TESTPLAN VERPLAATSEN

Experimenteel onderzoek voor het kiezen van een ideale manier van het verplaatsen van bloemen

Auteur: Ivo Bruinsma

Inlevermoment: 50% oplever-set

Docenten: Anne de Gier & Wouter Volders

4MT
Nijverheidstraat 10
2751GR Moerkapelle
079-5932121
www.4mt.nl

Revisiebeheer

Revisie	Auteur	Datum	Wijziging
	Ivo Bruinsma	14-10-2025	Document aanmaken testen uitvoeren en in vullen
	Ivo Bruinsma	10-11-2025	Ik instanties verwijderen

Bij het maken van een nieuwe versie, eerst kopie maken van oude versie en onder nieuwe bestandsnaam opslaan. Dan versienummers in document aanpassen en revisiebeheer invullen.

Voordat wijzigingen gemaakt kunnen worden, dienen de vorige wijzigingen eerst geaccepteerd te worden (Controleren -> Accepteren -> Alle wijzigingen accepteren). Zet vervolgens ‘Wijzigingen bijhouden’ aan.

Inhoudsopgave

Context 3
Verschillen tussen MoveJ en MoveCart 4
Test 5
Doel test 5
Test opstelling & benodigdheden 5
De test 5
Geteste onderdelen 5
Test resultaten 6
Conclusie 6
Bijlage 7

Context

Voor het bewegen van de robot arm kan er gebruik worden gemaakt van verschillende bewegingsfuncties. Er zijn 2 verschillende soorten: Cartesian space, via een coördinaten systeem (x,y,z,rx,ry,rz), en via joint space, hier wordt de hoek van elke as vast gesteld. Het verschil tussen de twee functies is dat bij de Cartesian space je de exacte coördinaten waar de kop van de arm komt te staan meegeeft. Bij de Joint space geef je de hoeken van elke as mee. Dit heeft beide zo zijn voor- en nadelen. Via de testen welke worden uitgevoerd is er een besluit genomen van welke functie er wordt gebruikt voor het aansturen van de arm. De test zal worden uitgevoerd doormiddel van het verplaatsen van een pen. Dit omdat er nog geen geschikte grijper is welke de hortensia's stabiel en betrouwbaar kan oppakken.

Verschillen tussen MoveJ en MoveCart

De verschillen tussen MoveJ en MoveCart voor het schrijven van de code niet gigantisch. Beide functies:

```
1. robot.MoveJ(joint_pos=joint_pos[J1, J2, J3, J4, J5, J6], tool=tool, blendT=blendT, vel=vel,
user=user)
```

```
1. robot.MoveCart(desc_pos=cords[x, y, z, rx, ry, rz], tool=tool, blendT=blendT, vel=vel,
user=user)
```

Zoals te zien is verschillen de twee functies niet tot nauwelijks. Het verschil zit zich in de werking van de twee. Bij MoveJ worden alle servomotors aangestuurd tot een bepaalde hoek. Dit geeft veel controle over hoe de arm gepositioneerd wordt. Bij de MoveCart worden de coördinaten waar de kop van de robot zich zal moeten bevinden doorgegeven. Ook wordt er meegegeven op welke hoek de kop zal staan te opzichten van dit punt. Dit zorgt voor meer precisie en minder complexiteit als je de coördinaten van de plek waar de arm naar toe moet al weet. Beide functies hebben dus eigen voordelen tegenover de andere. Maar ook nadelen; bij het gebruik van MoveJ wordt een verplaatsing op de z-as erg gecompliceerd. Dit omdat voor een verplaatsing op de z-as meerdere motoren moeten worden verplaatst bij MoveCart wordt dit uitgerekend door de arm zelf. Het grootste nadeel van MoveCart is het aansturen van de kop. Doordat het aansturen van het kopstuk werkt in graden is dit moeilijk onder de knie te krijgen.

Test

Om de twee functies met elkaar te vergelijken wordt er een experiment uitgevoerd waarbij de prestaties van beide functies met elkaar wordt vergeleken.

Doel test

Het doel van deze test is om erachter te komen welke van de twee bewegingsfuncties gebruikt zal worden voor het aansturen van de arm.

Test opstelling & benodigdheden

Om deze test zo eerlijk mogelijk uit te voeren zullen beide functies dezelfde bewegingen uitvoeren welke vergelijkbaar zullen zijn met die van het eindproduct worden verwacht. De test-opstelling is als volgt:

De robot arm zal worden opgesteld zoals deze volgens de instructies van Fairino beschreven is. Hierna moet de arm worden gepositioneerd in een open ruimte waarbij deze vrijuit kan bewegen. Er zal een pen worden neergelegd op 60cm links van de arm en 15,5cm onder het voetstuk, de arm is gemonteerd op een kleine pallet met een houtplaat erboven, de arm zal deze oppakken en verplaatsen naar een positie 45 cm voor het voetstuk van de arm en op gelijke hoogte van dit voetstuk. Deze punten zijn gelijk voor beide bewegingsfuncties.

De benodigdheden van de test zijn:

- Fairino FR5
- Fairino Python SKD Version 2.1.6
- JODELL ERG J1.0, gemonteerd op het kopstuk van de arm en aangesloten op de output port op de arm.
- Simpele balpen
- Open ruimte met diameter van minimaal 1800 mm
- Test codes
- Stopwatch

De test

De taak van beide functies zal het zelfde zijn. Er zal een pen worden opgepakt en verplaatst naar een specifieke locatie. Zodra de pen wordt opgepakt zal de arm eerst 30 cm omhoog gaan. Dit omdat de arm zal worden gebruikt op een sorteerband met meerdere bloemen en als deze niet eerst omhoog zal gaan zal de arm meerdere bloemen van de band af schuiven. Als de arm de pen op de eindlocatie heeft neergelegd zal deze weer 30 cm omhoog gaan om te voorkomen dat bij vertrek de pen of bloem zal worden verplaatst.

Geteste onderdelen

De twee functies zullen met elkaar vergeleken worden op twee punten: snelheid en complexiteit. Voor de snelheid worden alle stappen getimed en met elkaar vergeleken. Voor de complexiteit zal er gekeken worden naar de stappen welke ondernomen moesten worden om de robot voor te bereiden voor de test.

Test resultaten

De stappen welke doornomen moesten worden voor het voorbereiden van de test was het makkelijkst voor de MoveCart functies. Dit omdat voor de 30cm verplaatsing er geen extra punten opgenomen moesten worden. Het verkrijgen van de punten heb ik gaan doormiddel van de DragTeachSwitch functie. Deze code geeft beide waardes terug en kunnen worden overgenomen. Voor het opnemen van de 30cm verticale verplaatsing voor de MoveJ functie heb ik eerst de MoveCart functie gebruikt. Dit omdat het zekerheid geeft dat het punt echt loodrecht boven het oude coördinaat ligt. Via de webapp zijn de hoeken van elke as verkregen en deze ingevuld voor de test code. Deze stappen maakte het opzetten van de test voor de MoveJ functie extra complex daarom krijgt de MoveCart het voordeel voor de complexiteit.

Voor de snelheid heb ik de test 3 maal gedaan en het gemiddelde in de tabel hieronder gezet.

	MoveJ	MoveCart
Oppakken	2.41 seconden	3.56 seconden
Van oppak punt tot afleverpunt	1.24 seconden	3.09 seconden
Neerleggen	2.59 seconden	3.58 seconden

Conclusie

Beide functies doen de taken zoals dit bedoelt is. De MoveJ functie is over het algemeen sneller dan de MoveCart functie. Helemaal in het verplaatsen van 30cm boven het oppak punt en boven het afleverpunt. Dit maakt de MoveJ functie ideaal voor het verplaatsen van de bloemen tussen deze punten. Alleen het precieze werk is niet gemaakt voor de MoveJ functie. De MoveCart is makkelijker op te zetten voor het verplaatsen op de z-as, ook voor het gebruik van sensoren en/of camera's welke coördinaten zullen doorgeven aangezien deze coördinaten alleen doorgegeven hoeven te worden. Beide functies hebben belangrijke voor- en nadelen daarom ga ik ook beide functies gebruiken. De MoveJ zal worden gebruikt voor de grotere verplaatsingen zoals tussen het oppak en het aflever punt, waar de MoveCart zal worden gebruikt voor het oppakken en neerleggen van de bloem.

Bijlage

Code MoveJ

```
from fairino import Robot
import time

robot = Robot.RPC('192.168.58.2')
robot.ActGripper(2,0)
time.sleep(1)
robot.ActGripper(2, 1)
time.sleep(1)

joint_pos = [
    [9.724, -141.33, -101.05, -27.301, 90.259, -122.14], # pick up
point
    [-87.29, -110.09, -126.18, -33.232, 91.852, -137.205],    # Drop
point
    [9.724, -112.39, -94.249, -63.036, 90, -122.14],   # +30 cm
above pick up point
    [-87.29,-86.502, -101.31,-81.686,91.851,-137.205] # +30 cm above
drop point
]


# basic setup
vel = acc = ovl = 100
tool = user = 0
blendT = -1.0
robot.SetSpeed(100)

# Open gripper
robot.MoveGripper(2, 0, 100, 8, 10000, 0, 0, 0, 0, 0)

# Move to 30cm above the pickup point
robot.MoveJ(joint_pos=joint_pos[2], tool=tool, blendT=blendT,
vel=vel, user=user)

# Move to pickup point and close gripper
rtn = robot.MoveJ(joint_pos=joint_pos[0], tool=tool, blendT=blendT,
vel=vel, user=user)
print(rtn)
robot.MoveGripper(2, 78, 100, 8, 10000, 0, 0, 0, 0, 0)

# Return to 30cm above the pickup point
rtn = robot.MoveJ(joint_pos=joint_pos[2], tool=tool, blendT=blendT,
vel=vel, user=user)
print(rtn)

# Move to 30cm above drop point
rtn = robot.MoveJ(joint_pos=joint_pos[3], tool=tool, blendT=blendT,
vel=vel, user=user)
```

```
print(rtn)

# Move to drop point and open gripper
rtn = robot.MoveJ(joint_pos=joint_pos[1], tool=tool, blendT=blendT,
vel=vel, user=user)
print(rtn)
rtn = robot.MoveGripper(2, 0, 100, 8, 10000, 0, 0, 0, 0, 0)
print(rtn)

# Move back to 30cm above the drop point
rtn = robot.MoveJ(joint_pos=joint_pos[3], tool=tool, blendT=blendT,
vel=vel, user=user)
print(rtn)
```

Code MoveCart

```
from fairino import Robot
import time

robot = Robot.RPC('192.168.58.2')
robot.ActGripper(2,0)
time.sleep(1)
robot.ActGripper(2, 1)
time.sleep(1)

cords = [
    # 375 -> -275
    [625, 4.438, -32.246, -179, -0.404, 41.868], # pickup point
    [-76.302, -472.35, 123.796, -179.11, -1.699, -40.09], # drop
point
]

# basic setup
tool = user = 0
blendT = -1.0
flag = 0

rtn = robot.MoveGripper(2, 0, 100, 8, 10000, 0, 0, 0, 0, 0)
print(rtn)

# Calculate and move to 30cm above pickup point
cords[0][2]+=300
rtn = robot.MoveCart(desc_pos=cords[0], tool=tool, blendT=blendT,
user=user)
print(rtn)

#Calculate and move to pickup point
cords[0][2]-=300
rtn = robot.MoveCart(desc_pos=cords[0], tool=tool, blendT=blendT,
user=user)
print(rtn)

# close gripper and move back up to 30cm above pickup
rtn = robot.MoveGripper(2, 78, 100, 8, 10000, 0, 0, 0, 0, 0)
print(rtn)
cords[0][2]+=300
rtn = robot.MoveCart(desc_pos=cords[0], tool=tool, blendT=blendT,
user=user)
print(rtn)

# Move to 30cm above drop point
cords[1][2]+=300
rtn = robot.MoveCart(desc_pos=cords[1], tool=tool, blendT=blendT,
user=user)
print(rtn)
```

```
# Move to drop poit and open gripper
cords[1][2]-=300
rtn = robot.MoveCart(desc_pos=cords[1], tool=tool, blendT=blendT,
user=user)
print(rtn)
robot.MoveGripper(2, 0, 100, 8, 10000, 0, 0, 0, 0, 0)
print(rtn)

# Move 30cm above the drop point
cords[1][2]+=300
rtn = robot.MoveCart(desc_pos=cords[1], tool=tool, blendT=blendT,
user=user)
print(rtn)
```

DragTeachSwitch

```
from fairino import Robot
import time

robot = Robot.RPC('192.168.58.2')
robot.DragTeachSwitch(1)
try:
    while True:
        print("Current position")
        print(robot.GetActualJointPosDegree())
        print(robot.GetActualTCPPose())
        time.sleep(1)
except KeyboardInterrupt:
    robot.DragTeachSwitch(0)
    robot.CloseRPC()
```