Projet - Robot parallèle

CPGE – TSI → Sciences Industrielles de l'Ingénieur (SII)

Les applications de pick-and-place a hautes cadences requièrent des caractéristiques très élevées en terme de performances dynamiques, que seuls les robots parallèles sont capables d'atteindre

Présentation du projet

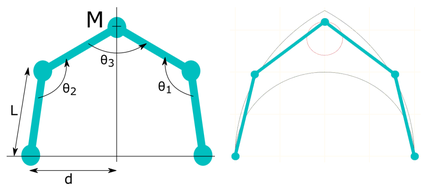
Le robot parallèle est un mécanisme plan qui se compose de quatre bielles liées entre elles par des liaisons pivot.

L'objectif de ce projet est de concevoir et de réaliser la commande d'un robot parallèle. Le système devra localiser en temps réel la position d'un objet et suivre sa position. Le projet se décompose en trois parties :

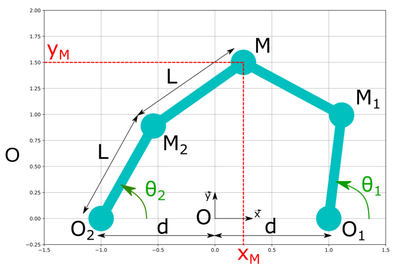
- PARTIE 1 - Conception mécanique
- PARTIE 2 - Analyse et mise en œuvre des actionneurs
- PARTIE 3 - Élaboration de la commande

Conception Mécanique

- Zone de travail - conception du premier prototype



- Détermination des lois de commande



- O

$$\theta_1 = \arcsin\left(\frac{\sqrt{(x_M-d)^2+y_M^2}}{2L}\right) - \phi_1 \text{ ou, } \theta_1 = \pi - \arcsin\left(\frac{\sqrt{(x_M-d)^2+y_M^2}}{2L}\right) - \phi_1$$

avec $\phi_1 = \arcsin\left(\frac{x_M-d}{\sqrt{(x_M-d)^2+y_M^2}}\right)$.

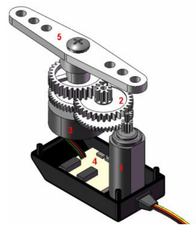
$$\theta_2 = \arcsin\left(\frac{\sqrt{(x_M+d)^2+y_M^2}}{2L}\right) - \phi_2 \text{ ou, } \theta_2 = \pi - \arcsin\left(\frac{\sqrt{(x_M+d)^2+y_M^2}}{2L}\right) - \phi_2$$

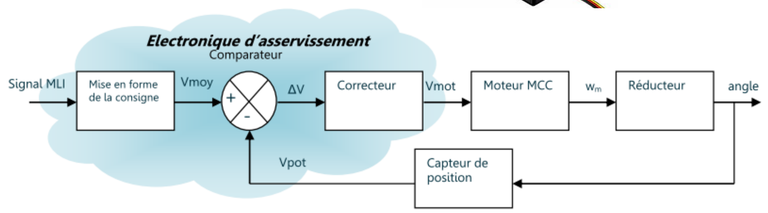
avec $\phi_2 = \arcsin\left(\frac{x_M+d}{\sqrt{(x_M+d)^2+y_M^2}}\right)$

Mise en œuvre des actionneurs

- Choix de l'actionneur :

Les actionneurs utilisés pour cette application sont des servomoteurs : 360 ° 2KG Geekservo





Une consigne angulaire, codée sous forme de signal électrique, est envoyée au servomoteur. Cette consigne est comparée avec une mesure de la position angulaire réelle du servomoteur obtenue par un potentiomètre rotatif (3) situé à l'intérieur du servomoteur. Une commande est alors élaborée par un contrôleur numérique intégré (4) qui alimente la machine à courant continu (1). Le moteur entraine un réducteur (2) en sortie duquel l'utilisateur peut brancher son système mécanique (5).

- Commande via Python – Arduino :

On souhaite réaliser l'intégralité de la programmation du système en langage Python. Pour cela, il est nécessaire de pouvoir communiquer via python avec la carte Arduino sur laquelle sera branché les servomoteurs.


```python
import sys
import time
from pymata4 import pymata4

board = pymata4.Pymata4() # Déclaration de la carte arduino (appelée board)
pos = 90 # Position angulaire souhaitée en degré
Tmin = 500 # Durée minimale de l'impulsion (en microseconde)
Tmax = 2500 # Durée maximale de l'impulsion (en microseconde)
board.set_pin_mode_servo(2,Tmin,Tmax) # Initialisation du servomoteur
board.set_pin_mode_servo(4,Tmin,Tmax) # Initialisation du servomoteur

try:
    board.servo_write(4, 22) # Envoi de la consigne pos au servomoteur
    board.servo_write(2, 22) # Envoi de la consigne pos au servomoteur
    time.sleep(1) # pause de 1 seconde
    board.servo_write(4, 45) # Envoi de la consigne pos au servomoteur
    board.servo_write(2, 45) # Envoi de la consigne pos au servomoteur
    time.sleep(1) # pause de 1 seconde
except KeyboardInterrupt: # Interruption
    board.shutdown() # Arrêt de la communication avec la carte
    sys.exit(0)
```

Élaboration de la commande

- Analyse d'images en temps réel

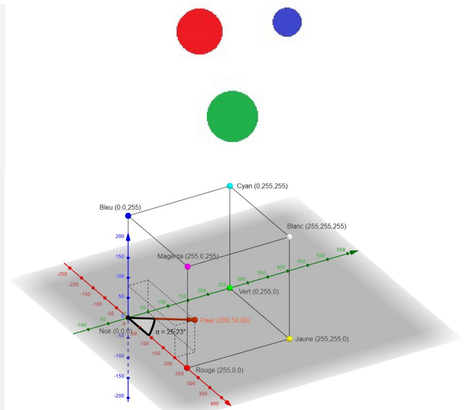
L'objectif est de détecter un objet par sa couleur en temps réel à partir d'images provenant d'une webcam. On utilise la bibliothèque opencv-python.

```python
# import the opencv library
import cv2
import numpy as np
import matplotlib.pyplot as plt

# define a video capture object
vid = cv2.VideoCapture(0)

while(True):
    # Capture the video frame
    # by frame
    ret, frame = vid.read()
    cv2.imshow('frame', frame)
    # the 'q' button is set as the
    # quitting button you may use any
    # desired button of your choice
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# After the loop release the cap object
vid.release()
# Destroy all the windows
cv2.destroyAllWindows()
```



$$\langle \vec{c}, \vec{r} \rangle = \|\vec{c}\| \times \|\vec{r}\| \times \cos(\alpha) = c_1 r_1 + c_2 r_2 + c_3 r_3$$

```python
while(True):
    # Capture the video frame
    # by frame
    ret, frame = vid.read()
    N_L,N_C,c = np.shape(frame)
    X, Y, n = 0, 0, 0
    for i in range(0,N_L,10):
        for j in range(0,N_C,10):
            if Cos_angle_R(frame[i,j]) > seuil :
                n, X, Y = n + 1, X + i, Y + j
    if n != 0:
        X = X/n
        Y = Y/n
        Xm.append(X)
        Ym.append(Y)
    Xm_moy, Ym_moy = Moyenne_glissante(Xm,Ym)
    cv2.circle(frame,(Ym_moy,Xm_moy),2,(0,0,255),5) # Point détecté
```

- Des coordonnées de l'objet aux angles des moteurs