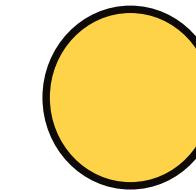
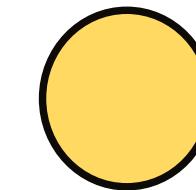
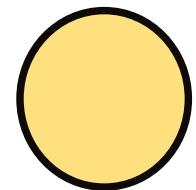
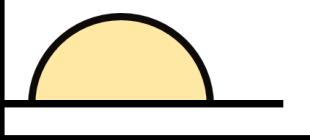
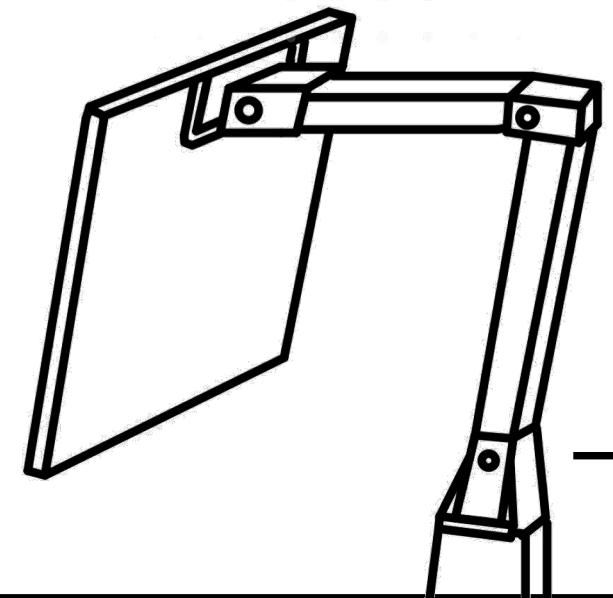


궤적 근사 알고리즘을 이용한

병렬 처리 탁구로봇



Team : 인삼팀



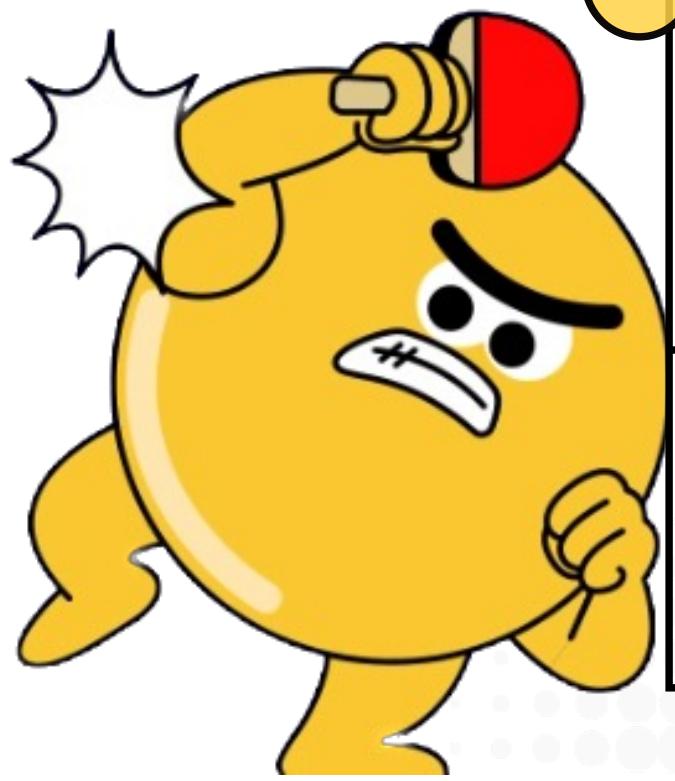
con tents

I. overview

- a. 간략한 소개글
- b. 실제 동작 모습
- c. System Overview
 - 4개 세부 시스템
 - 상호작용 모식도

II. 시스템 별 상세 설명

- a. 비전시스템
 - HW - 스테레오 카메라 설치
 - SW - 탁구공 감지
 - SW - 탁구공 위치 추출
- b. 예측시스템
 - SW - 궤적 근사 알고리즘
- c. 로봇시스템
 - HW - 로봇 구조 설계
 - SW - 로봇 모터 제어
 - SW - Linear Actuator 제어
- d. 통합시스템
 - SW - 멀티 쓰레딩 및 프로세싱



con tents

III. 시행착오 이야기

- a. 딥러닝
- b. 1 프레임 예측
- c. 허프 변환 검출
- d. 60fps로 입력
- e. Depth 문제

V. 기본검증~ing 이야기

- a. 기본검증 결과
- b. 보완점
- c. 현재 작업 중인 과제들
 - 로봇 모터 좌우 회전 추가
 - 정확한 스테레오 비전 완성
 - z축 궤적 예측 알고리즘
 - 비전시스템 통제 환경 보완

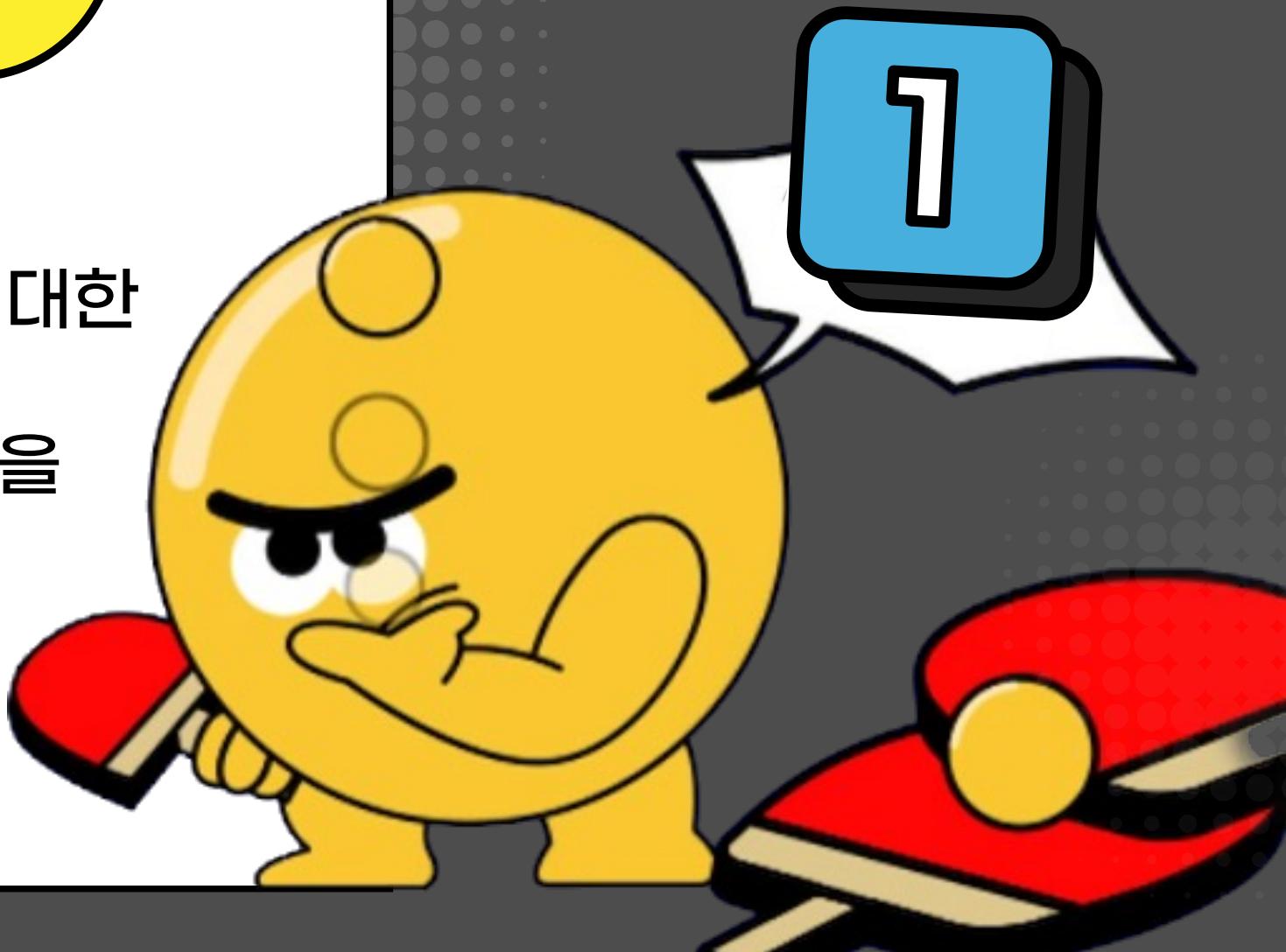
IV. 팀워크 이야기

- a. 정기 모임
- b. 노션 활용
- c. 깃허브 활용



overview

저희 인삼팀의 탁구로봇 전체 시스템에 대한
간략한 소개와 실제 동작 모습,
그리고 세부 시스템들 간의 상호작용을
overview 합니다.



간략한 소개글

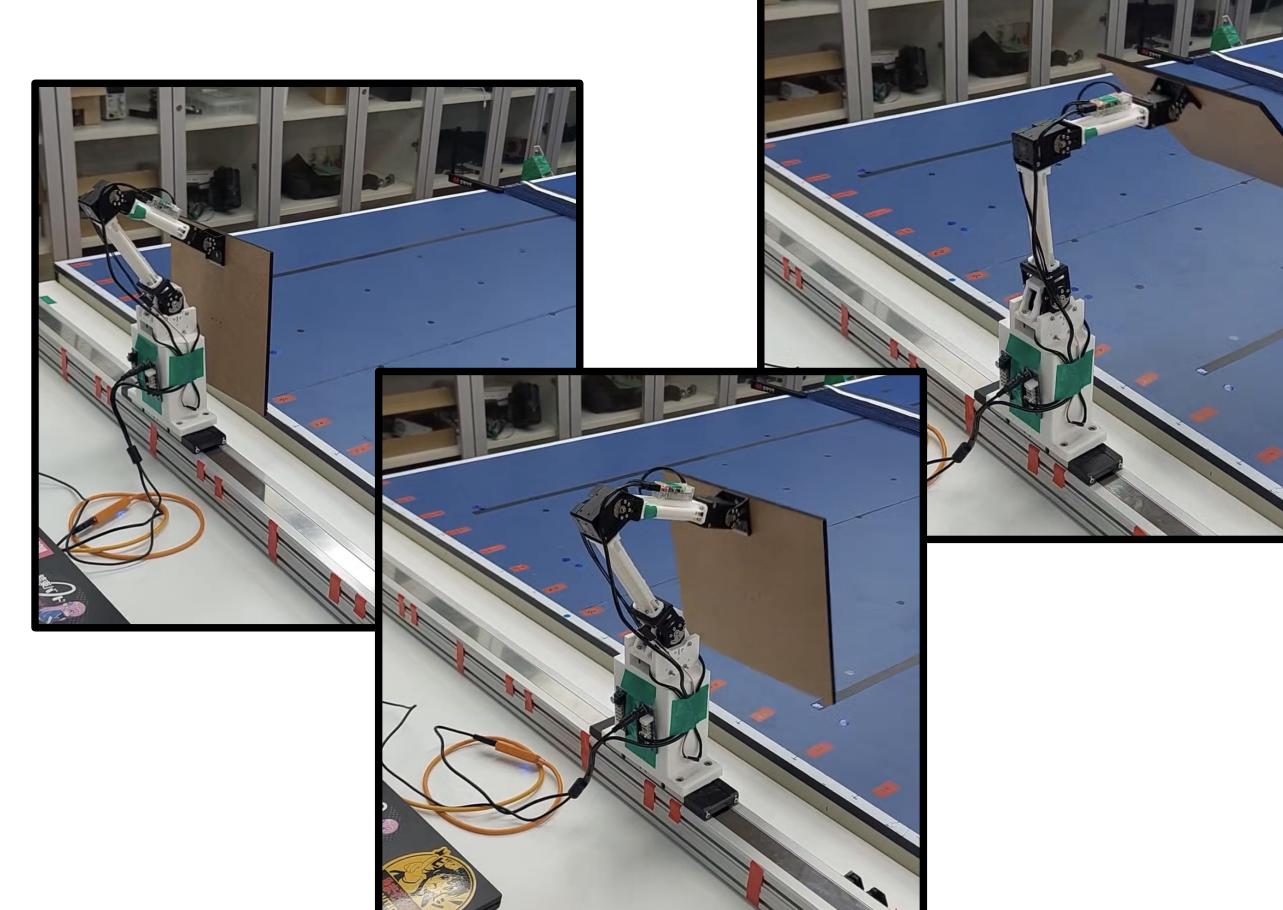
저희 탁구로봇은

카메라 영상의 초기 2 frame에서
탁구공을 감지하여 위치를 추출하고,

궤적 근사 알고리즘을 사용하여

탁구공의 최종 도착 위치를 예측한 후,

해당 위치로 로봇 몸체가 이동하여
타격 모션으로 공을 쳐냅니다.



이때 모든 작업을 **병렬 처리**로 수행하여
빠르게 날아오는 탁구공에 대해 신속히 대응합니다.

실제 동작 모습

8/4 기본검증 시에 촬영한 동영상입니다.



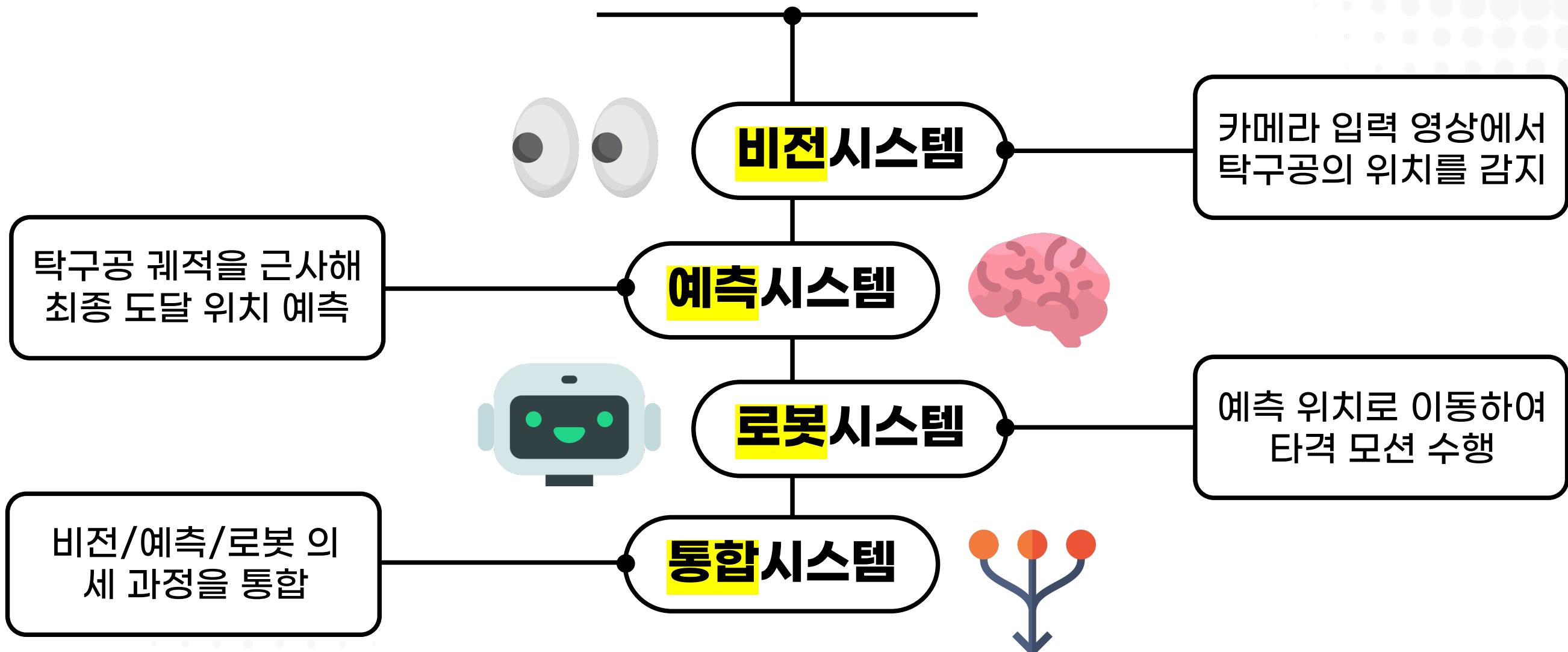
▲ 중앙 방향 공에 대한 작동 영상



▲ 오른쪽 방향 공에 대한 작동 영상

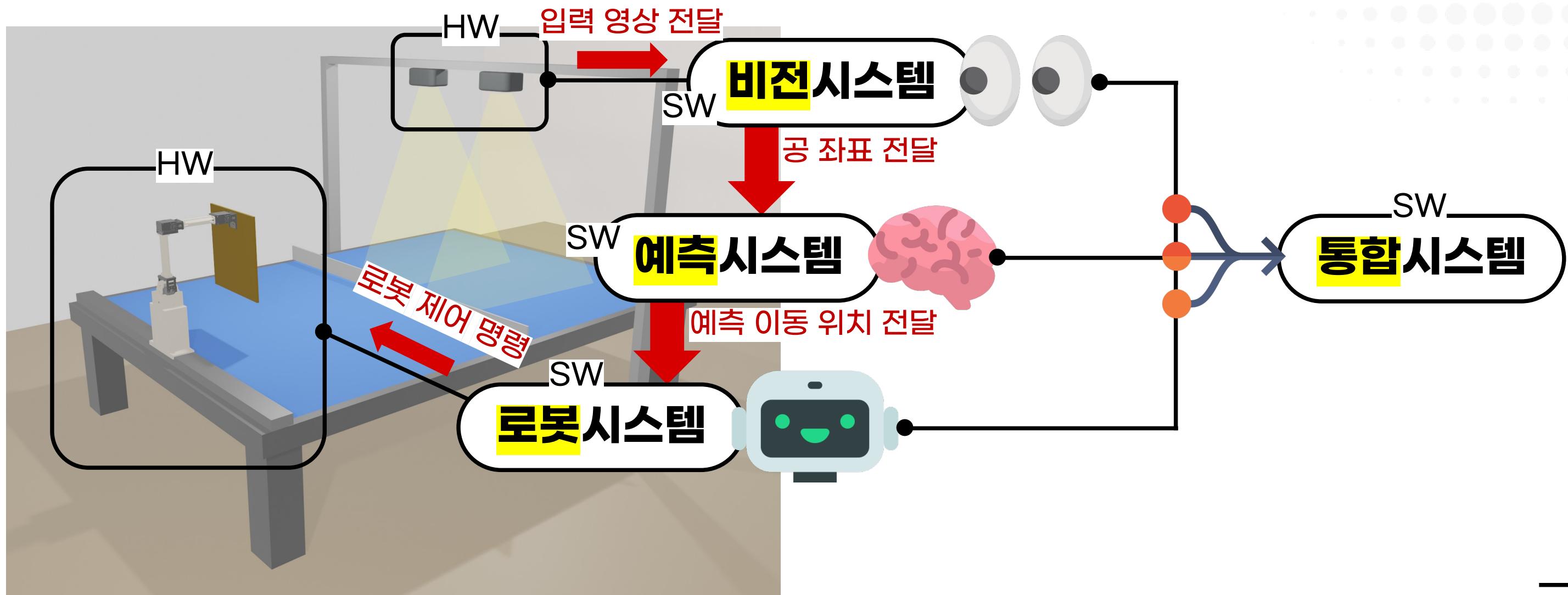
System overview

탁구로봇 전체 시스템은 총 **4개의 세부 시스템**으로 나뉘어져 있습니다.



System overview

각 시스템들 간의 상호작용을 시각화한 모식도입니다.



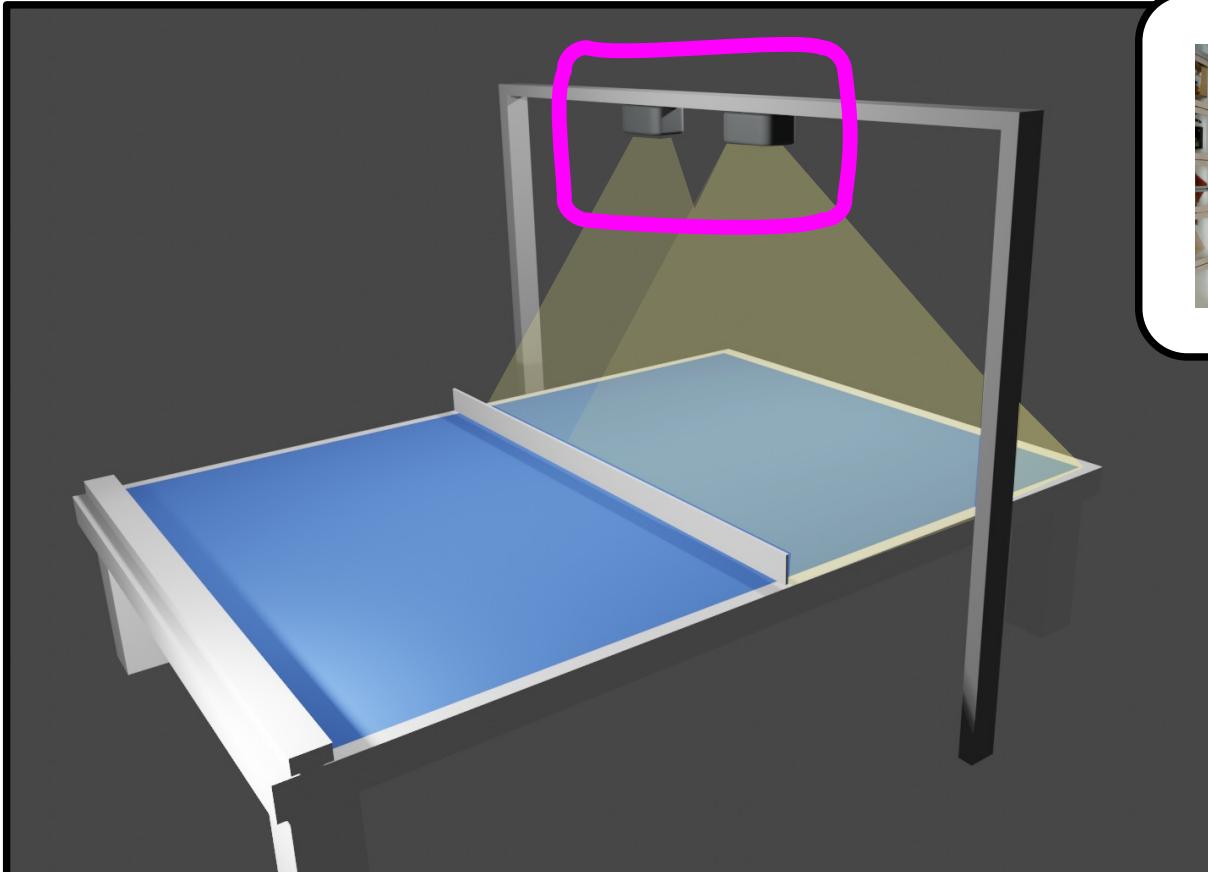
시스템별 상세설명

네 가지 세부 시스템
(비전/예측/로봇/통합) 각각에 대해,
구체적인 원리와 내용을 설명합니다.

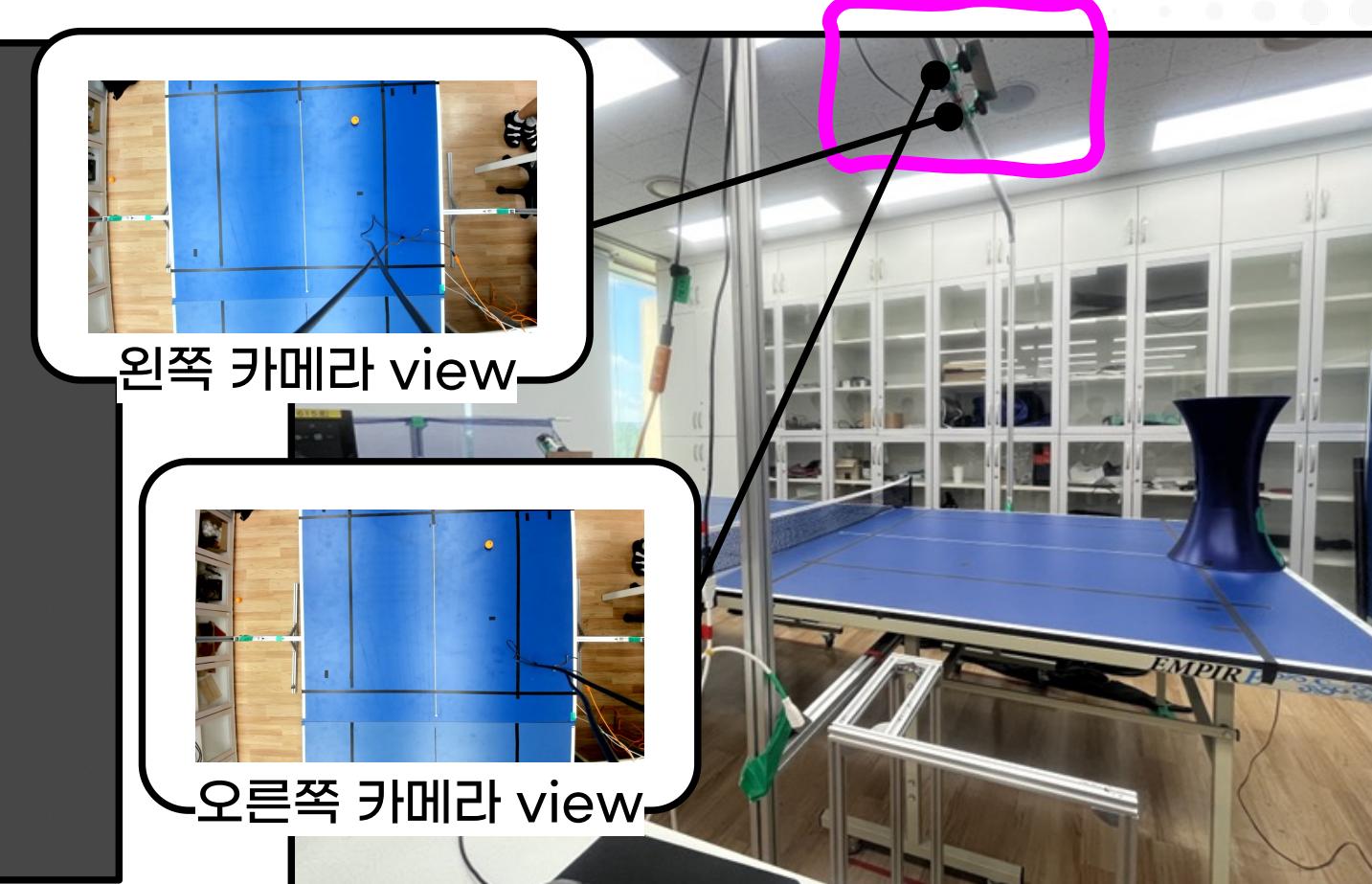


비전시스템 HW - 스테레오 카메라 설치

탁구대 첫 절반이 view에 보이도록,
두 대의 카메라를 알루미늄 프로파일에 고정하여 설치하였습니다.



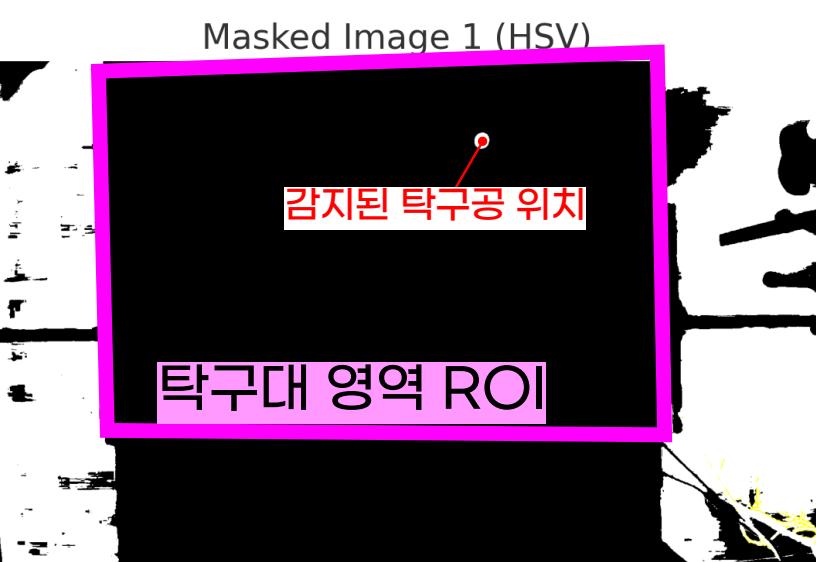
▲ 비전 시스템 위치 및 구조 설계



▲ 실제 설치한 모습

비전시스템 SW - 탁구공 감지

HSV색상 값 범위를 정해 주황색 물체만 masking한 후,
탁구대 영역 (ROI) 내 주황 픽셀의 무게중심을 탁구공 위치로 감지
: Region Of Interest



▲ 실제 탁구공 감지 처리 과정

(※ 실제 처리 과정에서는 탁구대 영역 ROI를 affine변환 한 후
탁구공 위치 값을 추출하게 됩니다)



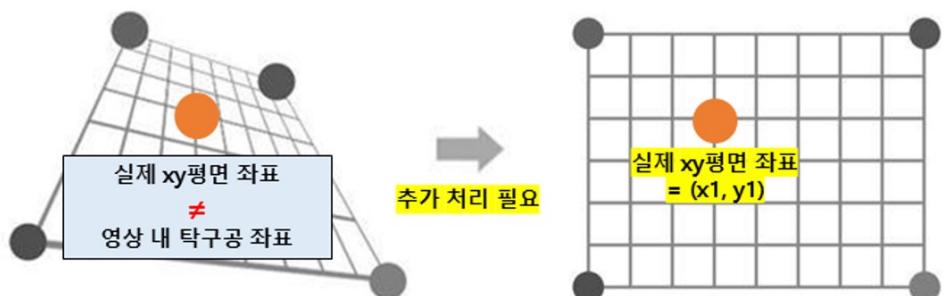
기본 검증에서는
로봇이 공을 주기 때문에
ROI 내에 발생 가능한
다른 노이즈(손, 팔 등)를
엄격히 처리하지 않았지만,
본선에서는 엄격히
처리할 예정입니다.

비전시스템 SW - 탁구공 위치 추출

탁구공의 frame 상의 위치에 대응되는
실제 3차원 좌표 (x, y, z)를 계산

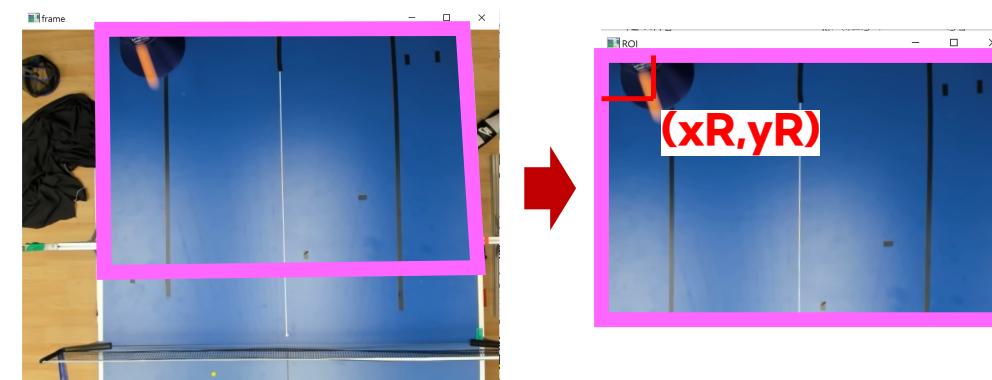
1

: ROI를 affine변환한 후의 frameL, frameR 의 픽셀 좌표 이용



▲ affine 변환을 하는 이유

: 현실의 xy평면과 영상에 찍히는 탁구대 평면이
평행하지 않아 보정이 필요하다



▲ 실제 탁구공 감지 처리 과정

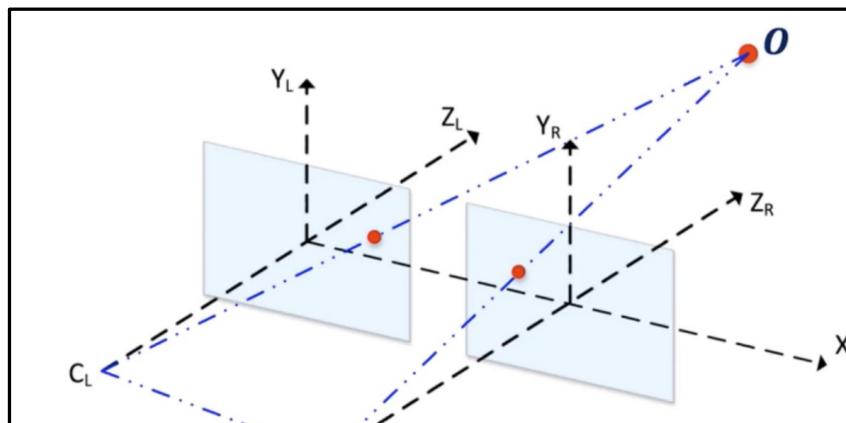
* 위 과정을 카메라 두 대에 대해 수행해
frameL의 (xL, yL), frameR의 (xR, yR) 를 추출

기본 검증에서는
z값을 사용하지 않고
(xL, yL)만을 사용하도록
단순화했습니다.
현재 본선 대비를 위해서
정확한 stereo vision을
구현하고 있습니다.

비전시스템 SW - 탁구공 위치 추출

2

: stereo vision을 이용해 (xL-xR)에서 실제 3차원 좌표값 추출

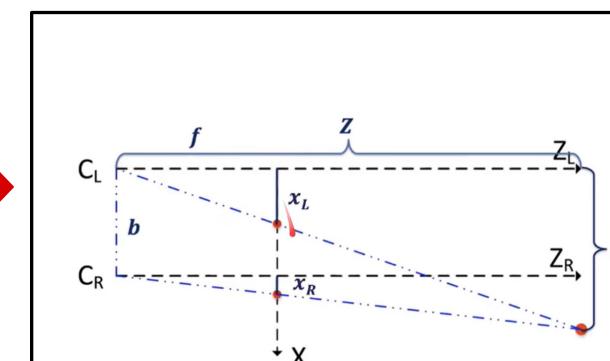


: 카메라 왜곡이 보정된,
두 카메라 상의 x축과 y축이 정렬된 상태



카메라+스테레오 calibration

: stereo vision 계산을 위해 보정이 필요



$$\frac{Z}{f} = \frac{X}{x_L} \quad \frac{Z}{f} = \frac{X-b}{x_R}$$

$$x_L = \frac{X}{Z} \cdot f \quad x_R = \frac{X-b}{Z} \cdot f$$

$$Disparity = x_L - x_R$$

: d(disparity, xL-xR)값을 이용한
triangulation으로 depth 측정 가능

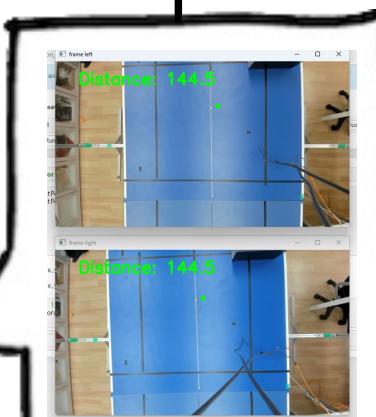
$$\begin{aligned} \text{depth } Z &= \frac{fb}{d} \\ \text{보정된 } X, Y \text{값} &= \frac{Zx_L}{f} \\ &= \frac{Zy_L}{f} \end{aligned}$$

: depth 값으로 보정된 x, y값 획득,
실제 z값은 depth 를
카메라에서 탁구대까지 거리를 뺀 값

실제 3차원 좌표값

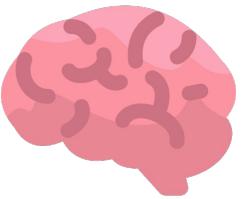


기본 검증에서는
z값을 사용하지 않고
(xL,yL)만을 사용하도록
단순화했습니다.
현재 본선 대비를 위해서
정확한 stereo vision을
구현하고 있습니다.



실제 depth값
추출 테스트

예측시스템 SW - 궤적 근사 알고리즘



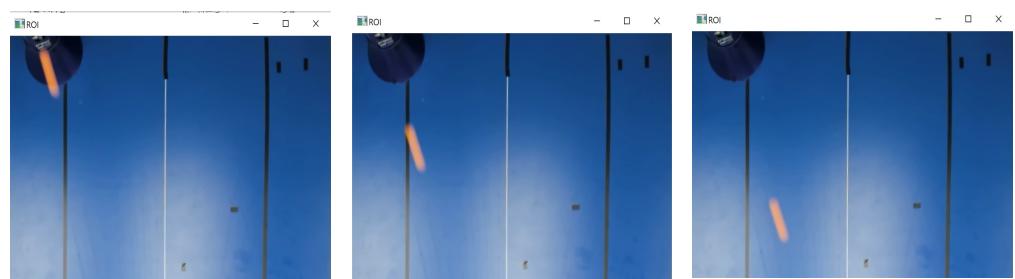
첫 2 frame 상의 (x, y, z) 를 바탕으로 전체 궤적을 근사하여 최종 위치 예측

※ 이때 2 frame은 ROI 내에서 검출된 주황색 객체의 y가 증가하는 방향($y_1 < y_2$)일 때만을 가져옵니다.

1

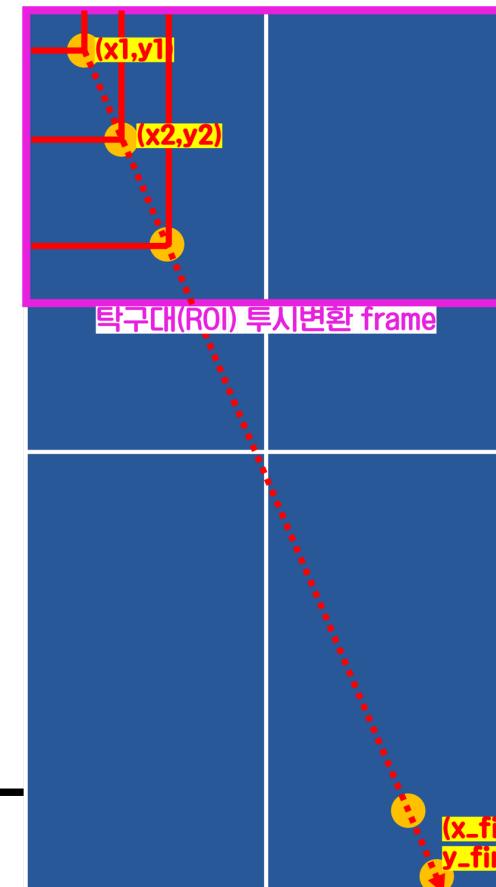
: xy평면상에서 직선 궤적으로 근사해 탁구공의 최종 도달 x위치 예측

기본 검증에서는 z값을 사용하지 않고 xy평면 직선궤적 근사만을 사용하여 단순화했습니다. 현재 본선 대비를 위해서 yz평면 포물선궤적 근사를 구현하고 있습니다.



▲ 1~3번째 ROI frame의 탁구공 위치

: overlap 해보았을 때 xy평면 상에서 직선 궤적으로 근사할 수 있음을 확인



2 frame 상의 $(x_1, y_1), (x_2, y_2)$ 를 이용, 직선의 방정식을 이용, x_{final} 을 예측

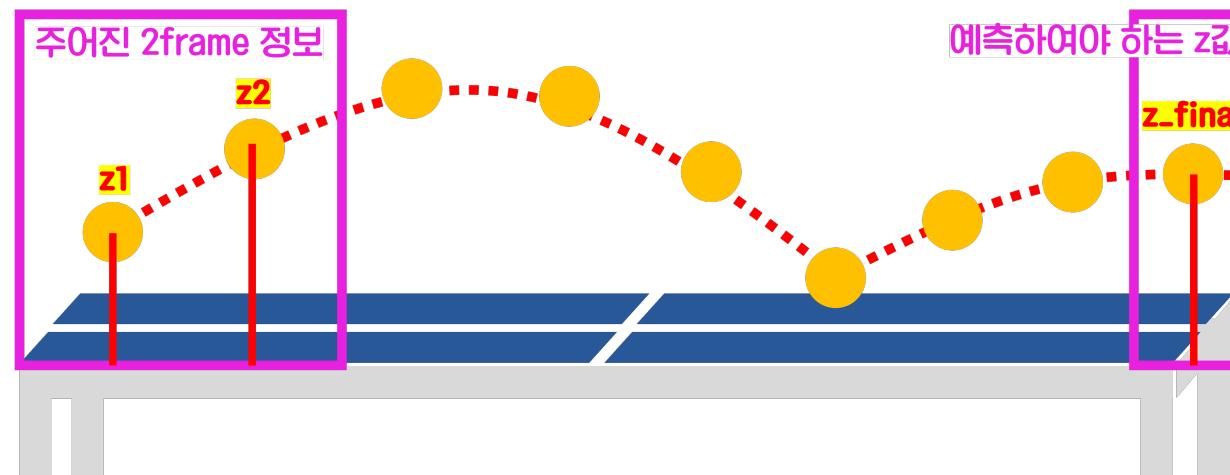
※ 공이 튀긴 후의 궤적 변화, 실제로는 완벽한 직선이 아님 등의 오차를 고려하기 위해 실험적으로 가중치와 편향을 추가로 부여했습니다.

예측시스템 SW - 궤적 근사 알고리즘



2

: yz평면상에서 포물선 궤적으로 근사해 탁구공의 최종 도달 z위치 예측



< 정사영된 포물선 공식을 사용 >

: $y'z'$ 평면상의 포물선을

$$y'^2 + ay' + bz' + c = 0, x' = x_0$$

yz평면으로 정사영한 포물선의 공식을 이용

$$x = x' \cos \theta - y' \sin \theta$$

$$y = x' \sin \theta + y' \cos \theta$$

$$z = z'$$

< 최소제곱법을 이용한 근사 곡선 방정식 >

: 제곱 오차 S 를 최소화하는 포물선 공식 이용

$$S = \sum_{i=1}^n (ay_i^2 + by_i + c - z_i)^2$$

$$\sum_{i=1}^n y_i^4 a + \sum_{i=1}^n y_i^3 b + \sum_{i=1}^n y_i^2 c = \sum_{i=1}^n y_i^2 z_i$$

$$\sum_{i=1}^n y_i^3 a + \sum_{i=1}^n y_i^2 b + \sum_{i=1}^n y_i c = \sum_{i=1}^n y_i z_i$$

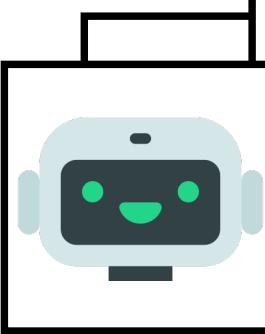
$$\sum_{i=1}^n y_i^2 a + \sum_{i=1}^n y_i b + nc = \sum_{i=1}^n z_i$$

$$z = ay^2 + by + c$$

기본 검증에서는
z값을 사용하지 않고
xy평면 직선궤적 근사만을
사용하여 단순화했습니다.
현재 본선 대비를 위해서
yz평면 포물선궤적 근사를
구현하고 있습니다.

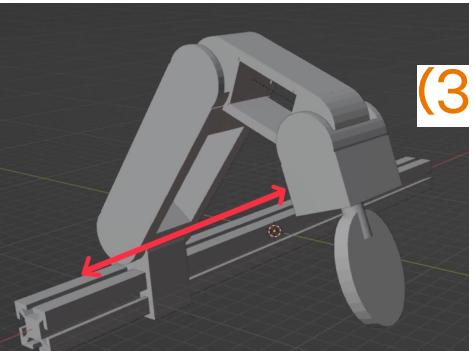
< 추가 방안 >

포물선 근사 궤적이
잘 구해지지 않을 경우,
차선책으로
수동 라벨 분류 방식을
고려하고 있습니다.



로봇시스템 HW - 로봇 구조 설계

초기 설계



: 모터 토크, 하단부에 가해지는
하중 등을 고려하여,
구조적으로 튼튼하면서도
3d 프린팅을 위해 단순하게,
그리고 모터 연결 **규격에 맞게**
베이스와 링크 프로파일을

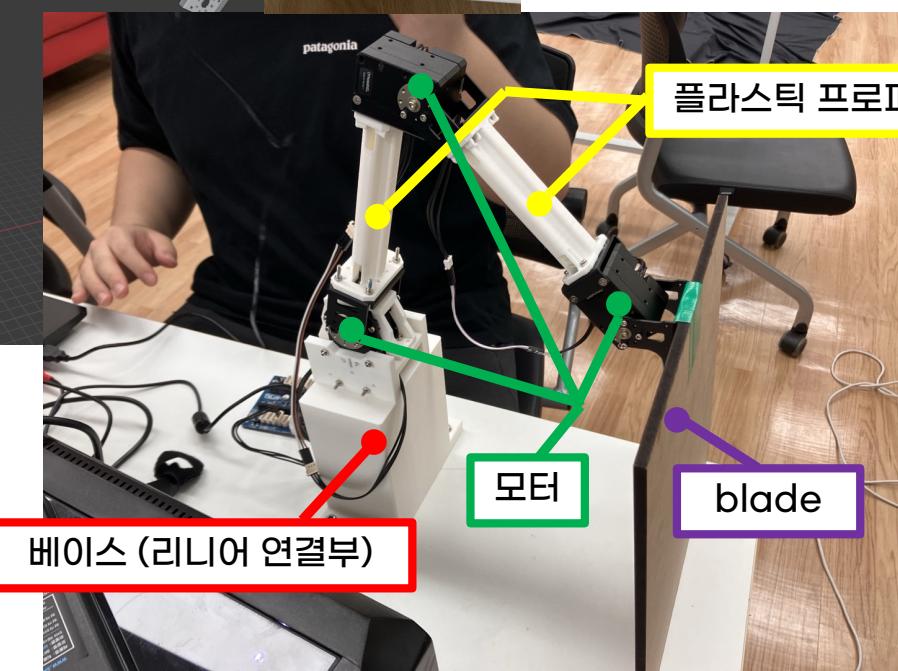
: 타격 blade의 면적을
넓힐 필요성이 있음

현재 설계

blender)



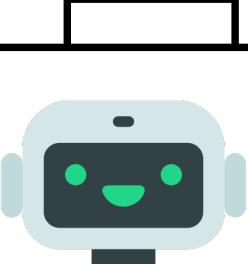
▼ 실제 로봇 부품 조립 결과



주
후

기본 검증에서는
blade의 좌우 회전 없이
설계하여 단순화했습니다.
현재 본선 대비를 위해서
좌우 회전을 넣은 설계로
제작하고 있습니다.

로봇시스템 SW - 로봇 모터 제어



타격 모션 탐구

: 초기에는 blade 회전만으로 타격하도록 설계했으나, 공이 네트를 넘기지 못하고 떠버리는 상황 발생



: 실제 탁구선수의 리시브를 분석하니 회전보다는 밀어서 침, 모터 3개의 종합적 운동으로 밀어서 치는 타격 모션 완성

모터 제어 코딩

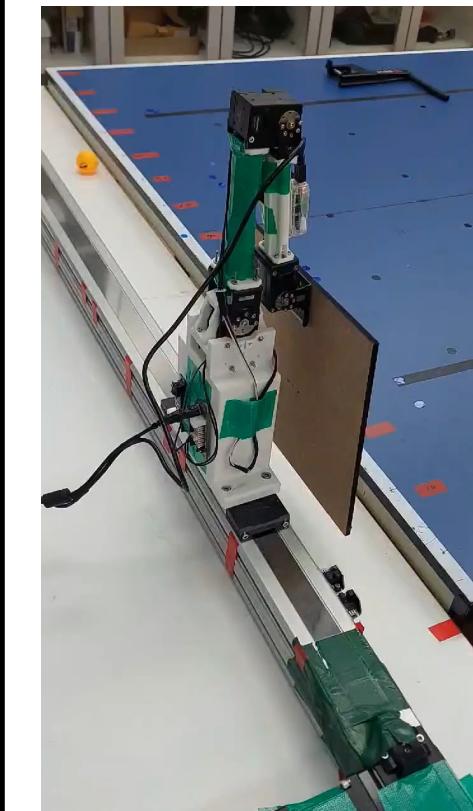


: 초기에는 사용이 간단한 pypot을 이용했으나, 각 모터의 섭세한 조절에 한계



: sdk를 이용해 정밀 조절 구현, 모터 회전과 속도 등의 조절 각각에 대해 함수를 만들어 사용하기 쉽도록 모듈화

실제 타격 모션



실제 기본검증 시의 타격 모션 작동

: 타격모션 과정을
(1) 준비모션,
(2) 전진모션 의
두 단계로 구성.

추후

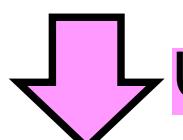
기본검증에서는 타격 z가 고정되어 있다고 가정하고 설계했지만, 현재 본선 대비를 위해서 z높이에 따른 다양한 모션을 제작하고 있습니다.

로봇시스템 SW - Linear Actuator 제어

다른 SW시스템들의 각 모듈들을 Python에서 통합적으로 제어했으므로,
C++로만 제어할 수 있는 Linear Actuator와 Python간의 연결이 필요

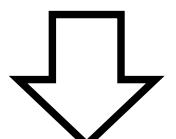
파이썬에서 소켓 통신으로 이동 값 송신

- : Socket 모듈 사용
- : Threading/queue 모듈로 데이터 입력 관리

 **UDP 통신** : 로컬 프로토콜로써 사용 (같은 컴퓨터 내의 파이썬 - C++간 통신)

C++에서 소켓 통신으로 이동 값 수신

- : 이동 값은 0부터 141까지의 integer value로, Linear Actuator가 이동해야 할 목적지를 가리킴



값 수신 시 Actuator 조작 함수 호출

정수 값을 부여하면 해당 위치로 Linear Actuator가 이동하는 방식으로 제어합니다.

통합시스템 SW - 멀티 쓰레딩 및 프로세싱

모든 SW시스템들의 모듈들을 Python에서 통합적으로 제어합니다.

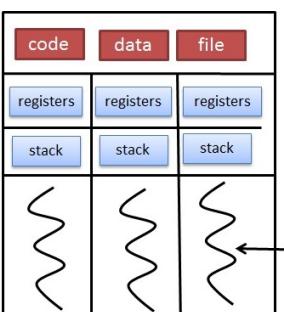
멀티 쓰레딩



- : 병렬처리를 도입하지 않으니 동시에 두 카메라 입력이 불가
- : 또, 프레임 처리 시에 시간 지연

멀티 쓰레딩
도입!

> cam1/2
동시 입력 해결



Multithreaded Process

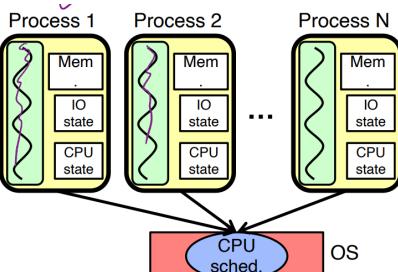
이 또한 문제

GIL 때문에 일어난 일

자물쇠는 하나뿐 ➔ 한 CPU만 일(계산)한다!

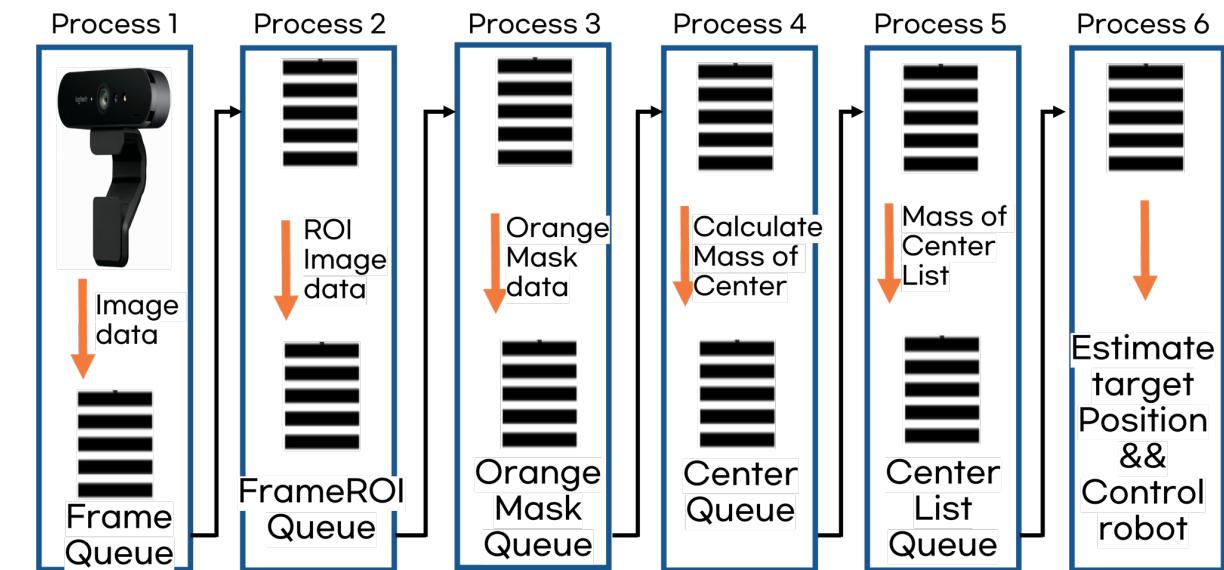


- : 파이썬 GIL 때문에 멀티 쓰레딩을 하더라도 I/O작업이 아닌 cpu중심 작업에 대해 실질적인 병렬 처리 불가



멀티
프로세싱
도입!
> GIL 영향 X,
속도 up

멀티 프로세싱



- : 실질적 병렬처리 가능, 총 6개의 프로세스로 분할.
값 전달 시 queue를 이용해 순차적 전달 수행

시행착오 이야기

기본 검증을 준비하며
여러 방안들을 시도했지만,
실패를 겪었던 시행착오들에 대해
이야기합니다.

(※ 이 중 몇몇은 현재 본선 대비를 하며
보완하여 해결해보고 있습니다.)

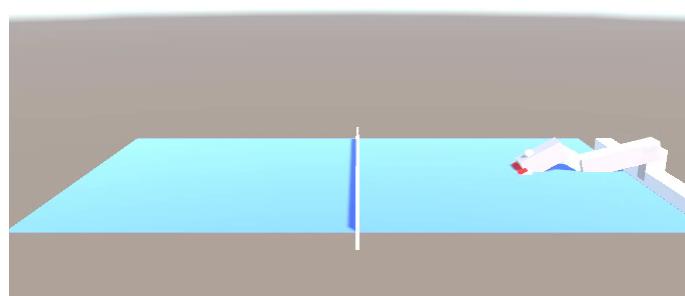


딥러닝



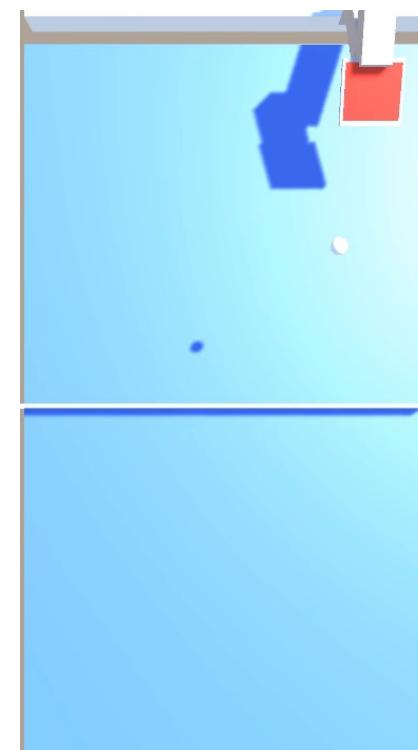
팀원 대부분이 딥러닝에 흥미가 있어 대회에서 여러 방안들을 적용하는 시도

1. sim2real

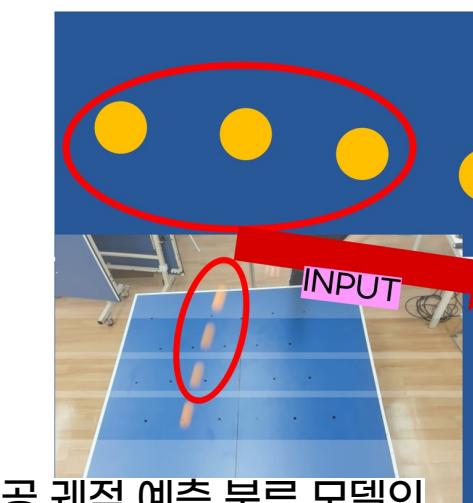


▲ 유니티로 환경 제작 후,
PPO 알고리즘을 이용해
학습시킨 탁구로봇 에이전트

- : 가상환경에서 로봇을 강화학습,
이를 추후 실제환경에 적용
- > 가상환경이 실제와 많이 다름
(공 탄성력, 로봇 동작 등)

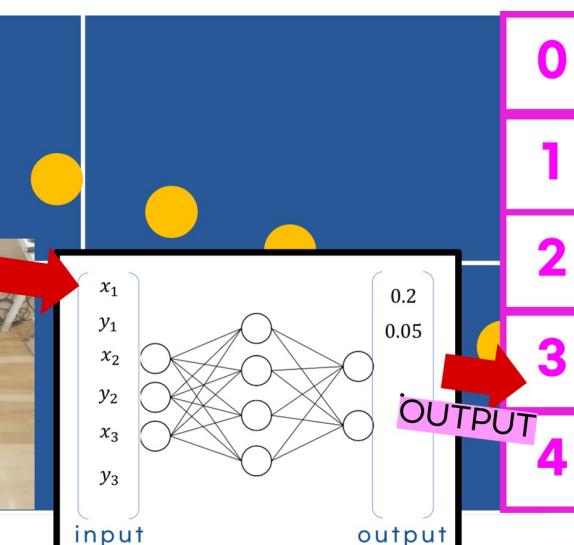


2. 공 궤적 예측



▲ 공 궤적 예측 분류 모델의
작동 방식

- : 리니어 이동 위치를 원핫인코딩한 output으로,
공 초기 위치 입력 시 궤적을 예측하는 분류 모델 학습



- > 모델 순전파에 시간이 상당히 소요됨,
데이터셋 부족으로 정확도가 좋지 못함

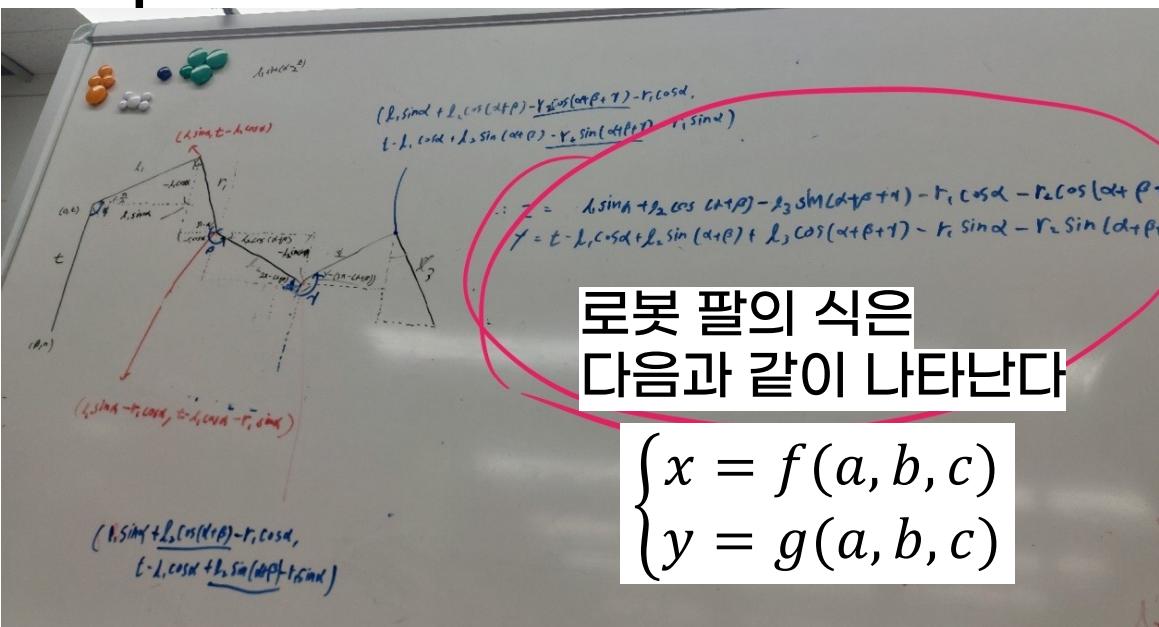
딥러닝



그러나 모두 실제환경 적용에서의 속도/정확도/전제오류 문제로 실패함.

3. 로봇 관절 제어

: 로봇 팔 끝점 좌표(x, y) 입력 시,
각 관절 모터의 각도(a, b, c)를
출력하는 함수를 만들고자 하였다.



! 최종식이 두 개인 반면, 미지수는 세 개이므로
계산상으로 해를 구할 수 없었다.

: (x, y) -> (a, b, c) 예측 모델 학습을 시도.

< 학습 데이터셋 생성 >

- Output인 (a, b, c)를 0~360 범위의 난수로 추출
- 추출한 (a, b, c)로 $x=f(a,b,c)$, $y=g(a,b,c)$ 계산
- 과정 1, 2를 반복하여 $[(a,b,c), (x,y)]$ 데이터셋 수집

< 학습 결과 >

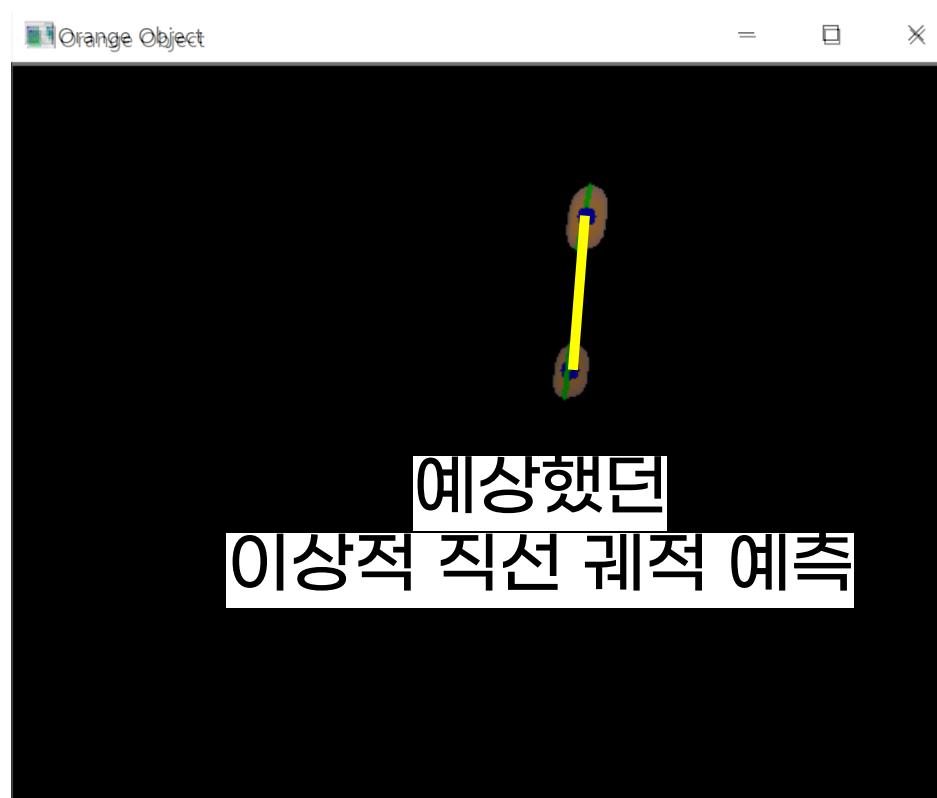
159.1728973388672 241.01614379882812 119.26670837402344
정답 : 10 15
예측 : 11.01191390431169 15.589777386701796
172.15322975976562 249.64210510253906 133.97789001464844
정답 : 20 20
예측 : 20.323160769585925 21.345557316857384
147.2723846435547 234.83004760742188 160.4669952392578
정답 : 25 10
예측 : 25.330159729015456 10.369973178861962

: 높은 정확도를 보였지만,
공의 최종 3차원 좌표값이
정확히 구해진다는 전제가
성립되지 못하여 실패

1 프레임 예측



하나의 프레임으로 탁구공의 최종 위치를 예측할 수 있다면
매우 빠르게 로봇을 조종할 수 있을 것이라 생각하여,
카메라에 찍히는 공의 잔상을 이용해 방향을 예측하는 것을 시도



예상했던
이상적 직선 궤적 예측



: 실제로는 masking된 공의 잔상에서
기대범위의 일부가 누락되는 상황이 빈번하게 발생했고,
이러한 불규칙한 모양으로 탁구공 잔상 영역이 감지되어
구해지는 직선의 기울기 오차가 매우 컸다.

허프 변환 검출

초기 설계 시에, 매 웹캠 frame 입력마다
(1) HSV 주황색 masking (2) 허프 변환 원 검출 의 고집합 중
가장 큰 영역을 탁구공으로 검출하려고 하였다



예상했던
이상적인 공 검출 과정

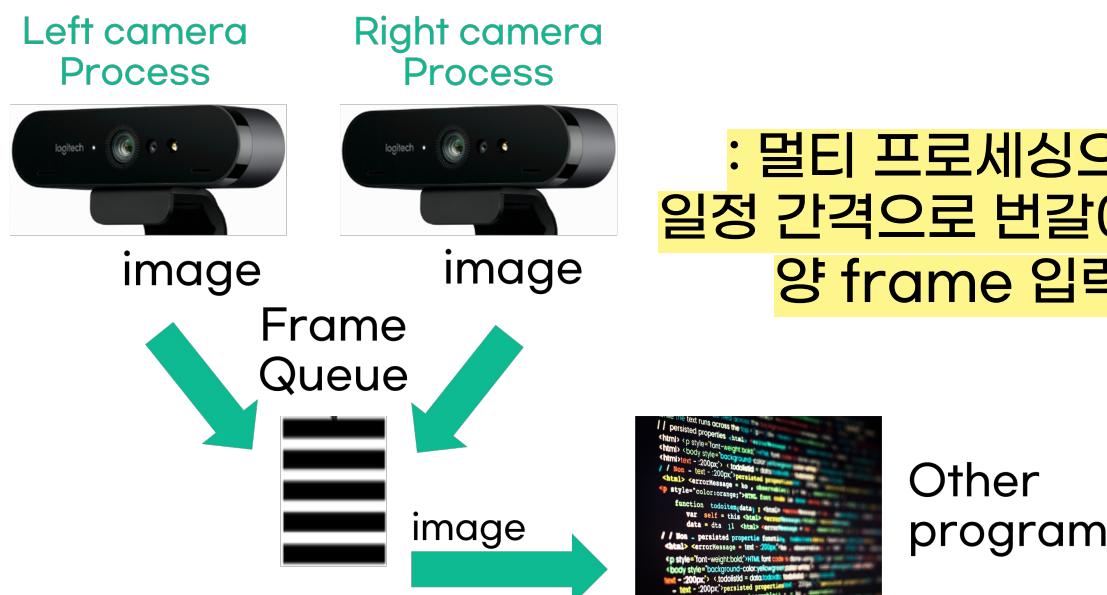
: 그러나 실시간 적용 테스트 결과,
1) 매 frame마다 허프 변환을
수행하니 처리 속도가 매우 지연
2) 탁구공이 빠르게 움직이며
잔상이 길게 남아
원으로 검출하기 어려움



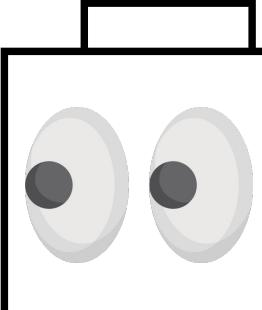
60fps로 입력

Logitech brio 4K pro webcam을 이용할 때 전용 소프트웨어를 이용하면 60fps까지 올리는 것이 가능했지만, 파이썬과 C++을 이용해 opencv로 프레임을 받아올 때는 수차례의 시도에도 불구하고 30fps를 넘기지 못하였다.

<두 개의 30fps 카메라로 60fps 구현하는 방법>

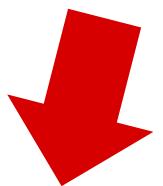


: 나름의 해결 방안을 찾았지만, 이 방식으로는 스테레오 비전을 사용하지 못하기 때문에 탁구공 위치 계산 방식이 바뀐다면 이용 가능한 방안이다.



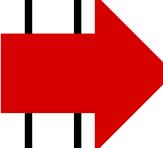
Depth 문제

stereo vision을 사용하여 3차원 좌표 (x,y,z)를 구하는 데에 있어 우선적으로 depth 측정 오차가 너무나 컸다.



< 오차 요인 후보 분석 >

- 1) 카메라 파라미터를 잘못 설정
- 2) 스테레오 calibration이 제대로 수행되지 않음
(이렇게 되면 x,y 축이 두 카메라에서 서로 어긋나기 때문에)
- 3) 이론상의 triangulation이 실제 상황에 딱 맞지는 않는 문제



< 개선 방향 >

- 1) 파라미터 설정과 calibration을 정확히 수행
- 2) Opencv 스테레오 sgvm 함수 사용
- 3) Triangulation 계산 없이, 비전 시스템에 z 축 카메라를 따로 설치해 바로 z 값을 측정

팀워크 이야기

저희 인삼팀 4명의 팀원이
대회 진행을 위해
상호 협업해온 방식들을 이야기합니다.



정기 모임

대회 프로젝트 개시 이후 매주 2-3번 이상 오프라인 정기 모임을 가졌습니다.

6 Jun							7 Jul							8 Aug						
S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S
														1						
														1						
4	5	6	7	8	9	10	2	3	4	5	6	7	8	6	7	8	9	10	11	12
11	12	13	14	15	16	17	9	10	11	12	13	14	15	13	14	15	16	17	18	19
18	19	20	21	22	23	24	16	17	18	19	20	21	22	20	21	22	23	24	25	26
25	26	27	28	29	30		23	24	25	26	27	28	29	21	22	23	24	25	26	27
							30	31						28	29	30	31			

▲ 정기 모임 일정 기록

또한 다신빌딩 513호에서의 오프라인 모임뿐만 아니라,
온라인으로도 꾸준히 관련 작업을 진행하고 진척을 보고하며 소통했습니다.

노션 활용

협업 상황에서 아래와 같은 다양한 목적을 위해 Notion 을 사용하였습니다.



The screenshot displays four Notion pages:

- 로봇 구조 설계 고민**: A page about robot structure design, featuring a list of points and a section on how to use Figma.
- 좌표 추출 : 생각 정리 노트**: A note-taking page for coordinate extraction, containing a list of steps and a section on using a webcam.
- 모델링 작업**: A modeling work page with sections on using open manipulator, Blender, and motors, along with a specific section on 3D printing a motor model.
- py 병렬처리**: A page on parallel processing with Python, comparing threading and multiprocessing, and discussing GIL and I/O-bound vs CPU-bound tasks.

Below these pages is a calendar view for July 2023, showing scheduled events like "전체 OT" and "교육".

- 팀 일정 관리
- 예산 사용 관리
- 매일매일의 시행착오를 기록하고 이전 맥락을 보존
- 각 작업의 진행사항과 Todo를 관리
- 아이디어 브레인스토밍
- 특정 주제에 대한 고민을 공유

깃허브 활용

분담하여 작업했던 코드 작성 결과를 단계적으로 Github에 기록하였습니다.

The screenshot shows a GitHub repository page for the user 'pingpong-insam'. The repository is private, has 1 watch, 0 forks, and 1 star. It contains 2 branches (main) and 0 tags. The 'Code' tab is selected. The commit history shows 16 commits from 'gyoenge' over the last week, with the most recent commit being 'VisionSystem to Vision, all deleted and add only calibration py for 8...' at 2ae4dac. Other commits include adding ballPosData Collecting autoVER, velocity change function in motor control, datacollection/camlearning, and a checkpoint. A blue button at the bottom encourages adding a README.

pingpong-insam Private

Watch 1 Fork 0 Star 1

main 2 branches 0 tags

Go to file Add file Code

About

2023학년도 GIST 창의융합경진대회 인삼 팀 탁구로봇 코드

Activity 1 star 1 watching 0 forks

Releases

No releases published Create a new release

Help people interested in this repository understand your project by adding a README. Add a README

Commit	Message	Date
gyoenge	VisionSystem to Vision, all deleted and add only calibration py for 8...	2ae4dac last week 16 commits
.idea	Vision : add ballPosData Collecting autoVER (needs Test)	2 weeks ago
ControlSystem	added velocity change function in motor controll	2 weeks ago
DataCollection/camLearning	added datacollection/camlearning	2 weeks ago
ML	add checkpoint	2 weeks ago
Vision	VisionSystem to Vision, all deleted and add only calibration py for 8...	last week

기본검증-ing 이야기

8/4 기본검증 결과 및
이를 통해 깨달은 보완점,
본선 대회 방식에 대비한 보완점,
그리고 현재 작업 중인 과제들을
이야기합니다.



기본검증 결과

기본 검증 결과, 90점 만점에 60점 기록, 통과!

결과 정리

- 중간으로 오는 공은 대부분 3점을 받았다.
 - 좌우로 오는 공은 네트는 넘겼지만 방향이 맞지 않아 반대편 탁구대에 도착하지 못하여 대부분 2점을 받았다. (이는 로봇에 좌우 회전이 없어서이다)
 - 테스트 때에는 거의 모든 공에 반응했지만 실제 기본 검증시에 공 자체를 인식하지 못하는 경우가 존재했다.
(이는 핸빛을 제대로 통제하지 못해서 일어난 상황으로 보인다)

보완점

기본 검증 결과 기반 보완사항

- 로봇이 좌우로 회전할 수 있도록 구조를 변경
- 햇빛을 더 꼼꼼히 통제할 수 있는 환경을 조성

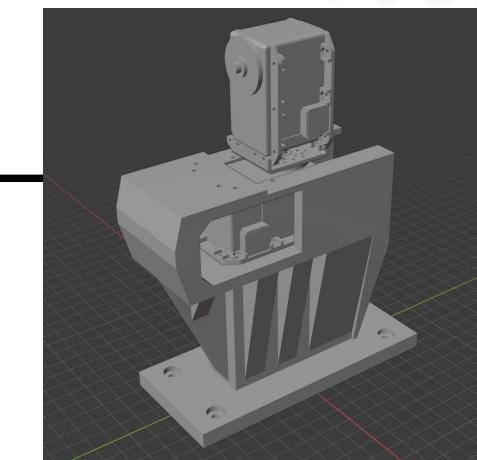
본선 환경 기반 중요 보완사항

- 사람 리시브의 불규칙한 z축 움직임을 대응하는 알고리즘/타격모션 구현
 - 비전시스템의 공 위치 추출 알고리즘을 더 정확하게 보완
- 공 탐지 주황색 masking에 사람의 손/팔이 인식되지 않도록 통제
 - 로봇 모션 제어, 웹캠 입력의 속도 향상 꾸준히 시도

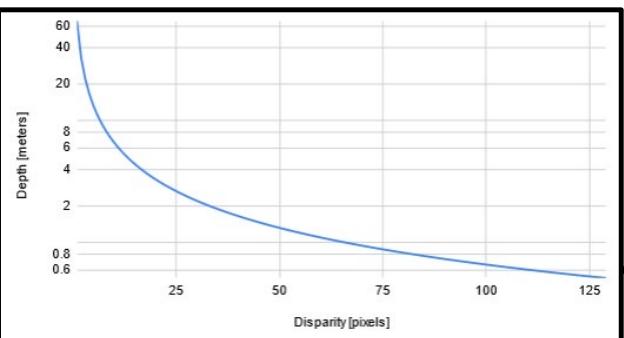
현재 작업 중인 과제들

기본검증 이후 **현재, 다음과 같은 과제들을** 작업하고 있습니다.

로봇 모터 좌우 회전 추가



정확한 스테레오 비전 완성



Z축 궤적 예측 알고리즘 완성



비전시스템 통제환경 보완

