

ROKEY BOOT CAMP

# SPORTS STACKING

창문 이용료 징수단  
이상우 양준혁 김주원  
류재준




# CONTENTS —

01 개요  
핵심 아이디어  
우리의 접근 방식

02 팀 소개  
역할과 책임  
핵심 아이디어 vs 비교군

03 데모 시연  
직접 시연  
참고 영상

04 코드&실험 리뷰  
핵심 아이디어 코드 위주 소개  
발생한 문제점과 해결한 방식



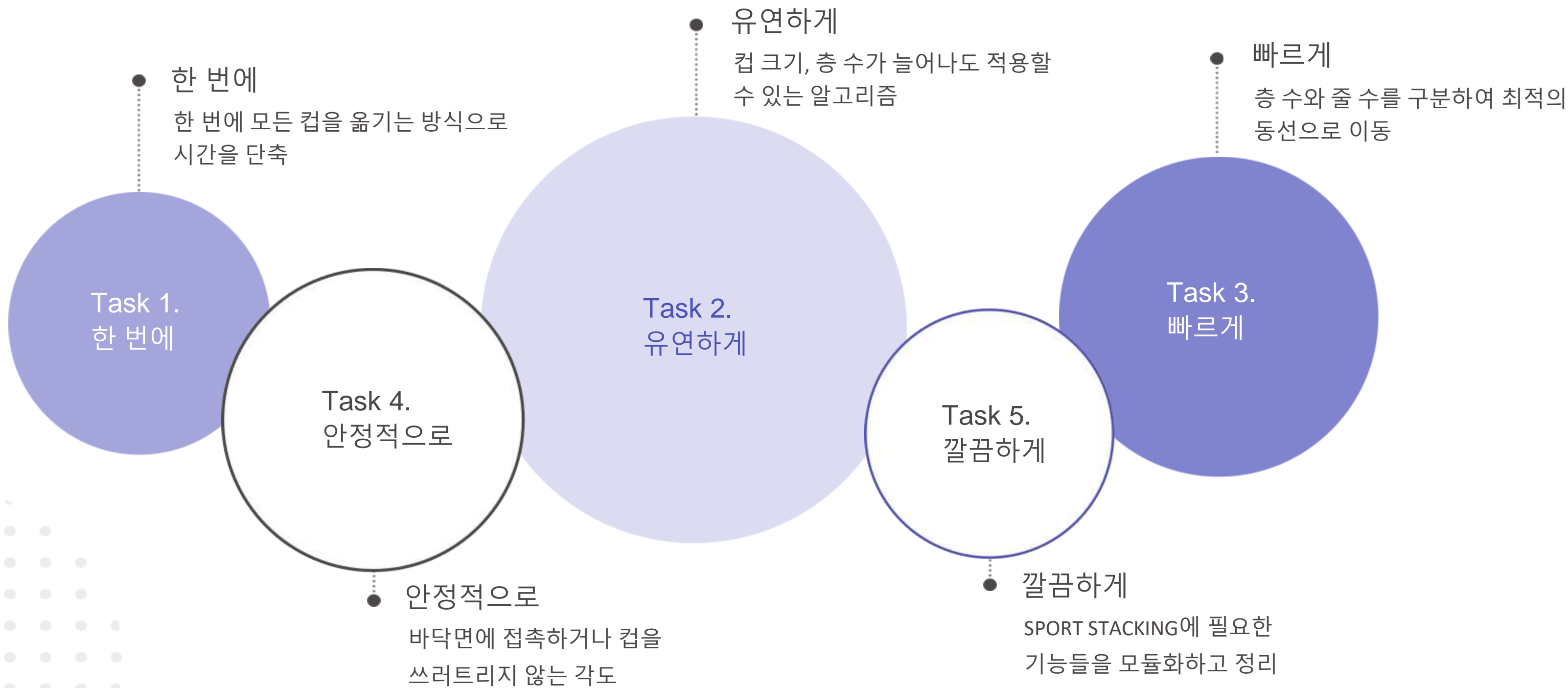


# 01

## 개요

핵심 아이디어  
우리의 접근 방식

# 핵심 아이디어



# — 02

## 팀 소개

역할과 책임

핵심 아이디어 vs 비교군



Key Word  
비교군 개발

## 김주원 류재준 아이디어 구현

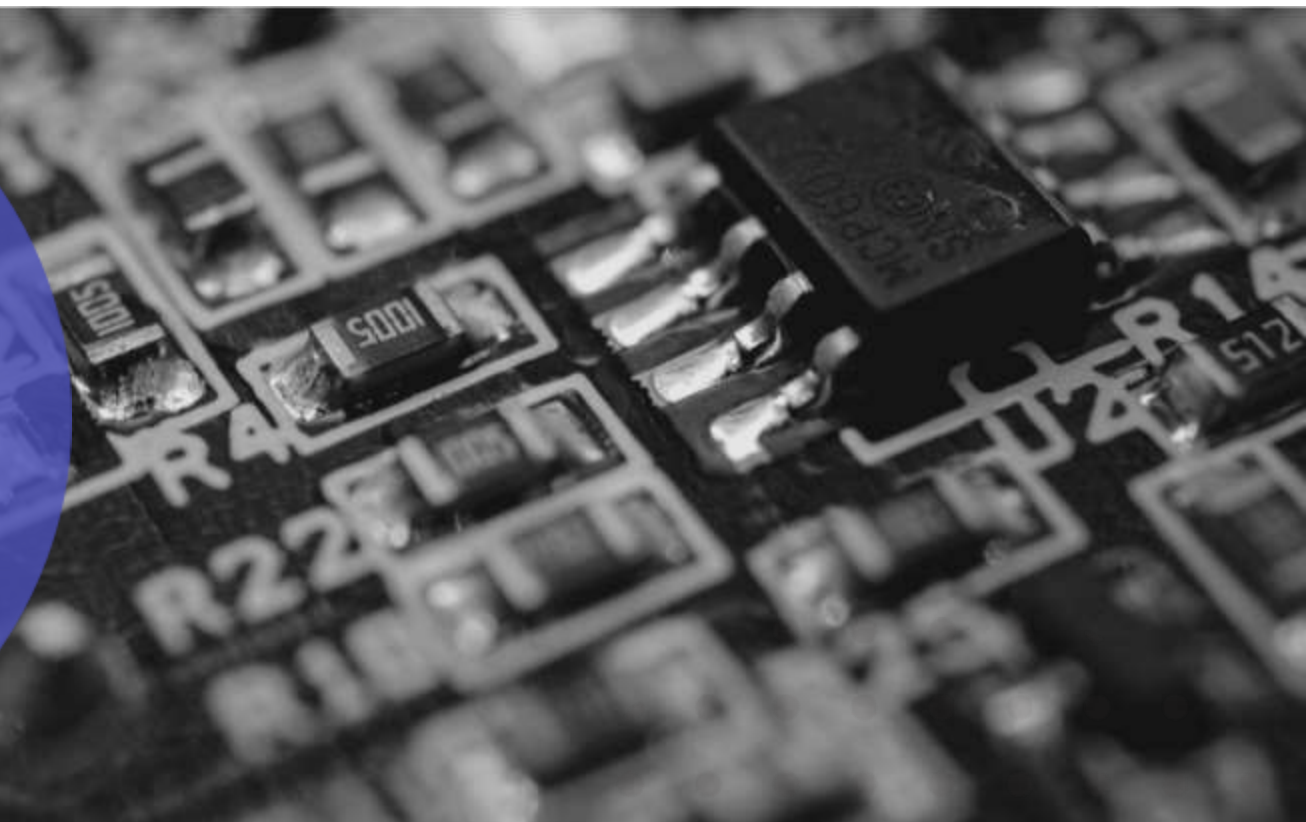
- 제시된 아이디어를 구현한 후 비교
- 사선 방향 접근, All at Once

## 양준혁 이상우 비교군 개발

- 아이디어의 결과를 검증할 비교 대상 개발
- 수직 방향 접근, One-by-One



Key Word  
아이디어 구현



03

## 데모 시연

직접 시연  
참고 영상

참고 영상 [only around 39 sec.](#)



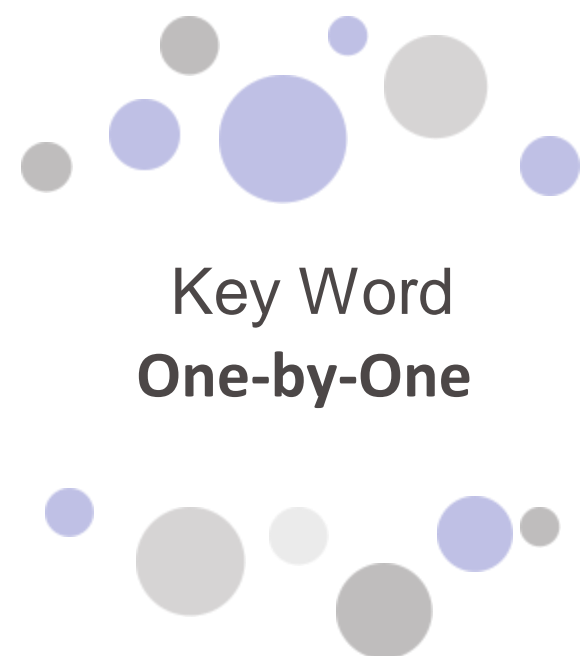


# — 04

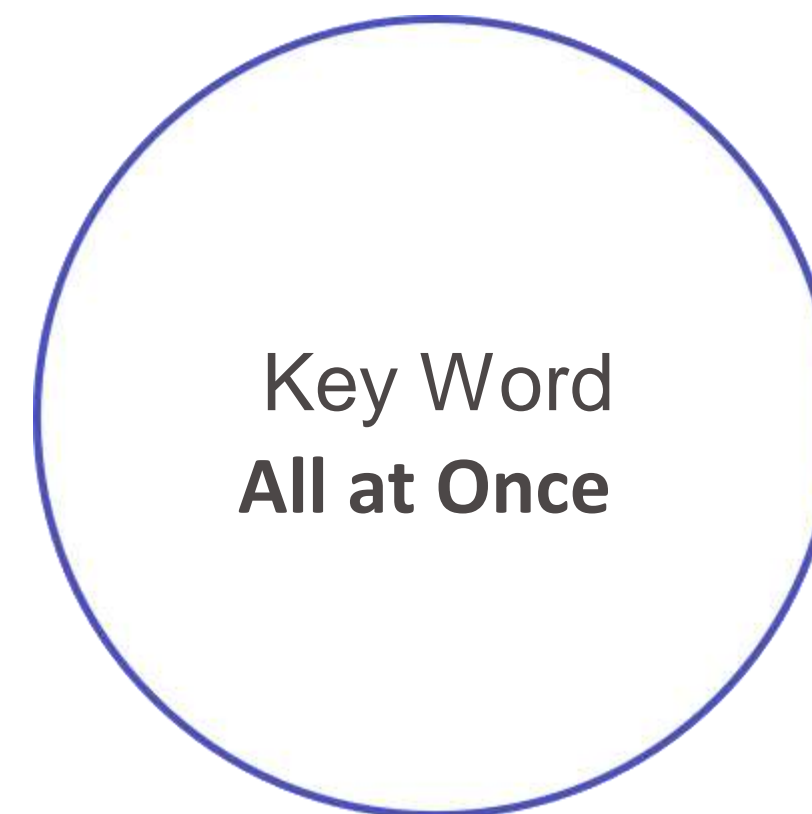
## 코드&실험 리뷰

핵심 아이디어 코드 위주 소개  
발생한 문제점과 해결한 방식

# 한 번에 목표를 달성하기 위한 최적값 탐색



Key Word  
One-by-One

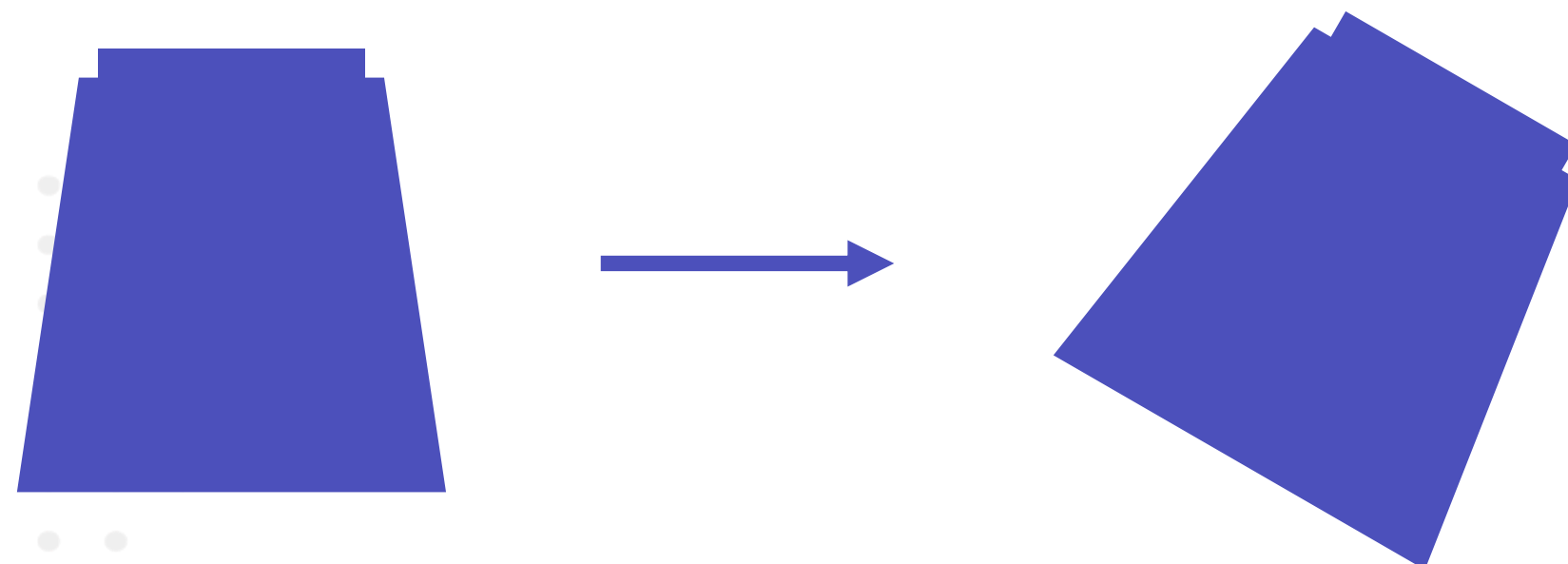
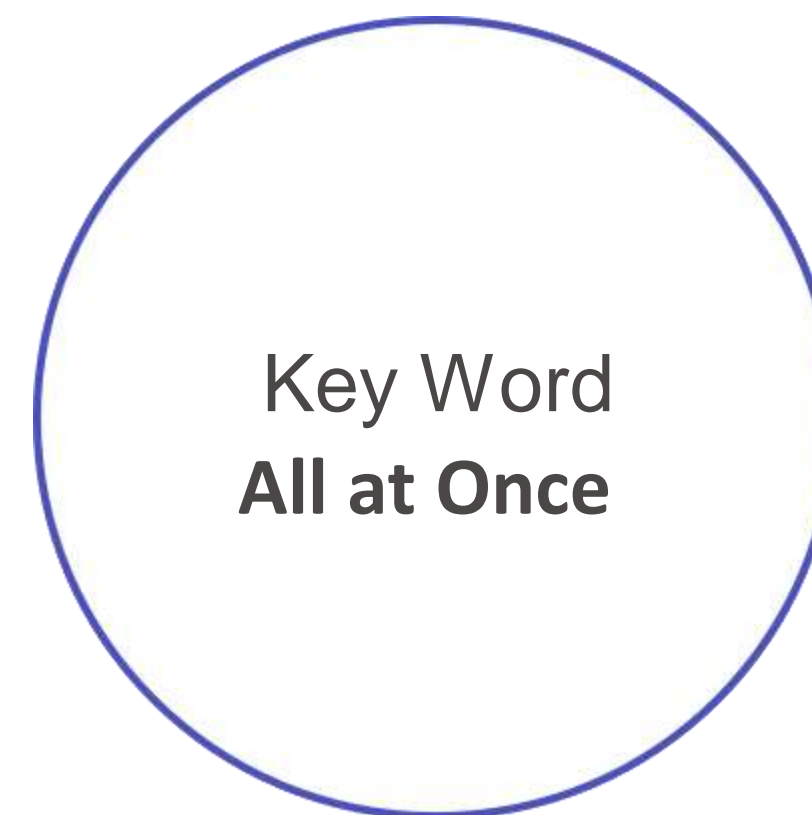
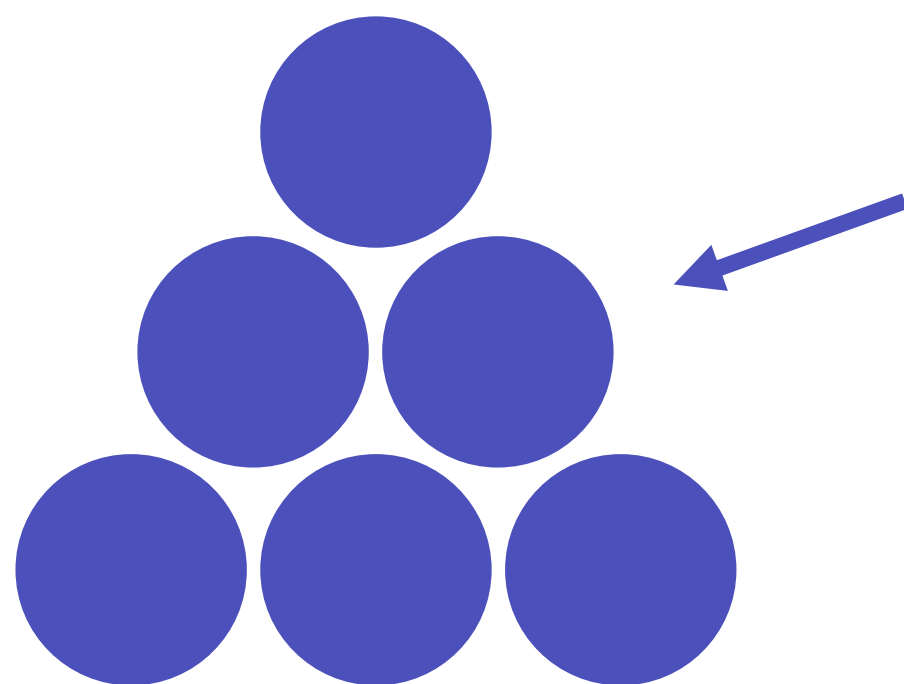


Key Word  
All at Once

- 수직 방향의 그리퍼
- 안정적인 힘제어
- grip : 50mm, 40N

- 사선 방향의 그리퍼
- 빠른 속도
- grip : 76mm, 8N

# 한 번에 목표를 달성하기 위한 최적값 탐색



- 사선 방향의 그리퍼
- 빠른 속도
- grip : 76mm, 8N

# 유연하게

## 목표와 규격에 따른 연산

- 컵의 반지름(r) : 38mm
- 컵의 높이(h) : 95mm
- 여유 공간(padding) : 3mm
- 시작점(base) : [707, 112.5, 5]
- 목표 층 수(N) : 3
- 그 외 적재 방향(x-y), 뿔 형태(3

```
# 컵 사이즈
HEIGHT = 94.7
RADIUS = 38

#타워 층
layer = 3

# 시작 좌표
starting_point = [707, 112.5, 5, 60, 125, 90]
second_point = starting_point # line 기준 좌표
cup_position = second_point # cup 기준 좌표

# 이격 거리
PADDING = 3
Z_OFFSET = 100

# 시간 변수
WAIT_SHORT = 0.4
WAIT_LONG = 1

# 작업 위치 변수
TRIANGLE_X = (RADIUS + PADDING) * 2
TRIANGLE_Y = math.sqrt(3) * (RADIUS + PADDING)
```

```
# x 좌표 업데이트
cup_position = t_trans(cup_position, [-TRIANGLE_X, 0, 0, 0, 0, 0])
logging.info(f"x 좌표 업데이트: {cup_position}")

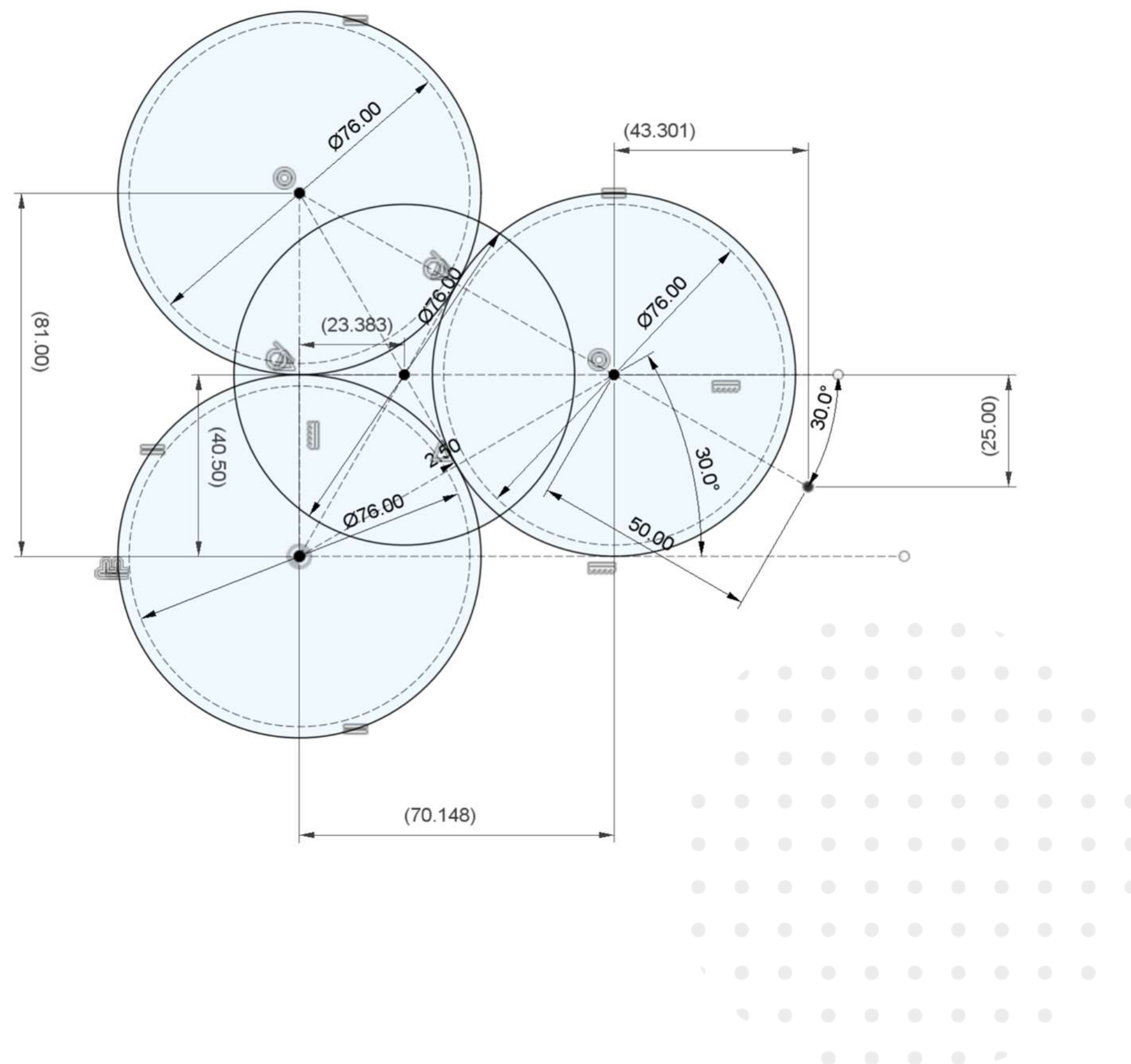
cup += 1

# XY 기준 좌표 업데이트
logging.info("XY 기준 좌표 업데이트")
second_point = t_trans(second_point, [-(TRIANGLE_X / 2), -TRIANGLE_Y, 0, 0, 0, 0])
cup_position = second_point
logging.info(f"XY 좌표 업데이트: {cup_position}")
```

# 유연하게

## 목표와 규격에 따른 연산

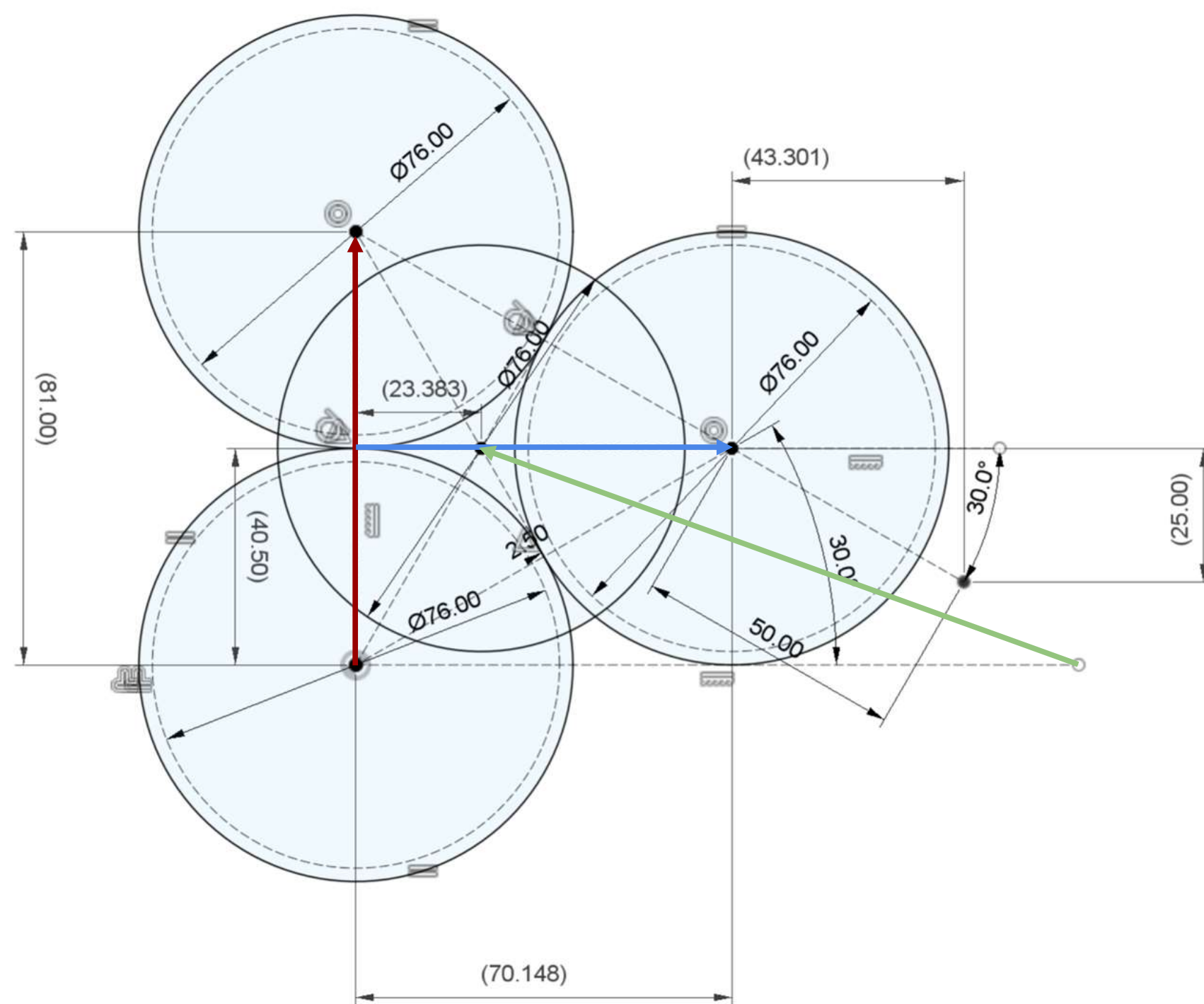
- 컵의 반지름( $r$ ) : 38mm
- 컵의 높이( $h$ ) : 95mm
- 여유 공간(padding) : 6mm
- 시작점(base) : [707, 112.5, 5]
- 목표 층 수( $N$ ) : 3
- 그 외 적재 방향( $x$ - $y$ ), 뿔 형태(3, 4, 5, ...)





# 빠르게

## 좀 더 효율적인 동선



- x축 이동 :  $3 * \sqrt{r + \text{padding}}$
- y축 이동 :  $2 * (r + \text{padding})$
- 홀수 줄 :  $(\text{line} - 1) * \text{x축 이동}$
- 짝수 줄 :  $-1 * (\text{line} - 1) * \text{x축 이동}$
- 기준점 이동 :  $(\text{layer} - 2) * \text{x축 이동} / 2$

$(\text{layer} - 2) * \text{y축 이동} + \text{y축 이동}$

# 빠르게

## 좀 더 효율적인 동선

- x축 이동 :  $3 * \sqrt{r + \text{padding}}$
- y축 이동 :  $2 * (r + \text{padding})$
- 홀수 줄 :  $(\text{line} - 1) * \text{x축 이동}$
- 짝수 줄 :  $-1 * (\text{line} - 1) * \text{x축 이동}$
- 기준점 이동 :  $(\text{layer} - 2) * \text{x축 이동} / 2$

$(\text{layer} - 2) * \text{y축 이동} + \text{y축 이동}$

```
# 층별 선(line) 생성
for line in range(layer, 1, -1):
    node.get_logger().info(f"Layer {layer}, Line {line} 시작.")
    if (layer % 2 == 0 and line % 2 == 1) or (layer % 2 == 1 and line % 2 == 0):
        node.get_logger().info(f"{layer, line}.")
        for _ in range(line - 1):
            move_vector = posx([TRIANGLE_BASE_LINE, 0, 0, 0, 0, 0])
            current = get_current_posx()[0]
            target_posx = trans(current, move_vector, ref=DR_BASE)
            movel(target_posx, vel=VELOCITY, acc=ACCELERATION, ref=DR_BASE)
            time.sleep(WAIT_LONG)
            set_cups(node, layer)
            pickup_cups(node)

        move_vector = posx([-TRIANGLE_BASE_LINE/2, -TRIANGLE_HEIGHT, 0, 0, 0, 0])
        current = get_current_posx()[0]
        target_posx = trans(current, move_vector, ref=DR_BASE)
        movel(target_posx, vel=VELOCITY, acc=ACCELERATION, ref=DR_BASE)
        time.sleep(WAIT_LONG)
        set_cups(node, layer)
        pickup_cups(node)

    else:
        node.get_logger().info(f"{layer, line}.")
        for _ in range(line - 1):
            move_vector = posx([-TRIANGLE_BASE_LINE, 0, 0, 0, 0, 0])
            current = get_current_posx()[0]
            target_posx = trans(current, move_vector, ref=DR_BASE)
            movel(target_posx, vel=VELOCITY, acc=ACCELERATION, ref=DR_BASE)
            time.sleep(WAIT_LONG)
            set_cups(node, layer)
            pickup_cups(node)

        move_vector = posx([TRIANGLE_BASE_LINE/2, -TRIANGLE_HEIGHT, 0, 0, 0, 0])
        current = get_current_posx()[0]
        target_posx = trans(current, move_vector, ref=DR_BASE)
        movel(target_posx, vel=VELOCITY, acc=ACCELERATION, ref=DR_BASE)
        time.sleep(WAIT_LONG)
        set_cups(node, layer)
        pickup_cups(node)

node.get_logger().info(f"Layer {layer}, Line {line} 작업 완료.")
```



# 안정적으로 바닥면과 컵 방향 고려





# 문제점과 해결 방식

## 코딩과 실험 중 발생한 문제점과 솔루션

### 01

#### Force control release

release가 종료되지 않은 상태에서 다음 움직임이 주어지면 로봇이 작동을 정지함.

### 03

#### Gripper Setting

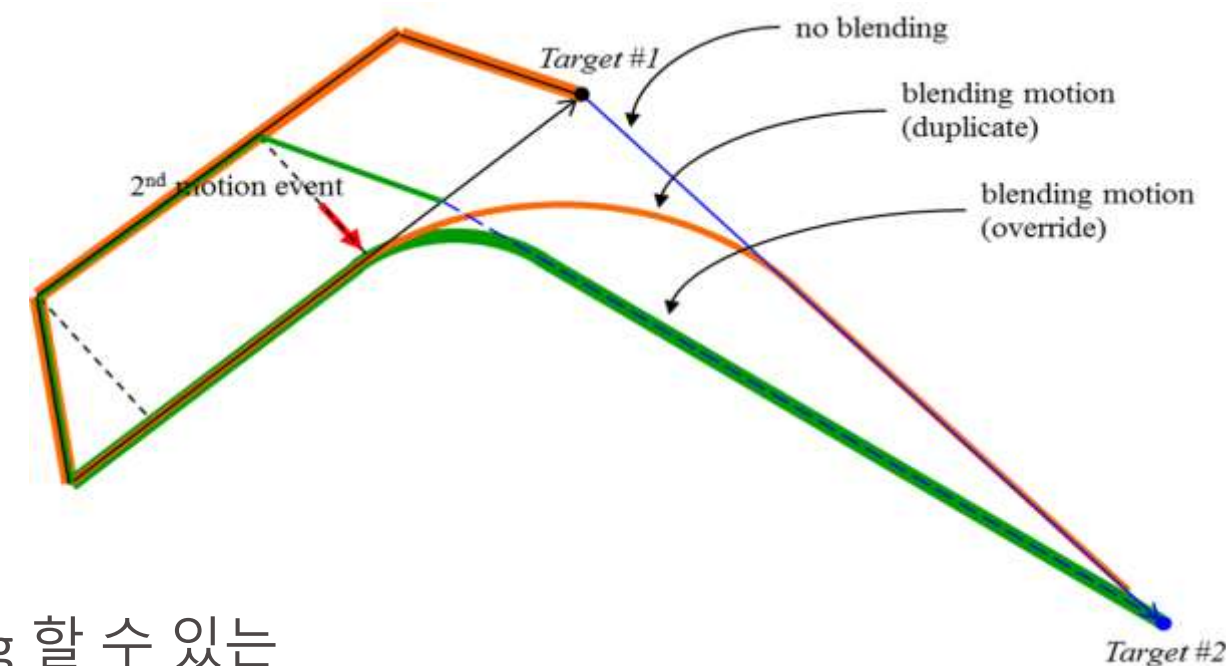
그리퍼의 I/O가 한정적이기 때문에 여러가지 그리퍼 세팅을 혼합하기 힘들. 더 많은 단자가 주어진다면 더 향상된 작업이 가능

### 02

#### 작업 동선 문제

`ra = DR_MV_RA_DUPLICATE`

이전에 주어진 움직임과 blending 할 수 있는 옵션



# 비교 분석



One-by-One



- 종이컵을 하나씩 집어서 옮기기 때문에 그리퍼 세팅을 미세하게 할 필요가 없음.
- 상대적인 시간 소요가 크지만 정확하게 작업을 성공시킬 수 있음.



All at Once



- 종이컵을 한 번에 옮기기 때문에 종이컵을 잡는 위치, 힘, 그리퍼의 너비 등을 미세하게 조정해야 함.
- 시간은 빠르지만 한 번에 옮기는 작업으로 인해 작업을 성공하지 못하는 경우도 발생함
- 하드웨어의 한계점이 보완될 경우 빠른 시간 내에 작업을 성공시킬 수 있음.

ROKEY BOOT CAMP

THANK  
YOU

창문 이윤록 짐솟단  
이상무 양준혁 김주원  
류재준