

AI SMART BULLDOZER

PODO



A2 POSCO 청년 Ai Big Data 아카데미 19기
김정원 김주연 이무동 이우철 인바다

CONTENTS

- 01 추진 배경
- 02 개발환경
- 03 적용 기술 및 기능
- 04 프로세스
- 05 시연영상
- 06 결론



1. 추진배경



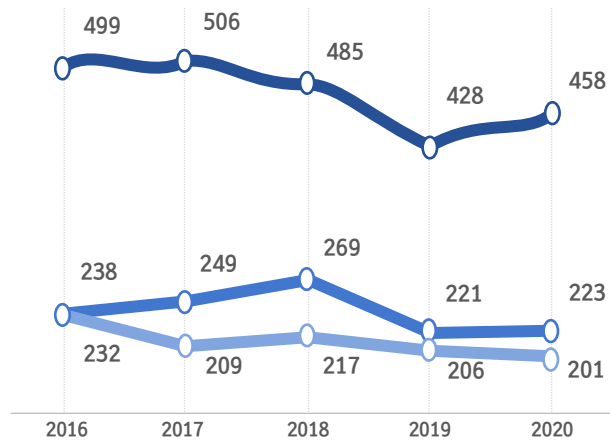
1. 추진배경

안전사고 증가와 그에 따른 경제적 손실 증가

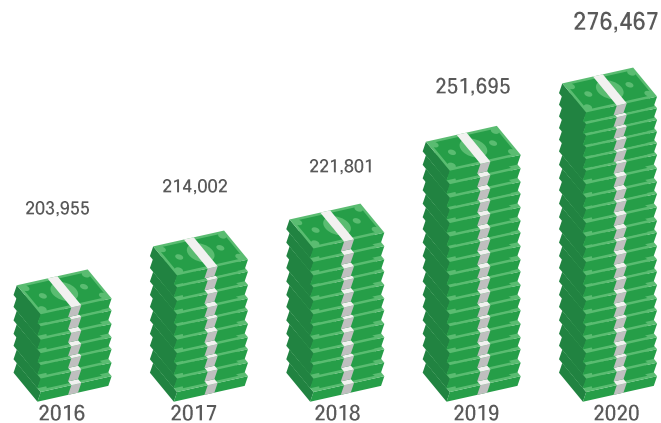
업종별 산재 사고사망 현황

단위 : 명

■ 건설업 ■ 기타산업 ■ 제조업



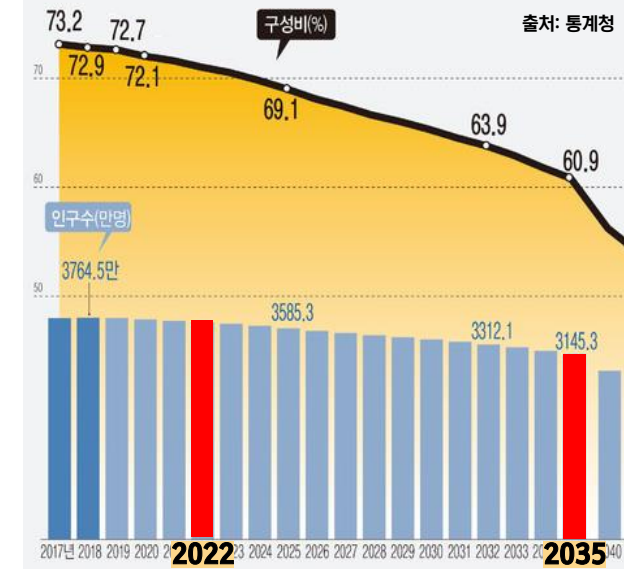
산업재해로 인한 경제적 손실액



생산가능인구 감소 ▶ 노동력 감소

생산가능인구(15~64세) 구성비 전망

출처: 통계청



1. 추진배경 _ 목표

AI smart 불도저

PODO



운송 무인화



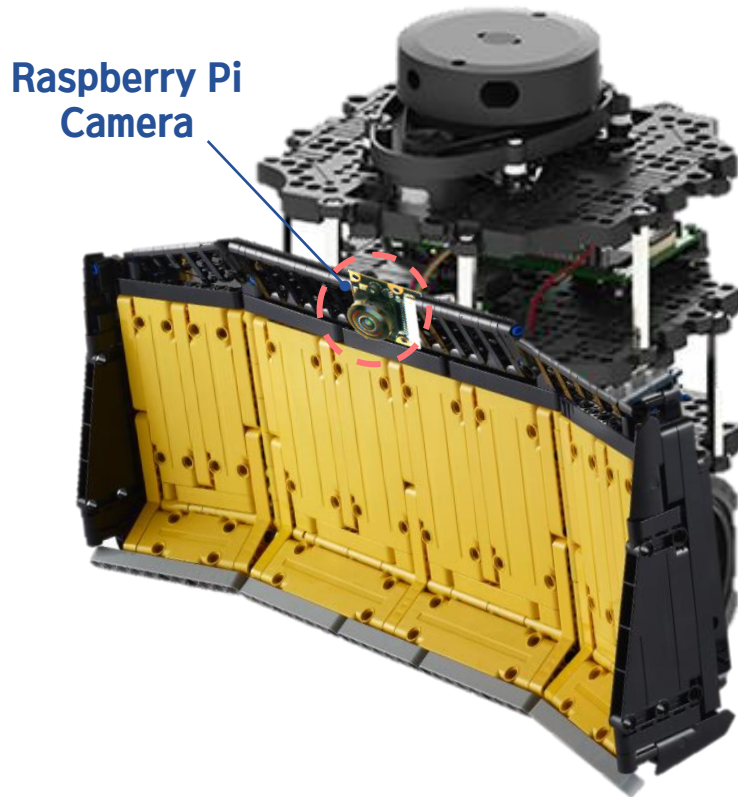
작업 효율성↑
노동자 안전성↑

2. 개발환경



2. 개발환경

Hardware Turtlebot3 + Bucket



LIDAR

Raspberry Pi 4

Open CR

Battery

Software

REMOTE PC OS

- ROS 2 foxy
- Linux
- Ubuntu 20.04



Robot OS

- ROS 2 foxy
- Linux
- Ubuntu 20.04



Language & Embedded

- Python
- Raspberry Pi 4



Computer Vision




- Open CV
- PyTorch
- YOLOv5
- nVIDIA



3. 적용 기술 및 기능

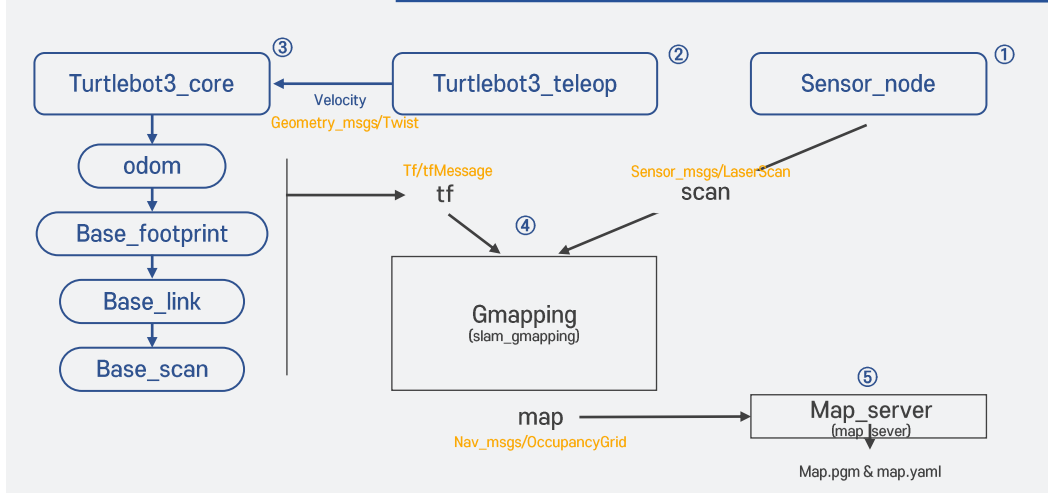


3. 적용 기술 및 기능 _ 개요

Robot Operating	Autonomous Driving		Object Detection	
ROS	SLAM	NAVIGATION	YOLOv5	OpenCV
<ul style="list-style-type: none"> - 로봇 응용 프로그램 개발 지원을 위한 소프트웨어 플랫폼 - Ubuntu : 20.04 LTS - ROS : Foxy 	<ul style="list-style-type: none"> - Simulation Localization and Mapping - 자율 주행 시 사용 - 주변 환경 지도를 작성하는 동시에 PODO의 위치를 인식 	<ul style="list-style-type: none"> - Mapping된 지도에서 로봇의 위치와 이동 방향 파악 - 목표 위치까지 경로 출력 - 장애물 회피 후 자율주행 실시 	<ul style="list-style-type: none"> - You Only Look Once - 단 한번의 객체 탐지 - 통합된 모델 사용 - 실시간 객체 탐지 	<ul style="list-style-type: none"> - Open Source Computer Vision - 실시간 이미지 및 영상 프로세싱 라이브러리 

3. 적용 기술 및 기능 _ Autonomous Driving : SLAM

ROS에서 SLAM 작동 방법



SLAM Mapping

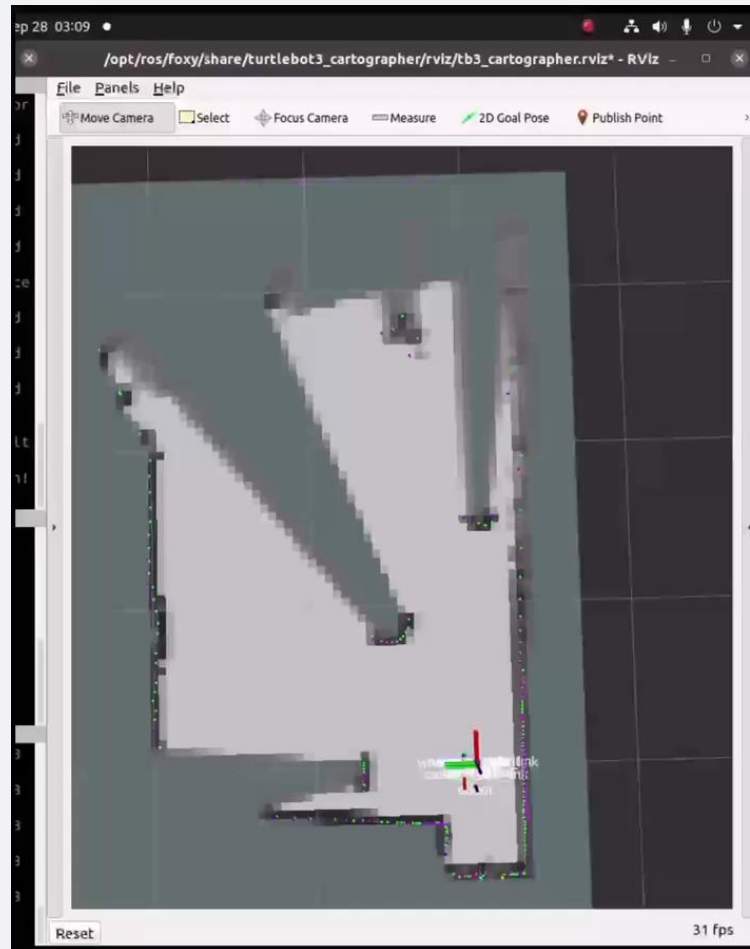


LIDAR SLAM

- 레이저 sensor로 획득한 데이터로 거리 값 추출.
- 레이저 센서 포인트 클라우드 데이터를 사용하여 고정밀 거리 측정 가능.
- 자율주행에는 3차원 라이다 포인트 클라우드를 이용하는 SLAM이 사용됨.

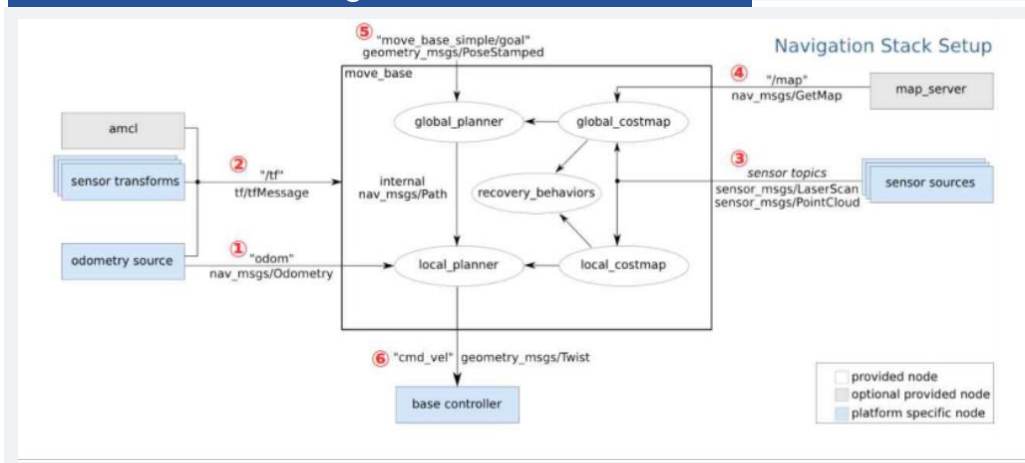
3. 적용 기술 및 기능 _ Autonomous Driving : SLAM

Mapping 시연 영상

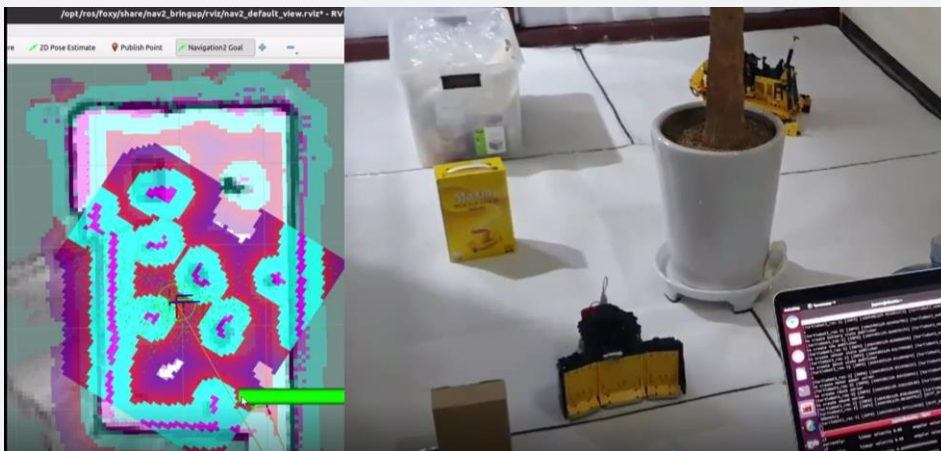


3. 적용 기술 및 기능 _ Autonomous Driving : Navigation

ROS에서 Navigation의 작동 방법



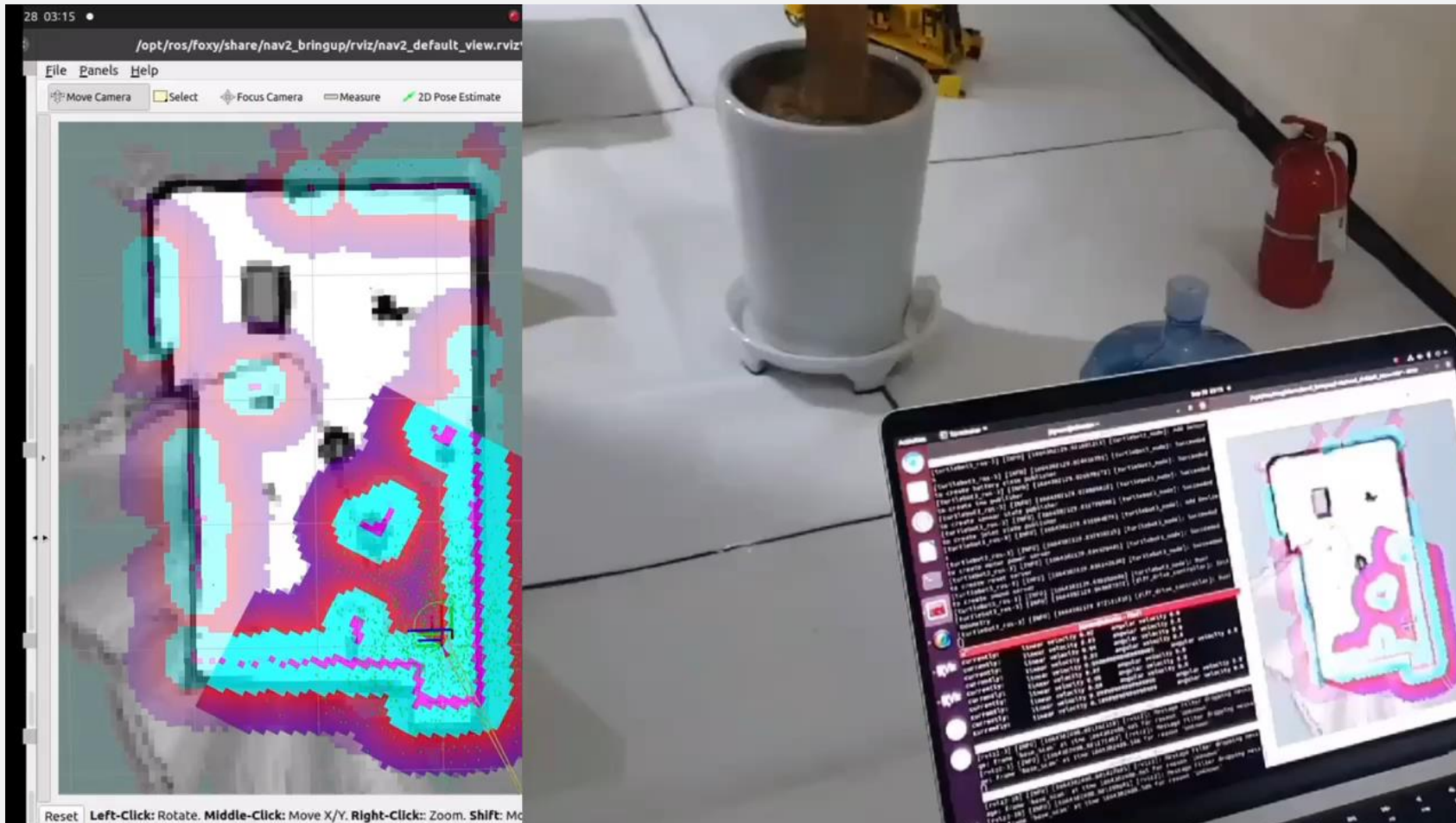
Navigation Working



- Mapping된 지도에서 로봇의 위치와 이동 방향 파악.
- 목표 위치까지 이동 궤적(Motion Planning) 생성.
- 모션 계획에서 작성된 이동 궤적을 따라서 목적지까지 이동, 또한 갑자기 나타난 장애물 회피하며 이동.

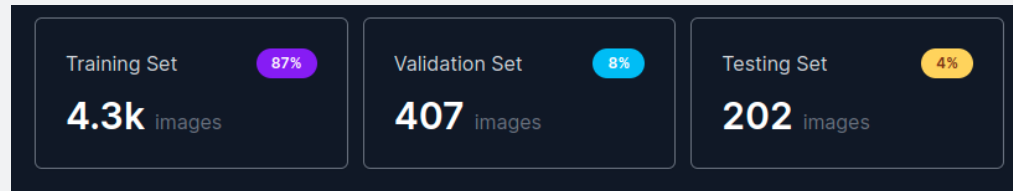
3. 적용 기술 및 기능 _ Autonomous Driving : Navigation

Navigation 시연 영상

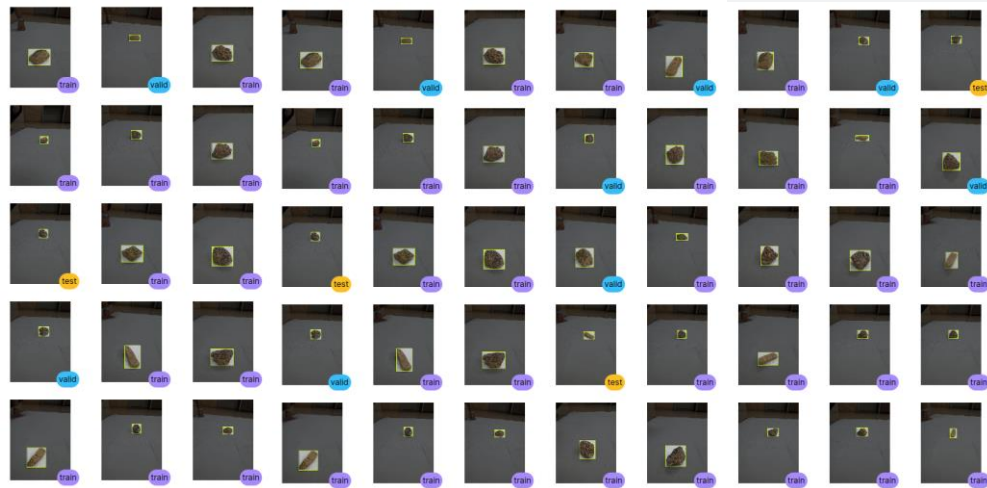


3. 적용 기술 및 기능 _ Object Detection

대용량의 rock 데이터



이미지 라벨링



YOLO 인식 정확도 확인

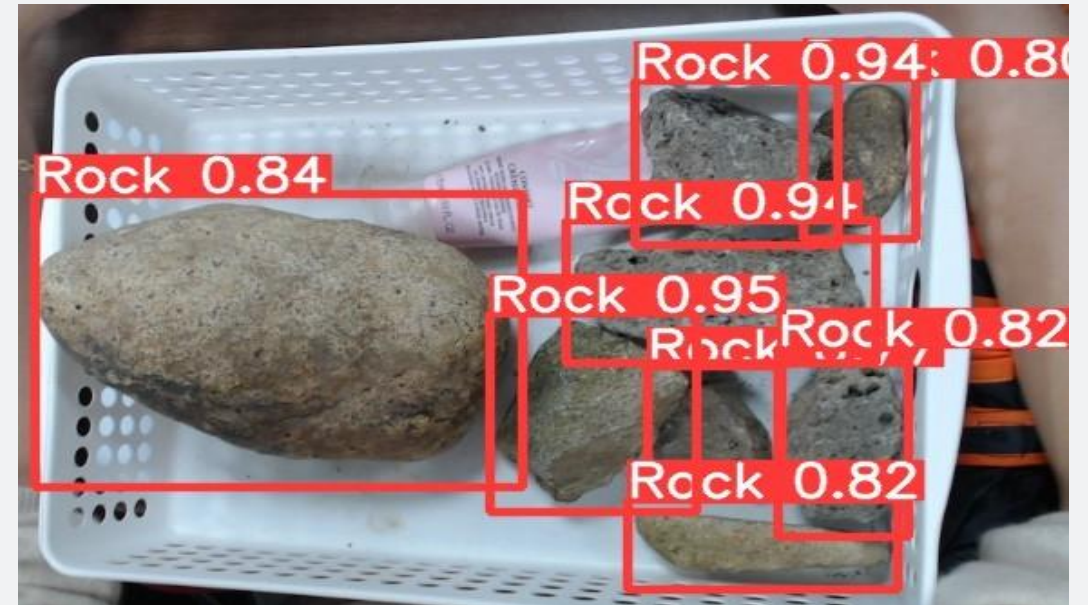
```
(cv.edu) pia@pia-Precision-7920-Tower:~/test/cv/yolov5$ python val.py --data /home/pia/test/cv/yolov5/final_data/data.yaml --weights ./runs/train/exp36/weights/best.pt --imgsz 416
val: data=/home/pia/test/cv/yolov5/final_data/data.yaml, weights=[./runs/train/exp36/weights/best.pt], batch_size=32, imgsz=416, conf_thres=0.001, iou_thres=0.6, task=val, device=, workers=8, single_cls=False, augment=False, verbose=False, save_txt=False, save_hybrid=False, save_conf=False, save_json=False, project=runs/val, name=exp, exist_ok=False, half=False, dnn=False
YOLOv5 v6.1-258-g1156a32 Python-3.8.13 torch-1.10.2 CUDA:0 (NVIDIA GeForce RTX 2080, 7982MiB)

Fusing layers...
YOLOv5m summary: 290 layers, 20852934 parameters, 0 gradients, 47.9 GFLOPs
val: Scanning '/home/pia/test/cv/yolov5/final_data/valid/labels.cache' images and labels... 805 f

```

Class	Images	Labels	P	R	mAP0.5	mAP0.5:95
all	805	2454	0.973	0.984	0.989	0.82

... Speed: 0.1ms pre-process, 3.6ms inference, 1.7ms NMS per image at shape (32, 3, 416, 416)



3. 적용 기술 및 기능 _ Object Detection

객체 인식 결과 영상

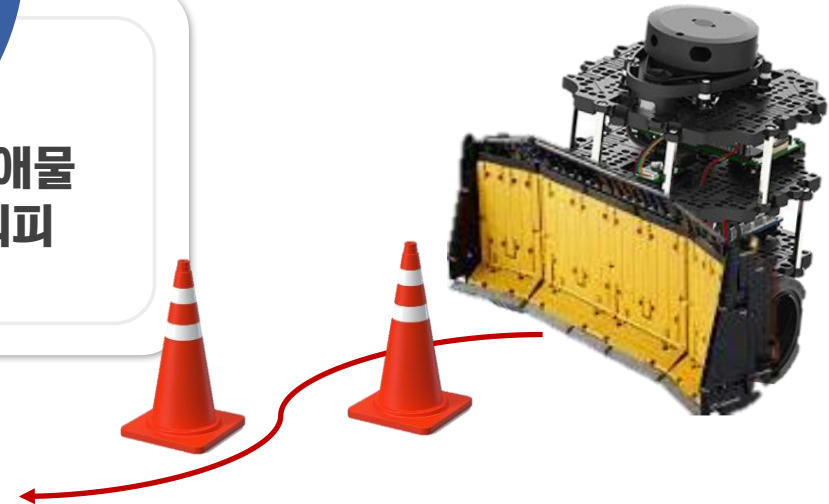
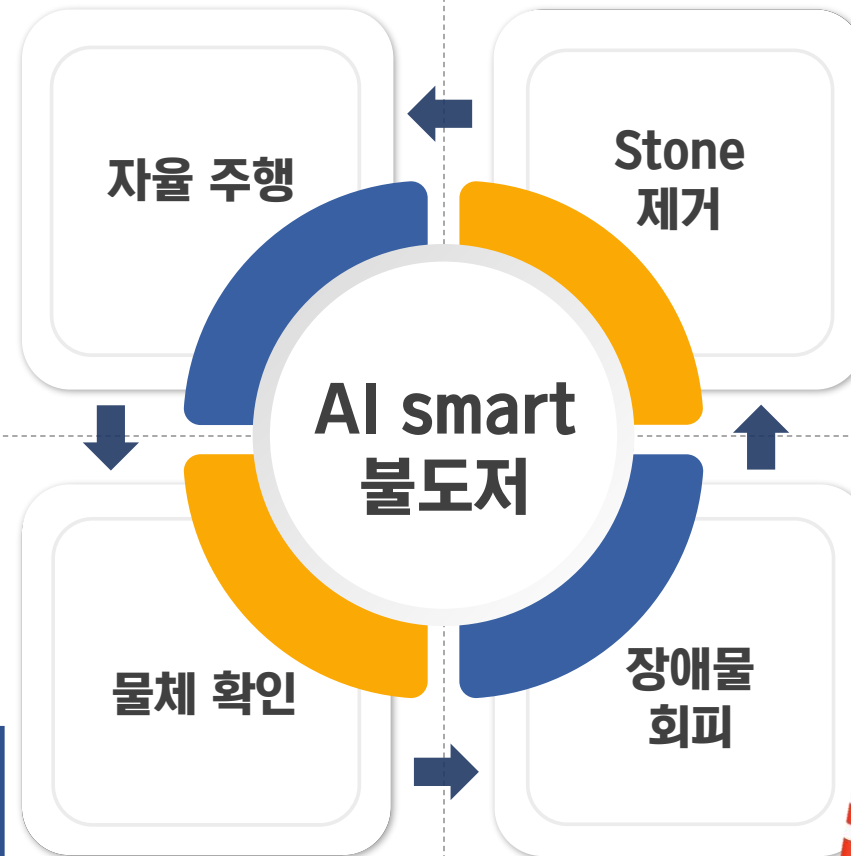
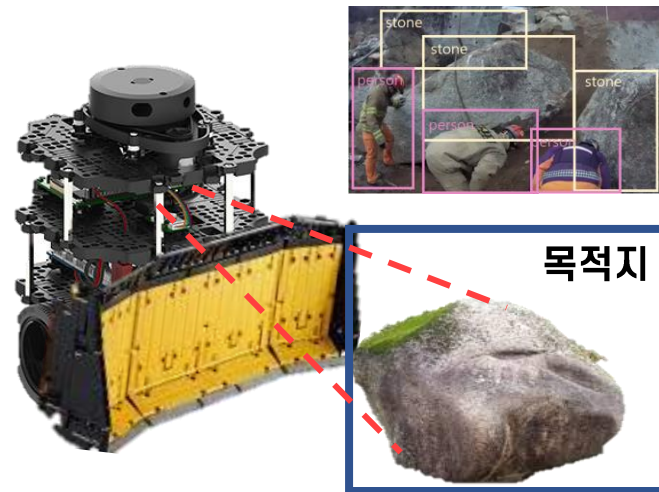
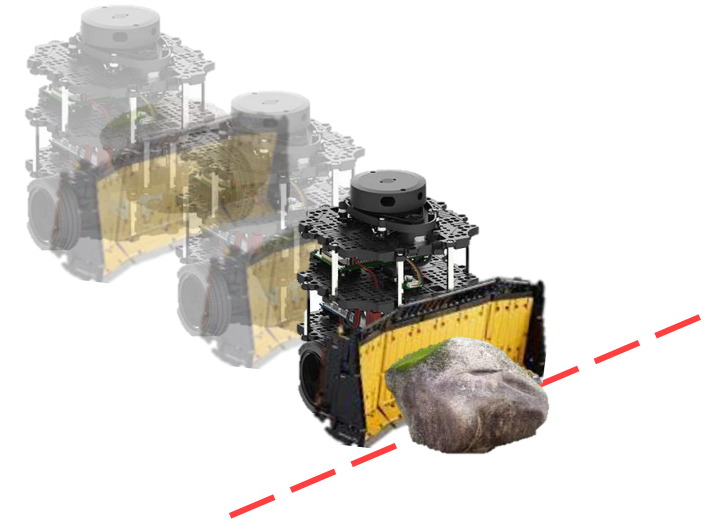
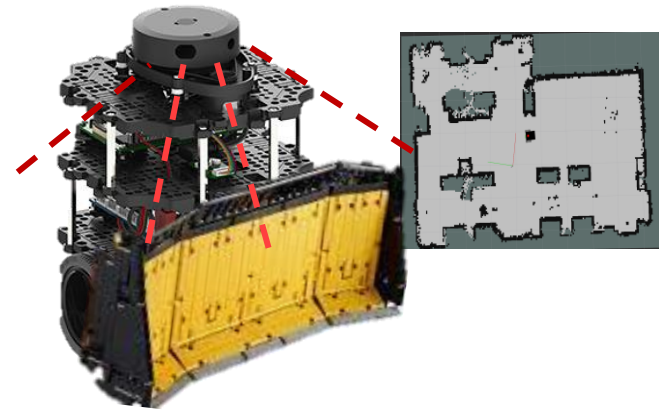
2.5배속된 영상



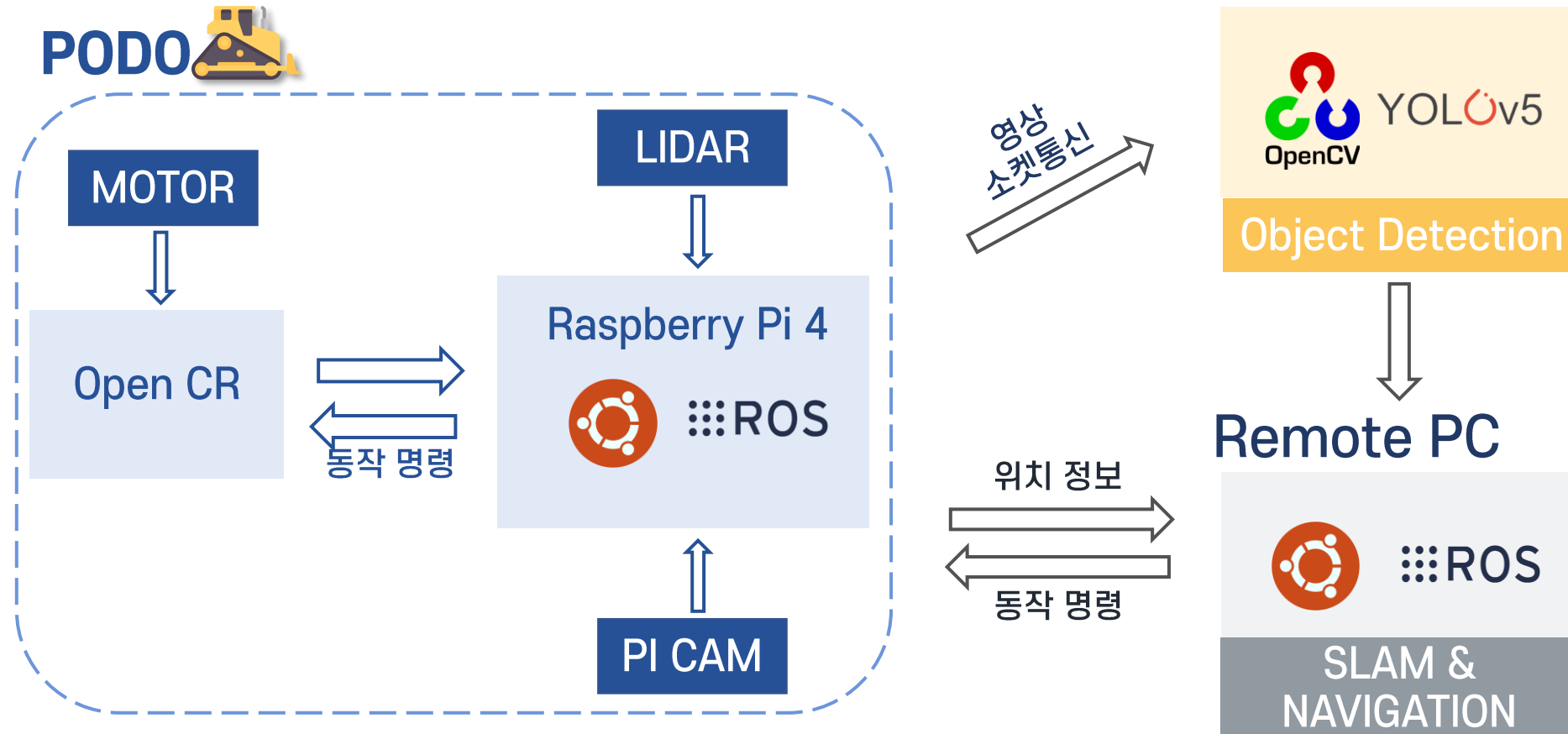
4. 프로세스



4. 프로세스 _ 주행



4. 프로세스 _ 통신



5. 시연 영상



5. 시연 영상

2.5배속된 영상



6. 결론



6. 결론_한계점 및 개선 기회

한계점

- ✓ 소켓통신을 통한 Object detection시 Delay를 해결하지 못함
- ✓ 돌의 무게가 너무 무거울 때에 PODO의 속도가 느려지는 문제에 직면함
- ✓ PI CAM의 화질 개선 필요
- ✓ PI CAM의 사각지대 존재

개선 기회

- ✓ SERVER를 사용하거나 웹 스트리밍을 사용
- ✓ 보다 더 출력이 높은 TurtleBot용 모터 사용
- ✓ PI CAM을 Web CAM으로 변경
- ✓ 시야각이 넓은 CAM을 사용하거나 높은 곳에 CAM 설치
- ✓ Hardware 외관에 보호막을 설치하여 Turtlebot에 이물질이 들어감을 방지

감사합니다

A2