ROS2 협동-2

Sports Stacking 구현 및 시연

C-1조

C-1 DR-01458 서재원
DR-01233 김근제
DR-01184 김차훈
DR-01325 정의재

발표자료

2024.11.25

CONTENTS

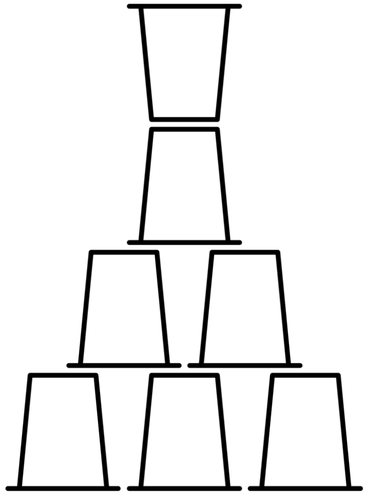
ROS2 협동 – 2 Sport Stacking

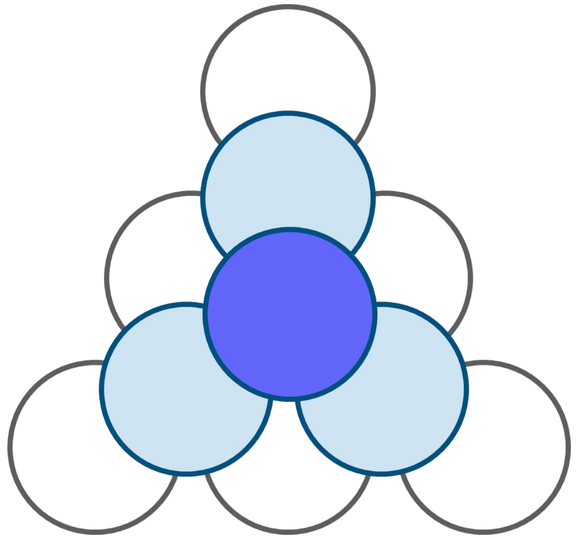
로봇 팔을 활용한 Sport stacking + 투상도(Projection)

Manipulator (M0609)

ROS2

측면도, 평면도





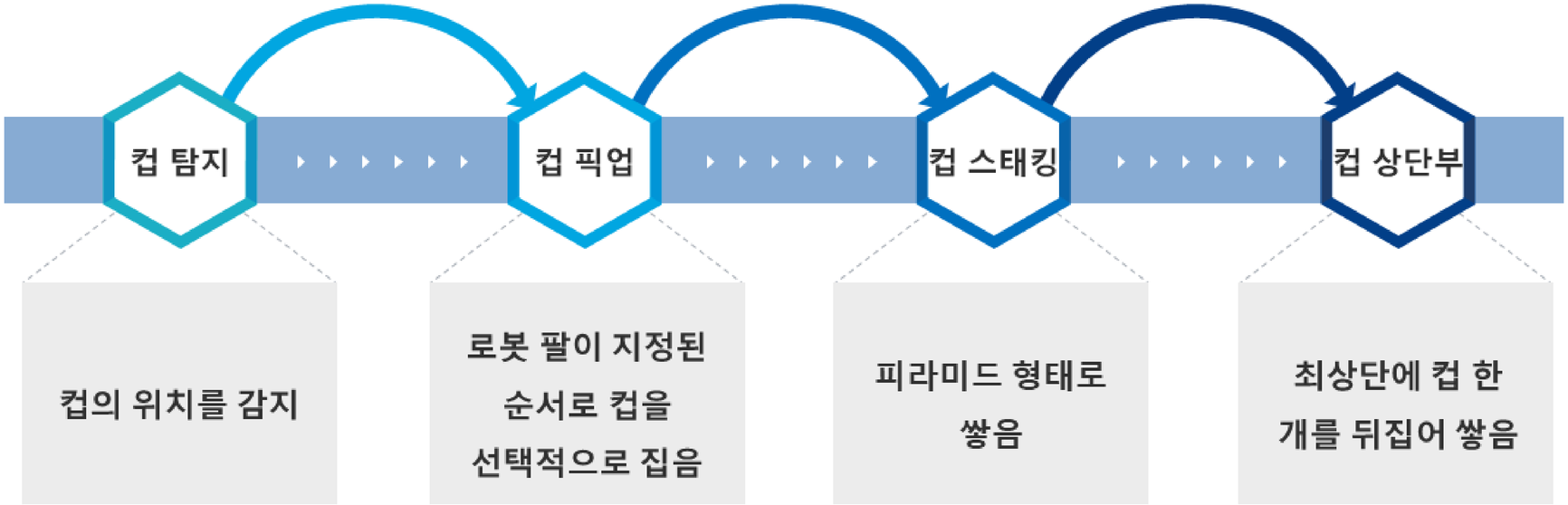
ROS2 활용 함수

환경 설정 함수
힘 제어 관련 함수
이동 관련 함수
좌표 및 힘 계산 함수

No.	사용한 함수	기능
1	set_tool	로봇에 장착된 툴을 설정하는 함수
2	set_tcp	TCP(Tool Center Point)를 설정하는 함수
3	release_compliance_ctrl	힘 제어(Compliance Control)를 해제하는 함수
4	release_compliance_ctrl	힘 제어(Compliance Control)를 해제하는 함수
5	check_force_condition	로봇이 설정된 힘 조건을 만족하는지 확인하는 함수
6	task_compliance_ctrl	작업의 힘 제어 모드를 설정하는 함수
7	set_desired_force	봇에 대해 원하는 힘 값을 설정하는 함수

No.	사용한 함수	기능
8	movej	조인트 공간에서 로봇을 이동시키는 함수
9	movel	선형 이동(linear movement)을 수행하는 함수
10	get_current_posx	로봇의 현재 X 좌표를 얻는 함수
11	trans_1d	좌표 변환을 수행하는 함수
12	fkin	Forward Kinematics (전방 기구학) 계산을 수행 (posj to posx) 함수
13	ikin	Inverse Kinematics (역기구학) 계산을 수행하는 (posx to posj) 함수

초기 시나리오


컵 탐지
컵 픽업
컵 스태킹
컵 상단부
컵의 위치를 감지
로봇 팔이 지정된 순서로 컵을 선택적으로 집음
피라미드 형태로 쌓음
최상단에 컵 한 개를 뒤집어 쌓음

학습경험을 통한 문제점 파악 및 개선

시행착오 01

최상단 컵 작업

- 최상단 컵을 상하반전 시에 수직으로 잡을 경우 기구학적으로 불가함

→ 최상단 컵 회전시에는 수평으로 그립

시행착오 02

작업 반경

- Z축으로 움직이기에. 기존 작업반경 유지 시, 특이점(Singularity) 도달

→ 매니퓰레이터 쪽으로 가깝게 쌓도록 작업공간 설정

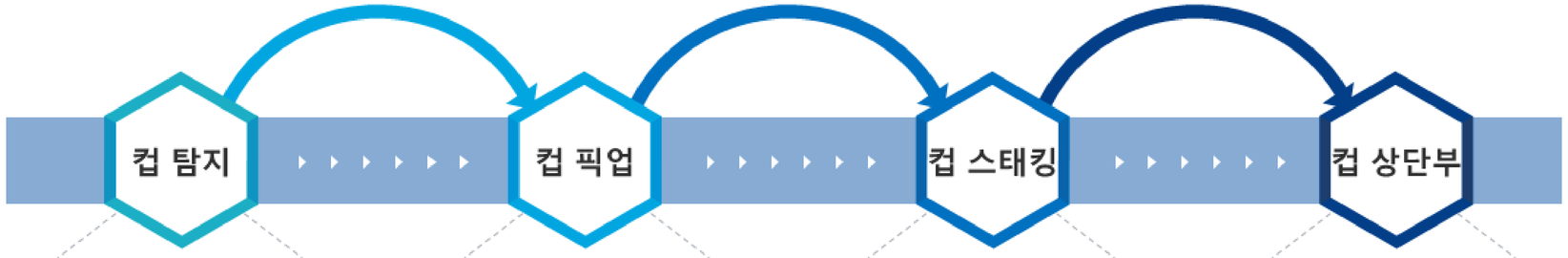
시행착오 03

컵 그립 – 스택 자세

- 컵을 수평으로 그립 – 스택 시에 메뉴퓰레이터가 최하단의 컵 작업 시, 바닥에 닿을 수 있음

→ 나머지 컵들을 컵의 수직방향으로 그립 - 스택

최종 시나리오



컵의 위치를 감지

- 지정된 순서로 컵을 집음
- 최상단에 놓일 컵을 상하반전
- 수평방향 그립

- 피라미드 형태로 쌓음
- 작업 반경 고려, 수직 방향으로 그립 후 스택

최상단에 컵 한 개를 뒤집어 쌓음

Code Review(1)

```
lass UpSideDownSpace:
    def __init__(self,upside_pos,moveing_pos):
        self.calculate_upside_down(upside_pos)
        self.calculate_move_spcace(moveing_pos)

    def calculate_upside_down(self,cur_pos):
        '''
        맨처음 pos를 기준으로 여러 포즈 계산
        1. pos1: down_pos에서 100mm 위로 이동
        2. pos2: pos1에서 100mm 위로 이동
        '''
        pos1 = cur_pos.copy() # 초기 위의 위치
        sol_space = 2 # 계산한 결과
        pos2 = trans_1d(pos1,'posx',DR_AXIS_Z,-100) # 100mm 아래로 이동
        pos3 = trans_1d(pos2,'posx',DR_AXIS_Z, 100) # 100mm 위로 이동
        pos3j = ikin(pos3, sol_space) # pos3의 joint space -- 주의 요망
        pos4 = trans_1d(pos3j,'posj',JOG_AXIS_JOINT_6,180) # 180도 회전
        pos4x = fkin(pos4, sol_space) # pos4의 x space
        pos5 = trans_1d(pos4x,'posx',DR_AXIS_Z,-57) # 100mm 밑으로 이동
        pos6  = trans_1d(pos5,'posx',DR_AXIS_Y,-150) # 150mm 뒤로 이동

        self.upside_down_space = [pos1,pos2,pos3,pos4,pos5,pos6]
```

```
def calculate_move_spcace(self,cur_pos):
    '''
    뒤집어진 컵 움직이기
    '''
    pos1 = cur_pos.copy() # 초기 위치
    pos2 = trans_1d(pos1,'posx',DR_AXIS_Z,-20) # 40mm 아래로 이동
    pos3 = trans_1d(pos2,'posx',DR_AXIS_Z,20) # 40mm 위로 이동
    pos4 = trans_1d(pos3,'posx',DR_AXIS_X,-200) # 200mm 오른쪽으로 이동
    pos5 = trans_1d(pos4,'posx',DR_AXIS_Z,-205) # 205mm 아래로 이동
    pos6 = trans_1d(pos5,'posx',DR_AXIS_Z,150) # 150mm 위로 이동
    self.move_space = [pos1,pos2,pos3,pos4,pos5,pos6]
```

상세 설명

- 컵을 뒤집기 위한 point 클래스로 크게 upside_down, move_space로 분류함
- upside_down은 뒤집는 과정에서의 pos, move_space는 뒤집은 컵을 옮기기 위한 pos들을 저장

Code Review(2)

```python
class UpSideDownController:
    def __init__(self,space:UpSideDownSpace,homing_pos):
        self.upside_down_space = space.upside_down_space
        self.move_space = space.move_space
        self.homing_pos = homing_pos
    def main(self):
        self.grip_process()
        self.upside_down()
        self.put_down()
        self.returning()
        movej(self.homing_pos,vel=VELOCITY,acc=ACC)
        self.moving()

    def grip(self): ...

    def ungrip(self): ...

    def grip_wide(self): ...

    def grip_process(self):
        movel(self.upside_down_space[0],vel=VELOCITY,acc=ACC) # 컵 피쳐 위치로 이동
        movel(self.upside_down_space[1],vel=VELOCITY,acc=ACC) # 아래로 이동
        wait(1.0)
        self.grip_wide()
        wait(1.5)

    def upside_down(self):
        movel(self.upside_down_space[2],vel=VELOCITY,acc=ACC) # 위로 이동
        movej(self.upside_down_space[3],vel=VELOCITY,acc=ACC) # 180도 회전

    def put_down(self):
        movel(self.upside_down_space[4],vel=VELOCITY,acc=ACC) # 아래로 이동
        self.ungrip()
        wait(1.5)
```

```python
def returning(self):
    movel(self.upside_down_space[5],vel=VELOCITY,acc=ACC) # 뒤로 이동

def moving(self):
    movel(self.move_space[0],vel=VELOCITY,acc=ACC) # 초기 위치로 이동
    movel(self.move_space[1],vel=VELOCITY,acc=ACC) # 아래로 이동
    self.grip()
    wait(1.0)
    movel(self.move_space[2],vel=VELOCITY,acc=ACC) # 위쪽으로 이동
    movel(self.move_space[3],vel=VELOCITY,acc=ACC) # 오른쪽으로 이동
    movel(self.move_space[4],vel=VELOCITY,acc=ACC) # 아래로 이동
    self.ungrip()
    wait(1.0)
    movel(self.move_space[5],vel=VELOCITY,acc=ACC) # 위로 이동
```

상세 설명

- 뒤집는 것과 움직이는 동작을 구현하기 위한 controller
- main함수로 구동이 되며
- 잡는 과정, 뒤집는 과정, 내려놓는 과정, 움직이는 과정 순서대로 작동하도록 구현

Code Review(3)

```python
class PyrmidSpace:
    def __init__(self):
        # Data in [x, y, z, 0, 0, 0] format
        data = [ ···

        self.first = data[:6] # 1st layer (6 points)
        self.second = data[6:9] # 2nd layer (3 points)
        self.second.reverse()
        self.third = data[9:] # 3rd layer (1 point)
```

```python
    def get_space_abs(self,top_pos):
        self.abs_space = []
        for point in self.first:
            pos = top_pos.copy()
            pos[0] += point[0]
            pos[1] += point[1]
            pos[2] += point[2]
            self.abs_space.append(pos)
        for point in self.second:
            pos = top_pos.copy()
            pos[0] += point[0]
            pos[1] += point[1]
            pos[2] += point[2]
            self.abs_space.append(pos)
        for point in self.third:
            pos = top_pos.copy()
            pos[0] += point[0]
            pos[1] += point[1]
            pos[2] += point[2]
            self.abs_space.append(pos)
        return self.abs_space
```

상세 설명

- 피라미드 쌓는 위치(pos)를 지정하기 위한 클래스
- data로 절대 위치를 지정하여 가운데 좌표가 (0,0,0)이 되게함
- get_space_abs를 통하여 base 좌표 기준으로 놓을 지정하여 좌표 변환을 시행

Code Review(4)

```python
class RobotController:
    def __init__(self,stack_pos,space: PyrmidSpace,move5,final_stack_pos):
        self.above_set = [VELOCITY,ACC] # 위로 올라가는 속도
        self.default_set = [VELOCITY,ACC] # 기본 속도
        self.stack_pos = stack_pos
        self.space = space # 전체 스테이지 index로 접근 가능
        self.final_stack_pos = final_stack_pos
        self.init_pos = move5

    def grip(self):...


    def ungrip(self):...

    def grip_wide(self):...

    def main(self):
        for index,point in enumerate(self.space):
            is_high = False if index != len(self.space)-1 else True
            self.stacking(self.stack_pos,point,is_high)
            if not is_high:
                self.return_home(self.stack_pos)
            self.stack_pos[2] -= GRIP_HEIGHT
        self.final_stacking(self.init_pos,self.final_stack_pos)
```

상세 설명

- stacking하는 부분과 뒤집힌 컵을 쌓기 위한 controller
- main함수로 pymidSpace로 받은 좌표와 stacked_pos 두 좌표로 옮기는 작업을 시행함
- 또한 마지막 두 좌표를 받아 뒤집힌 컵을 위로 쌓음

Q&A