Manuel d’utilisation
Projet KERAAL

Equipe: Philippe Baumstimler, Philippe Ngahbi, Ziqi Ma, Mohamed Aziz Ghaddab

ENSTA

IP PARIS

Sommaire

Robot POPPY **3**
Matériel et nomenclature 3
Raspberry Pi 4
Installation POPPY 4
Programmation POPPY 5
Simulateur Coppelia Sim 6
TroubleShooting installation 6

Algorithmes de détection de squelette **7**
Fonctionnement de BlazePose 7
Fonctionnement du code d’analyse des écarts 7
Archives : évaluation des mouvements 8

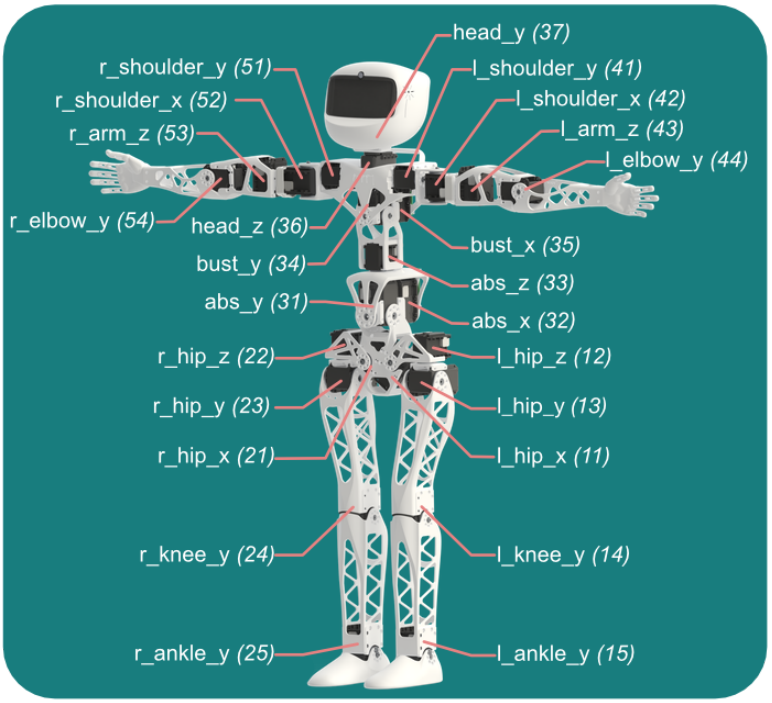
Vérin électrique **9**
Principe de fonctionnement 9
Branchement 9

Robot POPPY

Le projet KERAAL se fonde sur l'utilisation du système robotique POPPY, permettant le contrôle complet d'un robot humanoïde de manière simplifiée. Cette section recense les informations utiles et nécessaires à la compréhension et au fonctionnement du robot POPPY pour assurer le bon déroulement du projet. Vous trouverez ci-dessous une description des informations les plus importantes à savoir, mais voici le lien du site de POPPY recense en détail ce qu'il y a à savoir sur le robot, de son assemblage à son fonctionnement :
https://docs.poppy-project.org/en/getting-started/

a. Matériel et nomenclature

Le robot POPPY a été entièrement assemblé durant l'année et appartient à l'U2IS. Il est donc primordial de le manipuler avec précaution afin d'éviter la casse qui pourrait engendrer un coût et un retard conséquent sur le projet. Une liste du matériel est à votre disposition en version papier, décrivant l'ensemble du matériel et des pièces de rechange disponibles pour le robot, et qui se trouve également disponible sous la forme d'une feuille excel sur le drive ainsi que sur le groupe Teams.



Structure robot POPPY - nomenclature moteurs
datasheet POPPY

L'ensemble des pièces du robot ont une nomenclature propre au constructeur POPPY, qui permet de distinguer chaque pièce les unes des autres, notamment les

moteurs, dont la connaissance du nom est nécessaire pour la partie programmation du robot. Vous retrouverez directement sur chaque moteur du robot une étiquette précisant le nom qui lui est associé, comme indiqué sur le schéma précédent.

b. Raspberry Pi

Le robot POPPY est un robot humanoïde capable d'effectuer et de reproduire des mouvements complexes, grâce à la grande liberté de mouvements que lui procure ses 25 moteurs. Pour contrôler l'ensemble et assurer la liaison homme-machine, une carte électronique Raspberry Pi (RPi) est utilisée dans le système POPPY, situé dans la tête du robot. Vous trouverez ici les détails du fonctionnement de cette dernière.

Pour contrôler la carte de RPi avec votre ordinateur, il faut que les deux fonctionnent sur le même WIFI. Vous pouvez par exemple partager le réseau wifi via votre téléphone, puis vous pouvez alors utiliser les fonctionnalités ssh pour vous connecter à distance. les informations utiles sont ci-dessous:

ssh RPi
host: 192.168.43.69
porte: 22
nom: pi
code: qwer1234

NB: Ce qui est important à noter c'est que pour la connexion au WIFI, vous devrez utiliser un écran. L'écran LCD présent dans le matériel n'était pas fonctionnel au moment où nous avons configuré le robot. A la place, nous avons utilisé directement un écran d'ordinateur présent dans le labo de l'U2IS que nous avons branché directement sur la carte, ainsi qu'un clavier et une souris.

Une fois connecté en ssh vous pourrez contrôler votre raspberry à distance, et vous pouvez commencer à utiliser les fonctionnalités propres au système POPPY déjà installé.

c. Installation POPPY

Cette section n'est utile que si vous souhaitez remettre à zéro la RPi. Dans ce cas, voici les instructions à suivre, que nous rappelons sont disponibles sur le site POPPY. Il faut tout d'abord commencer par installer les différents packages POPPY nécessaires. Pour cela ouvrez un terminal connecté en ssh à la carte et exécutez les commandes suivantes :

```
pip install --upgrade pip setuptools wheels
sudo pip3 install opencv-contrib-python==4.5.3.56
pip3 install opencv-python
```

```
sudo apt install libatlas3-base
pip install -U numpy
pip install pypot
pip install poppy_humanoid
```

Pour vérifier la bonne installation des packages, lancez une console python et vérifiez l'exécution sans erreurs des librairies en effectuant la commande suivante:

```
from poppy_humanoid import PoppyHumanoid
```

Puis pour vérifier la bonne détection du système, téléversez et exécutez le script “check_ports.py” disponible sur le drive dans “*Informatique/Test_function”*:

```
python check_ports.py
```

Vous devez alors obtenir la sortie suivante :

```
/dev/ttyACM1 #Liste des moteurs du bas du corps
[11, 12, 13, 14, 15, 21, 22, 23, 24, 25]
/dev/ttyACM0 #Liste des moteurs du haut du corps
[31, 32, 33, 34, 35, 36, 37, 41, 42, 43, 44, 51, 52, 53, 54]
```

Chaque numéro correspond à un ID moteur. Si vous n’obtenez pas cette liste c’est qu’un ou plusieurs moteurs ne sont pas détectés, vérifiez alors que les connexions entre les moteurs et la carte électronique soient toujours opérationnelles.

Et voilà, vous pouvez maintenant utiliser POPPY !

d. Programmation POPPY

La RPi permet d’exécuter des script en python, et le système POPPY utilise ses propres librairies pour permettre de contrôler le robot. Ainsi, voici une liste des commandes principales à exécuter dans un terminal :

```
from poppy_humanoid import PoppyHumanoid #Importation de la librairie
poppy=PoppyHumanoid() #Initialisation d’un objet POPPY
PoppyHumanoid(camera=’dummy’) #L’option est à ajouter s’il n’y a pas de caméra
poppy = PoppyHumanoid(simulator = "vrep") #si vous souhaitez faire de la simulation sur Coppelia Sim
for m in poppy.motors #Parcours l’ensemble des moteurs de POPPY
m.compliant = False #Bloque le moteur m
m.compliant = True #Débloque le moteur m
m.goto_position(0, 0.5, wait=True) #Moteur m à l’angle 0, vitesse 0.5, s’exécute après que les autres mouvements parallèles soit effectués
```

Sur le drive, dans la section “*Informatique/Test_function”* se trouvent plusieurs fonctions qui nous ont permis de réaliser nos tests et qui vous permettront

d'observer le comportement de POPPY avec les commandes de bases. Une description de ces fonctions tests est disponible dans le rapport de tests.

e. Simulateur Coppelia Sim

Nous n'avons pas utilisé le simulateur cette année. Nous vous renvoyons donc directement au tutoriel disponible sur le site de POPPY et nous vous rajoutons un extrait du manuel d'utilisation de l'année précédente sur le simulateur, disponible dans la section "*Archives*" du drive:

Pour vous connectez au simulateur Coppelia Sim

- Télécharger le logiciel CoppeliaSim Edu

Sur linux :

- ouvrir un terminal et aller dans le dossier téléchargé

- entrer les commandes :

./coppeliaSim.sh

Le logiciel est alors ouvert.

- dans un autre terminal ouvrir python3 et taper :

from pypot.vrep import from_vrep

from pypot.creatures import PoppyHumanoid

poppy = PoppyHumanoid(simulator='vrep')

- taper les commandes voulues : par exemple poppy.r_hip_y.goto_position(-10,2)

le simulateur est ainsi prêt à être utilisé

f. TroubleShooting installation

Voici une liste non exhaustive d'erreurs que nous avons rencontrées lors de l'installation des différents logiciels, avec les solutions que nous avons trouvé pour les résoudre.

- En cas d'exécution infini lors de l'installation des packages avec le message suivant " Building wheel for opencv-contrib-python (pyproject.toml) ... ", voir la page suivante : https://stackoverflow.com/questions/63669752/building-wheel-for-opencv-python-pep-517-runs-forever

- En cas d'erreur de package indiquant une mauvaise installation de Numpy avec la commande `pip3 install numpy`: et renvoyant le message suivant libf77blas.so.**3:** cannot open shared object file**:** No such file **or** directory alors

voir le lien suivant : https://numpy.org/devdocs/user/troubleshooting-importerror.html

- Dans une situation où vous êtes amenés à utiliser l'écran LCD. Si l'écran reste blanc et que la carte ne semble plus s'allumer, se referrer à ce tutoriel: https://www.editions-eni.fr/open/mediabook.aspx?idR=ac500b67d93943f925242f10455cf8bd
- Les tutoriels d'installation des librairies pour l'utilisation de l'écran LCD se trouvent ici : http://www.lcdwiki.com/3.5inch_RPi_Display#Driver_Installation

Algorithmes de détection de squelette

Une grande partie du projet est consacrée à l'informatique, et en particulier à la détection du squelette. Cette section détaille le fonctionnement de l'algorithme de détection de squelette ainsi que du code d'analyse des écarts réalisé cette année, disponibles dans la section *"Informatique/Error/Code"* du drive.

a. Fonctionnement de BlazePose

- Les codes de traitement du squelette sous forme de BlazePose sont dans le dossier `'Error/Code'`.
- Pour faire fonctionner les codes, il faut mettre le dossier des vidéos dans le dossier `'Error'`
- Ouvrir une console dans le dossier `'Error/Code'`, taper `python BlazePose_Skeleton.py` dans la console.
- Les squelettes sont alors écrits dans un document `'.json'` du dossier `'Error/Skeleton_Blazepose'`.

b. Fonctionnement du code d'analyse des écarts

- Les codes de comparaison des écarts sont dans le dossier `'Error/Code'`.
- Pour faire fonctionner les codes, il faut mettre le dossier des squelettes sous forme d'Open Pose et celui sous forme de Kinect dans le dossier `'Error'`. Il faut leur donner les noms `'Skeleton_Openpose'` et `'Skeleton_Kinect'` .
- Ouvrir le fichier `'Error_comparaison.ipynb'` et cliquer le bouton `'Run All'`, le résultat apparait automatiquement, il est aussi stocker dans le dossier `'Error/Results'`

c. Archives : évaluation des mouvements

Nous n'avons pas pu tester les codes d'évaluation fournis par Mme Nguyen et l'ancienne équipe du projet, voici donc un rappel du manuel d'utilisation précédent sur ces fonctions qui pourront vous être utile afin d'évaluer si un exercice de kinésithérapie a été bien réalisé par le patient.

Après avoir généré les fichiers SkeletonSequence grâce à la reconnaissance de mouvements, il faut lancer le script mainLearning.py pour créer un modèle à partir de ces données. Plus il y a de fichiers générés grâce à la reconnaissance de mouvements, plus le modèle est permissif. Dans ce script il faut changer la valeur de la variable trainName qui représente la position du dossier contenant les fichiers que l'on veut traiter. Le modèle est alors sauvegardé dans un fichier model.txt. Pour évaluer un exercice, il faut alors générer le fichier SkeletonSequence correspondant grâce à la reconnaissance de mouvements. Nous allons ici utiliser le script mainEvaluation.py. Il y a aussi différents paramètres permettant de personnaliser l'évaluation :

- *seuil : tolérance de l'évaluation, plus il est proche de 0 plus l'évaluation est stricte*
- *minseuil : valeur par défaut du seuil*
- *registration (valeur par défaut : 1) : indique si l'on veut un alignement temporel ou non*
- *filt (valeur par défaut : 1) : indique si l'on veut filtrer les données ou non*
- *rem (valeur par défaut : 1) : indique si l'on veut retirer le début de la séquence (lorsqu'il n'y a pas de mouvement)*
- *ws (valeur par défaut : 21) : taille des segments*
- *fastDP (valeur par défaut : 1) : si on veut faire un alignement temporel rapide*
- *PrintResults : indique si l'on veut afficher ou non les résultats*
- *dirTrain : dossier d'un fichier correct pour l'alignement temporel*
- *fnameTrain : nom du fichier correct pour l'alignement temporel*
- *dirTest : dossier d'un fichier que l'on veut tester*
- *fnameTest : nom du fichier que l'on veut tester*

Les valeurs trouvées pour l'orientation sont généralement mauvaises même pour des exercices corrects. Cela est normal car l'estimation de l'orientation se fait uniquement par rapport aux positions, ce qui n'est pas totalement exact.

Vérin électrique

a. Principe de fonctionnement

Le fonctionnement du vérin électrique est assez simple. Celui-ci est contrôlé par le sens du courant par lequel il est parcouru. Dans un sens, le vérin rentre et dans l'autre il sort. En buté de vérin, c'est-à-dire quand le vérin est entièrement entré ou sorti, le courant est coupé, permettant d'éviter les surchauffes.

b. Branchement

Le vérin doit être alimenté par une alimentation **12V/5A**. Dans le matériel disponible se trouve une seconde alimentation, la première étant celle du robot POPPY. Nous avons soudé à la connectique du vérin deux fils électriques. Ces derniers sont fixés à un terminal LED à l'aide de vis, adaptateur permettant de brancher l'alimentation directement dessus. Pour tester le vérin il vous suffit de brancher le terminal dans un sens ou dans l'autre et de le brancher à l'alimentation pour faire rentrer ou sortir le vérin.