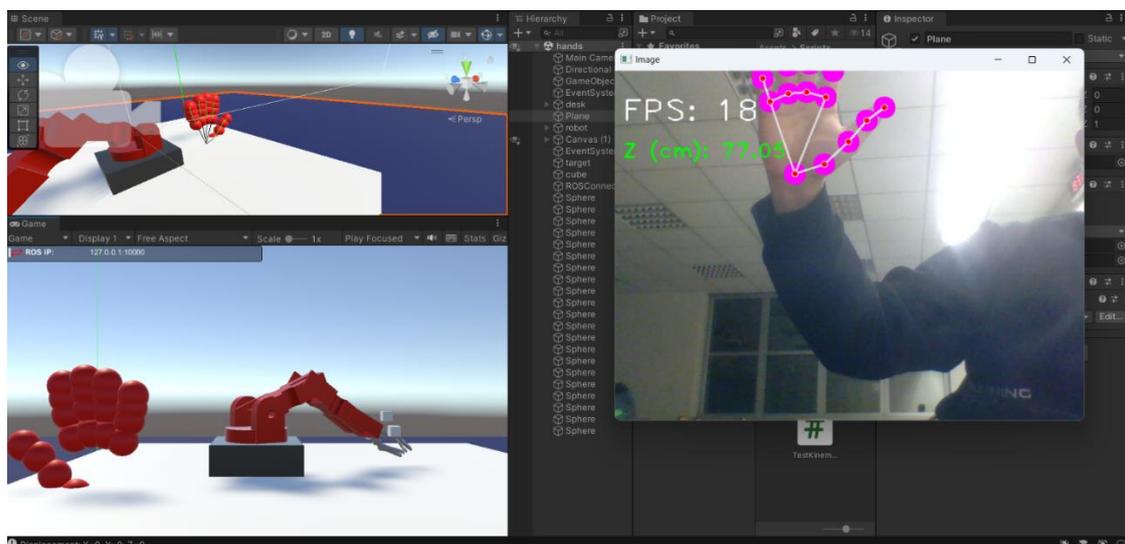


Figure 27 - Simulation de Braccio sur Unity. Source : Auteurs

On observe dans cette simulation :

- **Capture des mouvements humains (fenêtre en haut à droite) :**

- MediaPipe capture les mouvements de la main humaine en temps réel, détectant les points clés (landmarks) et affichant la distance Z (profondeur en centimètres) par rapport à la caméra.

- **Représentation du bras humain dans Unity (fenêtre à gauche) :**

- Le bras humain est modélisé par des sphères connectées, représentant les articulations et leurs positions dans l'espace 3D.
- Cette simulation permet de suivre les mouvements des articulations et de les mapper dans l'environnement virtuel.

- **Interface Unity et ROS (fenêtre en arrière-plan) :**

- Unity utilise ROS pour recevoir les données des mouvements capturés en temps réel.
- Le champ "ROS IP" illustre la connexion active entre le système de capture et l'environnement Unity.

b) Respect des contraintes

✓ **Précision de calcul angulaire** mentionnée dans le cahier de charges dans l'**annexe A**

La différence entre les deux mesures est calculée en degrés, et une contrainte de tolérance de $\pm 6^\circ$ a été appliquée pour évaluer la conformité des calculs. En effet, **un compas** a été utilisé pour mesurer l'angle entre les articulations du coude, après avoir dessiné des droites dans le milieu du bras et de l'avant-bras. Ces mesures ont été comparées avec celles calculées par le programme. Comme l'indique le graphe, tous les tests respectent cette contrainte ($\pm 6^\circ$). Cette validation est un indicateur

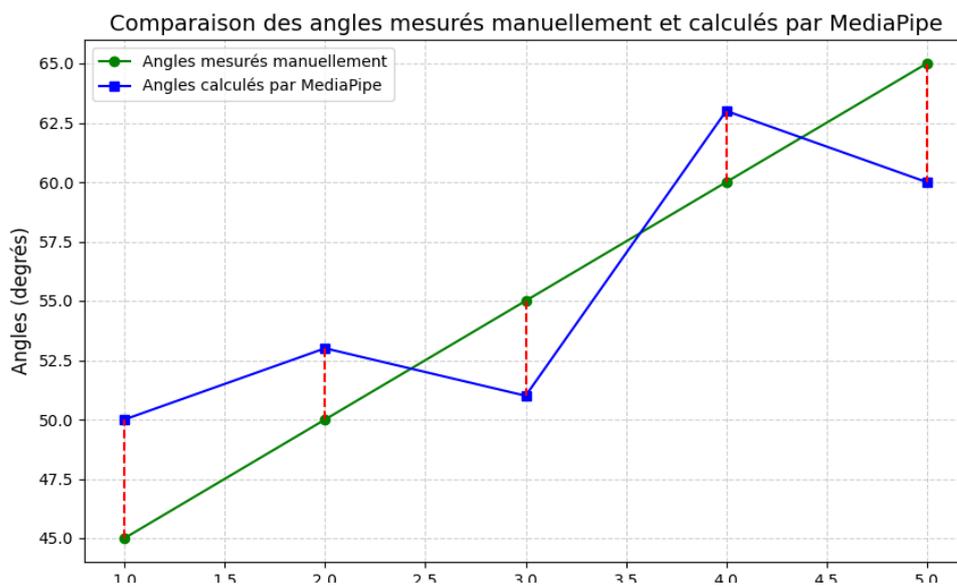


Figure 28 - Comparaison des angles mesurés manuellement avec celle calculés via Mediapipe

clé de la performance du système et de sa capacité à reproduire des mouvements humains avec précision.

- ✓ Décalage de mouvement entre la main humaine et Braccio mentionnée dans le cahier de charges dans l'annexe A.

Comme l'indique le tableau suivant, tous les tests respectent cette contrainte (décalage < 1 seconde).

id	Mesure (en s)	Moyenne(en s)
1	0.45	0.56
2	0.7	
3	0.5	
4	0.9	
5	0.2	

Figure 29 - Décalage entre mouvement de la main humaine et Braccio. Source : Auteurs

C. Propositions d'amélioration

1. Robot Pepper

Pour optimiser les performances de Pepper dans des applications futures, plusieurs améliorations peuvent être envisagées :

- **Optimisation des angles calculés** : Développer un algorithme d'ajustement pour adapter les angles calculés par MediaPipe aux contraintes mécaniques de Pepper.
- **Réduction de la latence** : Améliorer la communication entre MediaPipe et ROS pour réduire les délais de traitement et rendre les mouvements plus fluides.
- **Robustesse environnementale** : Ajouter des algorithmes de prétraitement pour stabiliser les données d'entrée de MediaPipe face aux variations d'éclairage ou d'angle.

2. Robot Braccio

- **Optimisation de la MGI** : Une meilleure optimisation des algorithmes de MGI pourrait réduire davantage les erreurs de positionnement, en particulier pour les mouvements complexes.
- **Ajout de capteurs de retour d'information** : Intégrer des capteurs, tels que des encodeurs, pour fournir un retour d'information en temps réel sur la position des articulations permettrait d'améliorer la précision des mouvements.
- **Amélioration des moteurs** : L'utilisation de servomoteurs plus performants pourrait réduire les erreurs liées à la mécanique et améliorer la fluidité des mouvements.

- **Anticipation des mouvements rapides** : En prédisant les positions futures du poignet humain, le système pourrait adapter les commandes envoyées à Braccio, réduisant ainsi les saccades pour les mouvements rapides.

VII. Compétences acquises

Au cours de ce projet, les compétences et connaissances suivantes ont été acquises :

1. Utilisation flexible du framework ROS

L'utilisation de ROS a été maîtrisée pour assurer la communication entre les nœuds, permettant ainsi la transmission en temps réel des angles des articulations humaines au robot.

2. Application de la cinématique directe et inverse

Dans le cadre du bras robotique Braccio, les fonctions de cinématique directe et inverse n'ont pas été simplement utilisées via des bibliothèques comme MoveIt, mais ont été programmées manuellement. Cette approche a permis une compréhension approfondie des concepts sous-jacents.

3. Utilisation de MediaPipe en vision par ordinateur

L'exploitation de MediaPipe a été réalisée pour obtenir les coordonnées tridimensionnelles des points clés du corps humain et les utiliser dans le calcul et l'analyse des angles articulaires.

4. Collaboration entre matériel et logiciel

Bien que les objectifs aient été atteints dans la simulation du bras Braccio, des ajustements du code ont été nécessaires pour assurer une coopération efficace entre le logiciel et le matériel lors de l'interaction avec le bras robotique réel.

5. Collaboration en équipe et résolution de problèmes

Les défis techniques et les problèmes de débogage rencontrés ont été surmontés grâce à une collaboration efficace au sein de l'équipe, permettant ainsi de trouver des solutions adaptées.

6. Capacité de présentation

La création d'outils de gestion, tels que des diagrammes de Gantt et des structures de répartition du travail (WBS), ainsi que la présentation de l'avancement du projet aux clients, ont été perfectionnées.

VIII. Conclusion générale

Ce projet aborde deux problématiques techniques majeures : comment utiliser MediaPipe pour obtenir des angles articulaires précis et les mapper en temps réel au système de mouvement du robot Pepper, et comment planifier les trajectoires du bras robotique Braccio en utilisant des algorithmes de cinématique directe et inverse afin d'effectuer une préhension d'objets avec précision.

Dans la première partie, une architecture ROS à deux nœuds a été mise en place. Un nœud calcule les angles articulaires à partir des coordonnées tridimensionnelles obtenues avec MediaPipe, tandis qu'un autre les convertit en radians pour contrôler les articulations du robot Pepper. Les résultats ont montré que Pepper peut imiter des mouvements humains simples et saisir un objet après plusieurs tentatives. Dans la seconde partie, les algorithmes de cinématique directe et inverse ont permis de contrôler le bras robotique Braccio pour atteindre des positions spécifiques avec une certaine précision.

Cependant, des limitations demeurent. Pour Pepper, des erreurs dans le calcul des angles articulaires apparaissent lors de gestes complexes, altérant la fluidité des mouvements. Pour Braccio, l'absence d'une correspondance exacte entre les coordonnées du poignet humain et celles de l'extrémité du bras robotique entraîne une marge d'erreur dans les mouvements.

Pour aller au-delà de ces résultats, il serait pertinent d'explorer l'intégration de capteurs de déplacement ou de systèmes de capture de mouvement avancés afin d'améliorer la précision des données et le contrôle des robots. De plus, le recours à des algorithmes d'apprentissage profond pour anticiper et optimiser les mouvements pourrait ouvrir de nouvelles perspectives dans l'interaction entre robots et humains.

IX. Bibliographie et webographie

Ouvrages et Articles :

[1] Lugaresi, Camillo, *MediaPipe: A Framework for Building Perception Pipelines*, arXiv, 2019, 13 pages. Disponible en ligne : <https://arxiv.org/abs/1906.08172>

Sites Internet :

[2] Open Robotics. *ROS Documentation*, [Type de support en ligne]. Disponible en ligne : <https://docs.ros.org/> (consulté le 15 janvier 2025).

[3] Google AI for Developers. *MediaPipe Solutions Guide*, [Type de support en ligne]. Disponible en ligne : <https://ai.google.dev/edge/mediapipe/solutions/guide> (consulté le 15 janvier 2025).

[4] SoftBank Robotics. *Pepper Overview*, [Type de support en ligne]. Disponible en ligne : <https://www.softbankrobotics.com/emea/en/pepper> (consulté le 15 janvier 2025).

[5] Arduino. *Braccio Robot Arm*, [Type de support en ligne]. Disponible en ligne : <https://store.arduino.cc/products/tinkerkit-braccio> (consulté le 15 janvier 2025).

[6] Unity Technologies. *Unity Documentation*, [Type de support en ligne]. Disponible en ligne : <https://docs.unity3d.com/> (consulté le 15 janvier 2025).

[7] ROS Wiki. *ROS Tutorials*, [Type de support en ligne]. Disponible en ligne : <https://wiki.ros.org/ROS/Tutorials> (consulté le 15 janvier 2025).

[8] GitHub. *MediaPipe Repository*, [Type de support en ligne]. Disponible en ligne : <https://github.com/google/mediapipe> (consulté le 15 janvier 2025).

[9] PyPI. *mediapipe Python Package*, [Type de support en ligne]. Disponible en ligne : <https://pypi.org/project/mediapipe/> (consulté le 15 janvier 2025).

[10] ROS French Users Group. *Documentation ROS 2 en Français*, [Type de support en ligne]. Disponible en ligne : https://ros-french-users-group.github.io/ros2_documentation/ (consulté le 15 janvier 2025).

[11] Google I/O. *MediaPipe : le machine learning intégré simplifié*, [Type de support en ligne]. Disponible en ligne : <https://io.google/2023/program/44701430-a72b-47e3-aeb9-a6f8d5faace4/intl/fr/> (consulté le 15 janvier 2025).

Table des matières

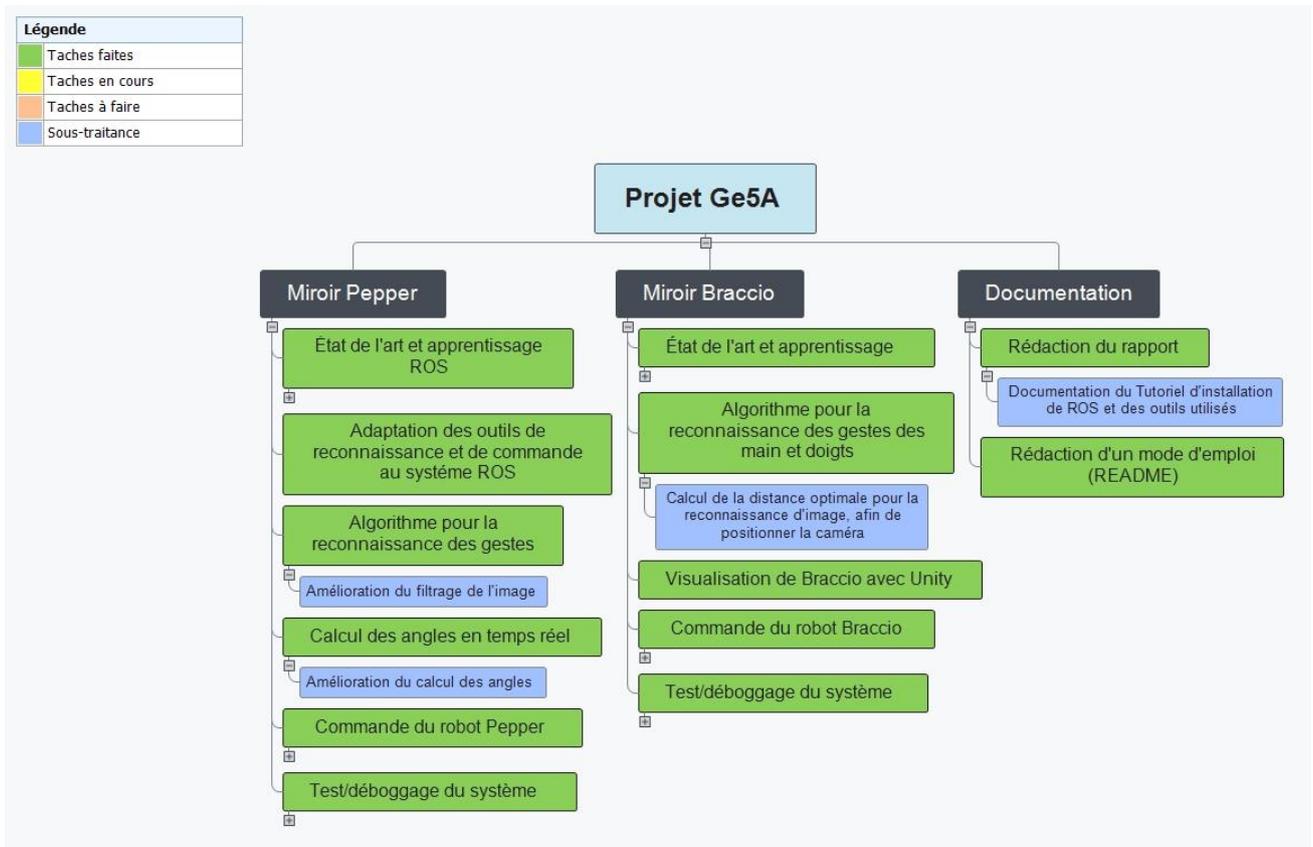
Résumé	
Abstract	
I. Introduction.....	1
II. Contexte	2
III. Problématique du sujet.....	2
IV. Gestion du projet.....	2
a) Organisation et chronologie avec Gantt.....	2
b) Organisation des taches.....	3
1. Miroir Pepper	4
2. Miroir Braccio	4
3. Documentation	4
V. Travail réalisé.....	5
A. Outils utilisés	5
1. Mediapipe	5
Pourquoi Utiliser Mediapipe ?	5
2. ROS (Robot Operating System)	5
Caractéristiques Principales	5
Composants Clés	6
Versions et Évolutions	6
3. Unity	6
Unity Simulator en Robotique et Technologie	7
B. Robot Pepper.....	7
1. Présentation du Robot.....	7
Caractéristiques Principales	8
1. Design Physique	8
2. Capacités d'Interaction	8
3. Capteurs	8
4. Connectivité	8
5. Logiciel	8
6. Applications	9
7. Personnalisation	9

2.	Objectifs réalisés.....	9
1.	Reconnaissance des mouvements des articulations humaines avec MediaPipe	9
2.	Mapping des mouvements humains par rapport à Pepper	10
3.	Calcul des angles pour la transmission à Pepper	10
4.	Contrôle du robot Pepper	10
5.	Tests et garantie d'une faible latence	11
3.	Fonctionnement du système	11
a)	Calcul des angles à transmettre à Pepper	12
b)	Programme et architecture du système Miroir Pepper.....	12
C.	Robot Braccio	13
1.	Présentation du robot	13
1.	Structure Physique.....	13
2.	Mouvements et Articulations	13
3.	Électronique et Programmation	13
4.	Alimentation	14
2.	Travail réalisé	14
a)	Calcul géométrique	15
1.	Cinématique Inverse (MGI)	15
	Structure Générale.....	16
•	Sorties	16
b)	Programme et architecture du système Miroir Braccio	16
➤	Simulation avec Unity	17
VI.	Résultats.....	18
A.	Etat d'avancement.....	18
B.	Analyse des résultats.....	19
1.	Robot Pepper	19
a)	Résultats.....	19
	19
	20
b)	Validation du cahier de charges	22
2.	Robot Braccio	23
a)	Résultats.....	23
b)	Respect des contraintes	25
C.	Propositions d'amélioration	26
1.	Robot Pepper	26

2.	Robot Braccio	26
VII.	Compétences acquises	28
VIII.	Conclusion générale.....	29
IX.	Bibliographie et webographie	
	Ouvrages et Articles :	
	Sites Internet :	
X.	Annexes.....	
A.	WBS.....	
B.	Cahier de charges.....	
C.	Gantt.....	
D.	Etat d'avancement.....	
E.	Etapes de calcul géométrique.....	
	1. Calcul des angles à l'aide du produit scalaire	
	2. Calcul des angles dans un plan spécifique	
	3. Calcul de l'inclinaison ou "pitch"	
	4. Calcul de l'angle du coude	
	5. Calcul de Yaw (rotation axiale autour du bras)	
	6. Utilisation du filtre de Kalman	
F.	MGD et MGI pour Braccio.....	

X. Annexes

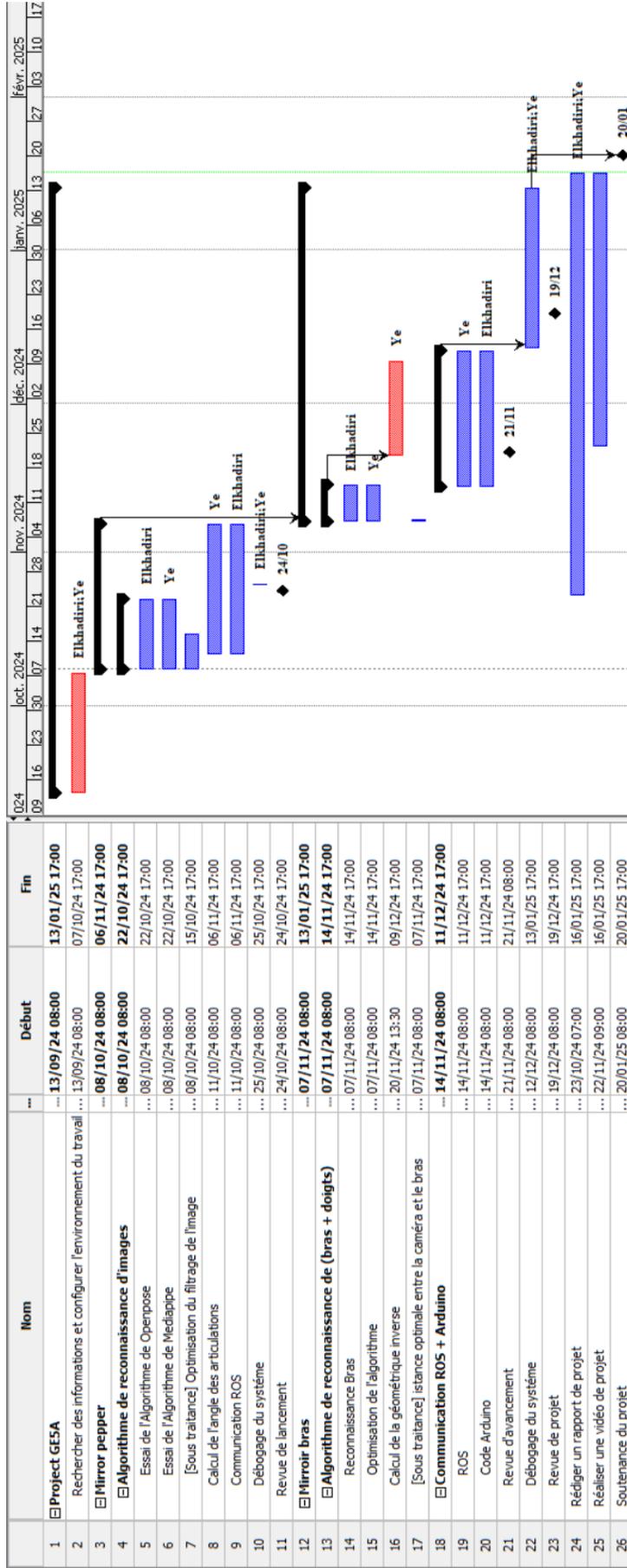
A. WBS



B. Cahier de charges

Fonctions	Description	Critères	Contraintes
Miroir Pepper			
FP1	Reconnaissance des Mouvements Humains	Précision de détection Temps de traitement Compatibilité des caméras	Environnement : ROS version 1
FP2	Mapping des Gestes Humains vers les Mouvements du Robot	Correspondance des gestes Calibration	Environnement ROS Précision de calcul : $\pm 6^\circ$ Contraintes angulaires liées au robot Pepper
FP3	Contrôle du Robot via ROS	Temps de réponse acceptable	Environnement : ROS version 1
FC1	La reconnaissance des angles constantes doit être indépendante des rotations et translations du corps humain en mouvement.	Précision de calcul Erreur minimale entre l'angle prévue et angle calculé	
FC2	Fluidité du mouvement de Pepper par rapport au mouvement du corps humain		Décalage du mouvement < 1s
Miroir Braccio			
FP1	Reconnaissance des Mouvements du bras humain	Zone de détection pertinente	Environnement : ROS version 1
FP2	Mapping des Mouvements des Joueurs vers les Mouvements du Robot	Reproduction exacte des mouvements Interaction fluide	Environnement : ROS version 1 Calcul des angles par la Modélisation géométrique inverse Précision de calcul des angles : $\pm 6^\circ$ Contraintes angulaires liées au robot Braccio
FP3	Contrôle du Robot via ROS	Précision des mouvements sur le matériel du jeu Modularité	Environnement : ROS version 1
FC1	Fluidité du mouvement de Braccio par rapport au mouvement du bras humain	Facteur d'échelle pertinent	Décalage du mouvement < 1s

C. Gantt



D. Etat d'avancement

Tâches	Durée(D) en heure	Criticité (C)	Priorité pondéré (D*C)	Taux de complétion (pa)	Progression Pondérée (D*C*pa)	C*pa
Miroir Pepper						
Etat de l'art et apprentissage ROS						
Outils de reconnaissance de gestes humains	8	4	32	1	32	4
Comparaison entre les différents outils et choix de l'outil optimal						
Outils de commande du robot	8	3	24	1	24	3
Comparaison entre les différents outils et choix de l'outil optimal						
Documentation Pepper	2	1	2	1	2	1
Identifier les gestes à reconnaître						
Identifier les limites du mouvements de robots						
Adaptation des outils de reconnaissance et de commande au système ROS						
Installer les dépendances Anaconda (Python 3)	7	2	14	1	14	2
Algorithme pour la reconnaissance des gestes						
Comparer les performances des différents algorithmes	10	3	30	0,85	19,5	1,95
Choix du meilleur algorithme en fonction de précision et de rapidité						
Calcul des angles en temps réel						
Conversion du système de coordonnées	16	4	64	1	64	4
Ajuster les angles perçus aux angles du Robot						
Utilisation du filtre de Kalman pour le filtrage du bruit						
Commande du robot Pepper						
Création du noeud ROS de reconnaissance des gestes	2	2	4	1	4	2
Création du noeud ROS pour la commande de Pepper	2	2	4	1	4	2
Test du système						
Détection des anomalies et des erreurs de précision	16	3	48	1	48	3
Amélioration des algorithmes	16	3	48	1	48	3
Miroir bras						
Etat de l'art et apprentissage ROS						
Outils de commande du bras	5	4	20	1	20	4
Comparaison entre les différents outils et choix de l'outil optimal						
Lecture de la documentation du bras	2	2	4	1	4	2
Identifier les limites du mouvements du bras						
Algorithme pour la reconnaissance des gestes de la main						
Comparer les performances des différents algorithmes	16	3	48	0,9	43,2	2,7
Choix du meilleur algorithme en fonction de précision et de rapidité						
Calcul des angles de mouvements du bras	8	3	24	1	24	3
Commande du robot Braccio (simulateur / test réel)						
Création du noeud ROS de reconnaissance des gestes de la main	16	1	16	0,8	12,8	0,8
Création du noeud ROS pour la commande du bras	16	1	16	0,8	12,8	0,8
Test de système						
Détection des anomalies et des erreurs de précision	10	3	30	0,8	24	2,4
Amélioration des algorithmes	8	3	24	0,5	12	1,5
Documentation						
Rédaction du rapport et réalisation video	50	2	100	1	100	2
Rédaction d'un mode d'emploi (PEADME)	2	1	2	1	2	1
TOTAL	220	50			Estimation d'avancement :	0,923

E. Etapes de calcul géométrique

1. Calcul des angles à l'aide du produit scalaire

La méthode principale repose sur le produit scalaire entre deux vecteurs v_1 et v_2 , défini comme suit :

$$\vec{v}_1 \cdot \vec{v}_2 = \|\vec{v}_1\| \cdot \|\vec{v}_2\| \cdot \cos(\theta)$$

où θ représente l'angle entre les deux vecteurs. L'équation est réarrangée pour isoler θ :

$$\theta = \arccos \left(\frac{\vec{v}_1 \cdot \vec{v}_2}{\|\vec{v}_1\| \cdot \|\vec{v}_2\|} \right)$$

Cette formule est utilisée pour déterminer plusieurs angles, tels que **le roll**, **le pitch**, et **les angles des coudes**, en fonction des vecteurs correspondants.

2. Calcul des angles dans un plan spécifique

Pour **le roll**, les vecteurs sont projetés dans un plan bidimensionnel (par exemple, x-y) avant de procéder au calcul. Le produit scalaire est ensuite utilisé pour déterminer l'angle entre les vecteurs projetés.

3. Calcul de l'inclinaison ou "pitch"

Le **pitch** est calculé dans un espace tridimensionnel. Une combinaison du produit scalaire et du produit vectoriel est employée :

1. Le produit scalaire permet de calculer l'angle principal :

$$\theta = \arccos \left(\frac{\vec{v}_1 \cdot \vec{v}_2}{\|\vec{v}_1\| \cdot \|\vec{v}_2\|} \right)$$

2. Le produit vectoriel est utilisé pour déterminer la direction de l'angle, en fonction de l'orientation des vecteurs :

$$\text{Cross product norm} = \|\vec{v}_1 \times \vec{v}_2\|$$

Ces calculs permettent de garantir la cohérence des résultats, notamment en cas d'ambiguïtés directionnelles.

4. Calcul de l'angle du coude

L'angle du coude est défini comme l'angle entre les vecteurs représentant le bras supérieur (shoulder → elbow) et l'avant-bras (elbow → wrist). Il est déterminé comme suit :

$$\theta = \arccos \left(\frac{v_{\text{shoulder} \rightarrow \text{elbow}} \cdot v_{\text{elbow} \rightarrow \text{wrist}}}{\|v_{\text{shoulder} \rightarrow \text{elbow}}\| \cdot \|v_{\text{elbow} \rightarrow \text{wrist}}\|} \right)$$

Ensuite, des ajustements sont appliqués pour limiter l'angle aux plages mécaniques définies (par exemple, -90° à 90°).

5. Calcul de Yaw (rotation axiale autour du bras)

Le yaw du coude est obtenu en projetant le vecteur de l'avant-bras sur un plan orthogonal au bras supérieur. Le vecteur projeté est comparé à un axe de référence, et l'angle est déterminé à l'aide de la fonction $\arctan2$, permettant d'inclure la direction :

$$\theta = \arctan 2(\text{z-component}, \text{x-component})$$

Cela garantit une mesure précise de l'angle de rotation autour de l'axe longitudinal du bras.

6. Utilisation du filtre de Kalman

Les valeurs calculées peuvent être bruitées en raison des variations dans la détection des points clés. Un filtre de Kalman est appliqué pour lisser ces variations et fournir des résultats stables. Le filtre corrige les angles calculés à chaque itération :

$$\text{Angle corrigé} = \text{Kalman.correct}(\text{Angle brut})$$

Cela permet d'améliorer la fluidité et la précision des mouvements reproduits par le robot.

F. MGD et MGI pour Braccio

La cinématique directe (FK) et la cinématique inverse (IK) reposent sur des relations mathématiques et géométriques permettant de décrire et calculer la position et l'orientation d'un bras robotique. Voici une explication détaillée de leur calcul.

1. Cinématique directe (FK - Forward Kinematics)

Objectif : Calculer la position et l'orientation de l'effecteur final (le bout du bras) à partir des angles des articulations θ et des longueurs des segments du bras.

Étapes

- Définition des paramètres DH (Denavit-Hartenberg) :** La méthode DH est utilisée pour modéliser la géométrie du bras robotique. Chaque articulation et segment est défini par :
 - θ : Angle de rotation autour de l'axe z.
 - d : Déplacement le long de l'axe z.
 - a : Longueur du segment, déplacement le long de l'axe xx.
 - α : Angle de torsion entre deux axes z.
- Matrice de transformation pour chaque articulation :** Pour chaque segment i , la transformation entre le segment i et $i-1$ est donnée par la matrice :

$$T_i = \begin{bmatrix} \cos \theta_i & -\sin \theta_i \cos \alpha_i & \sin \theta_i \sin \alpha_i & a_i \cos \theta_i \\ \sin \theta_i & \cos \theta_i \cos \alpha_i & -\cos \theta_i \sin \alpha_i & a_i \sin \theta_i \\ 0 & \sin \alpha_i & \cos \alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Matrice de transformation globale :** En multipliant les matrices de transformation de chaque articulation, on obtient la matrice finale :

$$T = T_1 \cdot T_2 \cdot T_3 \cdot \dots \cdot T_n$$

Cette matrice 4x4 contient :

- La position finale [px,py,pz] de l'effecteur.
- L'orientation finale, représentée par une matrice de rotation 3x3.

2. Cinématique inverse (IK - Inverse Kinematics)

Objectif : Calculer les angles des articulations nécessaires pour atteindre une position et une orientation spécifiques, définies par une matrice de transformation TT.

Étapes

1. **Extraire les informations de la matrice de transformation :** À partir de la matrice TT, on récupère :
 - [px,py,pz] : la position de l'effecteur.
 - [r11,r12,r13,...] : les éléments de la matrice de rotation.
2. **Calcul de l'angle θ_1 :** L'angle θ_1 est calculé à partir de la projection de la position sur le plan XY :

$$\theta_1 = \arctan 2(py, px)$$

3. **Calcul de l'angle θ_3 :** L'angle θ_3 est déterminé en utilisant la loi des cosinus sur le triangle formé par les segments l_2 et l_3 :

$$\cos \theta_3 = \frac{l_2^2 + l_3^2 - d^2}{2 \cdot l_2 \cdot l_3}$$

où d est la distance entre l'origine et [px,py,pz].

Calcul de l'angle θ_2 : En utilisant les relations trigonométriques et les distances projetées sur le plan XY et Z, on calcule :

$$\theta_2 = \arctan 2(p_z, \sqrt{p_x^2 + p_y^2}) - \text{correction due à } \theta_3$$

4. **Angles θ_4 et θ_5 :** Les angles restants sont calculés à partir de la matrice de rotation et des relations géométriques entre les segments.

Solutions multiples :

- Plusieurs solutions peuvent exister, car certaines articulations peuvent être dans différentes configurations (par exemple, coude haut ou bas).
- Le système retourne toutes les solutions possibles (en l'occurrence 4) pour que l'utilisateur choisisse celle qui convient.