PetBot : A Facial Recognition, Pose Detection, and Ball Tracking

Professor Euntai KIM
T.A Seokwon CHOI, Hongchan JO

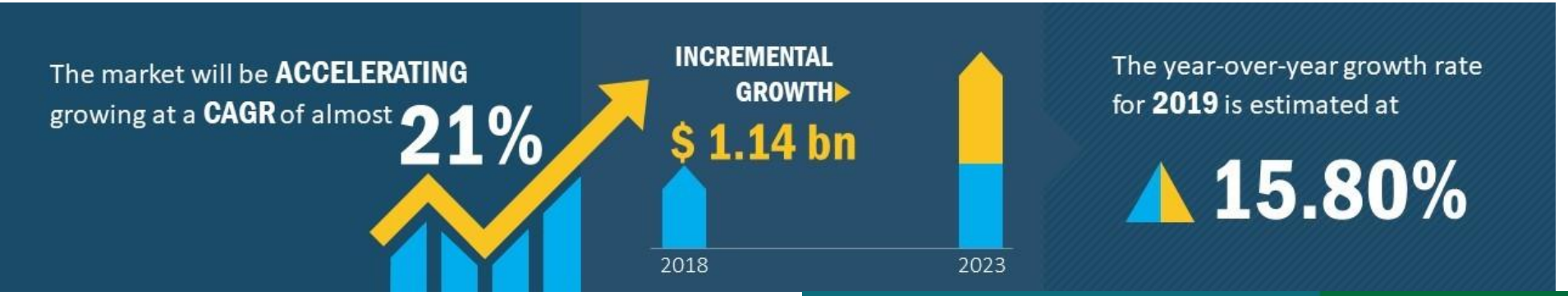
Team 15
Dokyu IM, Hokyun IM, Kyunghoon Jung

1

Introduction

Part 1

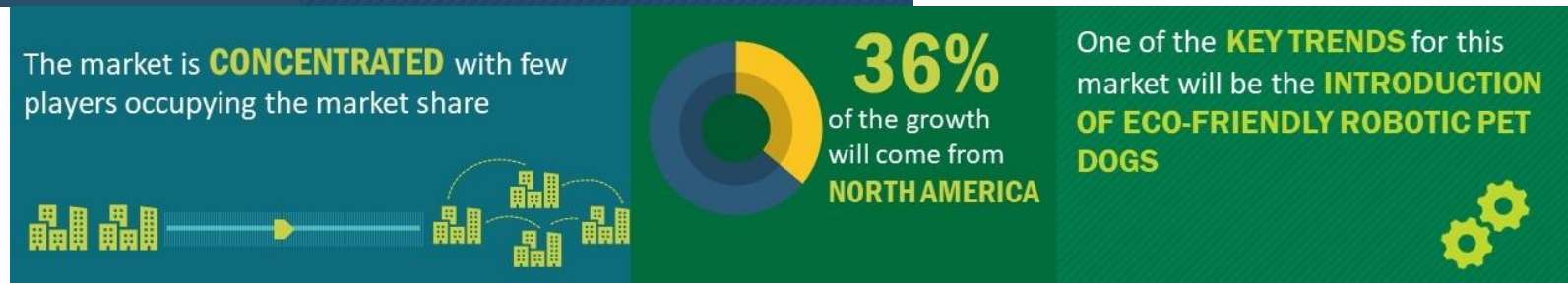
Introduction(Pet Robot Market)



→ CAGR 21%

5years Growth

$1.14 bn (1.6조 원)



→ CAGR 11.28%

5years Growth

$0.914 bn (1.3조 원)

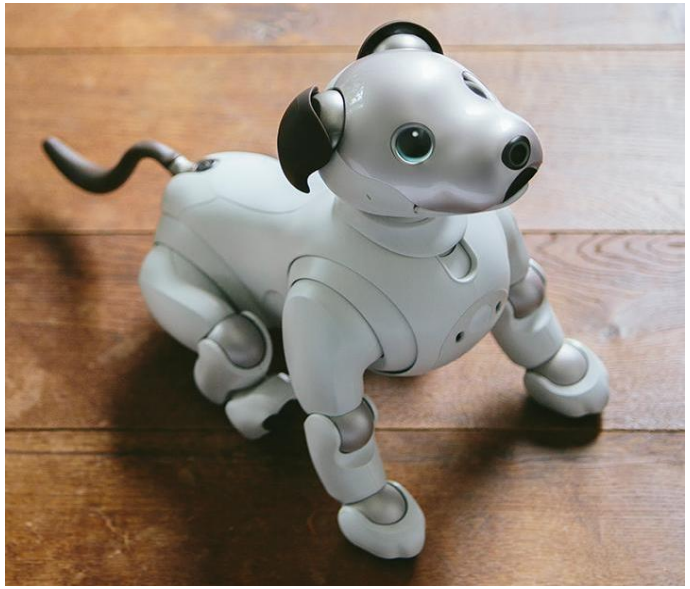
2

Related Works

Part 2

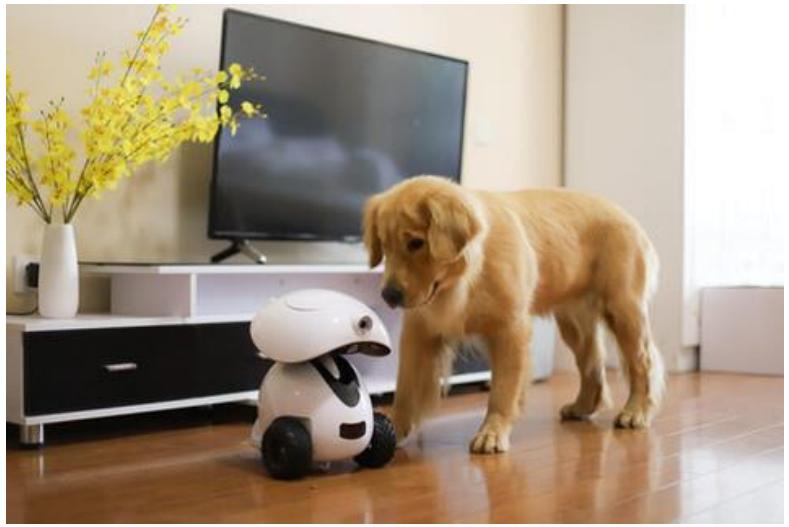
Related Work

Hasbro’s Joy(‘15.11, $99)



Sony’s AIBO(’19.02, $2,899)

Loona Pet Robot(’23.02, $299)



돌봄(CES 2019)

Guru IOT’s PEDDY(’19.06, $299)

3

Robot Configuration

Part 3-i

Software

Step 1

ubuntu

ROS

ROS Environment Setting

Step 2

ROS.org

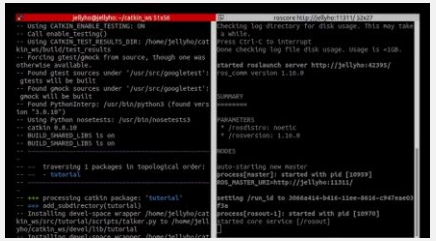
Documentation

kobuki

Only released in EOL distros: groovy

Kobuki Development Setting

Step 3

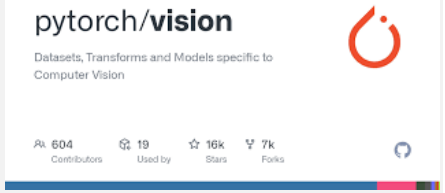


Migrate to Ros1 Noetic

Step 4

NVIDIA JetPack

PyTorch



Install DL Libraries

Part 3-ii

Hardware

RGB / Depth Camera

WIFI Module

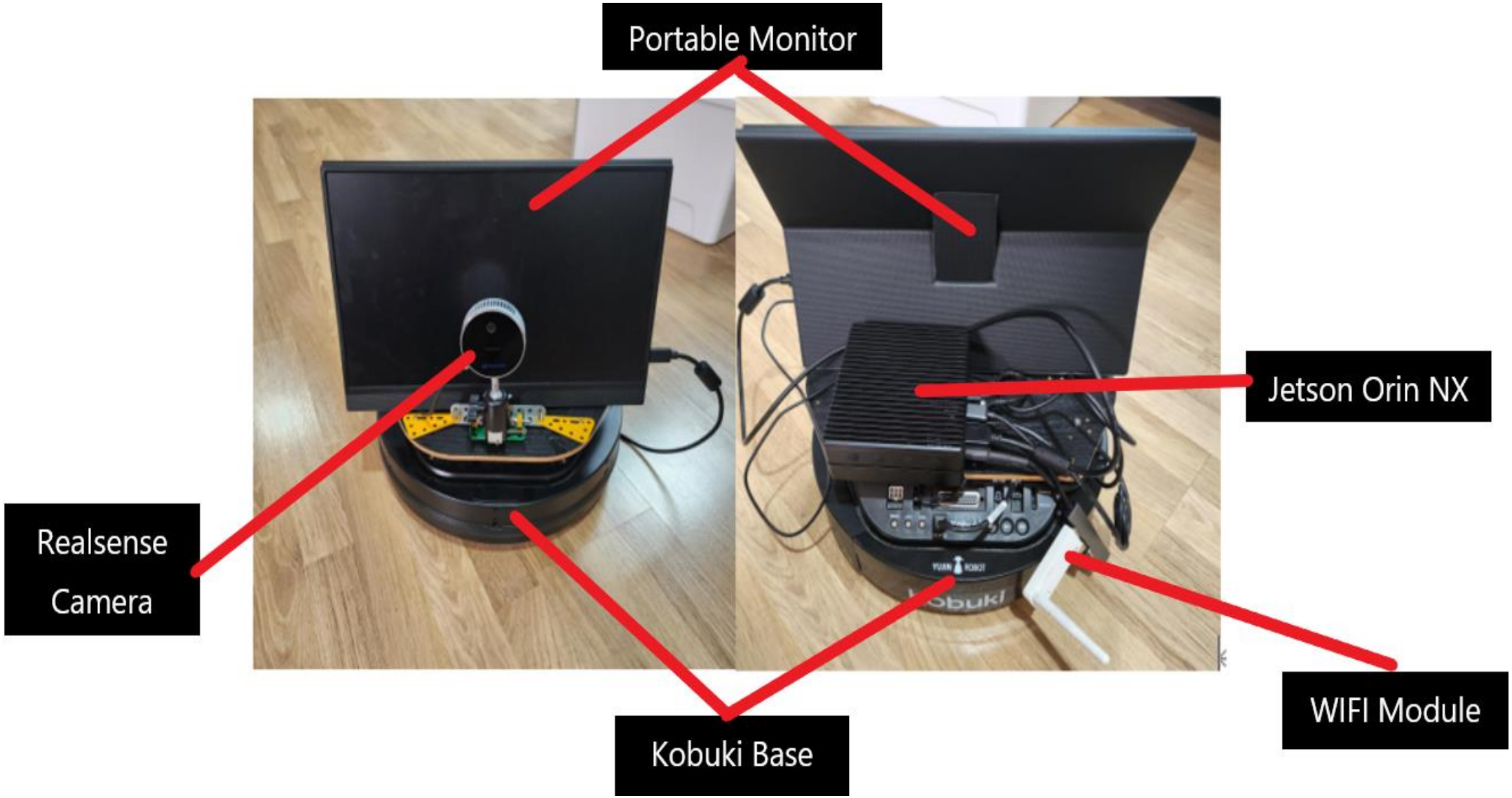
Robot Base

Edge GPU Device

Portable Monitor

Part 3-ii

Hardware



4

Function Implementation

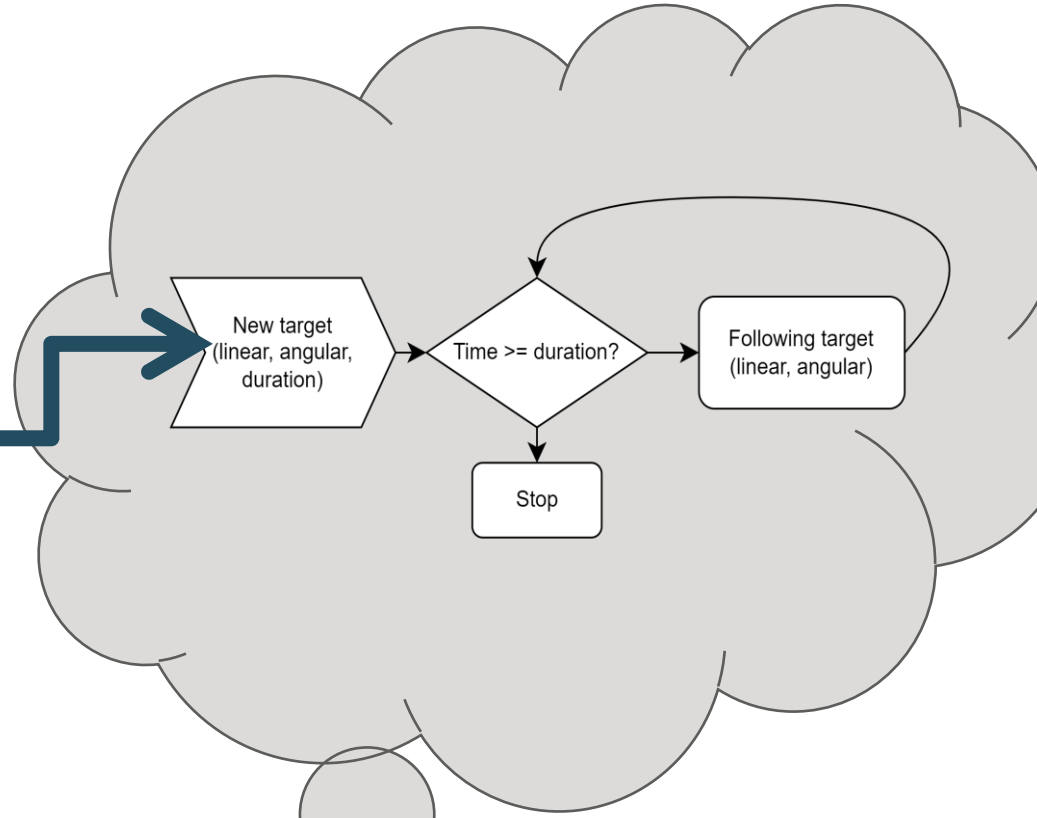
Part 4-i

Basic Movements

High-Level Decision

Movement Command

< 1.0m/s, 0.01rad/s, 2.0s >
<Linear, Angular, Duration>



Low-Level Controller

Kobuki API

Move!

Fail Safe

For the safe development and operation, robot needs fail safe.

If a hazard is detected, this must be operated before any code.

Part 4-ii

Fail Safe

Wall Detection

Wall Detection: Move-Collision-GoBack-Turn

Cliff Detection

Cliff Detection: Move-Detect-GoBack-Turn

Emergency Stop (PickUp Detection)

PickUp Detection: Move-Wrong Move-PickUp-Terminated

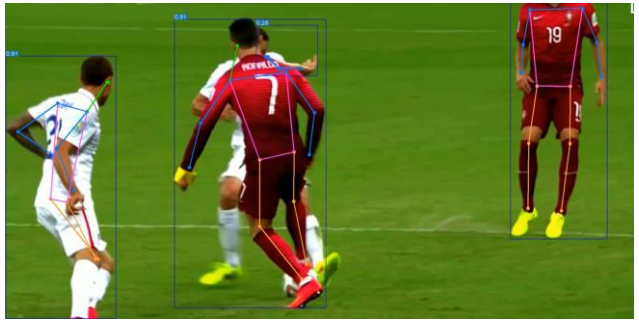
Real Time Pose Recognition/Classification

Pose Estimation Framework

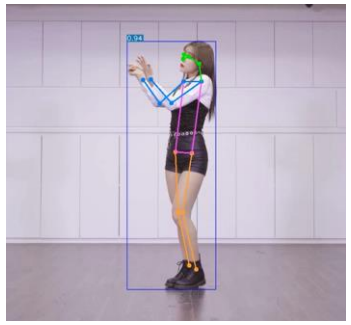
1. Yolo(You Only Look Once) Series
2. MediaPipe

Mediapipe

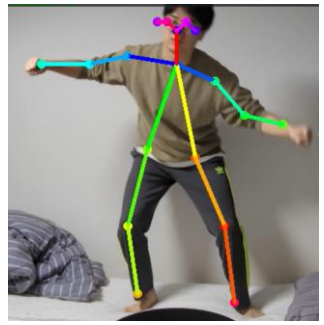
Yolo(You Only Look Once) Series



Test(Yolov7) : Football video



Test(Mediapipe) : Dance video



Test(Open-pose) : Realtime webcam

Real Time Pose Recognition/Classification

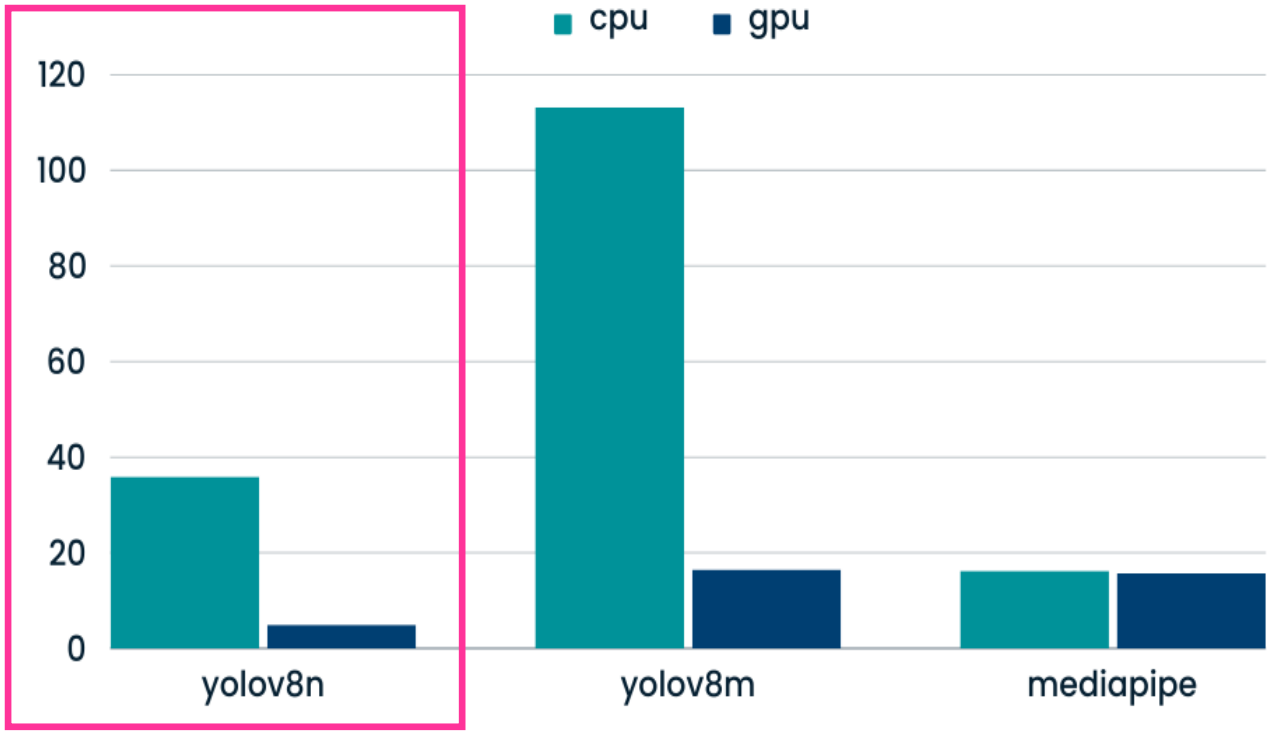
Measurement Environments

Input Size : 640 x 480
CPU/GPU WarmUp : Using Dummy Example(About 100)

CPU : 13th Gen Intel(R) Core™ i7-13700 2.10GHz
GPU : RTX 3060ti

	cpu	gpu
yolov8n	35.94	4.94
yolov8m	113.18	16.49
mediapipe	16.25	15.7

Result Table

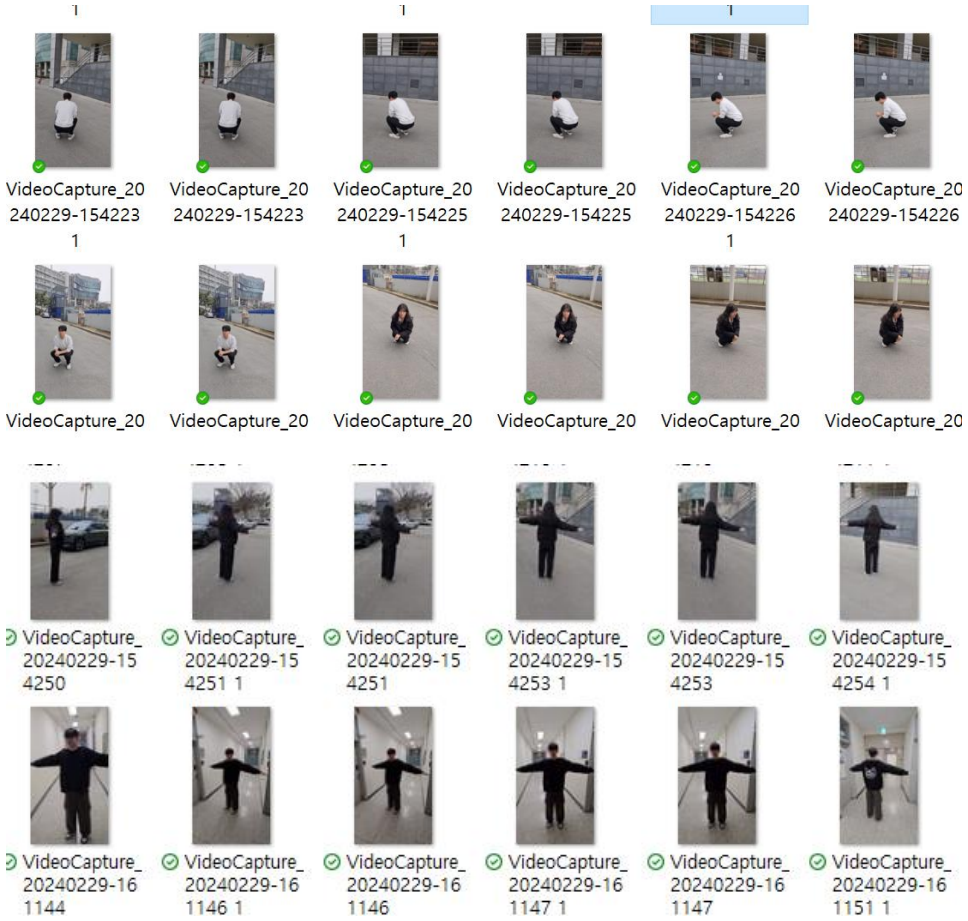


Result Graph

Real Time Pose Recognition/Classification

Data Collecting 1st

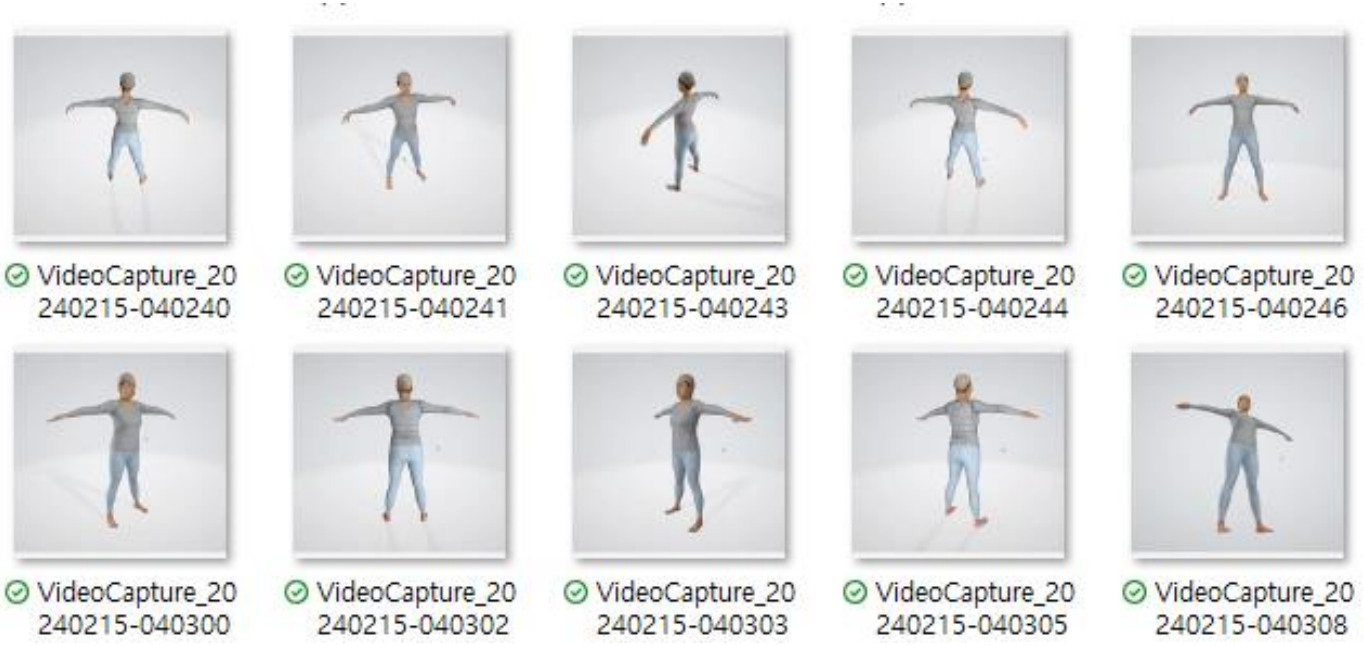
*3 Poses x 3 Person x 10 images(multi-view) = **93 images(Real World)**



*Attention Position(Stop)/Arm Stretch(Start)/Sit(Spin)

Korean 3D Scan Big Data :
3 Poses x 10 Person x 9 images = 274 **images**

Total 367 images



Real Time Pose Recognition/Classification

Data Collecting 2nd : Data Augmentation

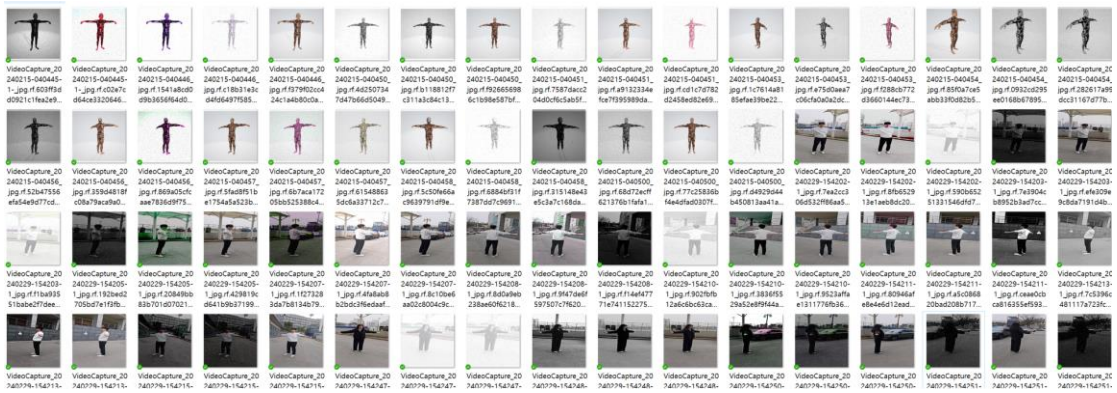
Original Data

Data Augmentation

Horizontal Stretching

Adding Noise & Color Filtering

Adding Noise & Removing Color



TestSet

(367 – 60) x 3.14 times = Total 966 images

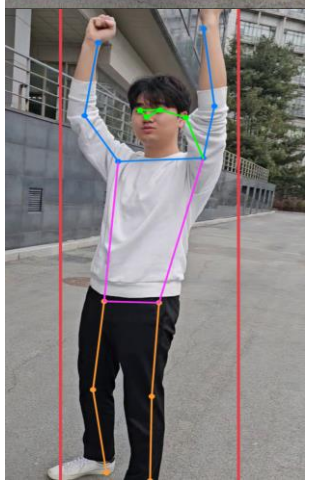
Data 1st

Augmentation

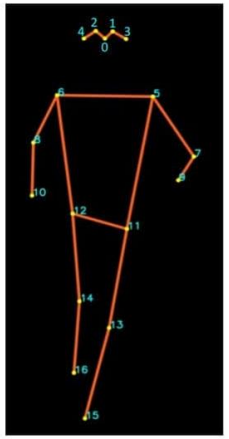
Real Time Pose Recognition/Classification

Pipeline

Extracting Pose Landmark → Pose Data → Translation for Data Format → CSV file → Train → Train Value → Inference



Index	Key point
0	Nose
1	Left-eye
2	Right-eye
3	Left-ear
4	Right-ear
5	Left-shoulder
6	Right-shoulder
7	Left-elbow
8	Right-elbow
9	Left-wrist
10	Right-wrist
11	Left-hip
12	Right-hip
13	Left-knee
14	Right-knee
15	Left-ankle
16	Right-ankle



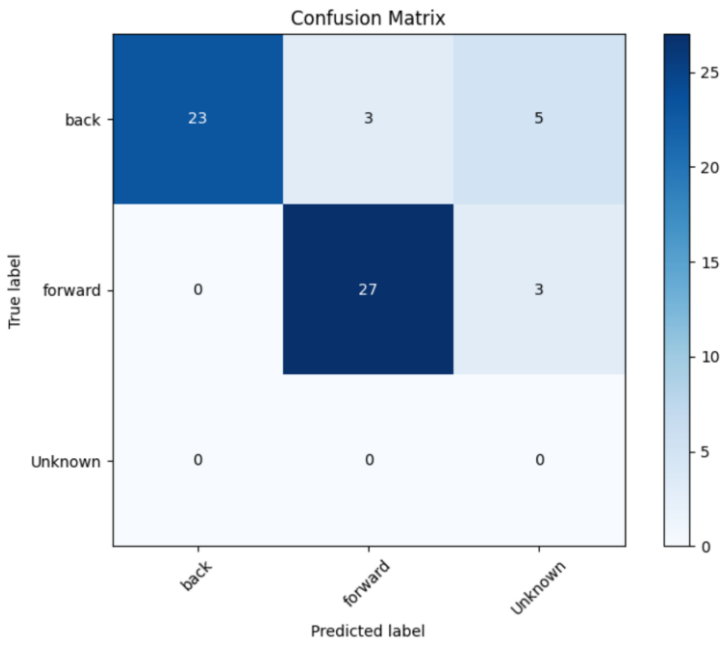
	BU	BV	BW	BX	BY	BZ	CA	
1	RIGHT_PINKY_X	RIGHT_PINKY_Y	RIGHT_PIN	RIGHT_PIN	LEFT_INDE	LEFT_INDE	LEFT_INDE	LEFT
2	-0.006938201	-0.837996503	-1.75142	0.986837	0.187401	-0.8056	-0.12362	0
3	0.000661393	-0.830895322	-1.36322	0.986289	0.201143	-0.80967	-0.01908	0.8
4	0.011974001	-0.829517011	-1.33139	0.985127	0.214762	-0.81581	-0.02978	0.8
5	0.061895834	-0.830839185	-1.21858	0.984827	0.265268	-0.80584	-0.105	0.8
6	0.083531456	-0.846780017	-1.2752	0.984739	0.298476	-0.80092	-0.27082	0.8
7	0.089020216	-0.845600605	-1.60286	0.984044	0.361031	-0.81022	-0.43974	0
8	0.09586136	-0.873492843	-1.53807	0.983557	0.397564	-0.82125	-0.64119	0
9	0.098163472	-0.900622638	-1.62103	0.983494	0.423056	-0.83504	-0.76917	0.8
10	0.09611963	-0.903610154	-1.37828	0.982563	0.432304	-0.84286	-0.94591	0.8
11	-0.113924796	-0.909183951	-1.34089	0.979972	0.40636	-0.8712	-1.01905	0.8
12	-0.180425932	-0.890726069	-1.36494	0.976898	0.338464	-0.87062	-1.41904	0.8
13	-0.207748481	-0.874283721	-1.44103	0.97428	0.301401	-0.8622	-1.5258	0.8
14	-0.238965673	-0.875736291	-1.4484	0.973952	0.252116	-0.86426	-1.62166	0.8
15	-0.275716671	-0.873891067	-1.56368	0.973068	0.201943	-0.86884	-1.70093	0.8
16	-0.287847896	-0.877702753	-1.5709	0.973586	0.184656	-0.87212	-1.73114	0.8
17	-0.300571186	-0.875959511	-1.63006	0.973792	0.16047	-0.88129	-1.76289	0.9
18	-0.315937456	-0.87229568	-1.57698	0.973864	0.13822	-0.87971	-1.69932	0
19	-0.333023801	-0.858728774	-1.57139	0.973383	0.125046	-0.87951	-1.65213	0.9

Optimizer : adam
Loss : categorical crossentropy

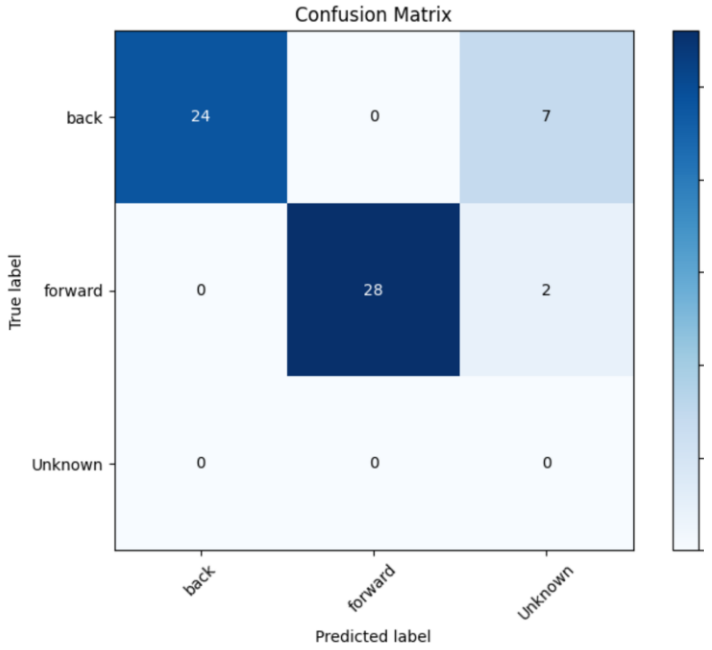
Model : Sequential
512, relu
256, relu
class_num, softmax

Real Time Pose Recognition/Classification

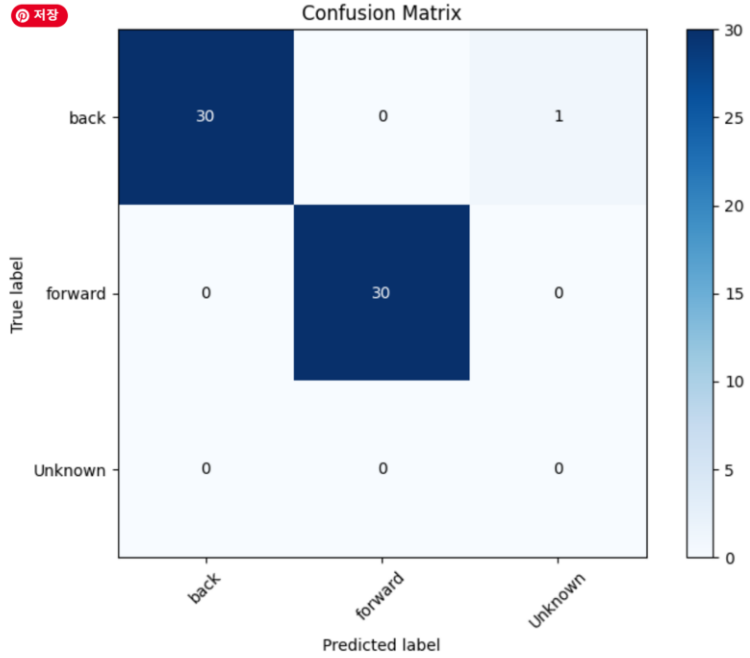
Result



Only 3D Scan Dataset



3D Scan + Real World Dataset



After Data Augmentation

	Before Augmentation	After Augmentation
Accuracy	81.97%	100%
Error Rate	18.03%	0%
Precision	100% (back), 90% (forward)	100%
Recall	74.19% (back), 90% (forward)	100%
F1 Score	85.19% (back), 90% (forward)	100%

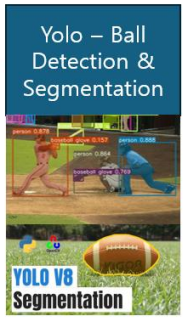
Real Time 3D Ball Detection

Trial And Error1(RGB-D Map Processing)

Ball Tracking PipeLine

RGB MAP

Same Resolution RGB & Depth Map



RGB MAP Segmentation

Depth MAP

Depth MAP Segmentatio

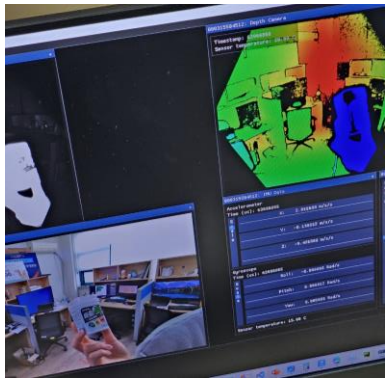
Calculate Distance
Between Robot & Ball

RGB-D MAP Super-resolution

Mode	Resolution	Fol	FPS	Operating range*	Exposure time
NFOV unbinned	640x576	75°x65°	0, 5, 15, 30	0.5 - 3.86 m	12.8 ms
NFOV 2x2 binned (SW)	320x288	75°x65°	0, 5, 15, 30	0.5 - 5.46 m	12.8 ms
WFOV 2x2 binned	512x512	120°x120°	0, 5, 15, 30	0.25 - 2.88 m	12.8 ms
WFOV unbinned	1024x1024	120°x120°	0, 5, 15	0.25 - 2.21 m	20.3 ms
Passive IR	1024x1024	N/A	0, 5, 15, 30	N/A	1.6 ms

Expand table

RGB Camera Resolution (HxV)	Aspect Ratio	Format Options	Frame Rates (FPS)	Nominal FOV (HxV)(post-processed)
3840x2160	16:9	MJPEG	0, 5, 15, 30	90°x59°
2560x1440	16:9	MJPEG	0, 5, 15, 30	90°x59°
1920x1080	16:9	MJPEG	0, 5, 15, 30	90°x59°
1280x720	16:9	MJPEG/YUY2/NV12	0, 5, 15, 30	90°x59°
4096x3072	4:3	MJPEG	0, 5, 15	90°x74.3°
2048x1536	4:3	MJPEG	0, 5, 15, 30	90°x74.3°



Trial And Error2(Marker Tracking)

Aruco Marker Size is constant
So, 3D localization possible!

Can we detect ball in 3d space?
With Constant Ball Size?

Real Time 3D Ball Detection

Result

$$\text{Distance} = \frac{R_{real} \times f \times W_{image}}{R_{img} \times W_{sensor}}$$

W_{image} : Width of the image

R_{real} : Actual radius of the Ball(0.15m)

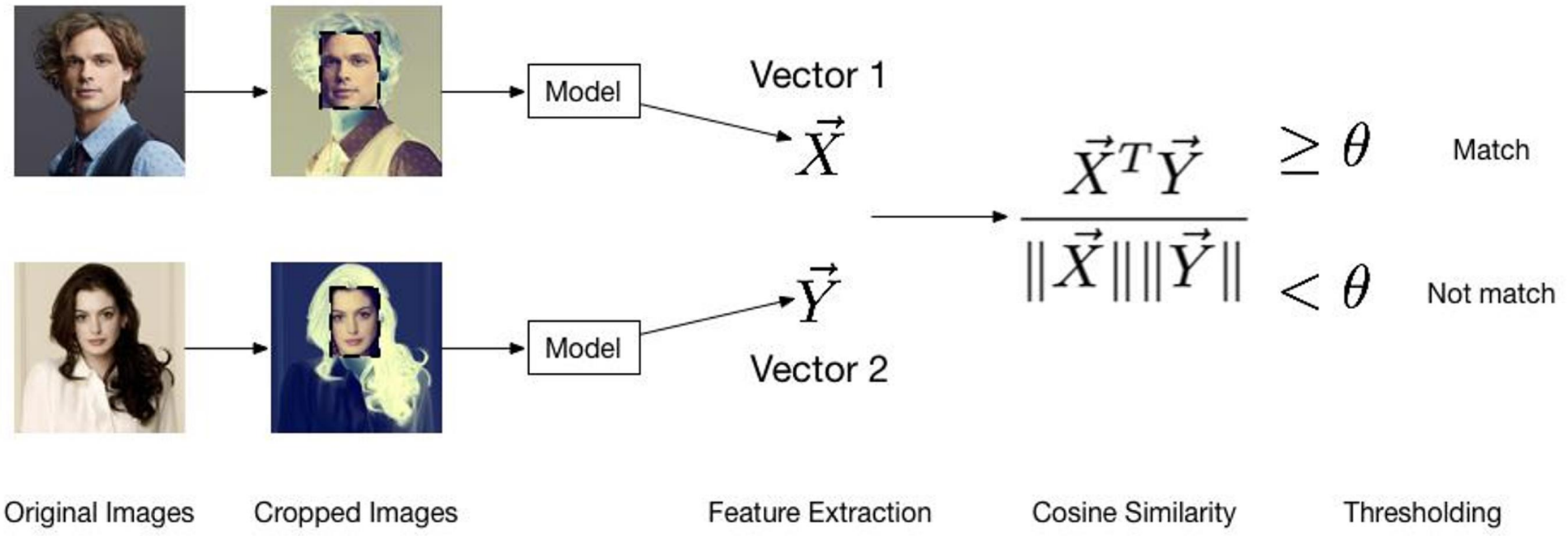
R_{img} : Radius of the ball detected in the imag

f : Focal length of the camera

W_{sensor} : Width of the camera sensor (0.036m, typical for a 35mm camera sensor)

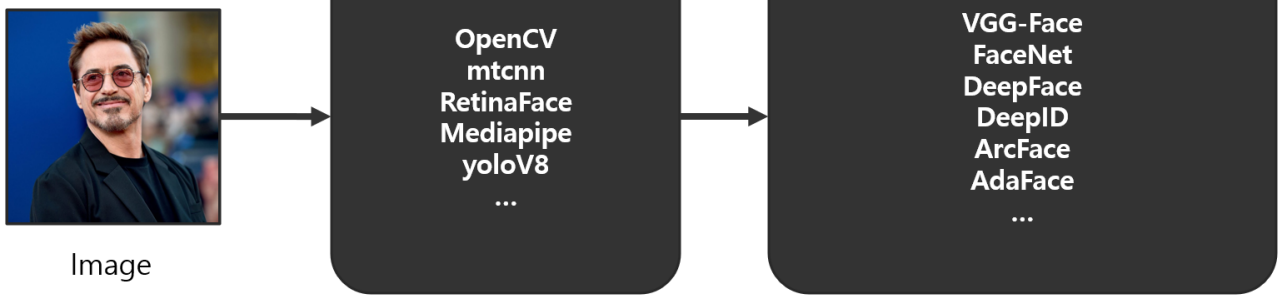
D : Distance between the camera and the ball (in meters)

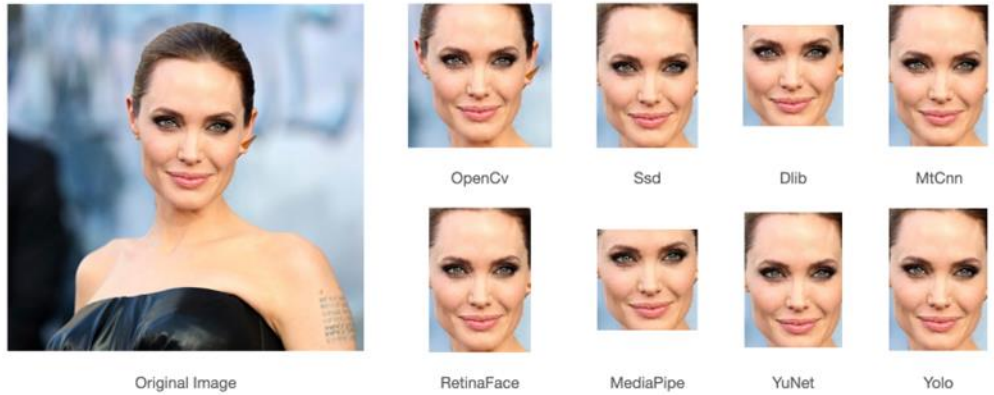
Face Recognition Pipeline



Real Time Face Detection/Recognition

Face Recognition Pipeline

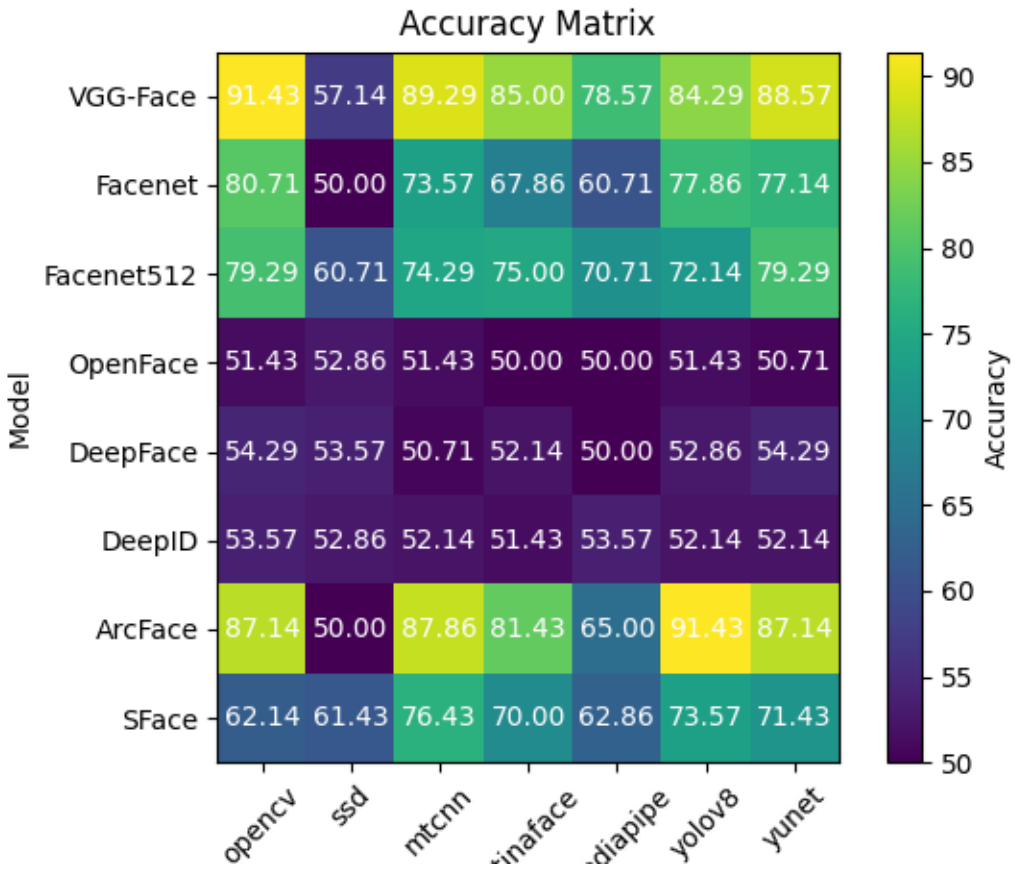




Model	LFW Score	YTF Score
Facenet512	99.65%	-
SFace	99.60%	-
ArcFace	99.41%	-
Dlib	99.38 %	-
Facenet	99.20%	-
VGG-Face	98.78%	97.40%
Human-beings	*97.53%*	-
OpenFace	93.80%	-
DeepID	-	97.05%

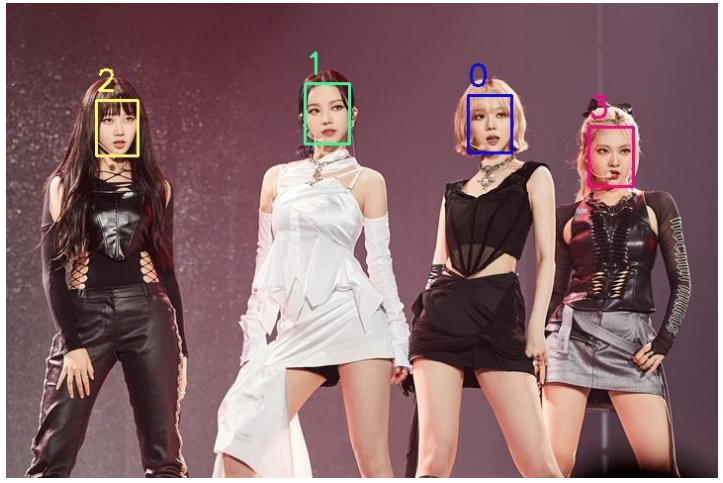
Real Time Face Detection/Recognition

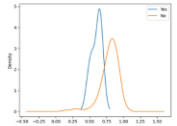
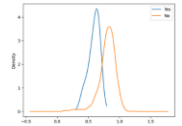
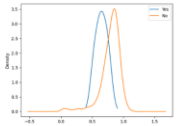
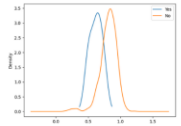
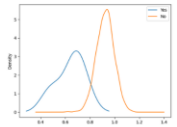
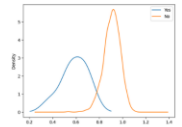
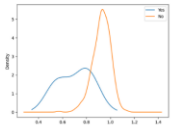
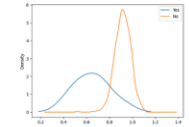
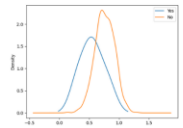
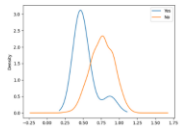
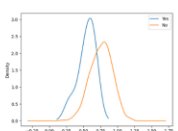
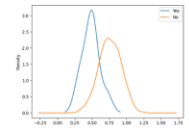
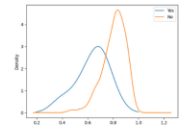
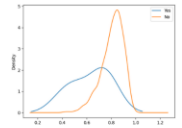
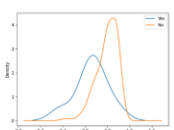
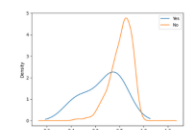
Quantitative Evaluation of Face Recognition Detection Model Combination



Real Time Face Detection/Recognition

Qualtitative Evaluation of Face Recognition Detection Model Combination



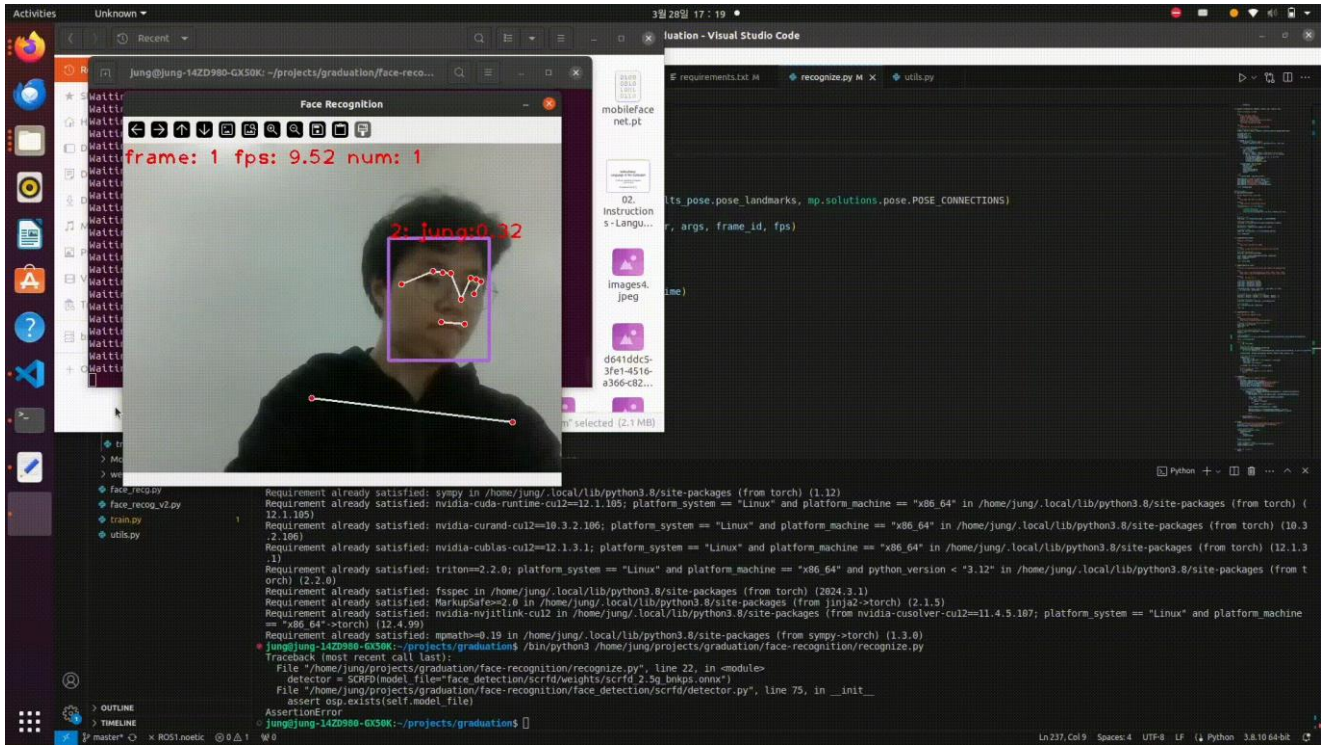
	mtcnn	opencv	retinaface	yunet
ARCFACE				
ADAFACE				
FACENET512				
VGGFACE				

Real Time Face Detection/Recognition

Real Time Face Recognition

- RTX 4090 (fps 30 or above)

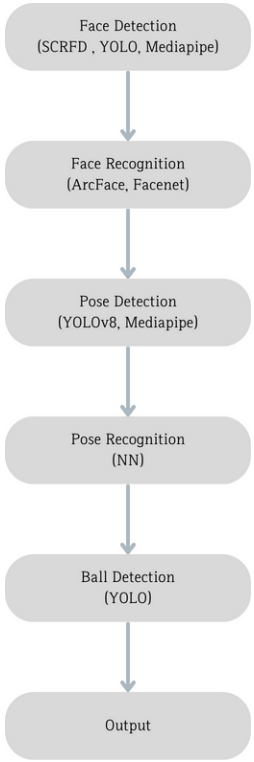
- No GPU (fps 10 or below)

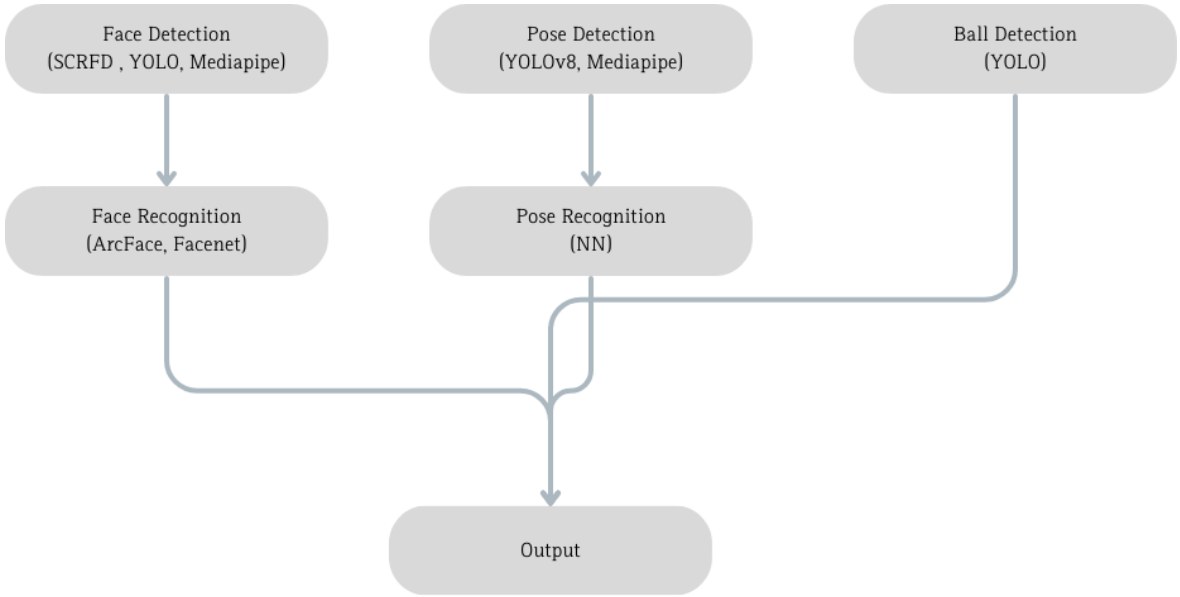


Real Time Face Detection/Recognition

Real Time Face Recognition

- the drop in FPS in the face accuracy measurement model can be addressed by running a lighter model
- However, it is hard to solve the FPS drop when multiple models such as pose detection object detection run together

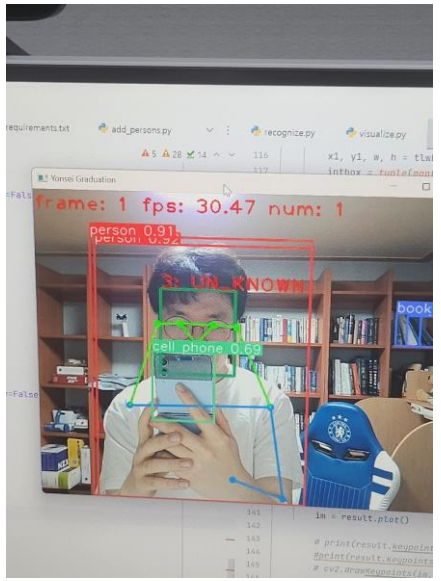


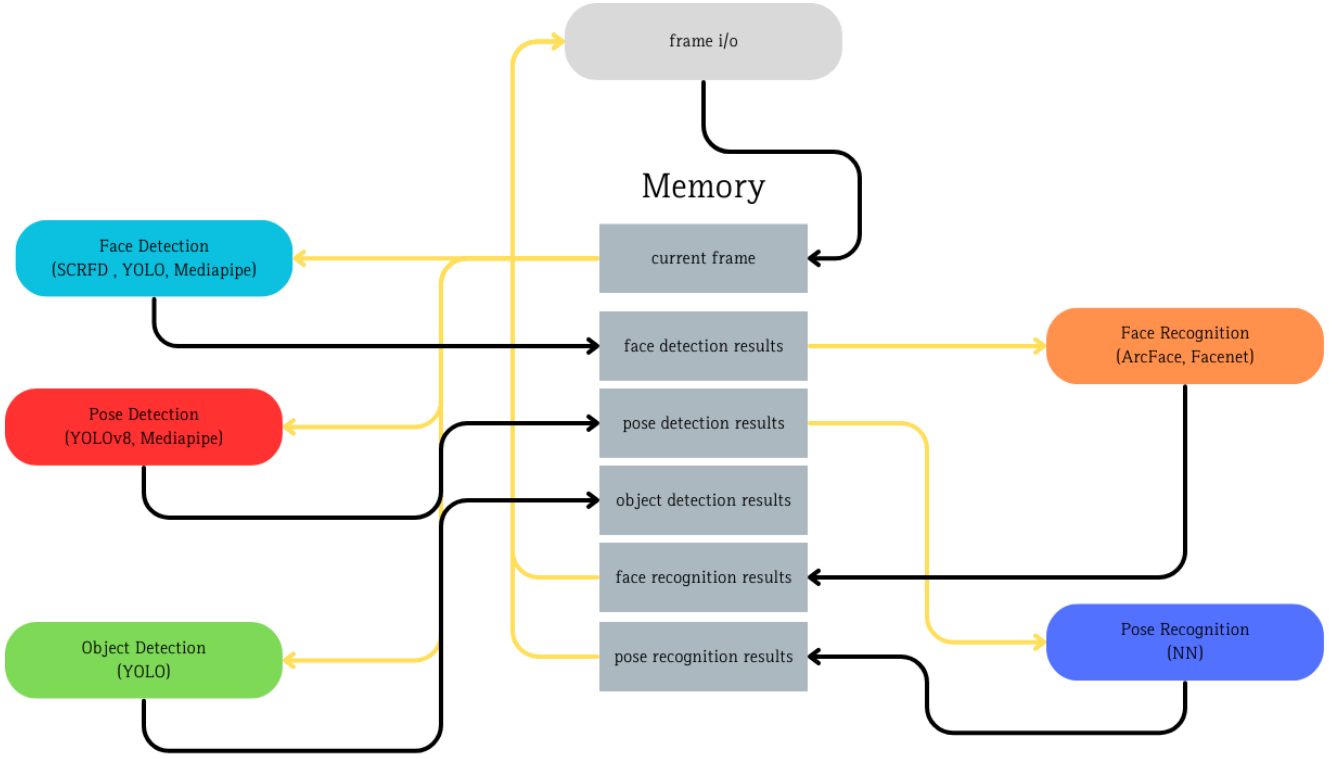


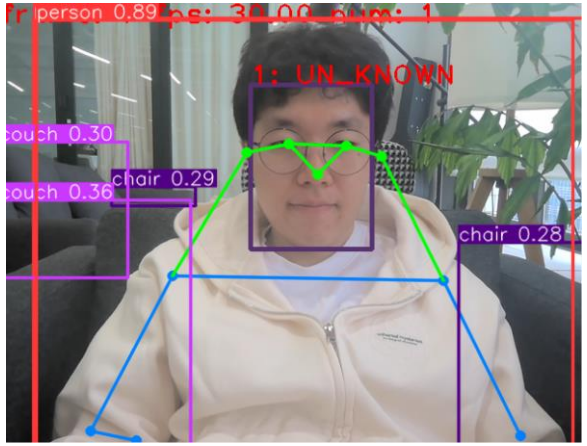
Real Time Face Detection/Recognition

Multithreading Multiple Models

- Each model operates in parallel by reading from or writing to its allocated memory
- Achieve 30 FPS in Laptop CPU, whereas without multithreading it would be under 1 FPS





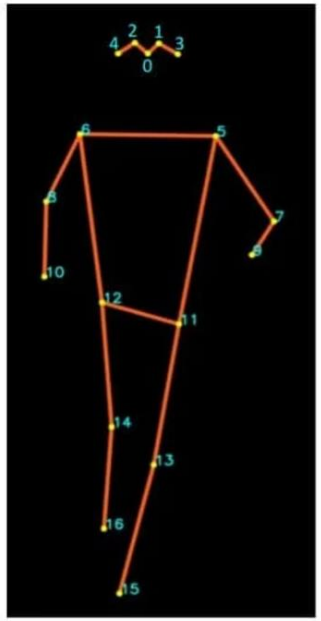


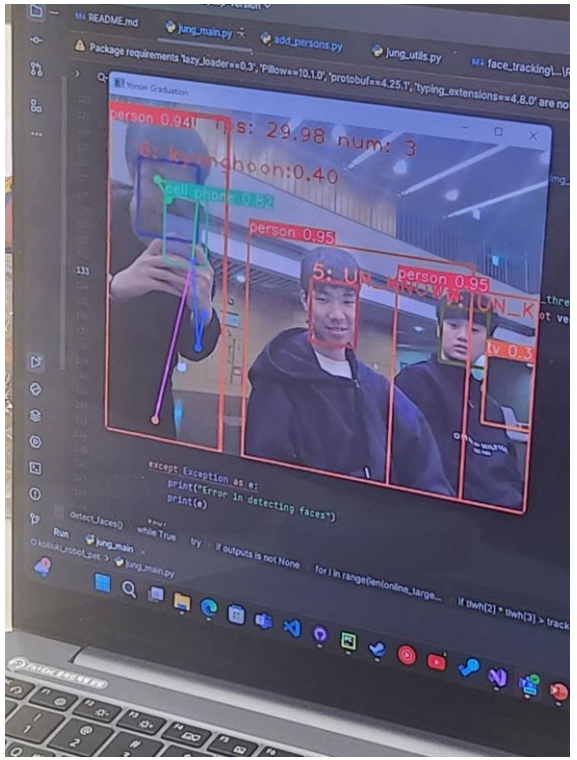
Real Time Face Detection/Recognition

Face-Pose Matching

- Since we are using independent face detection and pose detection models, it is necessary to align the same face and pose
- "Based on the fact that the YOLOv8 pose detection model extracts keypoints for the nose and eyes, if the nose and eye keypoints are within the bounding box of the face detection model, they are considered to belong to the same person

Index	Key point
0	Nose
1	Left-eye
2	Right-eye
3	Left-ear
4	Right-ear
5	Left-shoulder
6	Right-shoulder
7	Left-elbow
8	Right-elbow
9	Left-wrist
10	Right-wrist
11	Left-hip
12	Right-hip
13	Left-knee
14	Right-knee
15	Left-ankle
16	Right-ankle





Human / Ball Tracking and Following

How to follow detected human / ball?

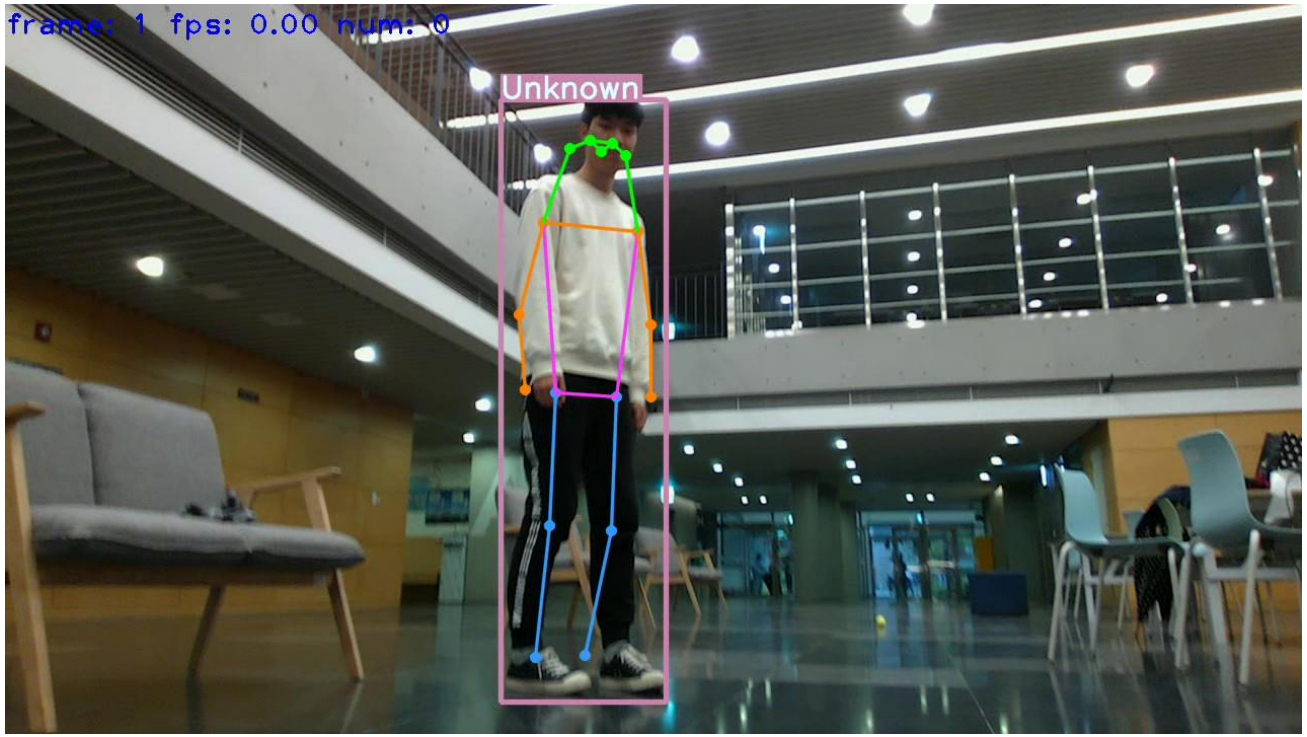


Image Space Detection

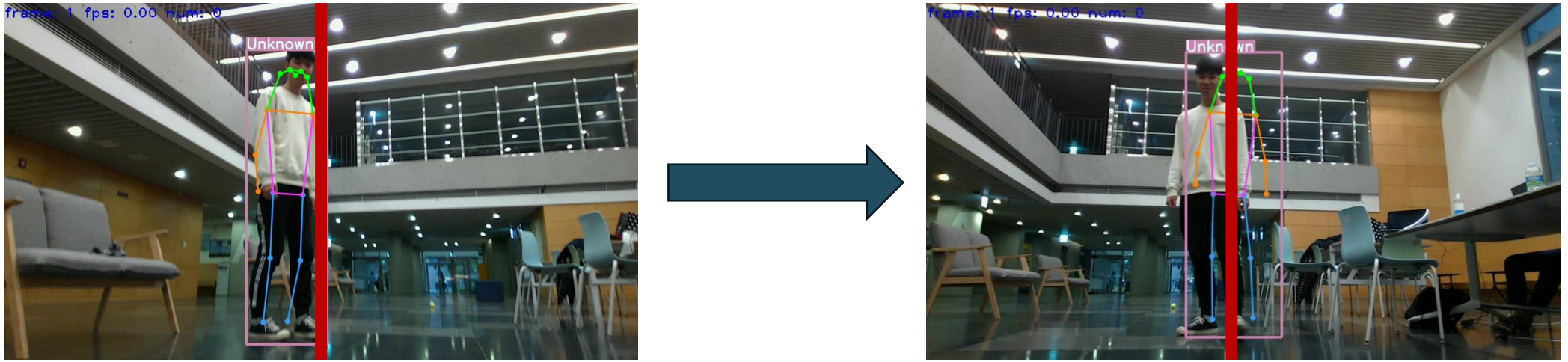
< 1.0m/s, 0.01rad/s, 2.0s >
<Linear, Angular, Duration>

Action

Part 4-vi

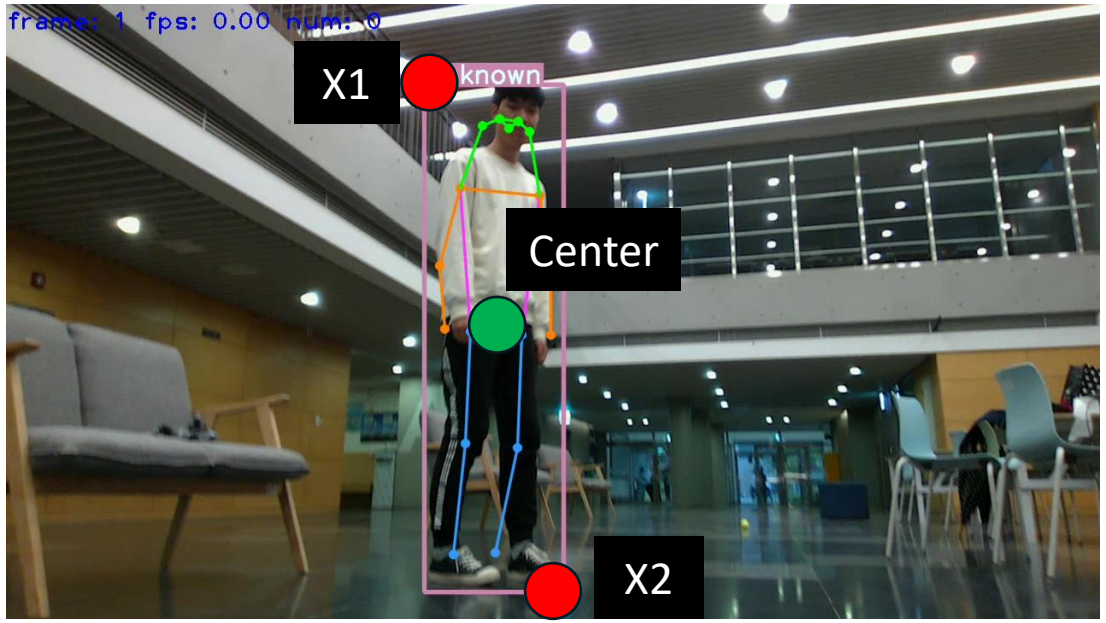
Human / Ball Tracking and Following

Goal : Keep the object in the middle of the screen



Human / Ball Tracking and Following

How far away is the object from the center of image?

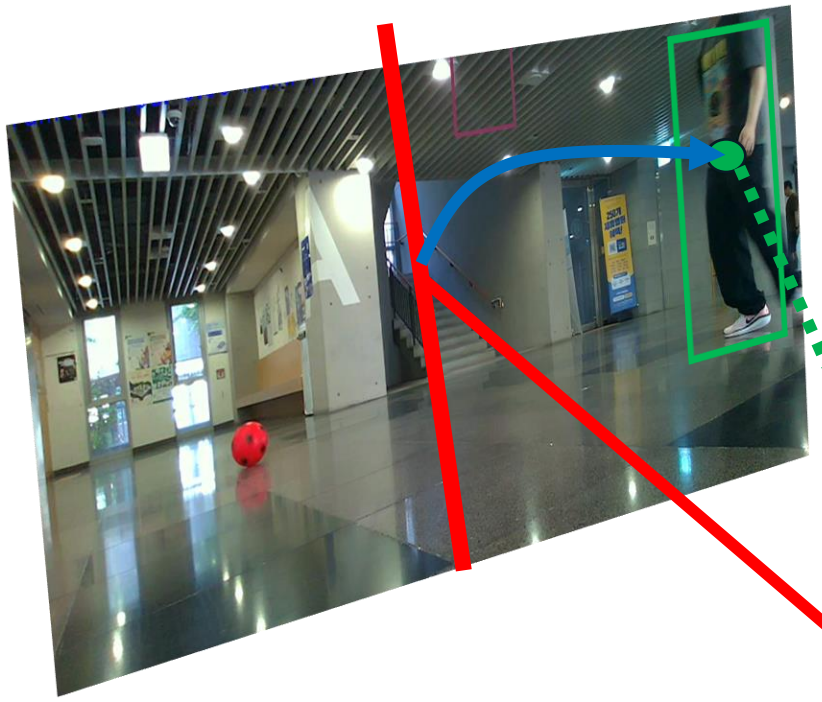


$$Center_{obj} = (box['x1'] + box['x2']) \,/\, 2$$

Part 4-vi

Human / Ball Tracking and Following

How far away is the object from the center of image?



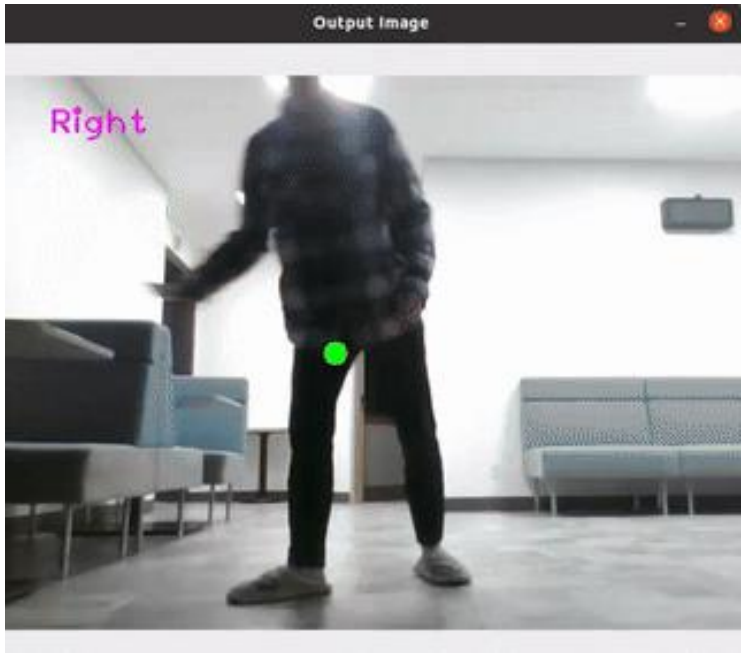
$$linear = 0.01m/s$$

$$\Delta angular = \alpha * \frac{\left(center_{obj} - \frac{width_{img}}{2}\right)}{width_{img}}$$

$$\Delta t = 0.01s$$

$$< linear, \Delta angular, \Delta t >$$

Send this to low-level controller (Part 4-i)

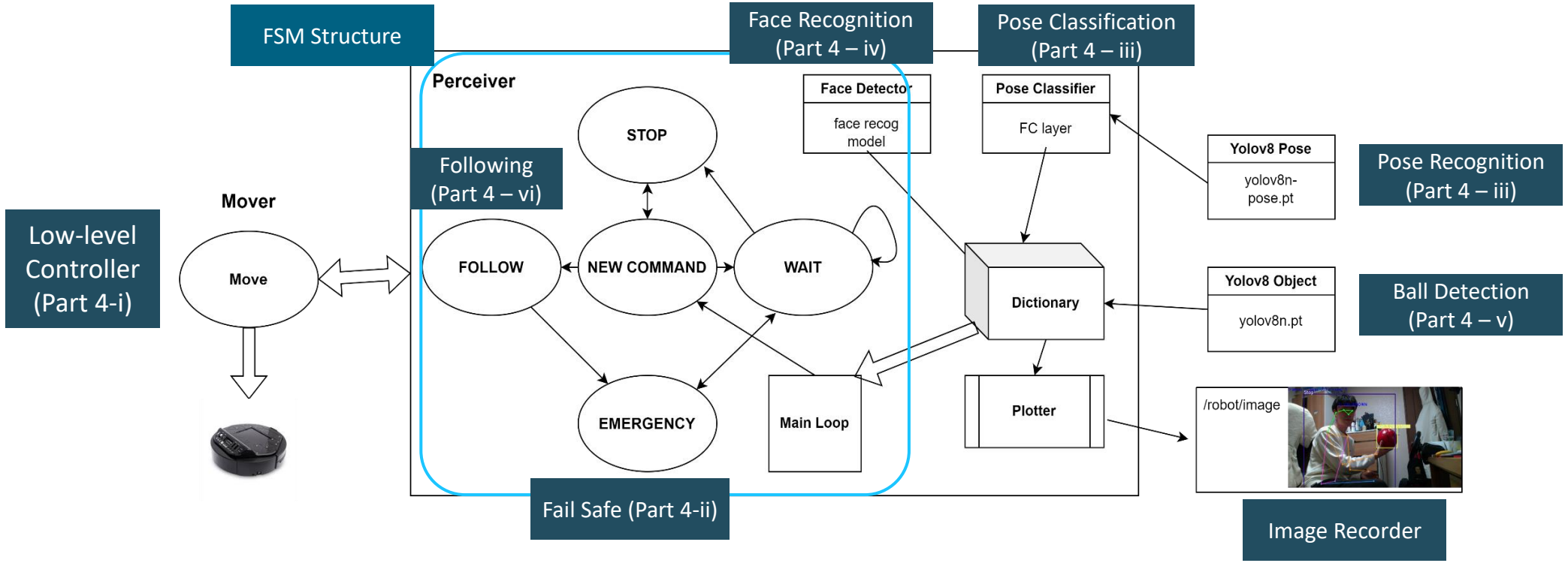


5

Integrating Functions

Node Design

1. Contruct ROS nodes for running each functions together.
2. Finite State Machine (FSM) design for internally making decisions.

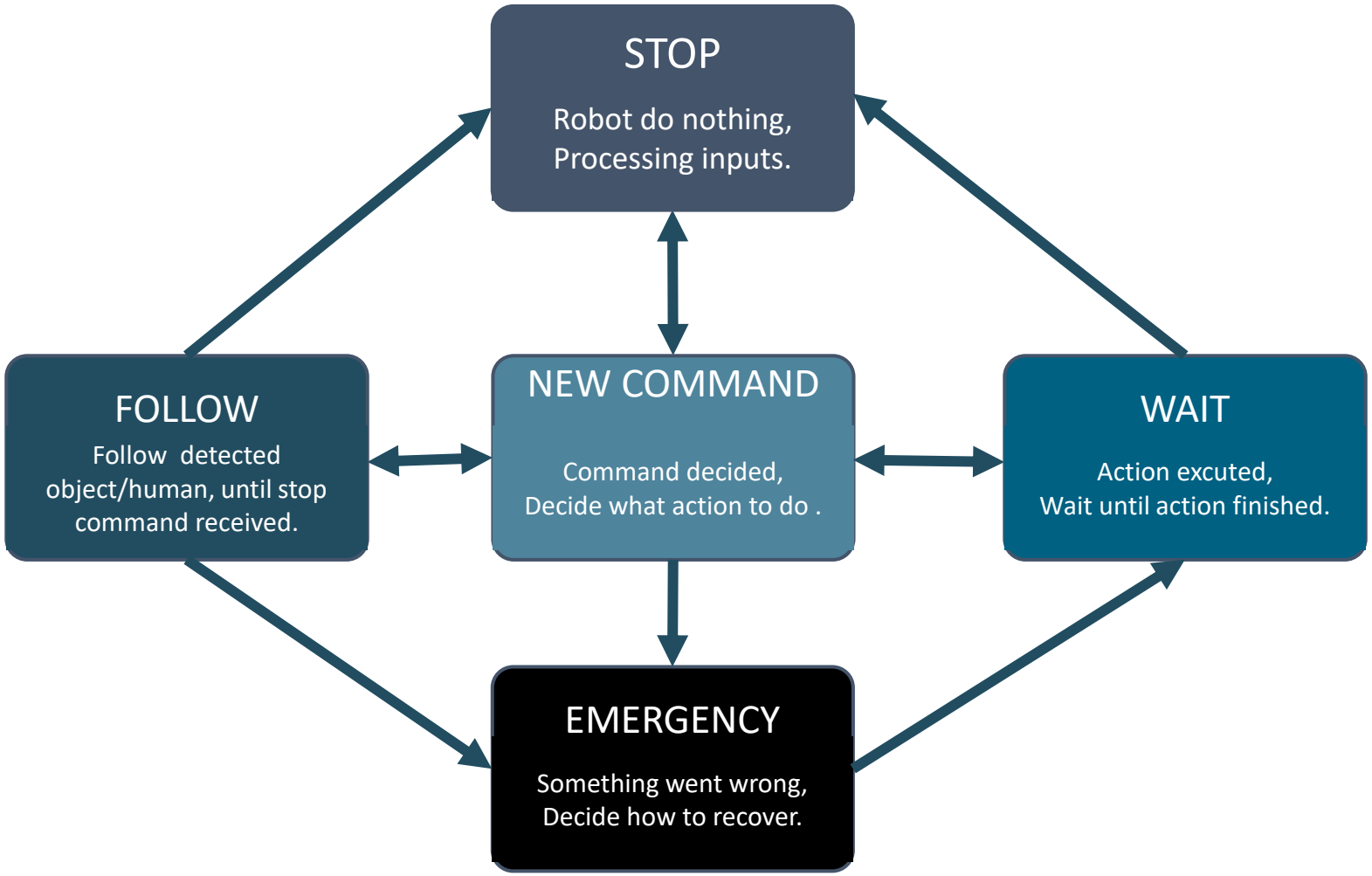


Part 5-I, ii

Node Design

```
def fsm(self):
    # implement FSM
    if self.state == State.STOP or self.state == State.COMMAND:
        if self.state == State.STOP: #STOP
            self.set_target(0, self.current.angular.z)
        if self.command == 0:
            self.state = State.STOP
        elif self.command == 1:
            self.state = State.FOLLOW
            self.set_target(x=0.1)
        elif self.command == 2:
            self.state = State.FOLLOW
            self.set_target(x=0.8)
        elif self.command >= 2:
            if self.command == 3:
                self.action_spin()
            ########## keyop #################
            elif self.command == 10:
                self.action_foward()
            elif self.command == 11:
                self.action_turn_left()
            elif self.command == 12:
                self.action_backward()
            elif self.command == 13:
                self.action_turn_right()
            ########## keyop end ##############
            self.state = State.WAIT
    elif self.state == State.EMERGENCY: #EMERGENCY
        if self.check_timer():
            self.set_target(x=0)
            self.set_timer(2)
            self.state = State.WAIT
    elif self.state == State.WAIT: #WAIT
        if self.check_timer():
            self.state = State.COMMAND
    elif self.state == State.FOLLOW:
        if self.command != 1:
            self.state = State.COMMAND
```

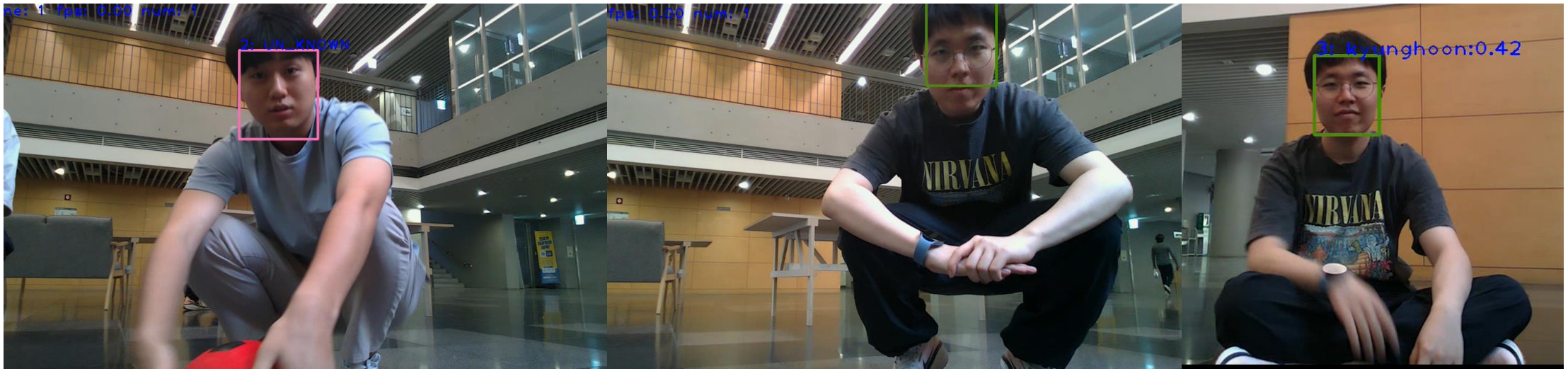
For reliable priority control and control,
Pet's brain has an FSM structure!



Part 5-iii~v

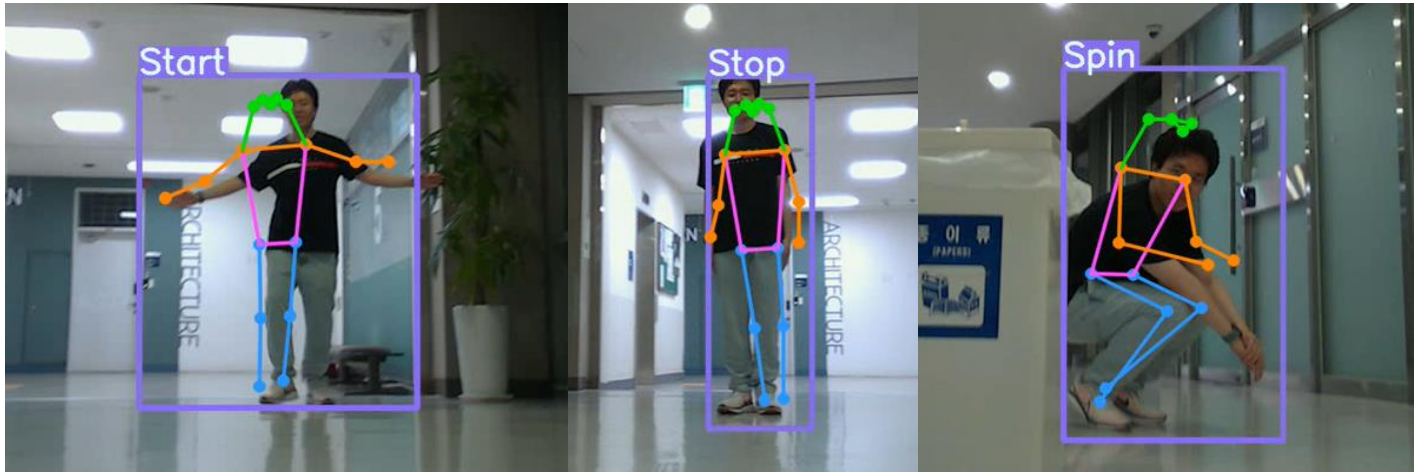
Pose Recognition/Face Recognition/3D Ball Detection

‘24.05



UN_KNOWN

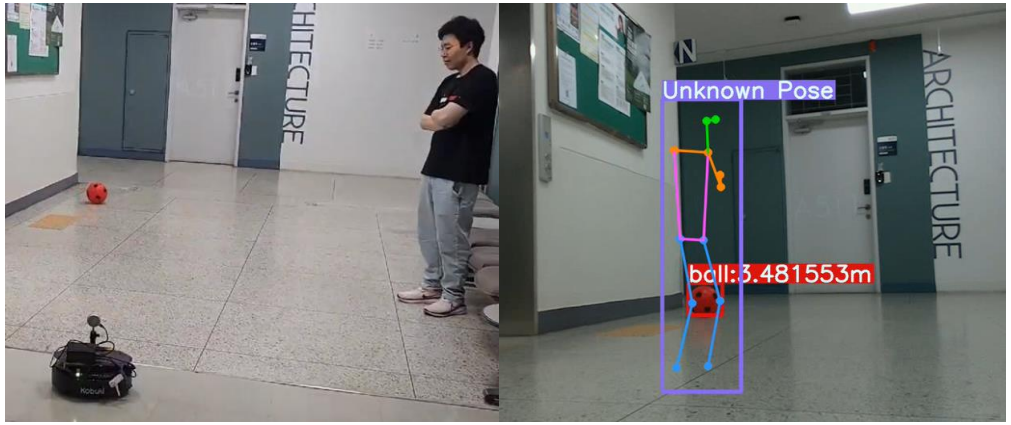
Owner(Kyunghoon)



Start : Arm Stretch

Stop : Attention Pose

Spin : Sit



3D Ball Detection

6

Results/Conclusions

Results1

Scene1 : Face Recognition -> Pose Recognition(Start/Stop)

Cam1 (Robot ver)

Cam2 (Global ver)

Results2

Scene2 : Face Recognition -> Pose Recognition(Start/Stop/ *Spin) -> Bump and Change Direction -> Ball Tracking

*Add This Week

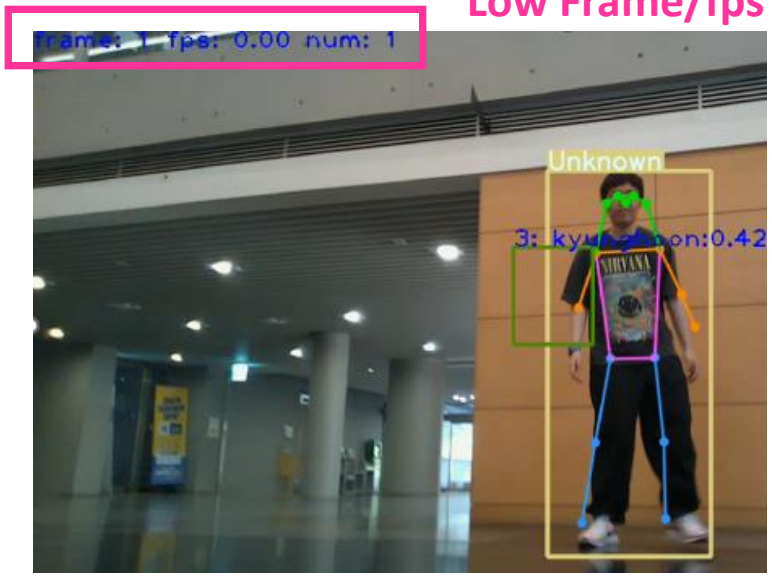
Face Recognition -> Pose Recognition -> Bump and Change Direction

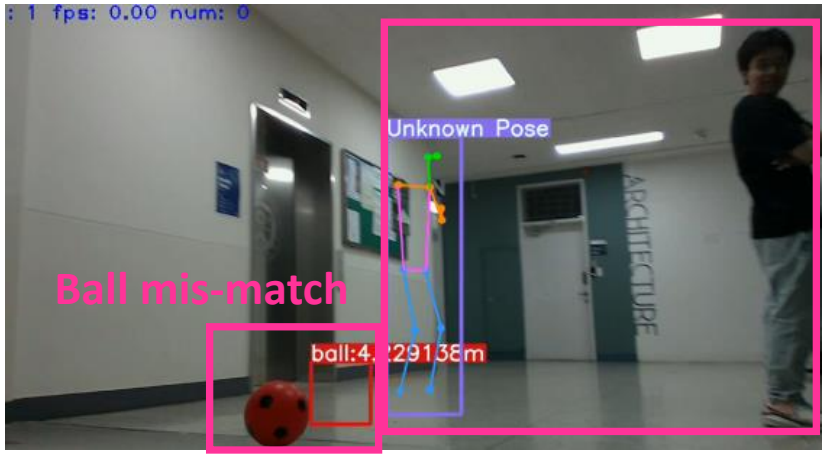
-> Ball Tracking

Conclusions1 : Why Robot Seems to be Not Realistic?

Problem1 : Robot Processor(Jetson)

Low Frame/fps 0 ~1



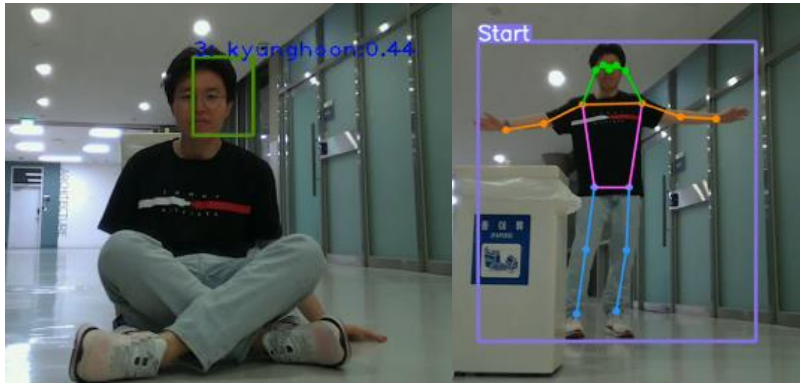


Human mis-match

Ball mis-match

Can not perceive
Abrupt Change

Problem2 : Camera’s Angle is limited

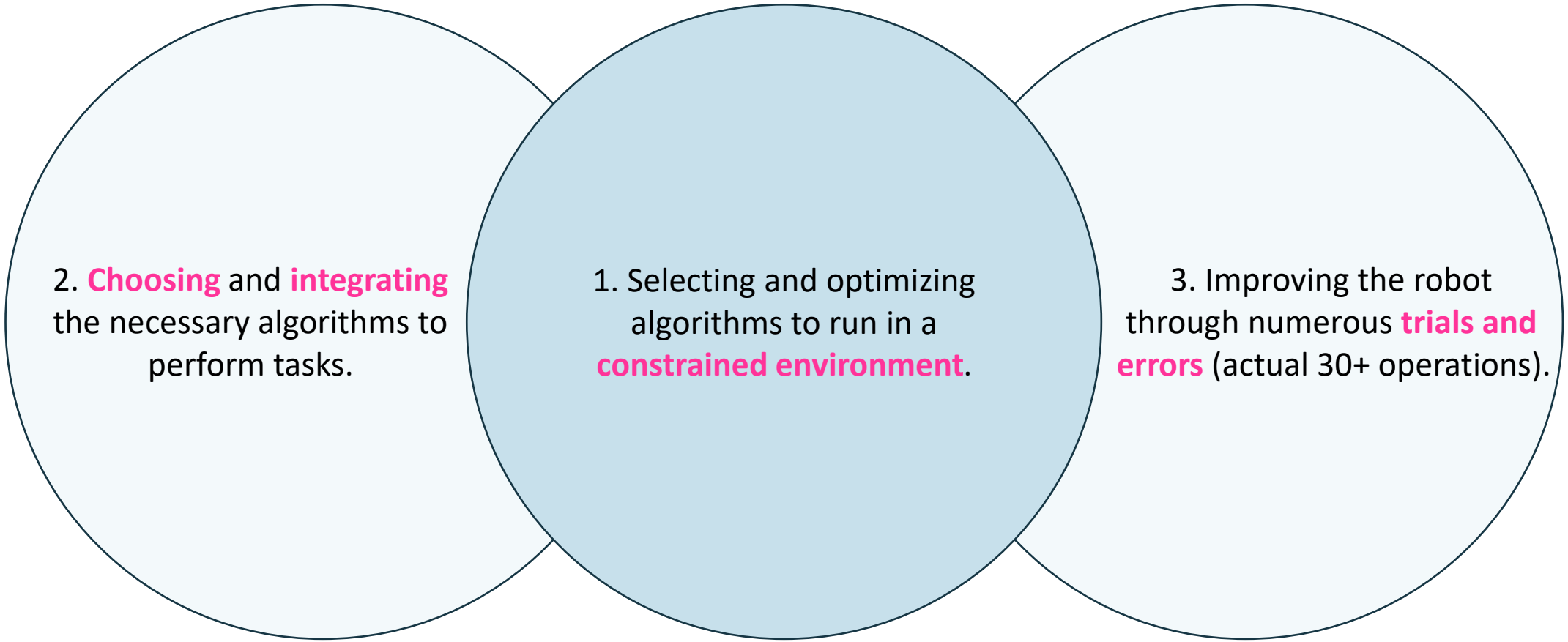


For Face Recognization(Real-time), Camera towards to up
For Ball Detection, Camera towards to down

→ Could only unlock robot from right in front of it

Problem3 : Robots should be generalists
as well as specialists?

Conclusions2 : What we have learned



Thank You
감사합니다