

Mémoire de stage de fin d'étude
Développement du robot assistant pour le maquillage
SU, Sichen

BICHON, Jean-Christophe tuteur industriel
Supervisé par BICHON Jean-Christophe, BA Sileye, LI Tao
BEAREE Richard tuteur pédagogique

L'ORÉAL
Recherche & Innovation

NOTICE BIBLIOGRAPHIQUE

ANNEE : 2022-2023 **N° :** LI-2022/12/11556

TYPE DE DOCUMENT : Rapport de SFE

CAMPUS DE RATTACHEMENT : Arts et Métiers Lille

AUTEURS : SU Sichen

TITRE : Développement du robot assistant pour le maquillage

ENCADREMENT : BEAREE Richard

ENTREPRISE PARTENAIRE : L'Oréal Recherche & Innovation

NOMBRES DE PAGES :

NOMBRE DE REFERENCES BIBLIOGRAPHIQUES :

RESUME : Le projet MURA (Make Up Robotic Arm) vise à développer un robot assistant capable d'appliquer du maquillage sur le visage. L'objectif de ce stage est de mettre en place un framework pour le contrôle du robot, en utilisant des produits cosmétiques pour réaliser des maquillages sur différentes parties du visage. Dans ce projet, nous nous appuyons sur le Robot Operating System (ROS) pour développer ce framework. Dans ce rapport présentera plus précisément les travaux suivants : • Définition et architecture du projet MURA (Make-Up Robot Arm) • Construction d'environnement de simulation dans le cadre de ROS. • Développement d'outils de contrôle pour un robot et la calibration de caméra et la calibration hand-eye • Mise en œuvre de la vision par ordinateur, en utilisant mediapipe réaliser une détection du visage et l'extraction de points caractéristiques. D'ailleurs, basé mediapipe, on est aussi arrivé à un maquillage virtuel • Contrôle du système robotique, utilisation de la cinématique directe et inverse pour piloter les servomoteurs en temps réel et planifier une trajectoire plus efficace. • Utilisation de l'apprentissage par renforcement pour améliorer les performances du robot dans l'application du maquillage.

MOTS CLES : Robot assistant, Robot collaboratif, Robot Real-Time contrôle, Computer vision, Machine Learning, ROS

Développement du robot assistant pour le maquillage



SU Sichen
15 janvier – 15 juillet 2023



Stage SFE

Lieu : Saint-Ouen, Paris (93)

Tuteur ENSAM BEAREE Richard (campus Lille)

Maître de stage : Jean-Christophe BICHON

Résumé : Ce stage de fin d'études a pour objectif de développer un système robotique capable de réaliser un maquillage automatique. Pour résoudre ce défi, nous avons développé un environnement de simulation afin de tester les algorithmes nécessaires tels que le contrôle du robot, la vision par ordinateur, graphique par ordinateur et l'apprentissage par renforcement. À la fin, nous sommes parvenus à réaliser une planification de mouvement pour le robot ainsi qu'un maquillage basique.

Mots clefs : Robot assistant, Robot collaboratif, Robot Real-Time contrôle, Computer vision, Force contrôle, Machine Learning, ROS.

I. INTRODUCTION

1. Structure du document

Dans ce rapport, nous présenterons la construction d'un environnement de simulation ainsi que les algorithmes que nous avons utilisés pour permettre au robot de détecter le visage et d'effectuer un maquillage sur un visage modèle.

Durant mon stage, je me suis basé sur ROS Melodic pour construire notre environnement de simulation. Dans cet environnement, nous avons développé et testé les algorithmes de robot contrôle et de vision par ordinateur.

II. METHODES MISES EN ŒUVRE

1. Make-Up Robotic Arm (MURA)

Le projet MURA (Make Up Robotic Arm) vise à développer un robot assistant capable d'appliquer du maquillage sur le visage. Pour accomplir cette tâche, nous devons combiner la vision par ordinateur, le contrôle du robot et les graphiques informatiques pour piloter notre robot. Par conséquent, nous avons divisé cette tâche en trois parties.

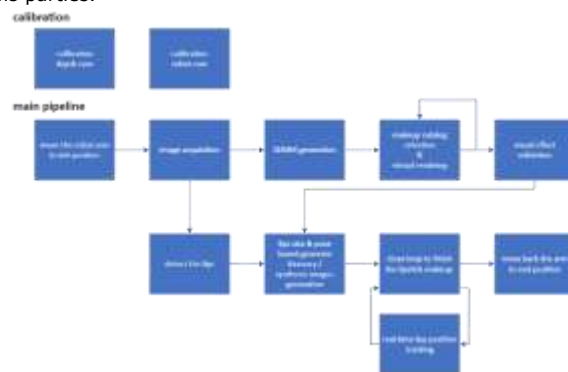


Figure 1 Pipeline de MURA

Dans ce projet, nous avons trois parties principales : **la vision par ordinateur**, utilisée pour détecter le visage et extraire les points caractéristiques ; **le contrôle du robot**, permettant de diriger celui-ci vers les cibles et d'éviter les risques de collision ; et **le graphique d'ordinateur**, utilisés pour visualiser formes de maquillage choisies par l'opérateur et le client.

2. Construction de l'environnement de la simulation

Au cours des dernières années, les avancées en robotique ont rendu les robots indispensables dans de nombreux secteurs industriels et applications variées. Afin d'assurer le succès du déploiement et de l'exploitation des systèmes robotiques, il est essentiel de les développer et de les tester dans des environnements à la fois réalistes et dynamiques. Durant mon stage, mon travail s'est concentré sur la création d'environnements de simulation



Figure 2 Structure de l'environnement simulation

En conclusion, les missions de ce stage m'ont permis d'approfondir mes connaissances en informatique, notamment en vision par ordinateur, graphiques informatiques et apprentissage par renforcement. Elles m'ont également donné l'opportunité d'améliorer mes compétences dans ces domaines et de comprendre comment piloter un projet de recherche.

III. RESULTATS

A la fin du stage, nous avons construit un environnement bien équipé avec les éléments nécessaires. Nous avons réussi la détection de visage et le maquillage virtuel. Enfin, nous avons réussi à piloter notre robot pour effectuer un maquillage dans l'environnement de simulation

RÉFÉRENCES

- [1] ROS Tutorial: Control the UR5 robot with `ros_control` - How to tune a PID Controller https://roboticscasual.com/ros-tutorial-control-the-ur5-robot-with-ros_control-tuning-a-pid-controller.
- [2] A., Yang, L., Hartikainen, K., Finn, C., & Levine, S. (2019). End-to-End Robotic Reinforcement Learning without Reward Engineering. ArXiv:1904.07854 [Cs.LG].
- [3] Head Pose Estimation using Python <https://towardsdatascience.com/head-pose-estimation-using-python-d165d3541600>.
- [4] Franceschetti, Andrea & Tosello, Elisa & Castaman, Nicola & Ghidoni, Stefano. (2021). Robotic Arm Control and Task Training through Deep Reinforcement Learning.

Make-Up Robotic Arm



SU Sichen
15 January – 15 July 2023



PFE

Location : Saint-Ouen, Paris (93)

ENSAM academic tutor Richard BEAREE (campus Lille)

Tutor : Jean-Christophe BICHON

Summary: *The aim of this internship is to develop a robotic system capable of performing automatic make-up. To solve this challenge, we developed a simulation environment to test the necessary algorithms such as robot control, computer vision, computer graphics and reinforcement learning. In the end, we were able to perform motion planning for the robot as well as basic make-up.*

Key words: *Robot assistant, Robot collaborative, Robot Real-Time control, Computer version, Machine Learning, ROS.*

I. INTRODUCTION

1. Structure of the document

In this report, we present the construction of a simulation environment and the algorithms we used to enable the robot to detect the face and perform make-up on a model face.

II. METHODES USED

2. Make-Up Robotic Arm (MURA)

The MURA (Make Up Robotic Arm) project aims to develop an assistant robot capable of applying make-up to the face. To accomplish this task, we need to combine computer vision, robot control and computer graphics to drive our robot. We have divided this task into three parts.



Figure 1 MURA Pipeline

In this project, we have three main parts: **computer vision**, used to detect the face and extract characteristic points; **robot control**, to direct the robot towards targets and avoid collision risks; and **computer graphics**, used to visualize make-up shapes chosen by the operator and customer.

3. Building the simulation environment

In recent years, advances in robotics have made robots indispensable in a wide range of industries and applications. To ensure the successful deployment and operation of robotic systems, it is essential to develop and test them in realistic and dynamic environments. During my internship, my work focused on the creation of simulation environments

During my internship, I used ROS Melodic to build our simulation environment. In this environment, we developed and tested robot control and computer vision algorithms.



Figure 2 Structure of simulation environment

In conclusion, this internship has enabled me to deepen my knowledge of computer science, particularly computer vision, computer graphics and reinforcement learning. They also gave me the opportunity to improve my skills in these fields and to understand how to manage a research project.

III. RESULTS

By the end of the course, we had built a well-equipped environment with the necessary elements. We successfully performed face detection and virtual make-up. Finally, we successfully piloted our robot to perform make-up in the simulation environment.

REFERENCES

- [1] ROS Tutorial: Control the UR5 robot with ros_control - How to tune a PID Controller https://roboticscasual.com/ros-tutorial-control-the-ur5-robot-with-ros_control-tuning-a-pid-controller.
- [2] A., Yang, L., Hartikainen, K., Finn, C., & Levine, S. (2019). End-to-End Robotic Reinforcement Learning without Reward Engineering. ArXiv:1904.07854 [Cs.LG].
- [3] Head Pose Estimation using Python <https://towardsdatascience.com/head-pose-estimation-using-python-d165d3541600>.
- [4] Franceschetti, Andrea & Tosello, Elisa & Castaman, Nicola & Ghidoni, Stefano. (2021). Robotic Arm Control and Task Training through Deep Reinforcement Learning.

Contents

1.	INTRODUCTION	7
1.1	Présentation de l'entreprise	7
1.2	Présentation du contexte du stage	7
1.3	Présentation de la problématique du stage	8
1.4	La structure du rapport	9
2.	Pipe-line de projet MURA	9
3.	Construction de l'Environnement Simulation	10
3.1	Contexte et Problématique	10
	Contexte	10
	Problématique	11
3.2	Méthodes utilisées pour répondre à la problématique	12
	Contexte	12
	Modélisation de robot et gripeur	13
	Construction du modèle de Realsense caméra D415	16
	Construction de force capteur	16
	Construction de gazebo world	17
	Création une configuration Moveit pour l'UR5 et un gripeur	18
3.3	Résultats obtenus	19
4.	Robot contrôle et camera calibration	19
4.1	Contexte et Problématique	19
	Contexte	19
	Problématique	20
4.2	Méthodes utilisées pour répondre à la problématique	21
	Contexte	21
	Robot cinématique et robot motion planning	26
	Camera calibration et hand-eye calibration	28
4.3	Résultats obtenus	30
5.	Computer vision et simulation de maquillage	30
5.1	Contexte et Problématique	30
	Contexte	30
	Problématique	31
5.2	Méthodes utilisées pour répondre à la problématique	31

Contexte.....	31
Configuration de media pipe pour s'adapter le ROS.....	32
Acquisition de l'information 3d de visage dans la coordonné de caméra.....	34
Acquisition de lèvres infos	35
Transformer les coordonnées de la caméra vers la base du robot.	36
Estimation de la pose de la tête.....	36
Généraliser une trajectoire pour robot	38
Colorisation de la model humain.	38
5.3 Résultats obtenus	42
6. Motion planning de robot	42
6.1 Contexte et Problématique	42
Contexte.....	42
Problématique	43
6.2 Méthodes utilisées pour répondre à la problématique.....	43
Contexte.....	43
Optimisation de la trajectoire	44
6.3 Résultats obtenus	46
7 Renforcement Learning Framework	47
7.1 Contexte et Problématique	47
Contexte.....	47
Problématique	47
7.2 Méthodes utilisées pour répondre à la problématique.....	47
Construction du cadre de RL	47
7.3 Résultats obtenus	49
8 Synthèse, conclusion et perspectives.....	49
9 Bibliographie	51

1. INTRODUCTION

1.1 Présentation de l'entreprise

L'Oréal Recherche & Innovation est le département de recherche et développement de L'Oréal, le plus grand entreprise cosmétiques au monde. L'Oréal Recherche & Innovation joue un rôle clé dans la création et le développement de nouveaux produits de beauté et de soins de cheveux. Leur objectif principal est d'innover continuellement pour répondre aux besoins et aux attentes des consommateurs du monde entier.

L'Oréal Recherche & Innovation compte plus de 4000 chercheurs, dont des chimistes, des biologistes, des experts en vision par ordinateur et en intelligence artificielle. Leur mission est de développer de nouveaux produits pour L'Oréal. Ils sont constamment à la pointe de la technologie et explorent des domaines tels que la beauté augmentée en utilisant des techniques optiques, la vision par ordinateur et l'apprentissage automatique. Ces avancées technologiques favorisent l'accélération du développement de produits innovants dans le domaine de la beauté. Les laboratoires sont équipés d'outils de pointe et emploient des techniques sophistiquées pour explorer de nouveaux domaines tels que la génomique, la biologie cellulaire, la robotisation en laboratoire, la modélisation informatique et la bio-imagerie.

En résumé, L'Oréal est un acteur majeur de l'industrie cosmétique, découverte de nouvelles technologies, de nouveaux ingrédients et de nouvelles formules pour offrir des produits de beauté de haute qualité aux consommateurs du monde entier.

1.2 Présentation du contexte du stage

Pendant mon stage, j'ai travaillé sur un projet robotique appelé MURA, qui signifie "Make-up RoboticArm". Ce projet était le fruit d'une collaboration entre les équipes suivantes :

- L'équipe de l'IA
Sileye BA, scientifique senior en apprentissage automatique, responsable de la direction intelligence artificiel.
Sichen SU, stagiaire en IA et robotique, responsable du contrôle de robot.
- L'équipe de vision par ordinateur
Tao LI, ingénieur senior en vision par ordinateur, responsable de la direction de la vision par ordinateur.
Vaibhav ELANGOVAN, ingénieur en robotique et vision par ordinateur.
- L'équipe de mécatronique. Notre équipe était composée de 6 personnes :
Jean-Christophe BICHON, mon tuteur professionnel et également le responsable du projet MURA.
Florian Dupuis, ingénieur R&D en mécatro-robotique, responsable de la gestion de projet.

Le projet MURA avait pour objectif de développer une plateforme avec un bras robotisé capable de maquiller le visage humain. Dans cette plateforme, nous souhaitons que le robot puisse effectuer les tâches suivantes :

- **Détection du visage humain et identification des zones d'intérêt.** Le système utilisait des images pour générer un maillage du visage humain, fournissant ainsi des informations géométriques telles que la position des pixels et l'orientation de la tête dans le référentiel de coordonnées.
- **Panification de trajectoire de robot.** Génération d'une trajectoire en fonction du maillage du visage du client, permettant d'appliquer différents produits cosmétiques.
- **Robot compliance control.** Contrôle de la force et du couple exercés par le robot afin d'éviter de blesser les humains, le système doit suivre précisément une trajectoire prédéfinie et réaliser une force control.

1.3 Présentation de la problématique du stage

Les robots ont un potentiel énorme pour assister les gens dans la vie quotidienne. Dans notre projet, nous cherchons à développer un système robotique capable d'effectuer du maquillage automatiquement, notamment pour les personnes handicapées, à l'autre côté, nous voulons réaliser une tâche de tester les produit maquillage dans un tête model.

Ce projet nous a confrontés à des défis à la fois passionnants et complexes, tels que la planification du mouvement du robot, le contrôle de la dynamique du robot et l'interaction physique, entre autres. Afin de tester différents algorithmes et divers modules, la première tâche que nous devons résoudre est la construction d'un environnement de simulation. C'est dans cet environnement que nous pourrions développer et tester les algorithmes

Dans cet environnement, il doit contenir les éléments basiques : un robot manipulateur et un modèle humain avec des propriétés physiques appropriées. En outre, nous devons ajouter des capteurs tels que des caméras RGBD, des capteurs de force pour détecter les données. Dans cet environnement, nous devons également effectuer le maquillage sur le modèle humain pour évaluer la qualité du maquillage.

Ensuite, une fois l'environnement de simulation construit, nous devons développer un module de vision par ordinateur. Ce module utilise une caméra RGBD pour acquérir des informations 3D. Nous devons également développer un algorithme pour détecter le visage humain et fournir les informations de Mesh pour déterminer la position des points clés. De plus, ce module doit calculer l'orientation de la tête en temps réel. Enfin, ce module doit me fournir une carte UV pour que je puisse appliquer le maquillage dans la zone qui m'intéresse.

Un autre défi est le module de contrôle du robot. Ce module doit s'appuyer sur la cinématique inverse et directe. Dans ce module, nous pouvons réaliser une

définition de collision, une définition de la model de mouvement de robot. Il peut bien calculer les configurations de robot pour réaliser un mouvement.

Le défi final concerne la mise en place d'un cadre pour l'apprentissage par renforcement. Notre objectif est d'obtenir un contrôle plus précis, et l'apprentissage par renforcement représente une excellente voie pour parvenir à un contrôle de robot à la fois plus sûr et plus précis. Nous souhaitons développer un cadre qui permettra au robot d'exécuter des actions spécifiques, et selon les actions effectuées, des récompenses seront attribuées. De plus, ce cadre sera également en mesure d'évaluer efficacement la politique de contrôle du robot

1.4 La structure du rapport

Dans ce mémoire, nous allons présenter l'évolution du projet MURA. En premier lieu, nous détaillerons le pipeline de MURA. Ensuite, nous discuterons de la construction de l'environnement de simulation avant d'aborder le développement de l'outil de contrôle et les processus de calibration. Par la suite, nous évoquerons le module de vision par ordinateur et examinerons les diverses fonctions que nous pouvons réaliser grâce aux informations provenant de la caméra. En outre, nous expliquerons comment générer une trajectoire plus efficace. Sur la base de notre système initial, nous montrerons ensuite comment réaliser un contrôle plus optimisé et précis. Enfin, nous conclurons sur le projet MURA.

2. Pipe-line de projet MURA

Le projet Mura vise à développer un système robotisé, assisté par la vision par ordinateur et l'intelligence artificielle, pour réaliser un maquillage automatique. L'objectif est de tester de nouveaux produits sur un modèle de tête humaine, et, en fin de compte, de déployer ce système sur de vraies personnes, permettant une interaction physique entre l'humain et le robot.

Pour mener à bien cette mission, nous avons mis en place un pipeline pour guider notre projet. Ce pipeline implique deux tâches de préparation essentielles : la calibration de la caméra et celle du robot. Les paramètres de calibration sont cruciaux pour assurer la précision du système robotisé. Une fois toutes les calibrations effectuées, notre objectif est de mettre en place un contrôle du robot lui permettant d'atteindre une position spécifique pour identifier les zones d'intérêt. À cet égard, nous nous appuyons sur la vision par ordinateur pour détecter les lèvres et réaliser un maquillage virtuel sur celles-ci. Lorsque le client ou l'opérateur choisit une couleur de rouge à lèvres, le robot suit une trajectoire préalablement générée pour accomplir l'application du maquillage.

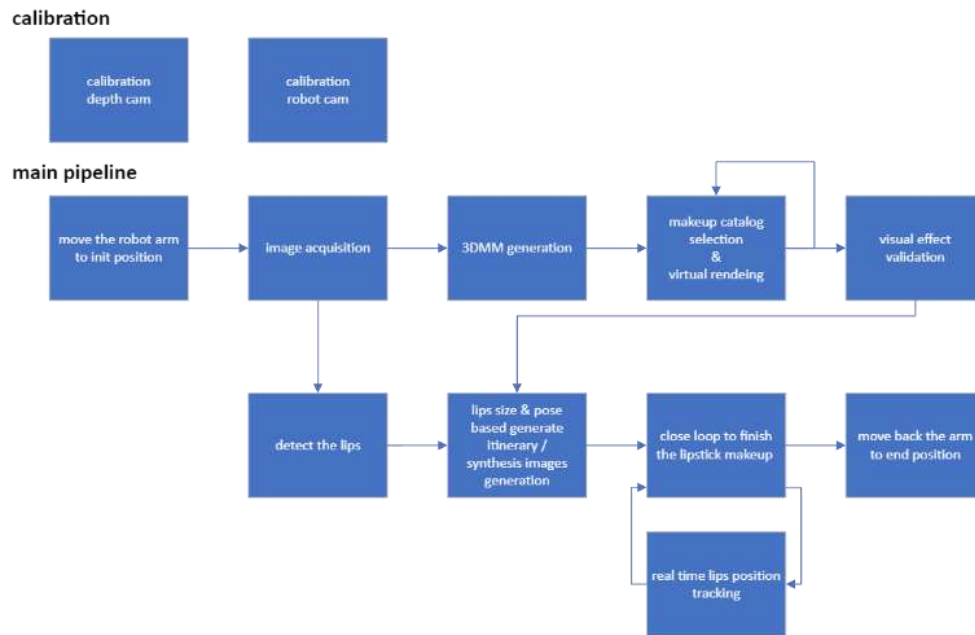


Figure 1 Pipeline de projet mura

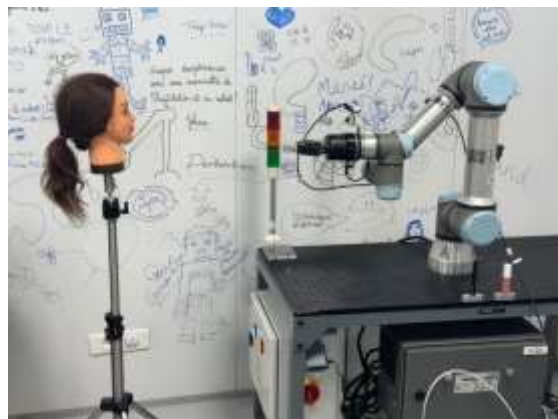


Figure 2 Système de projet mura

Dans notre système, nous avons intégré un robot UR5e, une caméra RealSense D415, une pince Robotiq, ainsi que divers produits de maquillage. Notre système est divisé en trois modules. Si nous tentions de tester tous ces modules simultanément sur un seul robot, cela pourrait entraîner du gaspillage. Par conséquent, nous avons décidé de mettre en place un environnement de simulation pour tester tous les algorithmes associés à notre projet.

3. Construction de l'Environnement Simulation

3.1 Contexte et Problématique

Contexte

Le développement technologique en robotique ces dernières années a fait des robots une composante intégrale de diverses industries et applications. Pour assurer le succès du déploiement et de l'exploitation des systèmes robotiques, il

est crucial de les développer et de les tester dans des environnements réalistes et dynamiques. Cependant, dans le processus de développement, si nous testons constamment un système robotique dans un environnement réel, cela peut entraîner une augmentation des coûts de développement et des risques de dommages au robot. Pour éviter ces problèmes, la création d'un environnement de simulation adapté aux systèmes robotiques est une composante essentielle du processus de développement.

Les environnements de simulation jouent un rôle crucial dans le développement et l'évaluation des systèmes robotiques. Ils offrent un moyen rentable et sûr de prototyper, tester et valider le comportement des robots avant leur déploiement dans des scénarios réels. Les simulations permettent aux chercheurs, aux ingénieurs et aux développeurs de concevoir et d'optimiser des algorithmes pour modéliser la dynamique des robots et d'évaluer leurs performances dans un environnement contrôlé et reproductible.

Dans le domaine de la simulation robotique, plusieurs logiciels populaires sont disponibles, tels que CoppeliaSim, Gazebo, Pybullet et MUJOCO. Pour notre projet, nous avons opté pour Gazebo, qui est un environnement de simulation open-source puissant et largement utilisé pour la recherche et le développement en robotique. Gazebo propose un moteur physique, des simulations de capteurs et des capacités de rendu réalistes. Il permet la simulation de systèmes robotiques complexes, y compris de multiples robots et environnements dynamiques. De plus, Gazebo s'intègre avec ROS (Robot Operating System), ce qui facilite la communication entre les robots simulés et d'autres composants compatibles avec ROS. Gazebo est également hautement extensible, grâce à une large communauté qui contribue à l'élaboration de plugins et de modèles pour divers robots et capteurs.

Problématique

La problématique de la création d'un environnement de simulation robotique pour le déploiement de notre projet Make-Up Robotique Automatique (MURA) englobe plusieurs questions essentielles à résoudre :

- **Création et affichage d'un modèle de robot**
Il s'agit de définir et de créer une représentation numérique du robot qui est suffisamment précise pour reproduire ses mouvements et ses interactions dans l'environnement de simulation. Cela implique de modéliser chaque partie du robot, y compris ses articulations et ses mécanismes d'actionnement. L'interface de simulation doit également être capable d'afficher le modèle de robot de manière visuelle, pour permettre une interaction intuitive avec les utilisateurs.
- **Construction d'un environnement de simulation :**
Un environnement de simulation réaliste doit être créé pour tester le robot. Cela doit inclure tous les éléments avec lesquels le robot interagira,

comme le visage humain sur lequel le maquillage sera appliqué, les accessoires de maquillage, et tout autre élément de l'environnement qui pourrait affecter la performance du robot. L'environnement doit également être dynamique, permettant de simuler différentes conditions d'éclairage, mouvements du visage, et différents types de peau.

- Mise en œuvre de l'interaction entre le modèle de robot et l'environnement de simulation :

Il est crucial de programmer le modèle de robot pour qu'il puisse interagir de manière réaliste avec l'environnement de simulation. Cela signifie qu'il doit être capable de percevoir l'environnement, d'effectuer des actions, et de recevoir des feedbacks sur ces actions. Par exemple, le robot doit pouvoir détecter la position et la forme du visage, appliquer le maquillage avec la bonne pression et précision, et recevoir un feedback visuel et tactile sur le résultat. La simulation doit également être capable de reproduire les effets de ces actions sur l'environnement, par exemple comment le maquillage est appliqué sur le visage.

Ces défis doivent être abordés en tenant compte des contraintes de performances et de ressources, et en veillant à ce que l'environnement de simulation soit suffisamment flexible et évolutif pour permettre des améliorations et des adaptations futures.

3.2 Méthodes utilisées pour répondre à la problématique

Contexte

Dans le cadre de notre projet de robot de maquillage automatique, nous nous sommes basés sur ROS, le Robot Operating System. ROS est un cadre flexible conçu pour faciliter le développement de logiciels pour les robots. Il offre un ensemble de bibliothèques et d'outils logiciels qui aident à la construction d'applications robotiques. La plateforme permet un partage et une réutilisation efficaces du code, ce qui facilite le travail des développeurs. Elle comprend également une variété de bibliothèques logicielles pour diverses fonctionnalités robotiques allant de la planification de trajectoires à la perception, en passant par le contrôle. De plus, ROS est compatible avec une large gamme de systèmes robotiques et de capteurs.

Pour ce projet, nous utilisons trois logiciels principaux : Gazebo, Rviz et Moveit.

- Gazebo est un simulateur dynamique 3D. Il est conçu pour afficher des modèles de robots et pour créer des environnements de simulation précis et efficaces. Ces environnements simulent des robots dans des contextes intérieurs et extérieurs complexes. À l'instar des moteurs de jeux qui fournissent des simulations visuelles d'haute-fidélité, Gazebo offre des simulations physiques d'haute-fidélité, un ensemble complet de modèles

de capteurs et une interaction conviviale pour l'utilisateur et le programmeur.



Figure 3 GAZEBO

- Rviz ou ROS Visualisation Tool, a pour objectif principal de visualiser les messages ROS en trois dimensions, ce qui permet une représentation visuelle des données. Par exemple, il peut afficher des modèles de robots, exprimer la distance entre un capteur et un obstacle, ou visualiser des données de nuages de points provenant de capteurs de distance 3D comme RealSense, Kinect ou Xtion, ainsi que les valeurs d'images provenant de caméras.



Figure 4 Ros Visualisation Tool (Rviz)

- MoveIt est un logiciel libre destiné à la planification de mouvements et aux tâches de manipulation en robotique. Il fournit un ensemble complet d'outils, de bibliothèques et d'API pour la manipulation du bras du robot, la planification de la trajectoire, le contrôle des collisions et l'exécution de la trajectoire. MoveIt est largement utilisé dans la communauté de la robotique pour simplifier le développement et le déploiement de systèmes robotiques. Il est conçu pour fonctionner avec une grande variété de plates-formes robotiques, soutenant les robots industriels et de recherche, contrôlant différents types de bras robotiques, y compris des manipulateurs sériels et parallèles, des robots mobiles et même des robots humanoïdes.



Figure 5 Moveit Motion Planning application

Modélisation de robot et gripeur

- Création de Robot model
Dans le cadre du projet MURA, nous avons utilisé un robot UR5e et une pince Robotiq 2F-85. Par conséquent, en fonction de ces deux types d'équipements, nous avons créé des fichiers URDF pour décrire les propriétés du robot et de la pince.

L'URDF (Unified Robot Description Format) est un format basé sur le langage XML qui permet de décrire les propriétés de la structure d'un robot. Il permet de définir les liens et les articulations du robot, d'ajouter des propriétés physiques et d'intégrer différents plugins pour les capteurs.



Figure 6 Robotiq Gripper 2F-85



Figure 7 UR5e collaborative robot

L'URDF permet de décrire un robot en termes d'éléments tels que :

- Les liens (links) : Ce sont les pièces rigides du robot. Par exemple, dans un bras robotique, chaque segment du bras serait un lien.
- Les articulations (joints) : Ce sont les connexions mobiles qui relient les liens entre eux. Les articulations peuvent avoir différents types de mouvements (par exemple, pivotant ou glissant).
- Les capteurs : Les informations sur les capteurs du robot peuvent également être incluses dans l'URDF. Par exemple, un robot pourrait avoir des capteurs de vision ou de force.
- Les propriétés physiques : Il est également possible d'inclure des informations sur les propriétés physiques des liens, telles que leur masse et leur centre de gravité.

En effet, l'utilisation de l'URDF nous a évité de faire des mesures manuelles. Nous avons cherché les sources ouvertes en ligne qui nous fournissaient les URDF du UR5e et du gripper Robotiq. Nos tâches principales ont donc été les suivantes :

- La création d'un fichier Xacro commun
Xacro (XML Macros) est un langage de prétraitement XML pour les fichiers URDF qui permet de créer des modèles de robots plus lisibles et maintenables en définissant et en utilisant des macros qui encapsulent des modèles URDF. Ce processus facilite la gestion des fichiers URDF, en particulier pour les grands modèles de robots.
Dans notre projet, nous avons deux fichiers URDF, l'un pour le UR5e (ur5e.xacro) et l'autre pour le Robotiq 2F-85 (robotiq_2F_85.xacro). Nous avons créé un nouveau fichier xacro pour les regrouper, puis nous avons importé ces deux fichiers dans ce nouveau fichier.
Pour simuler la connexion physique entre ces deux objets, nous avons également défini un "coupler" dans notre fichier xacro. Nous avons inscrit

les propriétés physiques de se coupler, y compris les caractéristiques des liens (links) et des articulations (joints). En définissant ce lien, nous avons pu connecter efficacement la pince au robot dans notre simulation.

Cela signifie que dans notre fichier xacro, nous avons défini comment le Robotiq 2F-85 est physiquement connecté au UR5e, en spécifiant où et comment ils sont attachés. En regroupant ces deux composants dans un seul fichier xacro, nous avons créé une représentation complète de notre système robotique qui peut être facilement utilisée et modifiée dans notre simulation.

- L'ajout de plugins Gazebo

Une fois la description des propriétés physiques du robot terminée, il nous a fallu ajouter des plugins ainsi que définir les propriétés de collision et d'inertie. En effet, comme Gazebo est un environnement de simulation, la détection des collisions est une caractéristique nécessaire. La définition des propriétés de collision fournit la base de la détection des collisions dans la simulation.

De plus, l'ajout de l'étiquette "inertielle" définit la matrice d'inertie d'une partie rigide du robot, qui est utilisée dans certaines simulations liées à la mécanique. Cela nous permet de simuler plus précisément le comportement dynamique du robot.

Nous avons ajouté un plugin de contrôle pour le Robotiq Gripper 2F-85. Ce plugin permet de contrôler le gripper à travers notre système logiciel ou notre environnement de simulation. Dans notre configuration, nous avons défini deux liens du gripper ("gripper_link") et un lien représentant la paume ("palm_link"). Ces liens spécifient les composants du gripper dans le simulateur Gazebo.

Enfin, nous avons ajouté un capteur de force-couple au poignet du robot (au joint 3). Ce capteur nous permet de mesurer les forces et les couples appliqués au poignet du robot, ce qui est essentiel pour réaliser des tâches de manipulation délicates, comme l'application de maquillage. Cela nous permet également de nous assurer que notre robot n'applique pas une force excessive lors de l'interaction avec l'utilisateur, garantissant ainsi la sécurité de ce dernier.

- Création de fichiers de lancement (launch files) :

Une fois toutes les propriétés du robot correctement définies et les plugins nécessaires ajoutés, nous devons créer des fichiers de lancement, aussi appelés "launch files". Ce sont des fichiers XML qui permettent de démarrer les nœuds ROS de manière organisée et coordonnée.

Dans notre projet, nous avons créé un fichier de lancement pour démarrer le nœud de simulation Gazebo avec notre robot UR5e et le gripper Robotiq. Ce fichier de lancement charge le modèle URDF du robot dans le paramètre du serveur ROS et lance le nœud de simulation Gazebo. Il lance également le nœud de contrôle du robot pour que nous puissions envoyer des commandes au robot depuis notre code ROS.

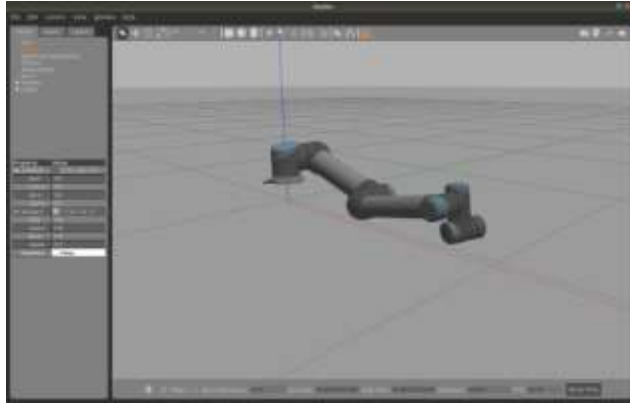


Figure 8 UR5e simulation model

Construction du modèle de Realsense caméra D415

Dans le cadre de notre projet, le module de vision par ordinateur est d'une importance cruciale. Par conséquent, l'intégration d'au moins une caméra à notre environnement de simulation s'avère essentielle. Nous avons opté pour le modèle de caméra RealSense D415, capable de fournir des images en couleur (RGB) et des images de profondeur.

Afin d'intégrer le RealSense à ROS, il a été nécessaire d'installer certaines dépendances. Suite à cela, nous avons conçu un fichier xacro pour notre module de caméra. Nous avons employé le plugin de caméra RGBD de Gazebo, inspiré du modèle de caméra Kinect, en ajustant uniquement les performances de la caméra pour les aligner sur celles du RealSense D415. Par ailleurs, nous avons incorporé un bruit gaussien à nos images pour simuler des conditions réalistes et augmenter la robustesse de notre système.

Après avoir défini les propriétés de la caméra, nous l'avons intégrée à notre fichier Xacro principal en créant des articulations pour la connecter au robot. Cette intégration nous offre une perspective de la caméra qui reflète ce que le robot perçoit dans la simulation, ce qui est crucial pour le succès de nos tâches de maquillage automatique.

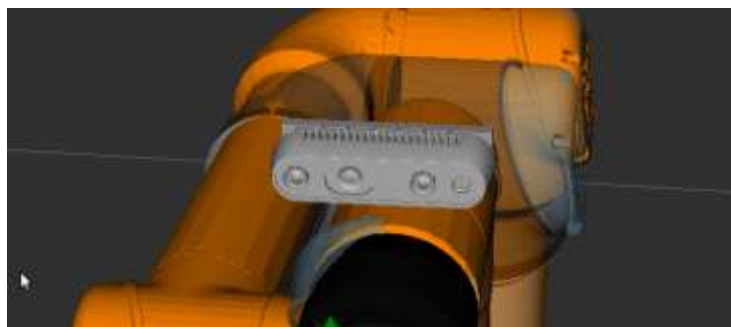


Figure 9 Des caméras module de RealSense D415

Construction de force capteur

Selon les besoins du projet, nous souhaitons mettre en œuvre un contrôle de force, par conséquent, un capteur de force-torque est nécessaire pour notre

environnement de simulation. Dans notre cas, nous avons ajouté un capteur de force au niveau du poignet 3. Il peut nous fournir une sortie de force et de couple dans les directions x, y, z.

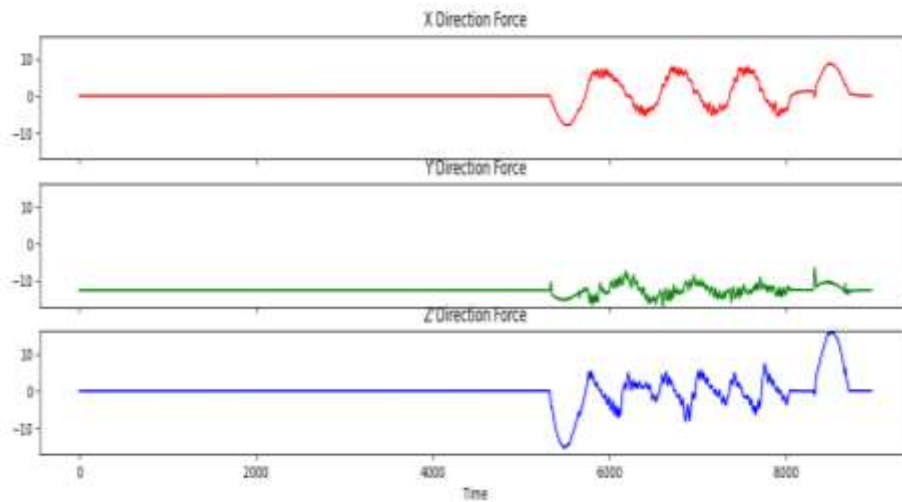


Figure 10 Force capteur

Construction de gezebo world

Lorsque nous avons fini de construire le modèle de robot, d'intégrer la caméra RGBD et les capteurs de force-couple, nous disposons d'un système de robot. Il est alors temps de commencer à construire un espace de travail pour installer notre robot. En fonction de notre espace de travail réel, nous avons créé un fichier "world" pour décrire notre espace de travail.

Dans cet espace, nous avons une table sur laquelle installer le robot, ainsi qu'un petit cylindre pour représenter un rouge à lèvres. De plus, nous avons écrit un fichier SDF (Simulation Description Format) pour ajouter un modèle 3D d'un être humain. Ce modèle humain a son propre maillage et squelette 3D, ainsi qu'une texture pour décrire son visage.

Avec ces éléments, nous pouvons créer un environnement de simulation réaliste pour notre robot de maquillage automatique. Cela nous permet de tester notre robot dans une variété de situations, tout en s'assurant qu'il fonctionne correctement avant de le déployer dans un environnement réel. C'est un élément essentiel du processus de développement de notre robot, car il nous permet de résoudre tout problème potentiel avant le déploiement réel.

En plus des autres éléments de l'environnement de simulation, nous avons également ajouté un chess board. Il est souvent utilisé en vision par ordinateur pour calibrer les caméras. Le tableau de calibration est généralement imprimé avec un motif spécifique, souvent une grille d'échiquier, qui permet d'aligner précisément la caméra et d'ajuster ses paramètres internes.

La calibration de la caméra est une étape cruciale dans la préparation de notre système de vision par ordinateur, car elle permet de corriger les distorsions d'image dues à l'objectif de la caméra et d'obtenir des informations précises sur la position et l'orientation des objets dans l'image. Une fois la caméra calibrée, nous pouvons utiliser cette information pour obtenir une image 3D précise du monde qui peut être utilisée par notre robot pour interagir efficacement avec son environnement.

En ajoutant cette boîte de calibration dans notre simulation, nous pouvons vérifier et ajuster les paramètres de calibration de notre caméra avant de la déployer dans le monde réel. Cela nous aide à assurer que notre système de vision par ordinateur fonctionnera de manière précise et efficace dans la réalité.



Figure 11 Espace de travail pour MURA projet

Création une configuration MoveIt pour l'UR5 et un gripeur

L'intégration de MoveIt! à notre projet a permis d'ajouter des fonctionnalités de planification de mouvements complexes à notre robot. MoveIt! est un logiciel puissant qui offre une interface de haut niveau pour la planification de mouvements, la manipulation d'objets, la perception 3D, la cinématique, le contrôle et la navigation. Il est capable de gérer la cinématique inverse et avancée, la détection de collision, et de nombreux autres aspects importants du contrôle de mouvement robotique.

Dans le cadre de notre projet, nous avons d'abord importé l'URDF que nous avons créé pour notre robot dans MoveIt. Cela a permis à MoveIt! de générer une configuration pour notre robot spécifique. Avec cette configuration, nous avons été capables de vérifier l'état des collisions entre les différentes parties de notre robot, ce qui est essentiel pour prévenir les accidents potentiels.

Nous avons ensuite défini des groupes de planification pour notre robot, allant du lien de base ("base_link") au lien d'extrémité effecteur ("ee_link"). Ces

groupes de planification permettent à MoveIt ! de planifier et de contrôler les mouvements de différentes parties de notre robot.

Nous avons également ajouté un groupe pour le gripper, nous permettant de contrôler ses mouvements de manière précise.

Ensuite, nous avons défini la position initiale de notre robot comme la position de départ pour toutes nos planifications de mouvement. De plus, nous avons ajouté des définitions pour les états d'ouverture et de fermeture de notre gripper.

Pour terminer, nous avons défini les joints passifs pour notre gripper et ajouté les contrôleurs de joints nécessaires. Enfin, nous avons généré le package de configuration pour notre robot, ce qui nous a permis d'implémenter un contrôle précis et complexe de notre robot à travers MoveIt !



Figure 12 Configuration de robot Moveit Package

3.3 Résultats obtenus

Une fois la construction de l'environnement de simulation achevée, nous avons créé un environnement de base. Dans cet environnement, un robot UR5E équipé d'un préhenseur Robotiq est installé sur une table. Sur cette même table se trouve un modèle de rouge à lèvres placé à côté du robot. De plus, notre robot est équipé de deux caméras : l'une est intégrée au préhenseur, tandis que l'autre est fixée à la tête du robot. En ce qui concerne notre modèle humain, nous disposons deux modèles d'humain, tous deux minutieusement conçus avec des propriétés de collision adéquates.

4. Robot contrôle et camera calibration

4.1 Contexte et Problématique

Contexte

À la suite de la création de notre environnement de simulation, les étapes suivantes consistent à contrôler notre robot afin qu'il réalise plusieurs mouvements simples. Par exemple, effectuer un contrôle de la cinématique

inverse, de la cinématique directe, ainsi que des actions de préhension et de dépose (picking-place). De plus, nous avons planifié diverses trajectoires adaptées au robot UR5, notamment des trajectoires linéaires et courbées, exécutées à vitesse constante. Pour ce faire, nous avons dû définir différentes méthodes de planification de mouvement, telles que "move_j", "move_p" et "move_l", afin de répondre aux besoins spécifiques de notre robot UR5.

Par ailleurs, nous avons intégré des caméras pour fonctionner en tant que système de servo-vision. Pour garantir le bon fonctionnement de ce système, une calibration de la caméra ainsi qu'une calibration œil-main sont nécessaires. Ces calibrations nous permettront d'acquérir des informations 3D précises et correctes.

Problématique

1. Le partie robot contrôle

Après avoir généré la configuration de notre modèle de robot, nous avons pu obtenir les groupes de planification en utilisant MoveIt. L'outil Rviz nous a permis de diriger le robot afin qu'il atteigne les poses que nous avons prédéfinies. Bien que nous ayons eu la possibilité de définir des objectifs via l'interface de Rviz, cette méthode ne nous a pas permis d'obtenir un contrôle suffisamment précis. Par conséquent, les défis que nous avons dû relever sont les problèmes suivants

- Comment utiliser l'API de MoveIt pour réaliser un contrôle cinématique de robot c'est-à-dire comment piloter le robot à l'aide de code en Python ou en C++, afin de réaliser un contrôle par cinématique directe et un contrôle par cinématique inverse.
- Comment effectuer un contrôle du préhenseur pour ouvrir et fermer l'effecteur final.
- Comment réaliser un évitement de collision, c'est-à-dire, comment ajouter des objets de collision dans MoveIt.
- Comment réaliser une planification de mouvement, c'est-à-dire comment élaborer une planification de trajectoire. Lorsque nous pilotons un robot, nous souhaitons que notre robot se déplace selon une trajectoire prédéfinie avec un vitesse constant et une accélération constante.

2. Le partie robot vision servo

Dans notre environnement de simulation, nous avons équipé les caméras pour des robots. Cependant, la question est de savoir comment exploiter efficacement ces informations pour notre robot. Pour que le robot puisse obtenir des informations correctes et utiles, nous devons résoudre les problèmes suivants

- Effectuer une calibration de la caméra : Cette étape est cruciale pour assurer que les images capturées par la caméra correspondent précisément à l'environnement réel. Elle consiste à déterminer les paramètres internes (comme la distance focale) et externes (comme la position et l'orientation) de la caméra.

- Effectuer une calibration hand-eye du robot et de la caméra : Cette étape vise à déterminer la relation spatiale entre la caméra et le préhenseur du robot (ou "main"). Cela permet au système de savoir exactement où se trouve l'objet saisi par le préhenseur par rapport à la caméra, et inversement. Cette calibration est essentielle pour permettre au robot d'interagir de manière précise avec son environnement en se basant sur les informations visuelles recueillies.

4.2 Méthodes utilisées pour répondre à la problématique

Contexte

1. Robot cinématique

Dans le domaine du contrôle robotique, nous avons deux types principaux de méthodes de contrôle : l'une est appelée cinématique directe, l'autre est connue sous le nom de cinématique inverse. Dans notre projet, nous avons utilisé ces deux types de méthode.

- **Cinématique Directe**

La cinématique directe est un processus qui permet de déterminer la position et l'orientation de l'effecteur final (la partie du robot qui interagit directement avec l'environnement, comme une pince ou une "main") en fonction des valeurs de ses articulations.

En termes plus simples, étant donné les angles de chaque articulation du robot, la cinématique directe nous permet de calculer la position et l'orientation exactes de l'effecteur final dans l'espace de travail. C'est donc une fonction qui prend les angles des articulations en entrée et donne la position de l'effecteur final en sortie.

$$Input = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6]$$

Équation 1 Input de cinématique direct

- **Cinématique Inverse**

La cinématique inverse, comme son nom l'indique, est le processus inverse de la cinématique directe. Au lieu de calculer la position de l'effecteur final à partir des angles des articulations, la cinématique inverse calcule les valeurs requises des angles des articulations pour que l'effecteur final atteigne une position et une orientation spécifiques.

Cette tâche est généralement plus complexe que la cinématique directe, car il peut y avoir plusieurs solutions possibles (plusieurs configurations d'articulations qui permettent d'atteindre la même position de l'effecteur final), ou aucune solution du tout, si la position cible est hors de la portée du robot.

$$Input = [x_{end}, y_{end}, z_{end}, R_{end}, P_{end}, Y_{end}]$$

Équation 2 Input de cinématique inverse

Ces deux types de cinématiques sont essentiels dans la programmation et le contrôle des robots pour effectuer des tâches précises et complexes.

2. Robot trajectoire planning

En effet, une fois que nous sommes capables de réaliser un contrôle cinématique inverse ou direct, nous devons définir les différents types de mouvements pour notre robot. Dans le cas pratique, et plus particulièrement pour les robots UR, il y a quatre types principaux de planification de mouvements : DéplacementJ, DéplacementL, DéplacementP et DéplacementC. Dans notre cas, nous nous concentrerons uniquement sur les trois premiers.

- **DéplacementJ (Joint Motion)**

Le DéplacementJ, ou mouvement des joints, est un type de mouvement où chaque articulation du robot bouge indépendamment des autres pour atteindre la position cible. Cela signifie que les articulations peuvent atteindre leur position finale à des moments différents. Ce mode de mouvement est souvent utilisé lorsque la trajectoire précise de l'effecteur final n'est pas importante, mais seulement sa position et son orientation finales.

- **DéplacementL (Linear Motion)**

Le DéplacementL, ou mouvement linéaire, est un type de mouvement où l'effecteur final du robot se déplace le long d'une ligne droite entre sa position initiale et sa position finale. Pour réaliser cela, toutes les articulations du robot doivent bouger simultanément et coordonnées. Ce mode de mouvement est souvent utilisé lorsque la trajectoire de l'effecteur final est importante, comme lors de la manipulation d'objets délicats ou de l'assemblage de pièces.

- **DéplacementP**

Le DéplacementP, ou mouvement de trajectoire, est une commande qui déplace l'outil du robot de manière linéaire à une vitesse constante. La trajectoire est lissée par des arcs de cercle à chaque coin, ce qui garantit une vitesse constante tout au long du mouvement. Cela est particulièrement utile pour les opérations de processus tels que le collage ou la distribution où la vitesse constante et le mouvement lisse sont nécessaires pour garantir la qualité du travail.

3. Camera calibration

Après avoir établi la planification des mouvements de notre robot, nous sommes maintenant capables de réaliser des mouvements plus complexes. Par conséquent, il est approprié d'aborder maintenant le système de servo vision. La vision par ordinateur constitue une composante majeure de notre projet. La caméra doit nous fournir des informations précises. C'est pourquoi nous devons effectuer une calibration de celle-ci avant d'exploiter ses informations. Cette calibration a pour but de déterminer les caractéristiques intrinsèques de la caméra, telles que la matrice intrinsèque de la caméra et les matrices de distorsion. Nous avons basé notre calibration sur le modèle de caméra sténopé (ou "pinhole"). Ce modèle est une simplification largement utilisée pour représenter les systèmes de caméras en vision par ordinateur. Selon ce modèle, tous les rayons de lumière passant par un point de l'objet

traversent un même point, le "trou d'épingle", avant d'être projetés sur le plan de l'image. Cela permet de représenter la projection de l'espace 3D sur le plan de l'image en utilisant des transformations linéaires, simplifiant ainsi les calculs.

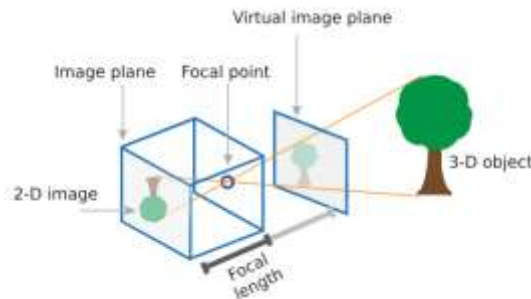


Figure 13 Modèle de caméra sténopé (Pinhole Camera Model)

Bien qu'il s'agisse d'une approximation (car il ne prend pas en compte des facteurs tels que la diffraction ou les aberrations optiques), ce modèle est généralement assez précis pour la plupart des applications en vision par ordinateur. Il est souvent combiné avec un modèle de distorsion pour corriger les déformations induites par l'objectif de la caméra. Dans ce modèle, les paramètres sont représentés par une matrice 3*4, qui projette les informations 3D des coordonnées du monde aux coordonnées de l'image.

$$w \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Scale factor Image points World points

$$P = K \begin{bmatrix} R & t \end{bmatrix}$$

Camera matrix Intrinsic matrix Extrinsic Rotation and Translation

Figure 14 Transformation du 2D au 3D

Grâce à la calibration de la caméra, nous pouvons estimer les paramètres intrinsèques (tels que la distance focale et le point principal) et extrinsèques (comme l'orientation et la position de la caméra) qui définissent ce modèle de projection. Ces paramètres sont essentiels pour mener à bien des tâches telles que la reconstruction 3D, la détection d'objets ou la navigation basée sur la vision. Les paramètres de ce modèle sont les suivants :

Nous avons utilisé l'algorithme de calibration calcule le camera matrix utilisant le matrix extrinsèque et matrix intrinsèque. Les paramètres extrinsèques présentent une transformation rigide du 3D coordonné du monde au 3D coordonné du caméra. Les paramètres intérieurs présentent une transformation de projection du 3D caméra coordonné au 2D image coordonné.

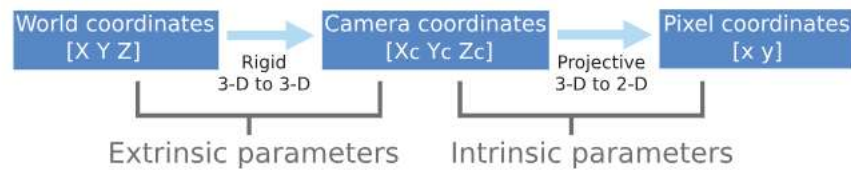


Figure 15 Relations de transformation de matrix de caméra

- **Paramètres extrinsèques**

Les paramètres extrinsèques consistent en une rotation, R , et une translation, t . L'origine du système de coordonnées de la caméra se situe en son centre optique et ses axes x et y définissent le plan de l'image.

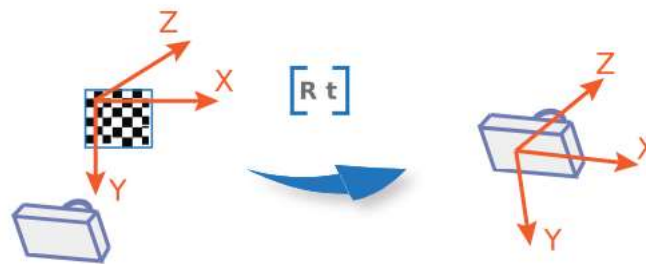


Figure 16 Transformation extrinsèque et la présentation de coordonné extrinsèque

- **Paramètres intrinsèques**

Le paramètre intrinsèque contient la longueur de focal, le centre optique et le coefficient de distorsion

$$\begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Équation 3 Matrix de paramètre intrinsèques

f_x et f_y sont les longueurs focales en pixels de la caméra dans les directions x et y . s est le coefficient de distorsion (habituellement nul pour les caméras modernes). c_x et c_y sont les coordonnées du point principal (habituellement le centre de l'image). Cette matrice est utilisée pour transformer des coordonnées de points dans le monde réel en coordonnées de points sur l'image de la caméra.

- La distorsion dans caméra calibration

La matrice de la caméra ne tient pas compte de la distorsion car une caméra idéale à sténopé n'a pas d'lentille. Pour représenter de manière précise une véritable caméra, le modèle de la caméra inclut la distorsion radiale et tangentielle de l'objectif.

La distorsion radiale se manifeste principalement sous forme de courbure de barillet ou de coussin, faisant que les lignes droites apparaissent courbes sur l'image. Elle est due au fait que plus un point est éloigné du

centre optique de l'objectif, plus il est agrandi. Elle est généralement caractérisée par trois coefficients (k_1 , k_2 , k_3) dans un modèle de distorsion radiale.



Figure 17 Distorsion radiale

La distorsion tangentielle est due à l'alignement non parfait de l'objectif par rapport au point de vue de l'image. Elle provoque un effet de "décalage" dans l'image qui peut généralement être corrigé à l'aide de deux coefficients (p_1 , p_2).

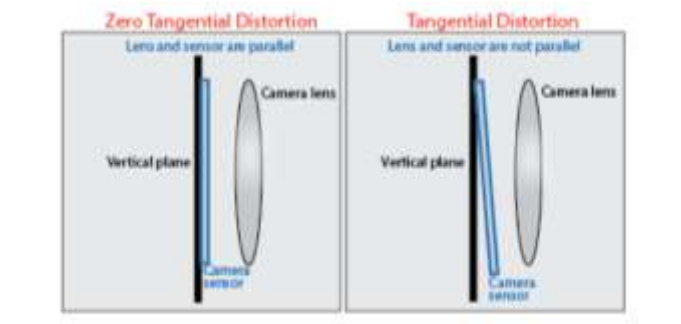


Figure 18 Distorsion Tangentielle

En résumé, pour obtenir un modèle de caméra précis, à la fois les paramètres intrinsèques (focal, point principal, skew) et la distorsion de l'objectif (radiale et tangentielle) doivent être pris en compte.

4. Hand-Eye calibration

Hand-eye calibration est utilisé pour relier ce que la caméra voit à l'endroit où le robot se déplace. Dans hand-eye calibration, nous avons deux genres de méthode de l'installation l'un Eye-in-hand, l'autre Eye-to-hand.

- Le eye-in-hand calibration est un processus permettant de déterminer la position et l'orientation relatives d'une caméra montée sur un robot par rapport à l'effecteur final du robot. Elle se fait généralement en capturant un ensemble d'images d'un objet statique de géométrie connue avec le bras du robot situé dans un ensemble de positions et d'orientations différentes.

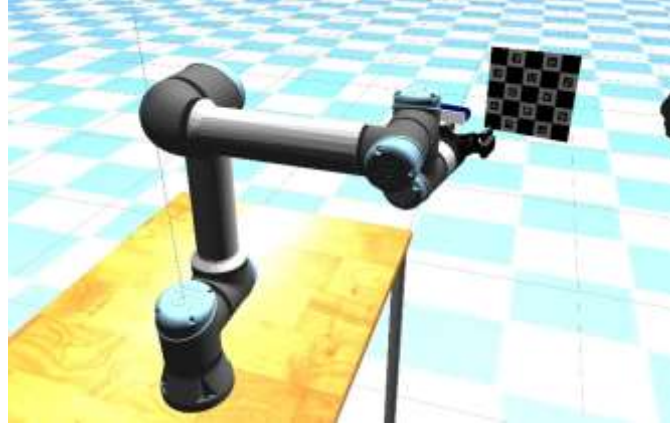


Figure 19 Modèle eye in hand

Dans ce modèle la relation entre la base du robot et la chesse board Les quantités résolues sont les relations de position entre la caméra et le système de coordonnées du robot. On a une équation

$$T_{EE2}^{ROB} \times T_{CAM}^{EE2} \times T_{OBJ}^{CAM2} = T_{EE1}^{ROB} \times T_{CAM}^{EE1} \times T_{OBJ}^{CAM1}$$

Après avoir déplacé des éléments vers la gauche et la droite

$$T_{EE1}^{ROB-1} \times T_{EE2}^{ROB} \times T_{CAM}^{EE2} = T_{CAM}^{EE1} \times T_{OBJ}^{CAM1} \times T_{object}^{CAM2-1}$$

Dans cette équation on sait la position de gripeur (End) dans coordonné de robot base, on sait aussi la position d'objet s dans coordonné de caméra, par conséquence, on peut obtenir une équation comment suivant :

$$A * X = X * B$$

Donc, l'objet de hand-eye calibration est résoudre le problème de $A * X = X * B$.

Robot cénématique et robot motion planning

- Robotique Cinématique

Pour réaliser un contrôle robotique cinématique, nous nous sommes basés sur MoveIt. MoveIt est une bibliothèque logicielle employée pour la planification de mouvements et le contrôle de robots. Elle propose des API en C++ et en Python. En se basant sur MoveIt, nous pouvons facilement réaliser un robot avec cinématique directe et cinématique inverse.

Cinématique Inverse : Pour la cinématique inverse, nous avons exploité les configurations prédéfinies de MoveIt afin de commander le robot. Nous avons utilisé le format de message ROS Pose pour définir les informations cibles en 3D. Dans ce contexte, le format d'entrée est la position (x, y, z) et le quaternion (x, y, z, w). Après avoir défini la cible, nous avons intégré cette pose à la fonction "set_pose_target" de MoveIt, puis utilisé la commande "plan ()" pour effectuer la cinématique inverse.

Cinématique Directe : En ce qui concerne la cinématique directe, la procédure est plutôt simple. Nous définissons simplement les angles désirés, puis nous utilisons la fonction "set_joint_value_target". Ensuite, en utilisant la commande "get_current_pose", le robot atteindra la position cible.

Pour la cinématique robotique, MoveIt offre une API simplifiée pour réaliser un contrôle facile.

- Robot motion planning

Dans le domaine de la planification des mouvements robotiques, nous aspirons à mettre en œuvre des méthodes de mouvement qui se distinguent de celles utilisées par les robots conventionnels. À cet effet, nous avons opté pour l'algorithme RRT (Rapidly-exploring Random Tree) au sein de MoveIt. Cet algorithme est capable de générer des trajectoires à la fois sûres et efficaces pour le robot. De plus, en nous appuyant sur MoveIt, nous pouvons définir directement l'accélération et la vitesse du robot. Plus spécifiquement, dans notre cas, nous avons défini les fonctions `move_J`, `move_P` et `move_L` pour le projet MURA. En outre, nous avons également défini des limites de collision pour assurer une sécurité accrue dans les mouvements du robot.

- **Move_J** : Concernant `Move_J`, nous nous sommes basés sur la cinématique directe. Nous fournissons simplement les valeurs d'angle pour chaque articulation, et le robot se déplace vers la position cible à une vitesse et une accélération constante.

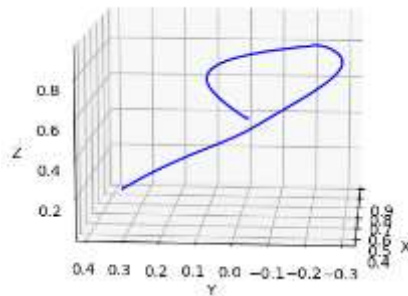


Figure 20 Trajectoire Move_J

- **Move_P** : En ce qui concerne `Move_P`, il s'agit d'un mouvement de position basé sur la cinématique inverse. Nous fournissons simplement les informations de position cible, et le robot suit une trajectoire sûre et efficace jusqu'à ces points à une vitesse constante.

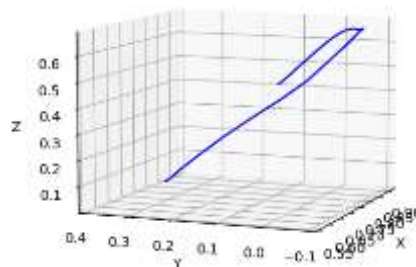


Figure 21 Trajectoire de Move_P

- **Move_L** : Pour `Move_L`, il s'agit également d'un mouvement de position basé sur aussi la cinématique inverse. Dans cette commande, nous pouvons fournir non seulement un point cible, mais aussi plusieurs

waypoints. Le système fait ensuite plusieurs itérations pour générer une trajectoire linéaire à une vitesse et une accélération constante.

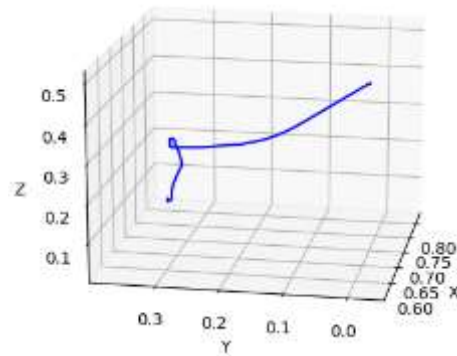


Figure 22 Trajectoire de Move_L

- **Collision** : Quand on a réalisé un motion planning, surtout pour la move_P, on a découvert qu'il y a des motion dangereux et indésirable. Et aussi pour notre simulation similaire notre env réelle, nous avons ajouté des collisions scène, En bas de robot, nous avons ajouté une scène 0.8×1.2 pour simuler la table, une même taille de scène pour simuler le mur, une scène en tête de robot pour simuler le plafond.

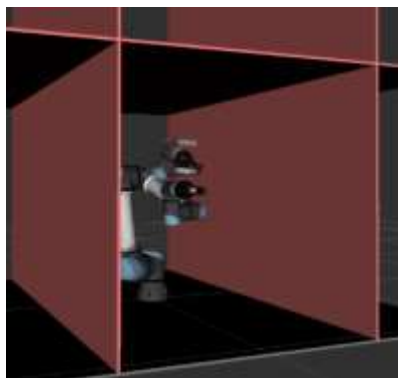


Figure 23 Ccollision scène définis

Camera calibration et hand-eye calibration

Pour la calibration de la caméra et la calibration "eye in hand" (œil dans la main), dans notre projet, nous avons combiné les deux afin d'acquérir à la fois les propriétés de la caméra et les informations sur la relation entre la caméra et le grippier. Nous avons utilisé un outil de calibration appelé ArUco. Dans notre simulation, nous avons créé un tableau ArUco comme référence de coordonnées.

Le processus de calibration se déroule en plusieurs étapes :

1. Localisation du tableau ArUco : L'image capturée par la caméra est utilisée pour localiser le tableau ArUco. En détectant les marqueurs ArUco, nous pouvons déterminer sa position et son orientation dans l'espace.

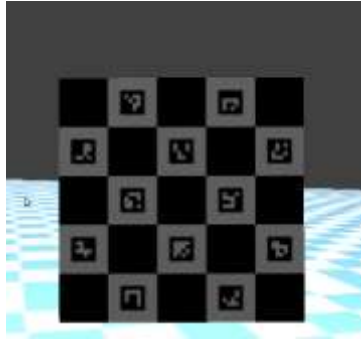


Figure 24 Tableau de ArUco

2. Extraction des caractéristiques : Les marqueurs ArUco détectés nous permettent d'extraire les caractéristiques nécessaires à la calibration, telles que les positions des coins du tableau et les identifiants des marqueurs.

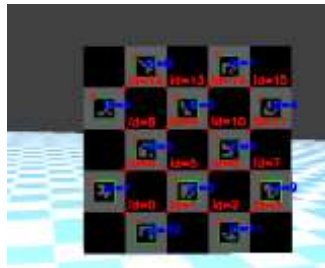


Figure 25 Extraction des caractéristiques

Acquisition d'images à partir de différentes positions du préhenseur : Pour réaliser une calibration de hand-eye qui est crucial pour un contrôle précis du robot, nous avons modifié la position du préhenseur afin d'obtenir des images sous divers angles. Ce processus a été répété 10 fois pour garantir un étalonnage précis et recueillir une quantité suffisante de données visuelles. Ces données nous permettent d'affiner la relation spatiale entre la caméra (l'œil) et le préhenseur (la main), facilitant ainsi un contrôle plus précis du robot.



Figure 26 les différents de pose

3. Calcul des paramètres de calibration : En utilisant les caractéristiques extraites, nous pouvons calculer les paramètres de calibration de la

caméra, tels que la matrice intrinsèque (focale, centre optique) et la distorsion radiale.

4. Calibration "eye in hand" : Une fois que les paramètres de calibration de la caméra sont obtenus, nous procédons à la calibration "eye in hand". Cela implique de déterminer la transformation entre la caméra et le gripeur. Nous utilisons les mouvements précis du gripeur réalisés 10 fois, ainsi que les relevés de pose de la caméra, pour calculer cette transformation.
5. Validation de la calibration : Afin de vérifier la précision de la calibration, nous utilisons des uns points mesurés par moi-même. Nous effectuons des mesures avec le gripeur dans des positions connues et comparons les coordonnées calculées avec les coordonnées réelles.

4.3 Résultats obtenus

Après avoir terminé le développement du contrôleur de robot ainsi que la calibration de la caméra et de la calibration "eye in hand", nous avons obtenu les résultats suivants :

- Nous sommes capables d'effectuer une cinématique inverse et une cinématique directe.
- Nous pouvons réaliser une planification de mouvements générant une trajectoire linéaire ou courbe avec une vitesse constante et une accélération constante.
- Nous avons réussi à éviter des mouvements indésirables pendant que le robot exécute ses tâches.
- Nous avons mis en œuvre un contrôle en temps réel. Lorsque le module de vision par ordinateur détecte les cibles et que ces cibles sont accessibles, notre robot est capable d'exécuter la tâche correspondante.
- Nous avons acquis les propriétés de la caméra, les matrices intrinsèques et la matrice de distorsion.
- Nous avons obtenu une matrice extrinsèque, qui contient la relation de translation et l'orientation entre la caméra et le préhenseur

5. Computer vision et simulation de maquillage

5.1 Contexte et Problématique

Contexte

Notre robot est désormais doté d'une caméra servo, ce qui lui confère la capacité d'acquérir des informations visuelles et de détecter des cibles par l'intermédiaire de cette caméra. La prochaine phase de notre projet consiste à mettre en œuvre une détection de visage afin d'extraire des informations faciales pertinentes. Dans le cadre de notre initiative orientée vers le maquillage, nous envisageons de développer un système de suivi du visage, d'extraction des caractéristiques faciales et de génération de trajectoires pour le robot. En appliquant la cinématique inverse, nous serons en mesure d'atteindre les points de passage nécessaires pour effectuer le maquillage.

Une fois que nous aurons réussi à accomplir cette phase de mouvement robotique, l'étape suivante consistera à évaluer la qualité du maquillage dans notre environnement de simulation. Il existe plusieurs méthodes d'évaluation possibles, comme la comparaison entre les trajectoires théoriques et les trajectoires réelles, ou l'application de couleurs sur notre modèle.

Problématique

Nous sommes actuellement confrontés à plusieurs problématiques :

1. Comment parvenir à une détection précise du visage humain et extraire les informations qui nous intéressent ?
2. Comment déterminer avec exactitude la pose de la tête ?
3. Comment effectuer une transformation de la 2D à la 3D, et passer d'un système de coordonnées à un autre ?
4. Comment contrôler le robot pour qu'il exécute la trajectoire que nous avons définie ?
5. Une fois la trajectoire terminée, comment appliquer la couleur sur la partie du visage que nous souhaitons maquiller ?

5.2 Méthodes utilisées pour répondre à la problématique

Contexte

La détection des visages est un domaine clé de la vision par ordinateur. Bien que de nombreux modèles aient été développés jusqu'à présent pour effectuer efficacement cette détection, notre projet requiert davantage. Nous visons non seulement à détecter les visages, mais aussi à extraire des points caractéristiques sur le visage. Par conséquent, nous avons décidé d'utiliser Media Pipe, un modèle basé sur l'apprentissage profond développé par Google.

Media Pipe nous offre des informations précieuses sur les visages et leurs caractéristiques. Il est capable de nous fournir 468 caractéristiques uniques, y compris des détails comme les contours des yeux, le nez et la bouche. En exploitant ces points caractéristiques et les relations entre eux, nous avons acquis une représentation structurée du visage. Ce mesh nous permet d'effectuer un maquillage virtuel dans notre environnement de simulation.

De plus, lorsque nous combinons MediaPipe avec une caméra RGBD, nous sommes en mesure d'extraire des informations 3D des points caractéristiques. Ces informations nous permettent de déterminer une trajectoire pour notre robot, augmentant ainsi la précision et l'efficacité de notre système de maquillage automatisé.

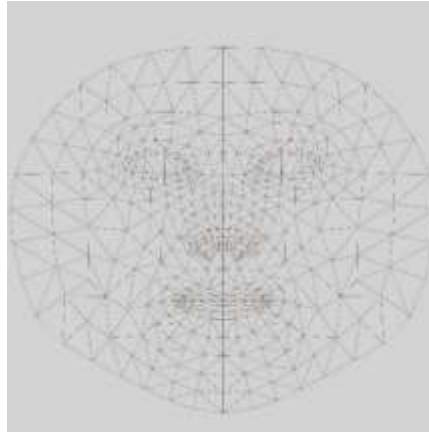


Figure 27 Mediapipe mesh avec 468 points caractéristiques

Configuration de media pipe pour s'adapter le ROS

Dans notre projet, nous nous appuyons sur ROS Melodic, qui est construit sur Python 2.7. Cependant, MediaPipe, l'outil que nous utilisons, nécessite un environnement Python 3. Par conséquent, notre premier défi a été de résoudre la configuration entre MediaPipe et ROS Melodic.

Nous avons utilisé Canda pour configurer notre environnement MediaPipe. Le premier problème dans cet environnement était de savoir comment envoyer les images de Gazebo à MediaPipe. Dans ROS, nous pouvons utiliser des topics pour permettre la communication entre différentes parties, mais cela repose sur différentes formes de messages ROS. Au départ, nous avons utilisé cette méthode, mais nous avons rencontré un sérieux conflit entre Python 2.7 et Python 3 à cause de cv_bridge.

Par conséquent, nous avons décidé de contourner ce problème et avons conçu deux méthodes pour transmettre des images à MediaPipe. L'une d'elles est basée sur la structure de communication serveur-client de ROS. Nous stockons les images et publions leur adresse. Lorsque MediaPipe a besoin d'une nouvelle image, il envoie une requête au ROS master, qui transmet cette demande au topic serveur. Le topic serveur envoie alors l'adresse de l'image à MediaPipe, qui peut ainsi acquérir une image.

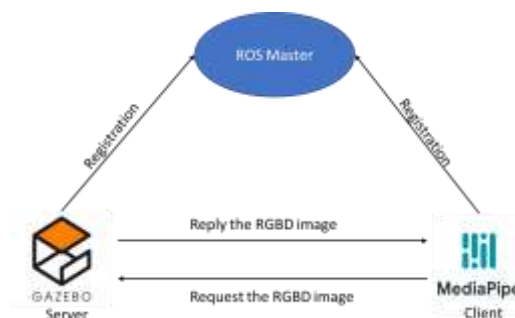


Figure 28 ROS server-client communication

La deuxième solution que nous avons développée consiste à créer une mémoire partagée dans un terminal. Nous avons dimensionné cette mémoire en fonction

de la taille de l'image, lui permettant de stocker temporairement à 10 images. Selon le test on a découvert qu'il a 0.002s retard pour recevoir une image. Nous avons souscrit au topic d'image et avons acquis l'image sous la forme d'un objet cv2. Ensuite, nous avons transmis cette image à la mémoire partagée. Ainsi, MediaPipe peut accéder à cette mémoire pour récupérer les images. Cette solution offre un moyen efficace de résoudre les problèmes de compatibilité entre différentes versions de Python tout en garantissant une communication fluide entre ROS et MediaPipe.



Figure 29 Mémoire partagé pour l'image acquisition



Figure 30 Comparaison de temps de retard

Les deux méthodes que nous avons mises en place permettent de transmettre efficacement les images, mais elles présentent des avantages et des inconvénients distincts.

La méthode client-serveur nécessite une grande quantité de stockage disque pour conserver les images. Son avantage réside dans la rapidité de transmission des images, avec un faible retard. Cela peut être crucial pour des applications nécessitant une réactivité en temps réel. Il est bien adapté le système réel.

D'un autre côté, l'utilisation de la mémoire partagée présente un retard plus important que la méthode client-serveur, mais elle offre un avantage considérable : elle ne nécessite pas la suppression des images conservées à chaque lancement de l'environnement de simulation. Cela peut faciliter la gestion des ressources et améliorer l'efficacité de l'environnement de travail.

Il est donc essentiel de prendre en compte ces considérations lors du choix de la méthode la plus appropriée pour un projet spécifique, en équilibrant les besoins en termes de performances, de ressources de stockage et de facilité de gestion.



Figure 31 Acquisition de l'image RGB et l'image depth

Acquisition de l'information 3d de visage dans la coordonné de caméra.

Une fois que nous avons résolu le problème de la transmission d'images, nous pouvons obtenir des images RGB et des images de profondeur. Pour le modèle Mediapipe, il utilise des images RGB pour détecter le visage et prédire les points caractéristiques. Il peut nous fournir le contour du visage et les valeurs des proportions de chaque point (P_x, P_y, P_z) , En multipliant ces valeurs par la taille de l'image ($weight, height$), nous obtenons la position des pixels des points clés.



Figure 32 Output de mediapipe

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} P_x \\ P_y \end{bmatrix} \begin{bmatrix} W_{image} \\ H_{image} \end{bmatrix}$$

Équation 4 Acquisition de Pixel coordonné

Lorsque nous disposons des coordonnées des pixels, nous pouvons utiliser les propriétés de la caméra, représentées par la matrice de calibration K, pour calculer une transformation des coordonnées des pixels vers les coordonnées de la caméra. Cependant, avant de procéder à ce calcul, il est important d'effectuer une étape de correction de distorsion pour éviter les influences indésirables de la distorsion optique.

Pour corriger ces distorsions, nous utilisons les coefficients de distorsion associés à la caméra D. Ces coefficients sont généralement déterminés lors d'une étape de calibration de la caméra, où des images d'un motif de calibration connu sont

utilisées pour estimer les paramètres de distorsion. Une fois que nous avons ces coefficients, nous pouvons appliquer des formules mathématiques pour corriger la distorsion dans les images.

$$Pixel_{undistorsion} = f(K, D, Pixel_{distortion})$$

Équation 5 Fonction pour undistorsion des pixels

Une fois que les pixels ont été corrigés pour la distorsion, nous pouvons procéder à la transformation du 2D au 3D. Pour cela, nous utilisons la matrice de calibration K, qui contient les paramètres intrinsèques de la caméra, tels que la focale, les coordonnées du point principal et les échelles des pixels. En utilisant cette matrice, et les informations de profondeur que nous avons extraites de l'image de profondeur, nous pouvons effectuer une projection des coordonnées des pixels dans l'image sur un plan 3D.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = [K]^{-1} \begin{bmatrix} U \\ V \\ 1 \end{bmatrix} * Depth$$

Équation 6 Transformation du 2D au 3D

Lorsque nous acquérons des informations 3D, nous pouvons effectuer les tâches suivantes :

1. Sélection des informations pertinentes pour notre projet, telles que les données 3D des lèvres.
2. Conversion des coordonnées de la caméra à la base du robot.
3. Génération d'une trajectoire pour le robot.
4. Évaluation de l'orientation de la tête.
5. Application de maquillage virtuel sur un modèle humain.

Acquisition de lèvres infos

Après avoir calculé les informations 3D, nous avons obtenu 468 points en 3D. Pour sélectionner les informations concernant les lèvres, nous avons utilisé les indices fournis par Mediapipe $Index_i$, sélection 60 point de clé (x_i, y_i, z_i) , et les relations de connexion pour regrouper des mesh $(index_i, index_j, index_k)$, ces infos sont la base pour notre projet, il peut utiliser dans la planification de trajectoire et les colorisations de model humain.



Figure 33 Bouche points sélection

Transformer les coordonnées de la caméra vers la base du robot.

Lorsque nous acquérons les informations dans les coordonnées de la caméra, la prochaine étape consiste à transférer les coordonnées de la caméra à la base du robot. Dans ce cas, nous utilisons les informations provenant de la calibration œil-main, à savoir les translations T et la rotation R. Nous pouvons ainsi obtenir la matrice de transformation du grippieur par rapport à la caméra :

$$T_{grippieur}^{camera} = [R | T]$$

Pour obtenir la pose de la tête dans les coordonnées de la base du robot, nous utilisons la transformation suivante :

$$T_{base}^{tête} = T_{base}^{grippieur} \times T_{grippieur}^{camera} \times T_{camera}^{tête}$$

Équation 7 Equation de transformation de la tête au robot base

Dans notre cas, nous enregistrons la pose du grippieur en utilisant le topic ROS pour calculer $T_{base}^{grippieur}$, ainsi que les poses de la tête. Lorsque nous avons acquis ces positions, nous pouvons calculer l'orientation de la tête, ce qui nous permet de calculer la matrice de transformation $T_{camera}^{tête}$.

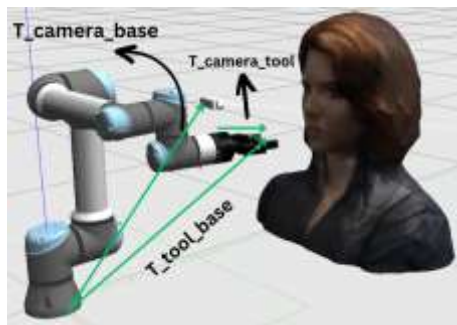


Figure 34 Transformation de coordonnées de la caméra au robot base

Estimation de la pose de la tête

En vision par ordinateur, la pose d'un objet fait référence à son orientation et à sa position relative par rapport à une caméra. La pose peut être modifiée en déplaçant soit l'objet par rapport à la caméra, soit la caméra par rapport à l'objet.

Dans notre projet, le problème d'estimation de pose auquel nous sommes confrontés est connu sous le nom de problème de perspective-n-points. Dans ce problème, lorsque nous disposons d'une caméra calibrée, ainsi que des positions 3D de n points et de leurs projections 2D correspondantes, notre objectif est de trouver la pose de cet objet dans les coordonnées de la caméra.

Par conséquent, quand on veut calculer 3d pose de l'objet, nous devons les informations suivantes :

- 2d pixel de points clé : Nous avons choisi 6 points $[U_{ref}, V_{ref}]^T$ pour évaluer la pose, mediapipe nous donne beaucoup de choix, nous utiliserons

l'extrémité du nez, le menton, le coin gauche de l'œil gauche, le coin droit de l'œil droit, le coin gauche de la bouche et le coin droit de la bouche.

- 3D position des mêmes points : ici, nous avons choisi emplacements 3D de quelques points dans un cadre de référence arbitraire $[x_{ref}, y_{ref}, z_{ref}]$.

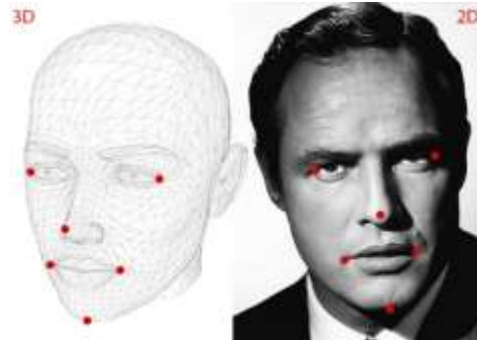


Figure 35 les points de référence de la tête

Pour estimer la pose de la tête, nous avons utilisé trois coordonnées : les coordonnées mondiales, les coordonnées de la caméra et les coordonnées de l'image. Les relations de pose représentent la transformation entre les coordonnées mondiales et les coordonnées de la caméra. Lorsque nous connaissons les informations en 2D d'un pixel, nous pouvons utiliser ces informations pour calculer les informations en 3D dans les coordonnées de la caméra. Ainsi, le problème d'estimation peut être simplifié par l'équation suivante :

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = [R | t] \begin{bmatrix} x_{ref} \\ y_{ref} \\ w_{ref} \\ 1 \end{bmatrix}$$

Équation 8 Equation de calcul de la pose tête

Dans cette équation, $[X, Y, Z]^T$ représente les coordonnées 3D dans le système de coordonnées de la caméra, $[x_{ref}, y_{ref}, z_{ref}]^T$ représente les coordonnées 3D dans le système de coordonnées de la mode. R et T, c'est la pose de la tête. C'est pourquoi nous devons choisir plusieurs fois de relation entre pixel points et 3D points pour calculer cette relation de transformation.

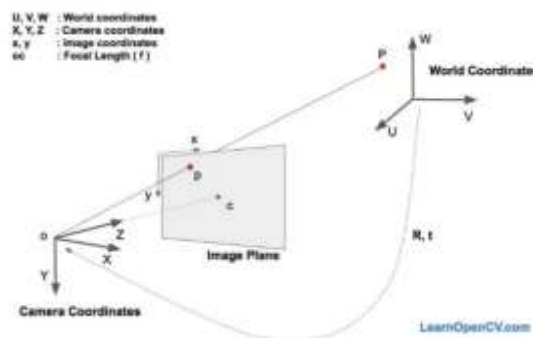


Figure 36 Transformations de coordonnées de la tête aux caméras



Figure 37 Estimation de pose de la tête

Généraliser une trajectoire pour robot

Une fois que nous avons obtenu la pose de la tête et les poses des points clés à l'aide de l'équation 7, nous pouvons calculer les poses des points caractéristiques dans le système de coordonnées de la base du robot.

Dans notre cas, nous avons choisi 60 points. La prochaine étape consiste à résoudre la question de la création d'un contour des lèvres. La trajectoire du robot n'est pas une trajectoire linéaire, mais une courbe. Pour obtenir une trajectoire qui s'adapte bien aux lèvres, nous avons décidé d'ajouter des points de passage afin de découper la courbe en segments.

Dans notre approche, nous utilisons les 60 points de base et itérons pour obtenir 120 points. Cela nous permet de créer une trajectoire plus détaillée et précise pour les mouvements des lèvres.

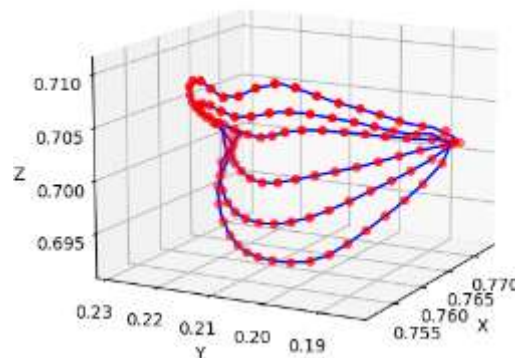


Figure 38 Trajectoire pour la partie lèvre

Colorisation de la model humain.

Lorsque notre robot a terminé son mouvement, nous souhaitons effectuer une coloration sur notre modèle, c'est-à-dire changer la texture de notre modèle. Pour réaliser cette tâche, nous devons effectuer les étapes suivantes :

- Comparer les maillages (mesh) de Mediapipe et du modèle dans Gazebo. Nous effectuons une projection du maillage de Mediapipe sur le maillage du modèle dans Gazebo, puis nous établissons une relation entre la texture de Gazebo et le maillage de Mediapipe. Cela nous permet de

comprendre comment appliquer la texture du modèle sur les parties correspondantes du maillage de Mediapipe

- Comparer les trajectoires réelles avec la trajectoire prédéfinie pour sélectionner la zone à colorier. Nous comparons les trajectoires suivies par le robot avec la trajectoire prédéfinie pour déterminer les parties du modèle qui nécessitent une coloration.

Mesh projection

Lorsque nous comparons les maillages (mesh) de Mediapipe avec ceux du modèle Gazebo, nous rencontrons un premier défi : l'extraction des informations contenues dans le fichier .dae. Ce fichier regroupe des informations cruciales dont nous avons besoin, telles que les cartographies UV, les données des sommets en 3D et les indices des groupes pour chaque maillage, ainsi que les groupes d'indices de pixels pour chaque triangle.

Afin d'améliorer la lisibilité du fichier 3D, nous nous sommes appuyés sur le logiciel MeshLab pour transformer le fichier .dae en un fichier .obj. Cela nous a permis de mieux comprendre la structure du document de maillage 3D et d'extraire facilement les informations relatives au maillage.

Une fois que nous avons terminé l'extraction des informations de maillage du fichier .obj, nous avons commencé à collecter les informations 3D spécifiques à la partie des lèvres. Nous avons sélectionné quatre points clés comme valeurs limites et projeté ces points 3D sur le plan y,z. Sur la base de ces quatre points, nous avons défini une forme ovale et nous avons recherché uniquement les points situés à l'intérieur de cette zone. Cela a constitué notre première fois sélection.

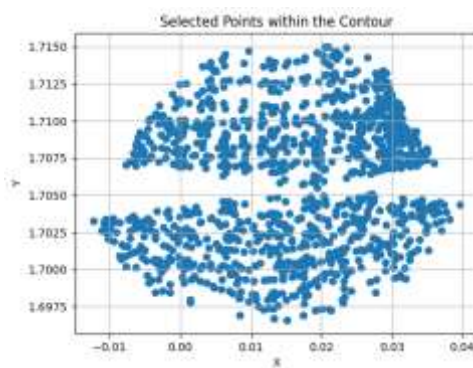


Figure 39 Projection de 2D pour la partie lèvre

Ensuite, nous avons utilisé les informations de profondeur pour éliminer les points superflus, dans le but de réduire la sphère d'analyse.

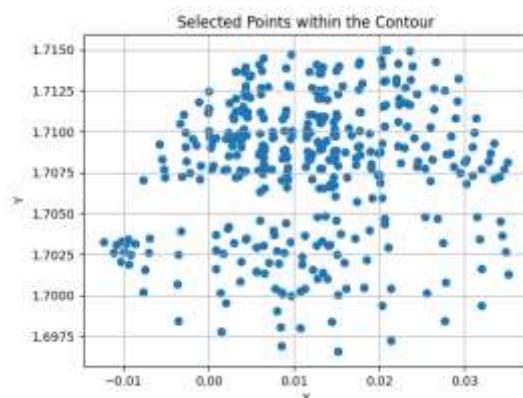


Figure 40 2^{ème} sélection par la depth de lèvres

Après ces deux étapes de présélection, nous avons normalisé à la fois les points 3D de Mediapipe et ceux du maillage Gazebo. Par la suite, nous avons comparé ces valeurs normalisées afin de sélectionner 60 points spécifiques parmi les points 3D du maillage Gazebo.

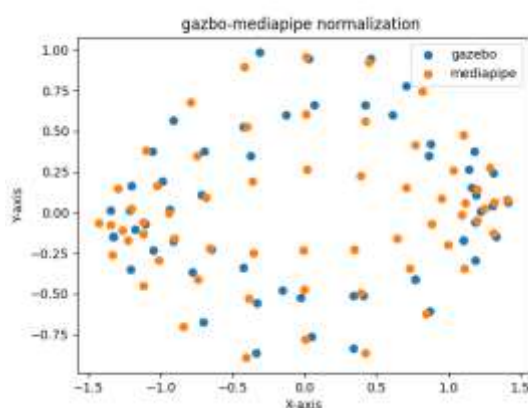


Figure 41 Projection du mesh mediapipe et mesh gazebo

Après avoir projeté les points 3D de Mediapipe sur les points 3D de Gazebo, nous nous sommes également basés sur la cartographie UV pour sélectionner les coordonnées des pixels correspondant aux 60 points dans Gazebo. En nous appuyant sur le maillage de Mediapipe, nous avons regroupé ces 60 points pour former des triangles. Grâce à ces triangles, nous avons pu effectuer la coloration.

Comparer les trajectoires réelles avec la trajectoire prédéfinie

En ce qui concerne la projection entre MediaPipe et Gazebo, nous avons choisi 60 points dans la trajectoire qui se rapprochent le plus des points cibles. Nous les avons ensuite comparés pour évaluer les performances de cette trajectoire, ou, autrement dit, pour évaluer la qualité du maquillage.

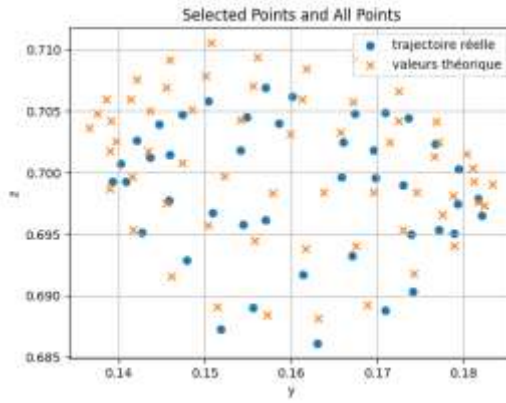


Figure 42 les points théoriques et les points réelle

Après avoir obtenu les points de trajectoire, nous les avons regroupés en utilisant les relations triangulaires du MediaPipe. Ainsi, nous pouvons obtenir des groupes comme suit.

$$\begin{bmatrix} x_i' & y_i' & z_i' \\ x_j' & y_j' & z_j' \\ x_k' & y_k' & z_k' \end{bmatrix} \text{ et } \begin{bmatrix} x_i & y_i & z_i \\ x_j & y_j & z_j \\ x_k & y_k & z_k \end{bmatrix}$$

Équation 9 Matrix de 3d information de mesh

Ici x_j' représente la valeur réelle, tandis que x_j représente une valeur théorique, Ensuite, on calcule les valeurs de chaque triangle et compare les moyennes théorique et réelle. Si $\Delta = |\bar{m} - \bar{m}'| < 0.002$, on peut définir le triangle réel est proche le triangle théorique, ce qui nous permet de réaliser une coloration. Ensuite, nous utilisé les groupe index de triangle pour obtenir les corrdonnées pixels puis réalisons une coloration dans la texture de Gazebo.



Figure 43 Coloration de texture de gazebo modèle

Ensuite, nous renouvllons le gaerbo simulation, nous pouvons decouvert le model est coloré per nous.



Figure 44 Maquillage virtuel en 3D

5.3 Résultats obtenus

Dans la partie de Computer vision et la simulation de maquillage, nous avons obtenu les résultats suivants.

1. En nous appuyant sur MediaPipe, nous avons mis en place un suivi du visage et réalisé une évaluation de la position de la tête. Nous avons aussi recueilli les informations 3D des points caractéristiques dont nous avons besoin.
2. Nous avons réalisé une transformation des coordonnées. Basée sur les propriétés de la calibration hand-eye et la position initiale du robot, nous avons calculé la distance entre la base du robot et la cible. Nous sommes ainsi en mesure de contrôler le robot pour atteindre les cibles que nous avons recueillies.
3. Nous avons défini une trajectoire pour la partie des lèvres à partir des points caractéristiques fournis par MediaPipe. Notre objectif est que le robot traverse tous ces points, c'est-à-dire que lorsqu'il traverse ces points, le robot réalise un maquillage sur le visage.
4. Nous avons également mis en œuvre un maquillage virtuel en utilisant MediaPipe. Tout d'abord, nous avons testé la faisabilité du maquillage avec une caméra RGB en nous basant sur le maillage de MediaPipe pour réaliser une coloration des lèvres. Ensuite, nous avons mis en place une projection de ce maillage entre MediaPipe et Gazebo. En utilisant les données du maillage, nous avons utilisé la texture de Gazebo pour réaliser un maquillage sur le modèle.

6. Motion planning de robot

6.1 Contexte et Problématique

Contexte

Une fois le computer vision partie terminée, nous avons entamé la planification des mouvements. Dans notre ensemble d'outils, nous disposons de différents

contrôles tels que move_J, move_P et move_L. Nous devons utiliser ces outils de mouvement pour optimiser nos trajectoires et les comparer afin de choisir un groupe de motions plus sûr et plus efficace.

Problématique

Les problèmes que nous devons résoudre sont les questions suivantes :

1. Comment améliorer la qualité du maquillage tout en réduisant le temps nécessaire ?
2. Comment éviter les mouvements brusques du robot et augmenter la sécurité lors des interactions ?

6.2 Méthodes utilisées pour répondre à la problématique

Contexte

Au début du projet, pour contrôler le robot, nous utilisions une simple cinématique inversée. Nous définissions un point et contrôlions le robot pour qu'il atteigne ce point en utilisant un solveur RRTs, ce qui nous donnait un mouvement moins énergique et plus rapide. Cependant, au fur et à mesure de l'avancement du projet, nous avons découvert qu'il était difficile d'obtenir une trajectoire plus sécurisée, et parfois il y avait des mouvements excessivement grands.

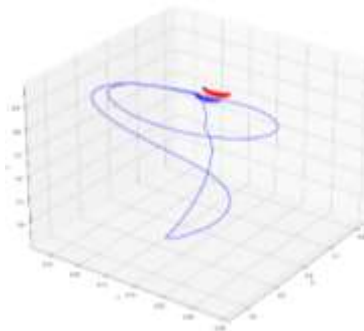


Figure 45 Trajectoire Move_P calculer par un RRTs

De plus, nous avons découvert que l'utilisation de Move_P pour contrôler le robot n'était pas une bonne approche, car la réalisation d'un mouvement de maquillage implique non seulement des problèmes cinématiques, mais aussi des problèmes dynamiques. Nous devons prendre en compte la vitesse dans les directions x et y. Pour Move_P, il présente un défaut majeur, à savoir qu'il atteint les points un par un. Tout au long du processus, il y a des phases d'accélération et de décélération, passant de 0 à 0. Cela entraîne de petites pauses lorsqu'il traverse tous les waypoints.

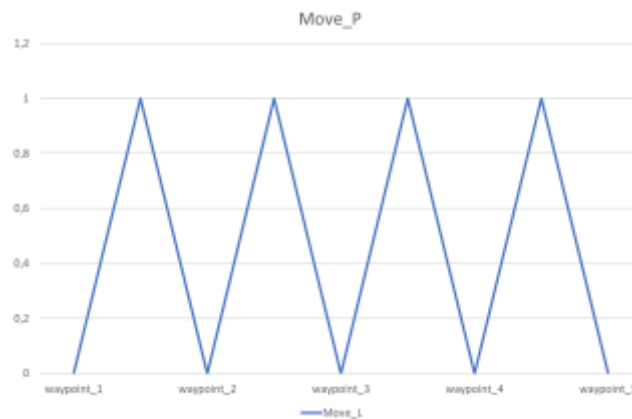


Figure 46 Variation de vitesse de Move_P pour tous les waypoints

Par conséquent, nous avons décidé d'utiliser l'outil Move_L. Avec cet outil, nous pouvons définir différents nombres de waypoints et définir leur résolution. Il peut calculer en une fois les groupes de configurations du robot pour traverser tous les waypoints à une vitesse constante.

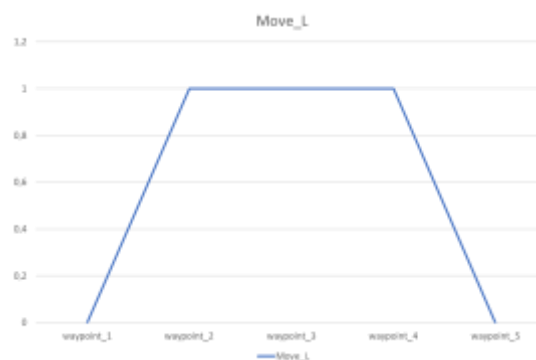


Figure 47 Variation de vitesse de move_L

Optimisation de la trajectoire

Dans notre projet, nous avons divisé la trajectoire du robot en deux parties : l'une pour la prise d'un rouge à lèvres, et l'autre pour l'application du maquillage.

Commençons par discuter de la trajectoire pour la prise du rouge à lèvres. Dans notre cas, nous avons défini la position du rouge à lèvres à gauche du robot à la position $[0, -0.3, 0]$ dans le système de coordonnées de la base du robot. Pour réaliser cette tâche, nous avons défini cinq waypoints : la position par défaut p_0 , le point initial p_1 , le point de préparation p_2 , le point de saisie p_3 et le point de détection p_4 . Nous avons comparé les méthodes Move_P et Move_L et avons constaté que le calcul de Move_P n'est pas toujours stable. Parfois, cela entraîne une trajectoire instable et le robot n'est pas capable de prendre correctement le rouge à lèvres. Par conséquent, nous avons décidé d'utiliser Move_L qui est plus stable et permet au robot de suivre correctement la trajectoire que nous avons définie.

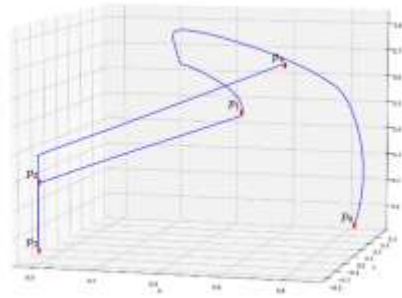


Figure 48 Move_L et Move_J trajectoire pour picking rouge-lèvre

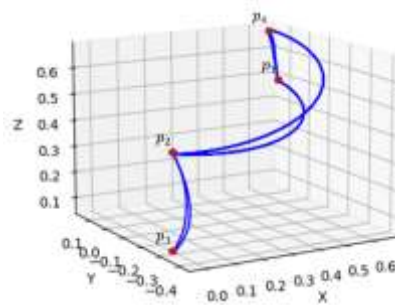


Figure 49 Move_P trajectoire pour picking rouge-lèvre

Dans le cadre du maquillage de trajectoire, nous avons fait une comparaison entre la move_P et la move_L. Nous avons comparé le temps dépensé depuis la réception de la commande jusqu'à l'achèvement de toutes les tâches. Nous avons également comparé la trajectoire pour chaque groupe de points de passage.

Nous nous sommes appuyés sur ROS (Robot Operating System) et avons utilisé la communication par topic pour publier et souscrire à des informations relatives aux commandes. Nous avons diffusé la trajectoire prédéfinie à notre robot. Lorsque le robot reçoit une commande, il suit cette commande et exécute les actions que nous avons définies.

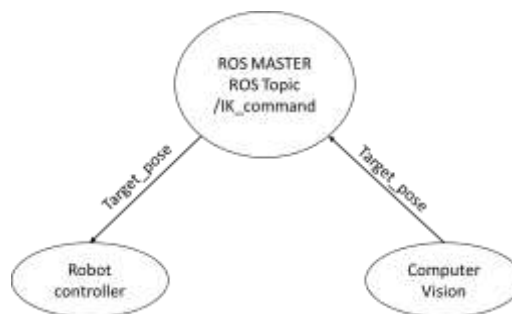
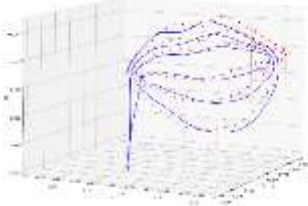
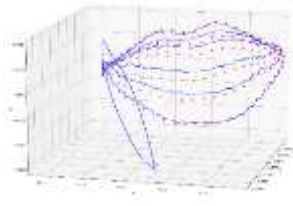
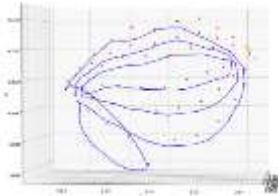
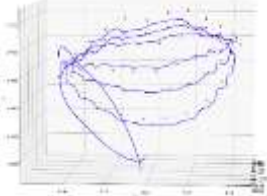


Figure 50 la structure de robot contrôle servo en temps réelle

Après que le robot a terminé la tâche, nous avons compté le temps d'utilisation et la trajectoire du robot. Dans notre système, lorsque le robot effectue le Move_P, il réalise une action dès qu'il reçoit une commande. Si le robot effectue le Move_L, il reçoit toutes les commandes, puis effectue les calculs avant d'exécuter les actions. Par conséquent, lorsqu'on exécute un Move_L, le robot

Il passe du temps à collecter les informations et à effectuer les calculs. Pour le Move_P, il ne fait pas tous les calculs en même temps, mais les réalise un à un. Cela fait que le Move_P est plus lent que le Move_L.

Tableau 1 comparaison de trajectoire pour différents de waypoints et différente move

	Move_L	Move_P
60 waypoints	23s	57s
120 waypoints	53s	247s
Trajectoire de 120 waypoints		
Trajectoire de 60 waypoints		

À la suite des tests réalisés, nous avons observé que l'augmentation du nombre de waypoints entraîne une hausse du temps nécessaire pour le calcul de la configuration du robot. C'est particulièrement vrai lorsque nous utilisons la méthode Move_L qui, bien qu'elle permette un gain de temps par rapport à Move_P pour le même nombre de waypoints, offre une précision moindre.

Effectivement, nous avons constaté que Move_P présente une précision supérieure à Move_L, étant capable de passer par plus de points clés pour un nombre de waypoints identique. Cependant, il est important de souligner que cette précision accrue s'accompagne d'une durée plus longue et peut présenter davantage de risques en raison des accélérations et décélérations nécessaires au cours des actions.

En conclusion, si Move_L est plus rapide que Move_P, c'est Move_P qui est le plus précis des deux. Toutefois, la lenteur et le niveau de risque plus élevé de Move_P doivent être pris en compte, notamment parce que notre solveur peut parfois proposer une configuration non optimale du robot, générant des mouvements significatifs lors de l'exécution d'une tâche.

6.3 Résultats obtenus

Une fois la planification de la trajectoire terminée, nous sommes en mesure d'effectuer une application de maquillage sur notre modèle, dans un scénario statique. Cependant, réaliser une application de maquillage sur un véritable être humain représente encore un défi majeur. Le premier défi est la gestion de la

force. Si nous ne pouvons pas contrôler la force, la réalisation en conditions réelles devient non viable.

7 Renforcement Learning Framework

7.1 Contexte et Problématique

Contexte

Comme nous l'avons discuté dans la cinquième partie, une fois que le robot est capable de réaliser une trajectoire à vitesse tangentielle constante, l'étape suivante consiste à contrôler la sortie du robot pour qu'il génère une force constante et sûre. En d'autres termes, nous devons maintenir une vitesse constante dans la direction normale à la trajectoire. Pour résoudre ce défi dynamique, nous avons deux solutions : une basée sur un modèle et l'autre basée sur des données.

Dans l'approche basée sur un modèle, nous supposons que les articulations du robot fonctionnent comme un système masse-ressort-amortisseur de second ordre. Nous devons définir une fonction dynamique pour représenter l'état du robot, et une fonction de sortie pour représenter la sortie du robot. C'est un processus relativement complexe, en particulier la construction du modèle dynamique.

Dans l'approche basée sur les données, la construction d'un modèle dynamique n'est pas nécessaire. Le robot peut simplement se baser sur une politique pour réaliser une action plus adaptée à l'état actuel. Cette méthode peut être plus flexible et adaptable, mais elle nécessite une grande quantité de données pour l'apprentissage.

Problématique

Les problèmes que nous devons résoudre sont les questions suivantes :

1. Développer un Renforcement learning cadre pour effectuer des actions précises, de recevoir des observations pertinentes, et d'acquérir une récompense.
2. Sélectionner une stratégie de formation pour entraîner une politique.

7.2 Méthodes utilisées pour répondre à la problématique

Construction du cadre de RL

Dans notre projet, nous avons construit notre cadre d'apprentissage par renforcement en nous basant sur OpenAI Gym. Nous avons intégré des éléments tels que Gazebo et Mediapipe, et ajouté divers sujets à notre cadre.

D'ailleurs pendant on a construit le cadre renforcement learning, on a rencontré un gros bouchon d'information, et un instable connections en utilisant la rostopic transmettre les informations, en suite on a utilisé un autre communication méthode. On a créé un client-serveur communication, pour réaliser un contrôle

en cinématique inversé. Ici, on a envoyé le robot contrôle message, et le serveur nous donné le positon de grippieur.

Dans notre cadre, nous disposons d'un espace d'action basé sur la cinématique inverse, qui permet d'exécuter des actions telles qu'avancer, reculer, tourner, en fonction de la décision prise par l'agent. Nous avons établi une position initiale $[x, y, z, x_q, y_q, z_q, w_q]$ et une valeur $\delta = 0.005$ qui permet au robot d'exécuter les actions, comme le démontre la fonction suivante :

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \pm \delta = robot\ action$$

Ici, nous ne voulons pas changer l'orientation de grippieur, pour simplifier le processus de Learning. On ne veut que notre robot puisse arriver les waypoints et trouver une trajectoire bien s'adapter la forme de lèvre.

En suite pour un observation espace, on veut acquit la pose de robot, et l'info de force senseur, et la qualité de maquillages et les points de caractéristiques de lèvre.

$$\begin{bmatrix} P^{6 \times n} \\ F^{1 \times n} \\ P_{target}^{6 \times n} \end{bmatrix}$$

Pour la partie de récompense, ici nous avons défini des bonus pour notre action.

- Récompense pour la force

$$\Delta_1 = F_{output} - F_{target}$$

Ici, nous pouvons définir une valeur désirée pour la force. Ensuite, nous pouvons lire les forces réelles dans l'espace d'observation et soustraire cette valeur cible pour obtenir une valeur Δ_1 . Dans ce cas, nous avons un seuil de force $\varepsilon=1$. Lorsque nous obtenons une valeur $\Delta_1 > \varepsilon$. Nous ne recevons pas de récompense mais si la valeur est inférieure ε , on peut avoir une récompense.

$$r_1 = \begin{cases} r_f, & |\Delta_1| \leq \varepsilon \\ 0, & |\Delta_1| > \varepsilon \end{cases}$$

- Récompense pour les points caractérisés.

Dans notre cas, nous avons 60 cibles, ici, on a faire une comparaison entre le point théorique et les points réelle, en calculant l'écart entre le positon de grippieur et celle de cible, si le robot approche les cibles, il va recevoir une récompense.

$$\Delta_2 = P_{gripper} - P_{target}$$

$$r_2 = \begin{cases} r_p, & |\Delta_2| \leq \varepsilon \\ 0, & |\Delta_2| > \varepsilon \end{cases}$$

- Récompense pour la qualité de maquillage

Ici, la récompense pour la qualité du maquillage représente un bonus important. Nous avons collecté les positions du gripper, puis, après l'entraînement, nous les avons comparées aux points cibles. Si nous avons traversé tous les points, nous recevrons un grand bonus proportionnel au nombre de points traversés par le robot.

$$r_3 = N_{points}$$

En suit, chaque fois, robot exécute une action, il est possible d'acquérir une récompense. Quand le robot finit les actions, ou il a traversé tous les points, il aura une grande récompense.

$$r_{naction} = \left(\sum r_1 + r_2 + r_3 \right)$$

7.3 Résultats obtenus

En ce qui concerne la partie sur l'apprentissage par renforcement, je n'ai pas encore obtenu de résultats qui peuvent être déployés sur un robot réel. Parfois, nous rencontrons des blocages d'information entre les différents sujets, ce qui interrompt l'entraînement du modèle. Pendant la rédaction de ce rapport, nous travaillons également à résoudre ces problèmes, comme surmonter les limites de Gym et éviter les processus complexes liés à la demande d'un sujet ROS. Nous essayons d'utiliser une communication serveur-client, où le robot, lorsqu'il exécute une action, peut envoyer la position du gripper une fois l'action terminée. Cette méthode de communication semble bien s'adapter aux besoins du cadre de l'apprentissage par renforcement.

8 Synthèse, conclusion et perspectives

Le projet "Make-up for Robot Arm" a pour objectif de réaliser un maquillage automatique grâce à un bras robotisé. Ce projet vise à créer une interaction physique entre l'humain et la robotique, et présente un potentiel prometteur pour le futur.

Dans le cadre de ce projet, nous avons développé un environnement de simulation synchronisé avec notre environnement réel, en utilisant le cadre ROS (Robot Operating System). En utilisant cet environnement, nous avons déployé un module de vision par ordinateur ainsi qu'un module de contrôle du robot. Cela nous a permis de détecter efficacement un visage et d'extraire ses points caractéristiques. Ensuite, grâce au module de contrôle, nous avons pu calculer les configurations des articulations du robot nécessaires pour réaliser des mouvements selon une trajectoire prédéfinie. Au cours de cette étape, nous avons réalisé non seulement des simulations, mais également des tests et déploiements des modules "media pipe" et "robot control" sur notre robot réel. Ces tests ont permis de vérifier le bon fonctionnement de nos algorithmes.

Dans l'ensemble, notre projet a atteint les objectifs fixés et a démontré la faisabilité d'un maquillage automatique par bras robotisé. Cependant, il reste encore des possibilités d'amélioration et des perspectives à explorer. Voici quelques pistes pour les développements futurs :

1. Ajout d'un module de contrôle de force : Dans notre projet, nous n'avons pas encore réalisé de contrôle de force. Cependant, dans l'interaction entre l'humain et le robot, il est essentiel de prendre en compte la force pour protéger l'humain et améliorer la qualité du maquillage.
2. Amélioration de la précision : Affiner les algorithmes de détection du visage et d'extraction des points caractéristiques pour des résultats encore plus précis et fiables.
3. Améliorer le module renforcement Learning, dans mon stage, je n'ai pas beaucoup de temps pour tester différents RL modèles, tels que Actor-Critic model. On compare les modèles value base, Policy model n'a pas besoin d'un grand espace pour mémoriser les actions. Il est plus adapté à un robot contrôle surtout pour les actions continues. D'ailleurs, je pense que le modèle hybride est aussi une bonne direction pour le modèle RL.
4. Pour la computer vision part, je pense qu'il faut réfléchir à comment faire une prédiction ou calculer l'orientation de chaque point caractéristique, dans notre cas, on a utilisé l'orientation de tête comme l'orientation de grippage, mais en idéal, l'orientation de grippage doit l'orienter normalement de chaque point.

En conclusion, le projet "Make-up for Robot Arm" a été une expérience enrichissante qui a abouti à la réalisation d'un maquillage automatique par bras robotisé. Les résultats obtenus ouvrent la voie à de nouvelles possibilités dans le domaine de l'interaction entre l'humain et la robotique dans le domaine de la beauté. Les perspectives d'amélioration et de développement futur sont prometteuses, et nous sommes impatients de voir comment ce projet évoluera et contribuera à l'avenir du maquillage automatisé.

9 Bibliographie

- [1] ROS Tutorial: Control the UR5 robot with ros_control – How to tune a PID Controller https://roboticscasual.com/ros-tutorial-control-the-ur5-robot-with-ros_control-tuning-a-pid-controller/
- [2] Camera calibration <https://fr.mathworks.com/help/vision/ug/camera-calibration.html>
- [3] Head Pose Estimation using Python <https://towardsdatascience.com/head-pose-estimation-using-python-d165d3541600>
- [4] Ros wiki Documentation http://wiki.ros.org/Documentation*
- [5] Gazebo Tutorials <https://classic.gazebosim.org/tutorials>
- [6] Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., Zhang, F., Chang, C-L., Yong, M.G., Lee, J., Chang, W-T., Hua, W., Georg, M., & Grundmann, M. (2019). MediaPipe: A Framework for Building Perception Pipelines. arXiv preprint arXiv:1906.08172.
- [7] Liu, W., Niu, H., Pan, W., Herrmann, G., & Carrasco, J. (2023). Sim-and-Real Reinforcement Learning for Manipulation: A Consensus-based Approach. arXiv preprint arXiv:2302.13423.
- [8] Franceschetti, Andrea & Tosello, Elisa & Castaman, Nicola & Ghidoni, Stefano. (2021). Robotic Arm Control and Task Training through Deep Reinforcement Learning.
- [9] Liu, Wenxing & Niu, Hanlin & Mahyuddin, Muhammad Nasiruddin & Herrmann, Guido & Carrasco, Joaquin. (2021). A Model-free Deep Reinforcement Learning Approach for Robotic Manipulators Path Planning.
- [10] Singh, A., Yang, L., Hartikainen, K., Finn, C., & Levine, S. (2019). End-to-End Robotic Reinforcement Learning without Reward Engineering. ArXiv:1904.07854 [Cs.LG].

MODELE DE DECLARATION DE CONFIDENTIALITE



DECLARATION DE CONFIDENTIALITE

ANNEE : 2022-2023 **N° :** LI-2022/12/11556

TYPE DE DOCUMENT : rapport de SFE

CAMPUS DE RATTACHEMENT : Arts et Métiers Lille

AUTEURS : SU Sichen

TITRE : Développement du robot assistant pour le maquillage

ENCADREMENT : BEAREE, Richard

ENTREPRISE PARTENAIRE : L'Oréal Recherche & Innovation

PARTIE A REMPLIR PAR LE TUTEUR PEDAGOGIQUE ENSAM

CONFIDENTIALITE DE CE RAPPORT

(Entourer la mention choisie) :

Classe 0 = accès libre

Classe 1 = Confidentiel jusqu'au _____ (date de fin de confidentialité obligatoire)

Classe 2 = Hautement confidentiel : je certifie par cette signature avoir vu le document et la note de synthèse et j'assure que :

. le document rédigé à l'occasion de ce SFE est conforme (rapport rédigé d'environ 70 pages annexes incluses) ;

. la note de synthèse est conforme

. les clauses de confidentialité nécessitent un classement du document en classe 2.

Date :

Nom du signataire :

Signature :

AUTORISATION DE MISE EN LIGNE DU RAPPORT DE STAGE DE FIN D'ETUDES

Je soussigné(e) Monsieur, Madame :

Su Sichen

Né(e) le 06/091996 - Demeurant : **1 Avenue
Pierre Masse**

Auteur, signataire de l'œuvre et à ce titre averti des droits d'auteur s'y rapportant, du Stage de Fin d'Etudes (SFE) en langue française intitulé :
**Tapez le titre de votre rapport Développement
du robot assistant pour le maquillage**

Ce rapport final est réalisé dans le cadre de la formation de l'élève ingénieur à Arts et Métiers. Il est classé en « accès libre » et déposé à la bibliothèque Arts et Métiers sous la forme d'un fichier pdf.

Encadré par BEAREE, Richard **BEAREE,
Richard, Professeur Arts et Métiers**

Soutenu le : 07/11/2023 à Arts et Métiers
Campus de : Arts et Métiers ParisTech Campus
Lille

(Entourez la mention choisie)

- **n'autorise pas** la mise en ligne de mon rapport de SFE
- **autorise**, agissant en l'absence de toute contrainte, après accord du jury du SFE et autorisation du chef d'établissement, la mise en ligne de mon rapport de SFE

dans les conditions suivantes :

- La mise en ligne est effectuée par Arts et Métiers dont l'objectif est de remplir ses missions de service public de diffusion de la

Signature de l'élève ingénieur précédée de la mention « lu et approuvé »

culture scientifique et de l'information scientifique et technique au sein de la communauté universitaire et de contribuer tant à la renommée de l'auteur qu'à celle du travail réalisé. Elle est réalisée à titre gratuit.

- La mise en ligne est autorisée sur les réseaux intranet, extranet et le réseau international internet.

- La mise en ligne inclut des opérations de reproduction du rapport notamment son téléchargement, son stockage ou tout acte de fixation temporaire qu'implique la transmission numérique et la diffusion du rapport sur le réseau intranet, extranet et internet, et ce, quel que soit son format et le procédé technique utilisé.

- Le rapport sera diffusé en version intégrale et en langue française, conformément à la loi 94-665 du 4 août 1994, accompagné d'un résumé en langue française et/ou étrangère. Une notice bibliographique pourra être établie par la bibliothèque, à partir du résumé fourni par l'étudiant, pour le signalement du rapport de Projet d'Expertise dans le catalogue informatisé de la bibliothèque régulièrement déclaré à la CNIL sous le n°892849.

- L'auteur demeure responsable, du contenu de son œuvre conformément au droit commun.

- Les autorisations de diffusion données à Arts et Métiers n'ont aucun caractère exclusif et l'auteur conserve toutes les autres possibilités de diffusion de l'œuvre.

- La mise en ligne est autorisée pendant toute la durée de protection légale accordée à l'auteur du rapport par la loi française.

- L'auteur peut retirer l'autorisation de diffusion à tout moment à charge pour lui d'en aviser le représentant légal d'Arts et Métiers par lettre recommandée avec accusé de réception. L'œuvre sera alors retirée dans les meilleurs délais.

Fait à, le

Signature du référent pédagogique Arts et Métiers précédée de la mention « lu et approuvé »