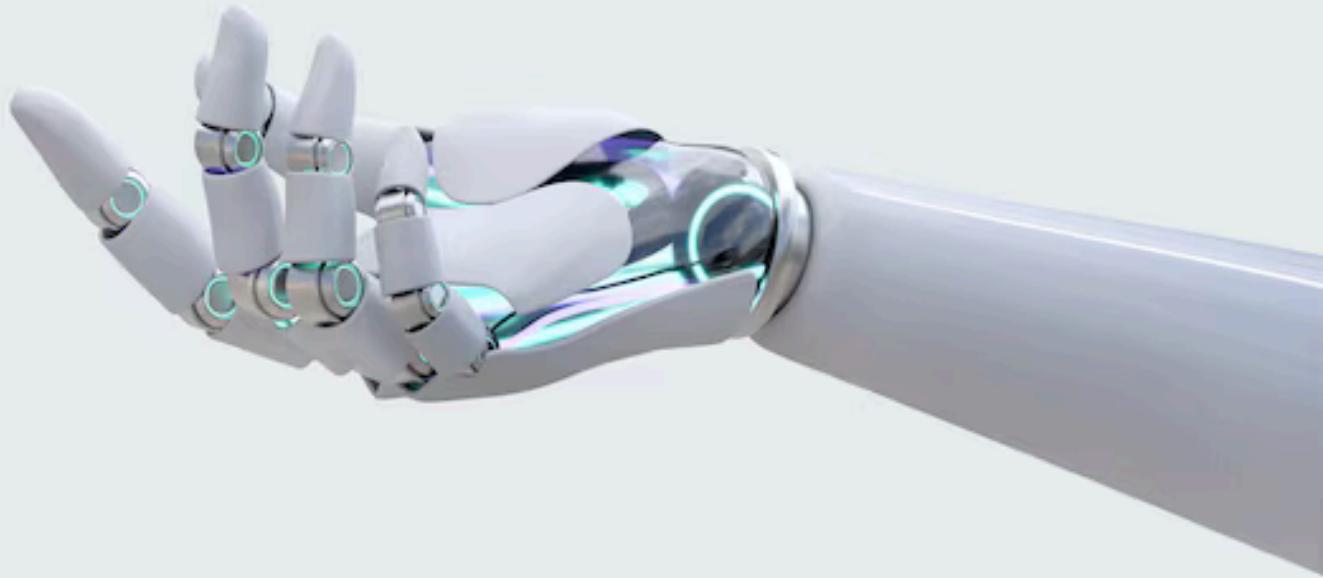

DeepTech 2024

HANDSOME HAND DIGITAL TWIN

Maxime CARRAZE
Arsène SEUILLOT
Inès LE KIM
Nora Madih

SOMMAIRE

01.	Introduction	3
02.	Définition et périmètre	3
03.	Pôles Techniques	5
	• Information	
	• Mécanique	
	• Puissance et commande	
04.	Pistes d'amélioration	8
05.	Conclusion	16



INTRODUCTION

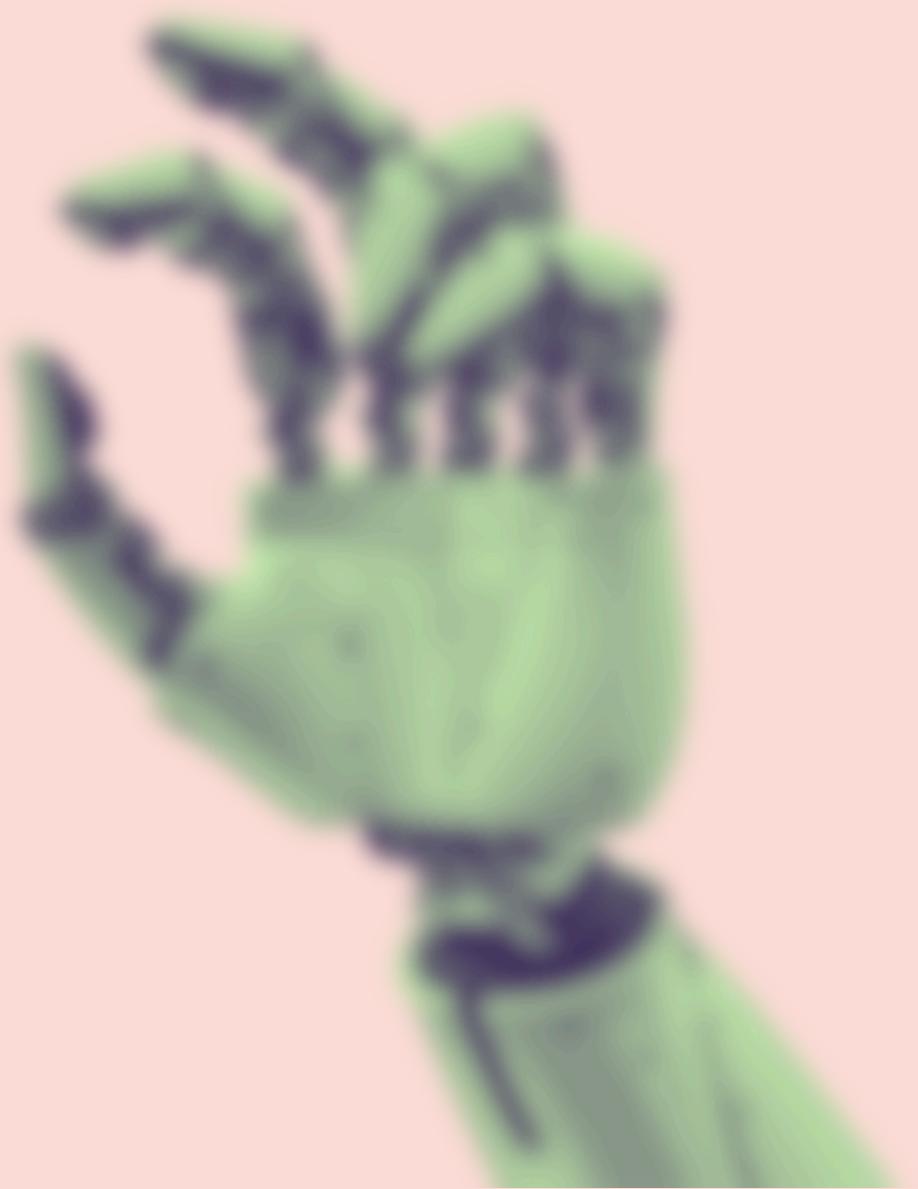
INTRODUCTION

Dans le but de repousser toujours plus loin les limites imposées par la nature, l'humain doit aujourd'hui effectuer des opérations complexes dans des milieux de plus en plus contraints. Il existe de nombreuses situations dans lesquelles nous avons besoin de faire des manipulations précises mais sans pouvoir faire les manipulations directement. Par exemple, dans des situations extrêmes de température ou de pression, ou alors des opérations spatiales en dehors de l'ISS, ou même pour reproduire les mouvements mais en plus petit, pour des opérations chirurgicales précises, ou en plus grand, pour des manipulations plus puissantes par exemple. Il existe également des situations dans lesquelles nous voulons reproduire les mouvements mais dans un environnement virtuel, que ce soit pour des activités pédagogiques, des opérations de ré-éducation ou même du divertissement.

Pour répondre à cette problématique, le projet HANSDOME propose de concevoir un système permettant de faire reproduire à l'identique à une main robotique les mouvements d'une main humaine.

Toutes les ressources informatiques liées au projet sont accessibles via le lien suivant : <https://github.com/arsene-seuillot/Handsome/>

DÉFINITION ET PÉRIMÈTRE



DÉFINITION

L'objectif du projet est de concevoir un système permettant de faire du mimétisme de mains sur une main humaine : nous voulons détecter les mouvements de la main et être capable de les reproduire sur une main robotique. Il existe de nombreuses méthodes permettant d'obtenir ce résultat, nous verrons donc dans cette partie quelle méthode sera retenue.

Discutons d'abord de la méthode de détection de la main. Il existe plusieurs manière de faire, et deux sont particulièrement notables. La première consiste en un jeu de capteurs sur la main permettant de capter ses déplacements et la position angulaire de chaque doigt. Des accéléromètres sur des points clé de la main, ou encore un gant équipé avec des capteurs de tension en sont des exemples. La deuxième méthode est purement informatique et consiste en l'implémentation d'un modèle d'intelligence artificielle détectant des points clés sur la main à partir d'une vidéo en temps réel. C'est cette seconde méthode que nous avons décidé de garder, pour maximiser la pluridisciplinarité du projet et limiter les retards techniques dûs à la complexité de faire fonctionner un jeu de capteurs complexes. Les modèles d'intelligence artificielle d'aujourd'hui étant très performants, ils permettent d'avoir des résultats très pertinents avec une complexité d'implémentation amoindrie.

Pour la conception mécanique de la main, nous voulons implémenter un maximum de mouvements des doigts pour avoir des mouvements réalistes. Nous verrons dans la définition du périmètre quels mouvements nous avons décidé de ne pas faire. Nous allons concevoir une main en taille réelle pour la démonstration.

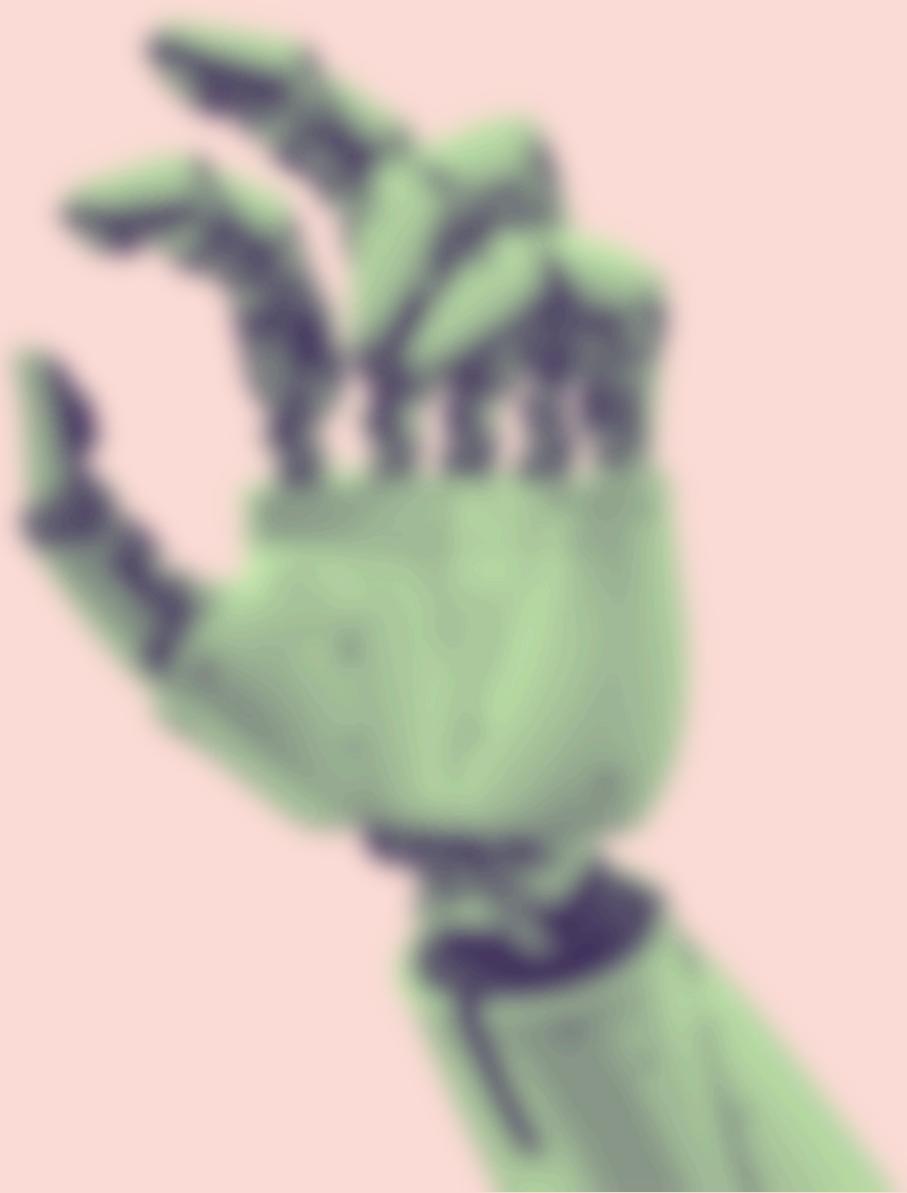
PRÉRIMÈTRE

Nous allons voir ici quelles parties du projet que nous allons omettre, pour des questions de réalisme et de temps.

Pour la partie information, l'idée est d'être capable de détecter complètement les mouvements de la main à partir d'une caméra embarquée reliée à une Raspberry Pi 5, qui fera les calculs. Nous voulons également être capable de reproduire une main en 3D dans Blender qui reproduit les mouvements en temps réel, mais nous n'allons pas insister sur cette partie là. Pour les programmes informatiques qui gèrent cette partie, nous voulons manipuler le tout avec un seul script C qui gère les scripts Python qui utilisent la caméra et font les calculs. Le "cerveau" du système complet, contenant la Raspberry et la caméra, ne sera pas intégrée dans le système mécanique.

Pour la conception mécanique de la main, nous avons décidé de faire deux degrés de liberté par doigts, en solidarisant la rotation des deux phalanges du bout du doigt. La dernière phalange bougera seule. Cela permet de faire des mouvements de doigts réalistes dans la plupart des cas, en retirant de la complexité dans le système. Nous n'allons pas nous occuper des mouvements du pouce, et nous ne prendrons évidemment pas en compte les mouvements de plis internes à la paume de la main. Nous allons donc nous concentrer sur les mouvements des doigts uniquement. Pour la structure finale, nous n'allons pas essayer de miniaturiser et nous nous concentrerons sur l'aspect fonctionnel. Nous nous concentrerons sur le fonctionnement interne d'un seul doigt, et nous verrons dans un dernier temps si nous arrivons à faire un assemblage fonctionnel.

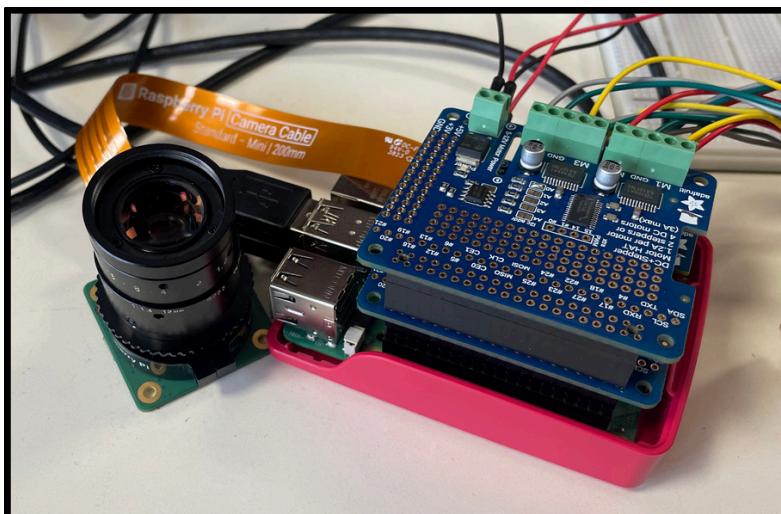
PÔLES TECHNIQUE



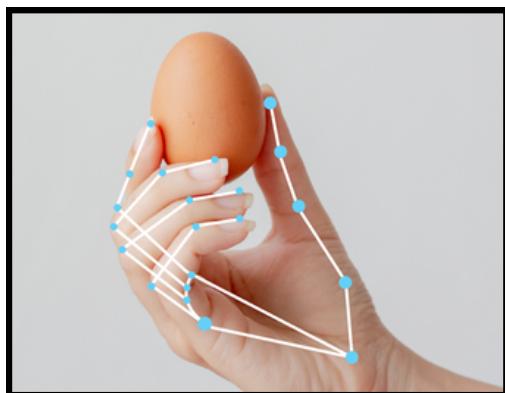
INFORMATION

Pour la partie information du projet, nous allons utiliser une Raspberry Pi 5, avec une caméra pour Raspberry. Nous utiliserons ensuite le logiciel Blender pour faire reproduire les mouvements à un modèle 3D de main. Enfin nous mettrons en place les programmes permettant de faire communiquer le script qui capture l'information, et le script qui gère les moteurs.

Nous gérons la caméra avec Picamera2, et nous utilisons également OpenCV pour faire le traitement d'images en temps réel. Le modèle d'intelligence artificielle que nous avons décidé d'utiliser est MediaPipe, qui une bibliothèque de solutions proposée par Google. Elle permet notamment de faire de la détection de repères sur la main.

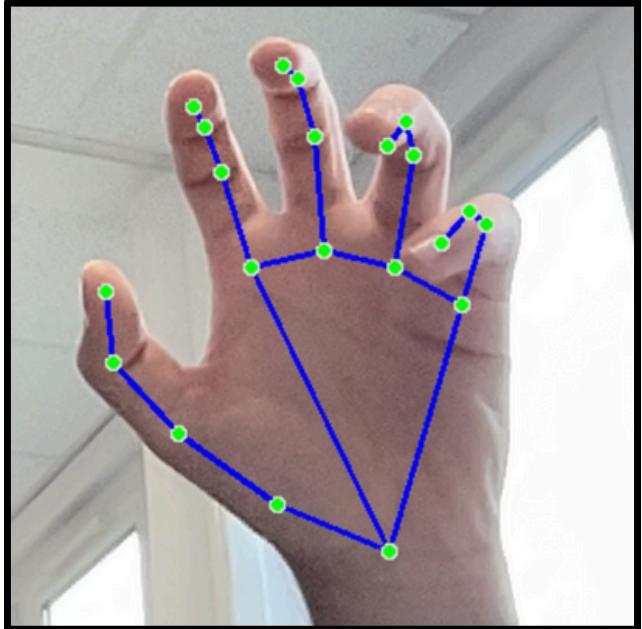


Notre système

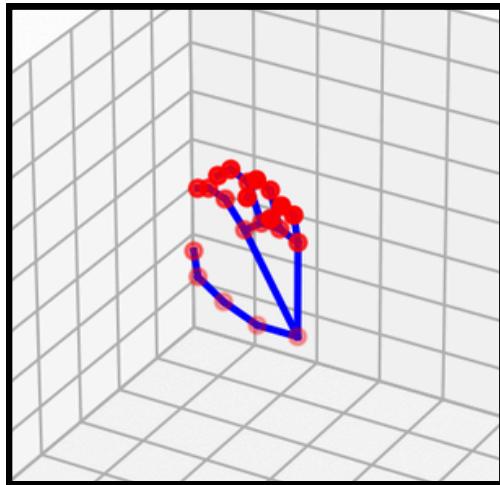


Démonstration de Google de MediaPipe

INFORMATION



Détection de la main



Reconstruction avec Matplotlib

Les résultats de MédiaPipe sont très convaincants et permettent d'avoir un débit d'information très élevé, près des 15 images par secondes. Cela se fait au détriment de la qualité de détection : certaines positions ont un peu de mal à être discernées, notamment lorsque les doigts sont de face de la caméra. Mais cela permet quand même d'avoir l'allure de la main en tout temps. De plus le modèle n'a pas de mal à détecter convenablement dans des conditions de basse luminosité, ou lorsque l'image n'est pas nette.

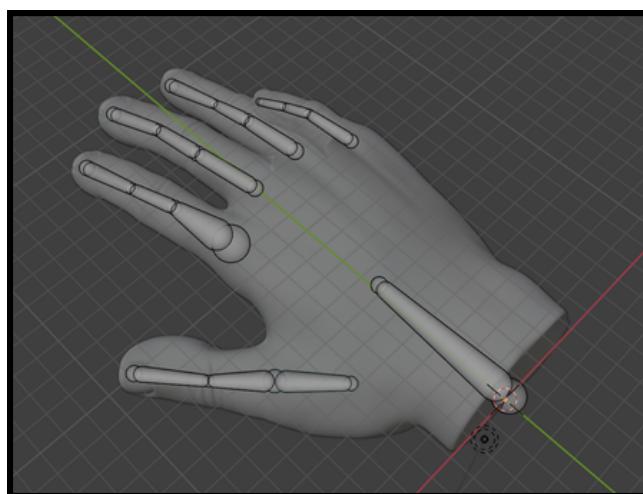
Il y a également parfois des sauts dans la détection des points qui peuvent faire des mouvements incohérents dans la construction. Pour limiter ce phénomène, et limiter les tropes grandes variations dans les positions des points, nous avons intégré un lissage des données par moyenne glissante.

INFORMATION

MediaPipe va placer des points de repère sur la main : un sur la base de la paume, un à la base de chaque doigts puis trois autres pour les phalanges et le bout du doigt. Le modèle va ainsi fournir les coordonnées cartésiennes de chacun de ces points de repère. Mais selon la chaîne d'information nous voulons avoir les angles entre chaque doigt : il faut donc programmer cette conversion. Il s'agit pour nous de la fonction `calculate_finger_angles` dans le programme principal. Elle prend en argument la liste de tous les points de repère, calcule les angles voulus et renvoie un dictionnaire contenant pour chaque doigt les angles entre les phalanges (c'est un dictionnaire de dictionnaire). Pour la suite, c'est ce dictionnaire-ci qu'il faudra transmettre en temps réel au script de gestion des moteurs.

Avant de nous intéresser à la transmission des données, voyons comment faire pour utiliser ces données sur Blender pour faire bouger une main virtuelle en temps réel.

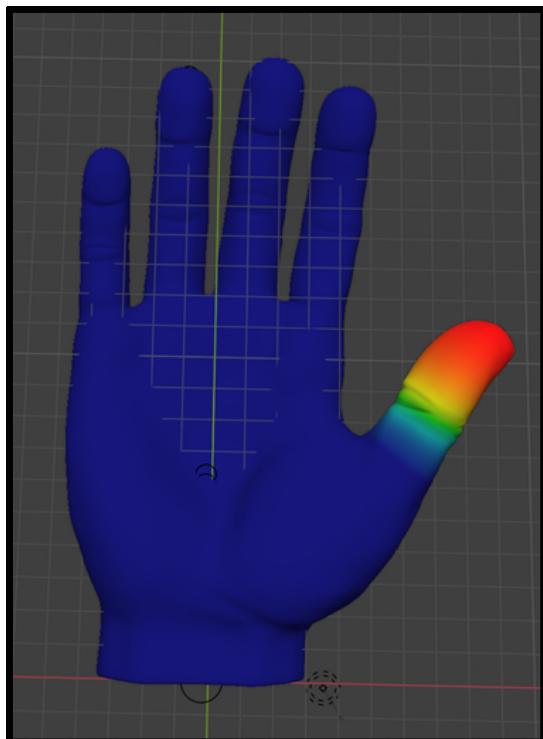
Blender possède un environnement python intégré qui permet de modifier tous les éléments à notre guise grâce à la bibliothèque `bpy`. Nous allons donc importer un modèle 3D de main dans Blender, le rigger pour avoir une armature cohérente avec les données que nous calculons. Ensuite grâce à un script python nous allons piloter chaque point de l'armature grâce aux données obtenues en temps réel par le programme de calcul. C'est en fait une simulation du système réel que nous allons concevoir par la suite.



Main riggée sur Blender

INFORMATION

Sur Blender nous pouvons également modifier la distribution des poids, c'est à dire l'effet qu'aura un mouvement de l'armature du modèle sur le maillage de la main. Cela permet de simuler les plis de peau aux alentours des articulations.



Peinture de poids pour le bout du pouce

Une fois la main configurée, nous pouvons faire le programme python qui relie chaque mouvement des articulations de la main aux angles envoyés par le programme de calcul. Il y a donc deux scripts python : un dans blender qui pilote les articulations et un autre, externe et relié "physiquement" à la caméra. Nous utilisons un socket python pour faire communiquer facilement les deux scripts en temps réel.

INFORMATION

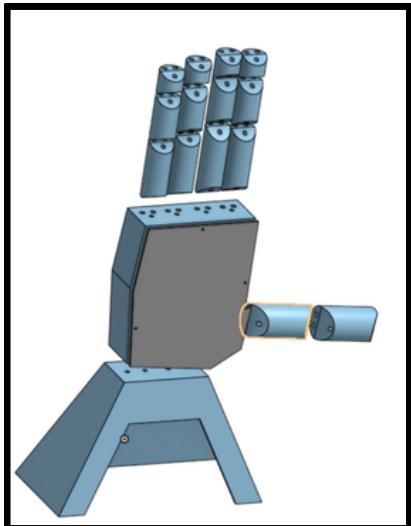
Pour finaliser la chaîne d'information, il faut envoyer les données à un autre script python qui se chargera de piloter les moteurs. Pour ce faire, nous avons fait un script C qui exécute les deux scripts pythons en reliant la sortie standard du programme de calcul à l'entrée standard du programme de pilotage grâce à un fichier temporaire. De cette manière nous pouvons facilement faire communiquer les deux programmes. Nous utilisons un formatage des données en binaire avec *pickle* sur python pour envoyer facilement le résultat du programme de calcul, soit un dictionnaire de dictionnaire.

Pour résumer cette section, grâce à une caméra reliée à une Raspberry nous pouvons filmer la main, et MediaPipe renvoie des positions dans l'espace de points clés sur la main. Nous convertissons ces données en un dictionnaire qui contient pour chaque doigts les angles entre les phalanges. Nous lissons les données pour éviter les sauts. Nous pouvons effectuer les mouvements calculés avec une main en 3D dans Blender pour visualiser le processus. Enfin nous envoyons ces données à un programme qui pilote les moteurs, pour faire les mouvements avec la main robotique.

CONCEPTION MÉCANIQUE

Les enjeux de la conception mécanique étaient multiples.

D'abord, il a fallu déterminer la manière de créer une main robotisée. Au vu de nos ressources, notre temps et nos compétences nous avons décidé de partir sur une main imprimée en PLA avec paume et 4 doigts (divisé en 3 phalanges). Un doigt est rigidifié par un câble tenser de 4 mm de diamètre, pour garantir un bon maintient des doigts à leur état de repos, assurer un retour rapide, mais aussi pour garantir une élasticité lors de sollicitations. Pour les mouvements, nous avons décidé d'utiliser du fil de pêche qui imitera le rôle des tendons fléchisseurs. Nous ne prenons donc pas en compte le mouvement de retour des tendons extendeurs, le qui sera



Premier prototype



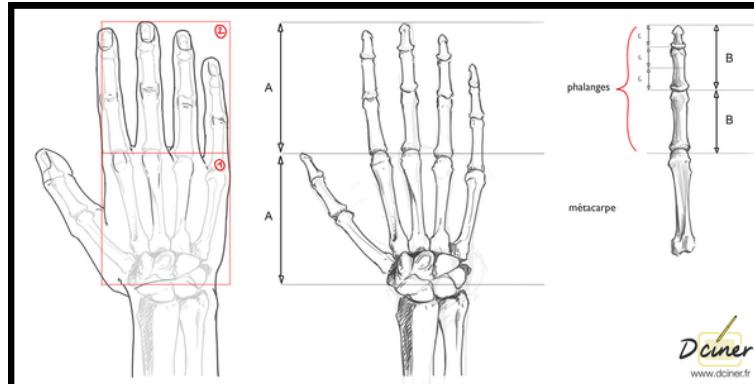
Choix du câblage



Le premier prototype a permis d'avoir un point de départ concret. A partir de là, nous avons pu améliorer empiriquement les prototypes.

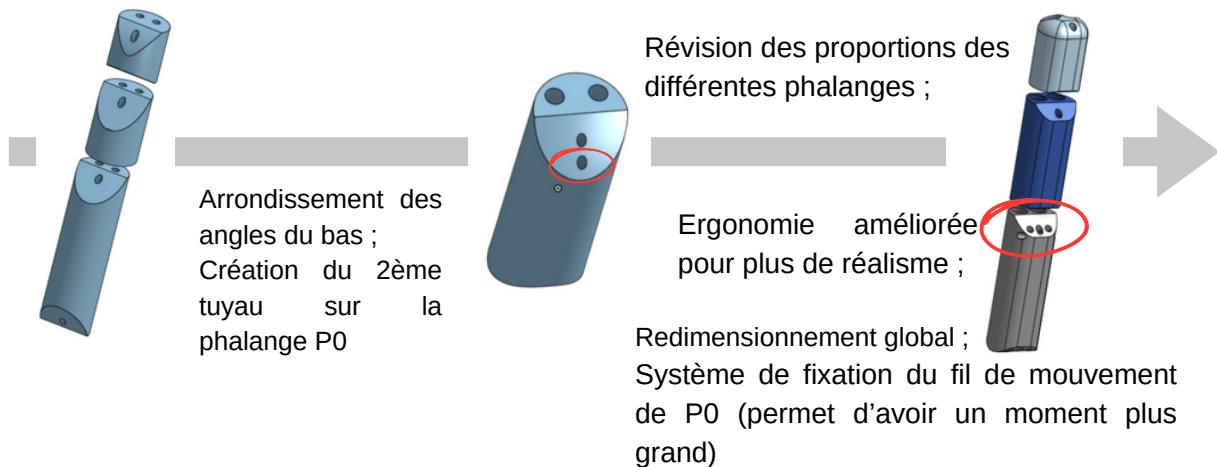
Pour au mieux comprendre les enjeux du design de la main, nous nous sommes appuyés sur des cours de dessin d'anatomie.

CONCEPTION MÉCANIQUE

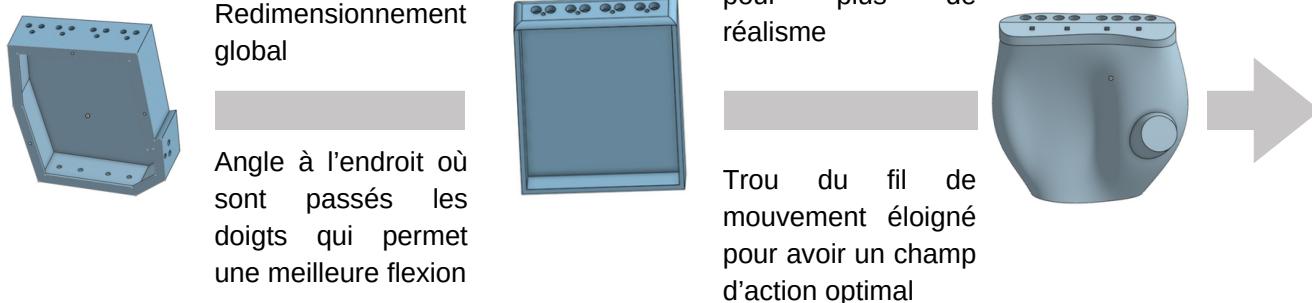


Modèle de dessin anatomique de la main

PROCESSUS D'AMÉLIORATION DES DOIGTS

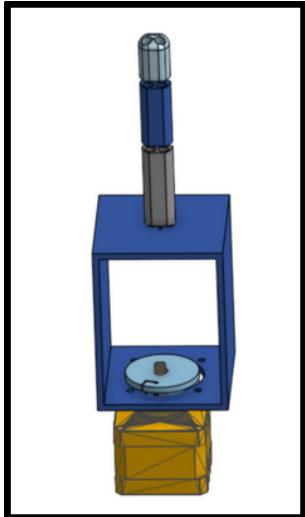


PROCESSUS D'AMÉLIORATION DE LA PAUME



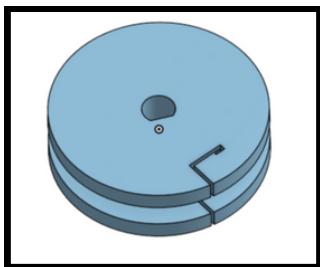
CONCEPTION MÉCANIQUE

STRUCTURE GLOBALE



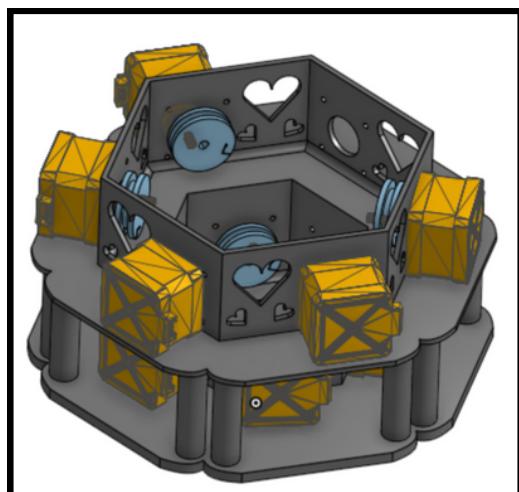
Premier prototype :

Afin d'avancer dans les tests mécaniques, il a fallu créer une première structure simple avec un doigt et un moteur.



Création des bobines :

Afin de créer un lien entre le moteur et les fils, il a fallu créer des bobines. On a imprimé 5 bobines avec des rayons différents pour déterminer à l'aide de test mécanique, le rayon optimal.



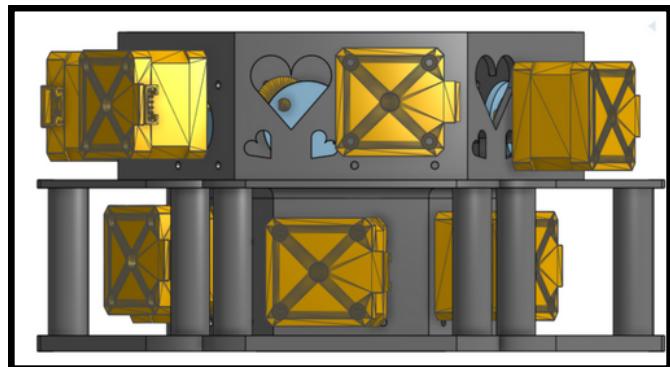
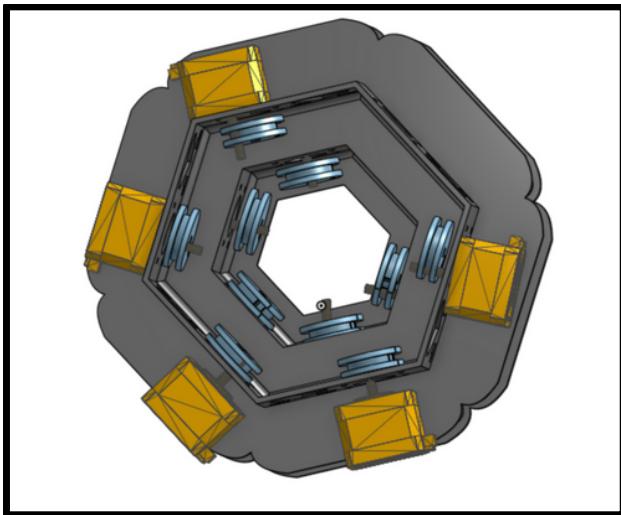
Tour moteurs :

Le but était de trouver un moyen de fixer les moteurs tout en regroupant les bobines sur un axe le plus verticalement possible. D'où l'idée de la tour en forme d'hexagone à 2 étages.

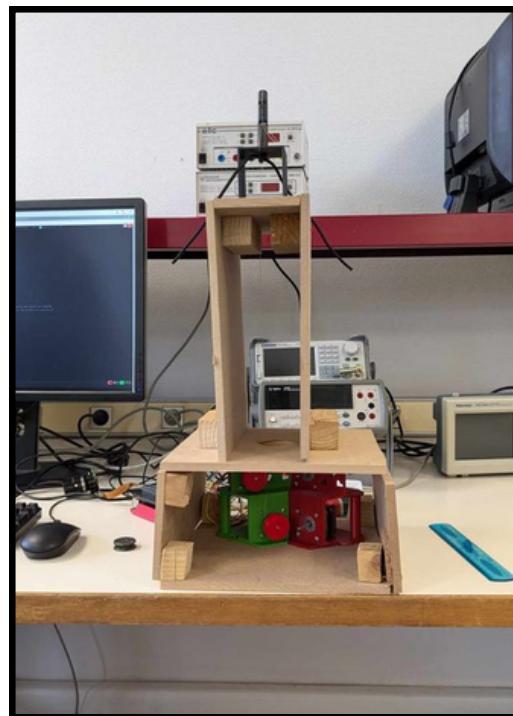
La première version manquait de rigidité d'où l'ajout des colonnes et des congés.

CONCEPTION MÉCANIQUE

STRUCTURE GLOBALE



Tour moteur vu de haut et de côté



Structure finale

PIUSSANCE ET MOTEURS

Pour reproduire les mouvements réalistes d'une main humaine, nous avons opté pour l'utilisation de moteurs afin d'actionner les articulations des doigts et de simuler la flexibilité naturelle et la fluidité des gestes. Pour atteindre un mimétisme aussi précis que possible, nous avons choisi de placer deux moteurs par doigt, permettant ainsi un contrôle fin des angles et des positions de chaque articulation. Grâce à cette configuration, la main robotique est capable de reproduire une large gamme de mouvements, y compris les gestes subtils et complexes, tout en assurant une synchronisation qui rend les gestes plus naturels.

CHOIX DES MOTEURS



28BYJ-48

VS



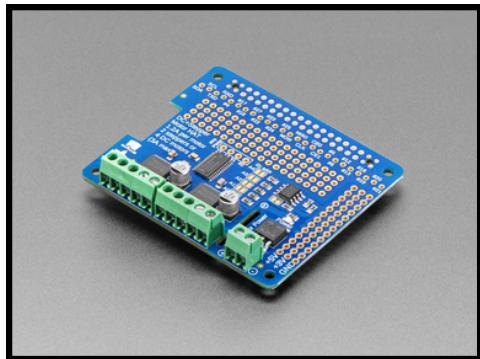
NEMA-17

Pour ce projet, nous avons opté pour le moteur NEMA 17, qui présente des atouts significatifs en termes de couple, de précision et de durabilité, surtout par rapport au 28BYJ-48.

Le couple élevé du NEMA 17 s'avère essentiel pour stabiliser les positions des doigts avec précision et résister aux efforts mécaniques liés aux mouvements répétés. Ce moteur fournit aussi une excellente **RÉSOLUTION**, permettant des ajustements angulaires très fins, indispensables pour reproduire chaque geste de manière réaliste et fluide, sans vibrations indésirables ni décalages. De plus, le NEMA 17 se distingue par sa **robustesse** et sa fiabilité, des qualités primordiales pour une application exigeante comme celle-ci, où les mouvements sont non seulement répétés mais doivent rester fidèles aux gestes humains. Le 28BYJ-48, conçu pour des charges légères, n'aurait pas offert cette stabilité sur le long terme.

PIUSSANCE ET MOTEURS

CHOIX DU CONTRÔLEUR

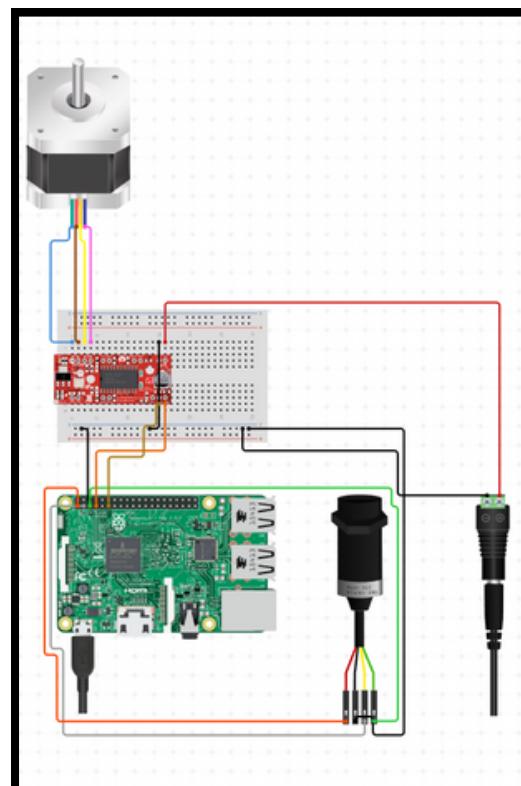


Pour contrôler nos moteurs pas à pas, nous avons choisi l'**Adafruit Stepper Motor HAT** pour Raspberry Pi. Ce contrôleur, parfaitement compatible avec le Raspberry Pi, simplifie la gestion et la programmation des moteurs. Il permet de contrôler plusieurs moteurs simultanément avec une précision élevée grâce à la modulation PWM, essentielle pour reproduire des gestes fluides.

De plus, il fournit un courant stable aux moteurs NEMA 17, nécessaires pour le couple élevé, et intègre des protections contre les surcharges et les courts-circuits, assurant ainsi la sécurité et la fiabilité de notre montage.

CIRCUIT GLOBAL

Le circuit global de ce projet combine plusieurs composants pour détecter et reproduire les mouvements de la main. La caméra Raspberry Pi capture les mouvements de la main humaine en temps réel, puis un modèle d'IA analyse ces images pour identifier les positions des doigts. Les données sont ensuite envoyées au Raspberry Pi, qui utilise le Adafruit Stepper Motor HAT pour piloter les moteurs NEMA 17. Ces moteurs, reliés aux articulations de la main robotique, reproduisent les mouvements détectés avec précision. Une alimentation adaptée assure une puissance constante aux moteurs, garantissant des mouvements fluides et bien synchronisés.



PUISANCE ET MOTEURS

COMMANDÉ DES MOTEURS

coming soon

PISTES D'AMÉLIORATION

PISTES D'AMÉLIORATION

INFORMATION

Pour la partie information, le principal point d'amélioration est la performance du modèle d'intelligence artificielle. Il est très rapide à l'exécution mais cela se fait au détriment de la qualité : nous aurions pu travailler à rendre le modèle plus performant, voir même faire notre propre modèle avec nos données d'entraînement pour le rendre performant dans les conditions dans lequel nous l'utilisons. Mais au vu de la durée du projet, il n'était pas raisonnable de perdre du temps à entraîner un modèle.

Dans l'optique d'avoir un système le plus uniifié possible, nous aurions également pu faire en sorte que le système s'exécute à l'allumage de l'ordinateur. De cette façon, dans le produit fini nous n'avons qu'à brancher la structure et le programme se lance directement, sans avoir à exécuter des programmes à la main

CONCEPTION MÉCANIQUE

La conception mécanique était la partie la plus limitante du projet et est celle qui aurait eu besoin du plus de temps pour être finalisée. Avec plus de temps nous aurions pu finaliser la structure globale, avec tous les doigts assemblés et motorisés. Nous aurions pu également plus terminer le modèle réel de main, que nous n'avons pas réussi à faire en temps imparti. Nous aurions également pu essayer d'implémenter la Raspberry et la caméra directement dans la structure mécanique, éventuellement avec un emplacement pour un petit écran, pour avoir une seule structure autonome qu'il aurait suffit de brancher pour faire fonctionner. Enfin, si tous ces points avaient été faits, nous aurions pu essayer de miniaturiser au maximum et avoir la chaîne de transmission de puissance la plus petite possible : en effet à cause du câblage des doigts nous devons avoir une structure assez haute pour tendre les câbles et pouvoir les manipuler facilement.

PISTES D'AMÉLIORATION

PUISANCE ET MOTEURS

coming soon