Documentation ROBOT-Module d'ouverture :PRM

BENNANI Boubker

IMT Nord Europe
École Mines-Télécom
IMT-Université de Lille

Contents

1 Fonctionnement du code test5.py

Le code fourni est écrit en Python et implémente une série de fonctions pour contrôler la position et l'orientation d'un robot dans l'environnement de simulation **Coppélia**

1.1 orientation()

La fonction orientation() permet de récupérer l'orientation actuelle du robot. Elle récupère les angles d'Euler correspondant aux axes x et y et détermine l'orientation du robot en fonction de ces angles. Si l'angle correspondant à l'axe x est plus petit que celui correspondant à l'axe y, le robot se déplace selon l'axe x, sinon il se déplace selon l'axe y.

1.2 move_y(dy, velocity=5,angle=0.5)

La fonction move_y(dy, velocity=5,angle=0.5) permet de déplacer le robot le long de l'axe y. Elle prend en paramètre la distance à parcourir (dy), la vitesse de déplacement (velocity) et l'angle de rotation nécessaire pour que le robot se déplace dans la bonne direction (angle). La fonction utilise la fonction rotate(angle, velocity=10) pour effectuer une rotation si nécessaire. La position initiale est stockée dans une variable globale y0 qui est mise à jour après chaque déplacement.

1.3 move_x(dx, velocity=5,angle=0.5)

La fonction move_x(dx, velocity=5,angle=0.5) est similaire à move_y, mais déplace le robot le long de l'axe x.

1.4 rotate(angle, velocity=10)

La fonction rotate(angle, velocity=10) permet de faire tourner le robot d'un certain angle (angle) à une certaine vitesse (velocity). La fonction utilise la fonction simxGetObjectOrientation pour récupérer l'orientation actuelle du robot et la fonction simxSetJointTargetVelocity pour contrôler les moteurs du robot et effectuer la rotation.

1.5 go(x, velocity=1)

La fonction go(x, velocity=1) permet de déplacer le robot en ligne droite sur l'axe x jusqu'à atteindre une certaine position x. Elle utilise la fonction simxGetObjectPosition pour récupérer la position actuelle du robot et la fonction simxSetJointTargetVelocity pour contrôler les moteurs du robot et le faire avancer.

1.6 rotatediagsens1()

Cette fonction fait tourner le robot sur lui-même ($\frac{\pi}{4}$) dans le sens anti-horaire en activant les moteurs des roues diagonales dans des directions opposées.

1.7 rotatediagsens2()

Cette fonction fait tourner le robot sur lui-même$-\frac{\pi}{4}$) dans le sens horaire en activant les moteurs des roues diagonales dans des directions opposées.

2 Chemin choisi

```
0000000----000000
0000000-11-000000
0000000-11-000000
0000000----000000
0000000000000000
liste des sommets = [0, 1, 2, 3, 4, 5, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 28, 29, 30, 33, 34, 35, 36, 37, 38, 39, 45, 46, 47, 50, 51,
 52, 53, 54, 55, 56, 62, 63, 64, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 9
7, 98, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130,
 131, 132, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163,
164, 165, 166, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 1
97, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 232, 233, 23
4, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 266, 267, 268, 269, 270, 271
, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288]
ligne de depart = 8
colonne de depart = 8
ligne d'arrivee = 0
colonne d'arrivee = 16
We found the following best path with a value of 19.65685424949238.
144 -> 145 -> 146 -> 147 -> 165 -> 183 -> 201 -> 202 -> 186 -> 169 -> 152 -> 135 -> 118 -> 101 -> 84 -> 67 -> 50 -> 33 -> 16
000000-----00000*
000000-111-000--*
000000-111-000--*
000000-----000--*
00000000000000--*
00000000000000--*
00000000000000--*
00000000000000--*
00000000****00--*
00000000000*0--*
0000000000000*--*
00000000000000**0
0000000----000000
0000000-11-000000
0000000-11-000000
0000000----000000
0000000000000000
liste des coordonnees de cellules a rejoindre : [[8, 8], [8, 11], [11, 14], [11, 15], [10, 16]]
PS C:\Users\o\Downloads\ROMOB>
```

Le code commence par une initialisation qui configure la position du robot et définit la vitesse de ses moteurs à zéro. Ensuite, le robot est programmé pour effectuer une série de mouvements qui sont les suivants :

orientation(): cette fonction permet de faire tourner le robot de sorte qu'il se dirige vers la droite.

move_y(0.4,1): le robot avance d'une distance de 0.4 unités en y (vers l'avant) à une vitesse de 1 unité/seconde.

move_x(0.6,1) : le robot avance d'une distance de 0.6 unités en x (vers la droite) à une vitesse de 1 unité/seconde.

rotatediagsens2() : le robot effectue une rotation de 45 degrés dans le sens antihoraire (rotation diagonale sens 2).

go(1.6,1) : le robot avance jusqu'à ce que sa coordonnée x atteigne 1.6.

rotatediagsens1() : le robot effectue une rotation de 45 degrés dans le sens horaire (rotation diagonale sens 1).

move_x(0.46+0.92+3*0.23,1) : le robot avance d'une distance de 1.835 unités en x.

rotatediagsens1() : le robot effectue une rotation de 45 degrés dans le sens horaire (rotation diagonale sens 1).

go(0.46+0.92+3*0.23+0.2,1) : le robot avance jusqu'à ce que sa coordonnée x atteigne 2.525.

rotatediagsens2() : le robot effectue une rotation de 45 degrés dans le sens antihoraire (rotation diagonale sens 2).

move_x(0.46+0.92+3*0.23+0.2,1) : le robot avance d'une distance de 2.725 unités en x.

move_y(2.2,1.5) : le robot avance d'une distance de 2.2 unités en y (vers l'avant) à une vitesse de 1.5 unités/seconde.

Entre chaque mouvement, le robot est programmé pour faire une pause de 0.2 secondes, sauf dans le cas de la fonction **time.sleep(0.1)** où la pause est de 0.1 seconde.