

2024년 한이음 ICT멘토링 프로젝트 결과보고서

프로젝트명	Development of Vision-Based Intelligent Driving Lane Recognition and Forward Vehicle Collision Avoidance Technology
-------	---

요 약 본

프로젝트 정보	
프로젝트명	비전 기반 지능형 주행 차선 인식 및 전방 차량 충돌 회피 기술 개발
주제 영역	<input type="checkbox"/> 생활 <input type="checkbox"/> 업무 <input checked="" type="checkbox"/> 공공/교통 <input type="checkbox"/> 금융/핀테크 <input type="checkbox"/> 의료 <input type="checkbox"/> 교육 <input type="checkbox"/> 유통/쇼핑 <input type="checkbox"/> 엔터테인먼트
기술 분야	<input checked="" type="checkbox"/> SW-AI <input type="checkbox"/> 방송·콘텐츠 <input type="checkbox"/> 블록체인·융합 <input type="checkbox"/> 디바이스 <input type="checkbox"/> 차세대보안 <input type="checkbox"/> 미래통신·전파
달성 성과	<input type="checkbox"/> 논문게재 및 포스터 발표 <input type="checkbox"/> 앱등록 <input type="checkbox"/> 프로그램등록 <input type="checkbox"/> 특허 <input type="checkbox"/> 기술이전 <input type="checkbox"/> 실용화 <input type="checkbox"/> 공모전() <input checked="" type="checkbox"/> 기타(프로젝트 완성)
프로젝트 소개	기존 ADAS는 차선이탈 경보, 추돌경보, 교통신호 알람, 보행자 인식 등의 다양한 기능을 여러 센서와 카메라를 이용하여 구현 중이다. 해당 프로젝트에서는 센서를 사용하지 않고, 비전 기반의 여러 영상 처리 기법을 통해, ADAS 기술을 발전시킨다.
개발 배경 및 필요성	ADAS 기술은 운전자의 안전과 편의성 측면에서 중요한 요소로 부상하고 있다. 날씨 정보를 고려하며, 더욱 지능화된 비전 기반 ADAS 기술은, 운전자의 안전뿐만 아니라 자율주행 기술의 발달, 사회 전반적으로 경제적인 이익을 가져와 줄 것이다.
프로젝트 특징	· 주행차선인식 : Canny Edge Detection 알고리즘을 통해 차선을 인식하고 이탈 방지 · 전방객체충돌회피 : 전방에 차량, 사람 등이 인식되면 속도를 줄이거나 멈춤 · 도로교통표지판인식 : 객체 탐지를 이용해, 교통 표지판을 인식하여 안전 운전
주요 기능	· 객체검출 : 차량 인식 / 보행자 인식 / 교통 표지판 인식 · 차선인식 : 라인 검출, 허프 변환, 소실점 계산 / 딥러닝 기반 차선 인식 · 거리측정-충돌경고알림 : 객체와의 거리 측정 및 추돌 경고
기대효과 및 활용 분야	· 더욱 지능화된, 비전 기반 ADAS 기술 기대 · 효율적인 운전으로 인한, 연비 향상 및 운전자 스트레스 감소 · 자동차 운송 산업, 항공, 철도, 선박 등에 활용 및 스마트시티 구현

(본문) 프로젝트 결과보고서

I. 프로젝트 개요

1. 프로젝트 소개

- 첨단 운전자 보조 시스템 (ADAS)은 자율주행 자동차의 기반이 되는 기술로 운전자가 운전 중에 발생할 수 있는 위험을 센서나 카메라를 통해 감지해 운전자가 대처할 수 있도록 도와주는 안전장치의 일종이다.
- 날씨 변화에 따라 교통사고의 치사율과 빈도가 많은 차이가 일어난다는 것을 고려해 날씨 정보를 수집 후, 실제로 구현하고자 하는 ADAS 기술에 여러 가지 변수를 추가해보고자 한다.
- ADAS 기술의 주요 기능으로는 전방 충돌 경고 기능, 차로 이탈 경고, 속도 제한 등의 여러 가지 기능들이 있는데, 라이다, 초음파 센서 등을 사용하지 않고, 이중 딥러닝 비전 영상 처리를 기반으로, YOLO 객체 인식 등의 기술을 이용하여 전방 객체 및 차선 인식, 거리 측정 등의 기술을, 수집한 날씨 등의 정보를 고려하여 구현해보려 한다.

2. 개발 배경 및 필요성

- ADAS 기술은 운전자의 안전과 편의성 측면에서 중요한 요소로 부상하고 있다. 통계상 교통사고로 인해 약 25초에 1명씩 사람이 사망하고 있는데, ADAS 시스템을 통해 운전자를 보조하여 운전자의 집중력을 올려줄 수 있고 그에 따른 부가 효과로 운전자의 안전뿐만 아니라 자율주행 기술의 발달 및 사회 전반적으로 경제적인 이익을 가져와 줄 것으로 예상된다.
- 통계적으로 우천, 안개 및 눈 등의 기상 상태에 따라 맑은 날 대비 교통사고의 사망률 등이 더 높게 나타나는 경향이 있는데, 이를 방지하기 위해 날씨 정보를 고려하여 ADAS 시스템에서 여러 가지 변수를 추가하는 것이 향후 자율주행 기술에 필요한 기술 및 기능이다.

- 현재 위치에 따른 날씨 정보를 수집하여 카메라를 통해 입력되는 영상을 날씨 정보와 함께 고려하며, 이를 통해 운전자가 해당 정보를 바탕으로 운전하여 발생할 수 있는 여러 상황에 대해 인지하게 되고, 이를 통해 사고 확률을 낮출 수 있게 된다.

3. 프로젝트 특·장점

1) 주행 차선 인식

- Canny Edge Detection, Hough Transform을 이용하여 차선을 인식하고, 해당 차선을 이탈하지 않도록 주행을 도움을 준다.

2) 전방 차량 충돌 회피

- 차량 전방에 차량, 사람 등 물체가 인식되면 속도를 줄이거나 멈추어 충돌을 방지한다.

3) 도로 교통 표지판 인식

- 객체 탐지를 이용하여 도로상의 정지, 속도 제한 표지판 등의 교통 표지판을 인식하여 스스로 속도를 조절하거나 정지한다.

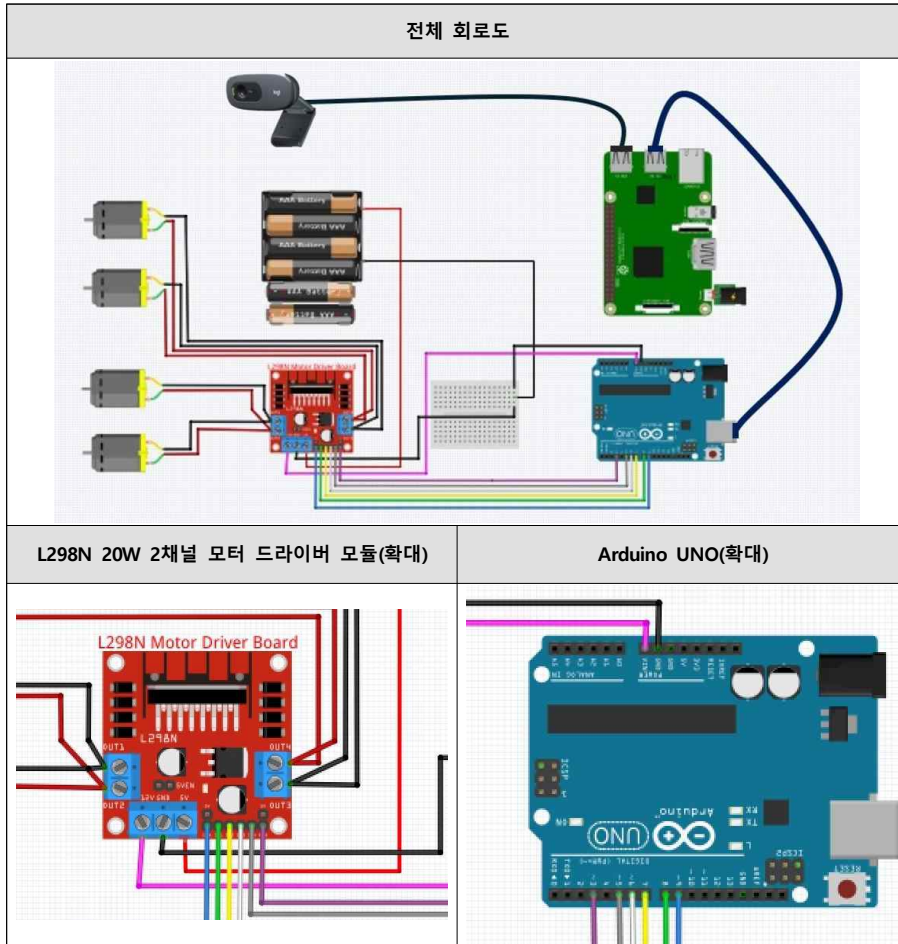
- 기존 프로젝트에서는 차선 인식만을 목표로 하였지만, 이 프로젝트에서는 YOLO 객체 인식을 이용한 객체 탐지 기능을 추가하여, 차선을 따라 주행하며 전방의 물체를 감지하는 등의 더 진보된 형태의 자율주행을 선보인다.

- 기존 자율주행 기술에서, 라이다와 초음파 센서, 라인 트래이서 센서 등을 사용하는 것과 다르게, 오로지 카메라를 통해 들어오는 영상의 비전 기반 정보를 통해, 전방 차선-객체를 탐지하고, 상황에 맞는 동작을 하여, 자율주행을 수행한다.

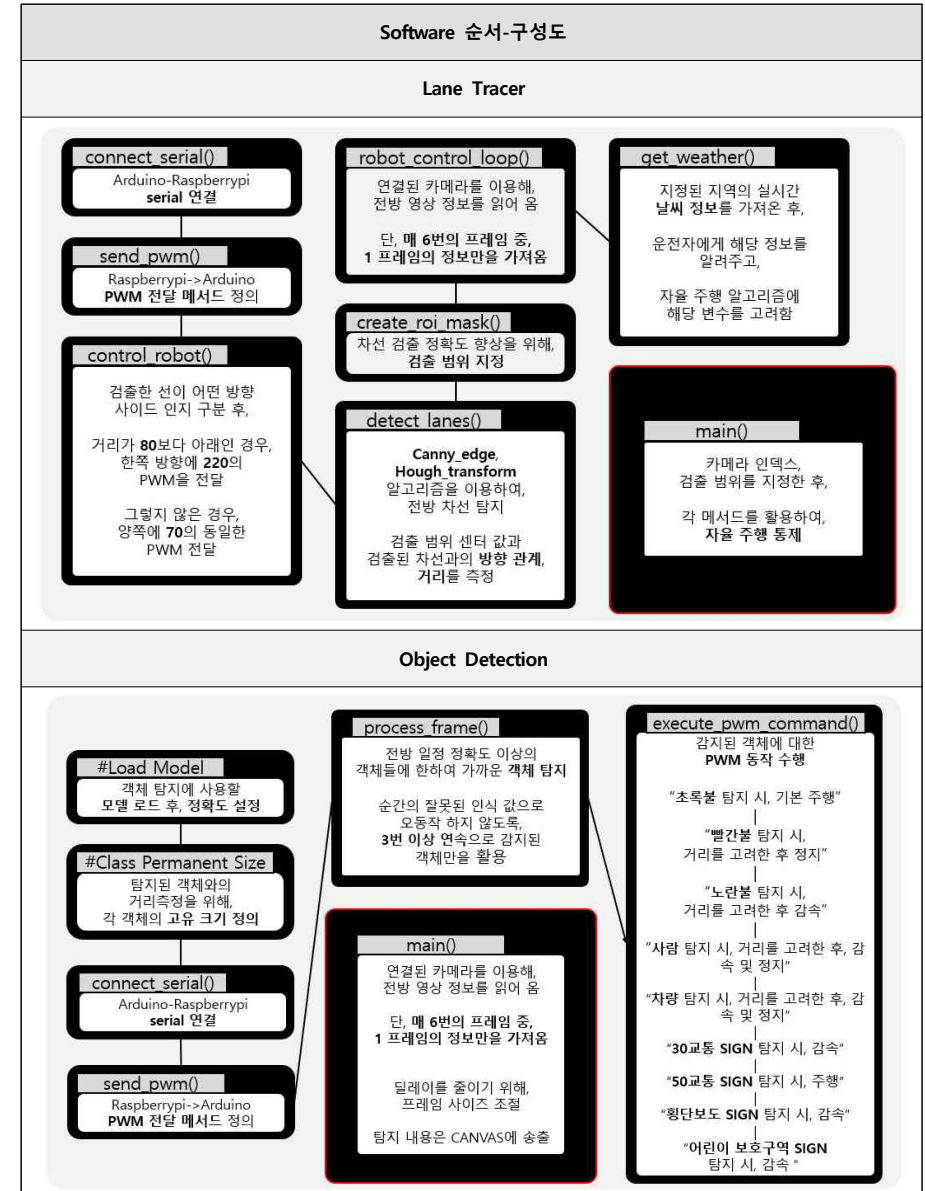
II. 프로젝트 내용

1. 프로젝트 구성도

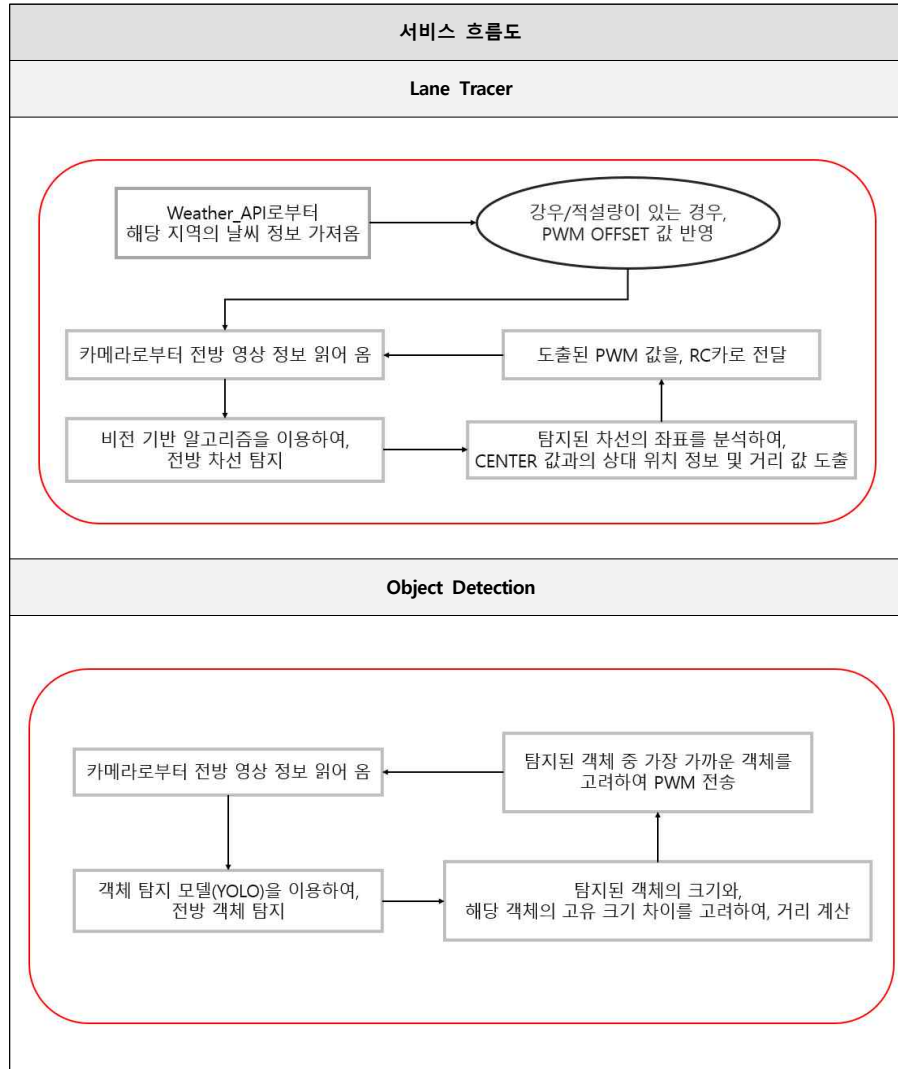
1) H/W 구성도



2) S/W 구성도



3) 서비스 흐름도

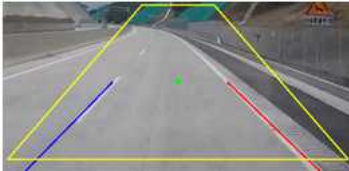



2. 프로젝트 기능

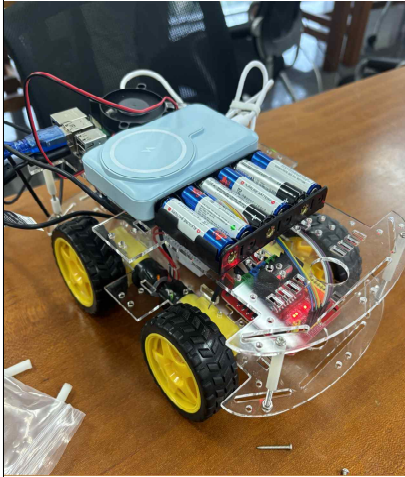
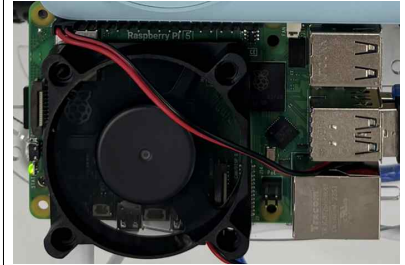


1) 전체 기능 목록

구분	기능	설명
S/W	영상 처리	Canny Edge or object Detection, Hough Transform 등 기본적인 영상 처리 이론(차선, 사람, 차, 표지판 등 탐지) -차선: LDWS 알고리즘을 이용한 차선 이탈 방지 -차량: FCWS 알고리즘을 이용한 전방 추돌 방지
	YOLO AI Engine 및 OpenCV 활용	-사람: 주행 시 전방 보행자 탐지하여 추돌 방지 -표지판: 속도 표지판, 어린이 보호구역, 보행자 표지판 등 탐지하여, 상황에 맞는 동작 요구
	날씨 정보 크롤링	해당 지역의 날씨 정보를 실시간으로 받아와서, 운전자에게 안내
H/W 및 S/W	차선 인식 및 이탈 여부 확인	직선 및 곡선 차선 인식 및 차량의 차선이탈 여부 확인. 카메라를 이용한 전방 장애물과의 거리 계산으로, 회피 시나리오 개발 (OpenCV + 카메라기반)
H/W	RC카 2채널 PWM 제어	Raspberry Pi에서 주는 PWM을 Arduino에서 처리하여, RC카를 알맞게 움직이도록 제어
	Raspberry Pi 발열 제어	기존 15분 정도밖에 버티지 못하던 Raspberry Pi의 기능 가능 시간을, FAN을 추가하여 수명 연장
	전방 영상 정보 읽기	카메라를 활용하여, 전방의 영상 소스를 Raspberry Pi로 전달

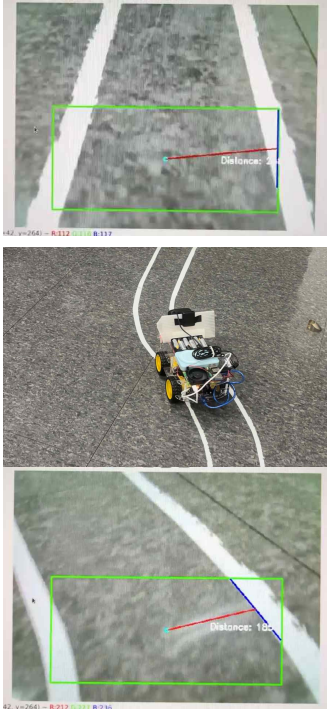

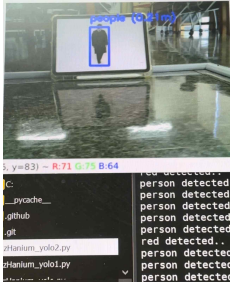
2) S/W 주요 기능

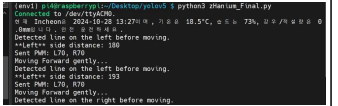
기능	설명	프로젝트 실물 사진
차선 인식 알고리즘	Canny Edge detection, Hough Transform을 이용하여 전방 차선을 감지. 차선과 주행 방향의 차이를 탐지하는 알고리즘을 이용해 차선을 따라 주행	
객체 탐지 및 거리 측정 알고리즘	YOLOv5 Engine을 이용하여 차량과 사람, 표지판, 신호등을 탐지하고, 각 객체의 고유 특성을 고려하여 거리 측정 후 RC카 PWM 제어 ex. 신호등을 탐지한 경우, 빨간불을 인식할 시, 정지하고, 초록불을 인식 시에 주행 표지판을 탐지하는 경우에는, 속도 표지판, 혹은 동작 표지판 등을 구별하여 그에 맞는 동작을 수행 (*오른쪽 상단 사진은, 사람을 인식한 후에, 해당 객체까지의 거리를 측정한 모습)	 
날씨 정보 크롤링	날씨 정보 웹사이트로부터 실시간으로 정보를 받아, 해당 지역의 날씨를 분석하여, 운전자에게 알맞은 조언을 수행	<pre>(lenv) python3 ~/Desktop/yolo5_5/python3/planum/final.py Connected to /dev/ttyACM0. ===== 2024-10-28 13:27:00, time 18.5°C, air 73%, 0.0 m/s ===== [0m0.0] ... 0.0 m/s Detected line on the left before moving. *Left* side distance: 180 Sent PWM: 170, 500 Moving Forward gently... Detected line on the left before moving. *Left* side distance: 193 Sent PWM: 170, 500 Moving Forward gently... Detected line on the right before moving.</pre>

3) H/W 주요 기능

기능/부품	설명	프로젝트 실물 사진
RC카 본체	Arduino에서 전송하는 PWM으로 바퀴(모터) 동작	
Raspberry Pi5 + FAN	카메라로부터 실시간 영상 정보를 받아, 차선 탐지 분석, 객체 탐지 분석 후, 알맞은 PWM을 Arduino로 전송 + Raspberry Pi 발열로 인한, 성능 저하를 방지하기 위해, 3V FAN 추가 설치	
Arduino UNO	바퀴(모터)로 전송할 수 있는 PWM을 사전에 정의 Raspberry Pi로부터 받은 PWM을 미리 배정한 핀으로 전송	
Logitech C270	실시간 전방 영상 정보를 받아, Raspberry Pi로 전송	

3. 주요 적용 기술

기술	설명	기술 사진
차선 검출 및 차선이탈 방지 (영상 전처리)	-RoI(Region of Interest) 마스크 생성 ▶ 특정 관심 영역만 처리하기 위함	
	-HSV 색상 공간 변환 ▶ 조명 변화에 상대적으로 민감한 RGB 대신, HSV 색상 공간을 적용	
	-이진화: 흰색과 노란색 마스크 생성 ▶ HSV 범위가 다른 두 색상 차선을 각각 검출하고, 비트 OR 연산으로 결합하여, 모든 차선 색상을 포함하는 마스크 생성	
	-가우시안 블러(Gaussian Blur): 노이즈 제거 ▶ 검출된 마스크에 가우시안 블러를 적용하여 작은 노이즈를 제거	
	-Canny Edge Detection: 경계선 검출 ▶ 블러링 된 이미지에서, Canny 알고리즘을 이용해 경계선을 검출	
전방 객체 탐지 및 충돌 방지	-Hough Transform: 직선 검출 ▶ 허프 변환으로 에지 사이의 연속된 직선 검출	
	-중심점과 직선과의 거리측정 후, PWM 제어 ▶ 중심점의 X좌표와, 검출된 직선의 X좌표를 비교하여, 어느 방향의 직선이 검출되었는지 확인. + 해당 방향 직선과의 픽셀 거리를 측정하여 거리가 가깝다면, 방향 제어 PWM 전송	
	YOLO 객체 탐지 엔진을 기반으로, '0_파란불', '1_빨간불', '2_노란불', '3_사람', '4_자동차', '5_30 Sign', '6_50 Sign', '7_Crosswalk Sign', '8_Childrenzone Sign' 탐지 + 각 객체의 고유 크기를 고려하여, 객체와 RC카 사이의 거리 측정 후, PWM 제어	

Weather API를 이용한 실시간 날씨 데이터 활용	기상 예보 사이트에서 실시간으로 날씨 정보를 받아와, 비나 눈이 오는 경우, PWM 값을 보정	
---	--	---

4. 프로젝트 개발 환경

구분		상세내용
S/W 개발환경	OS	Linux(Raspbian), Window
	개발환경(IDE)	PyCharm, VS(Visual Studio), IDLE(Python)
	개발도구	Raspberry Pi5, Arduino UNO
	개발언어	Python, C
	기타사항	X
H/W 구성장비	디바이스	Raspberry Pi5, Arduino UNO, Logitech C270
	센서	X
	통신	UART Serial Communication
	언어	Python, C
	기타사항	RC Car Kit
프로젝트 관리환경	형상관리	자택 보관, 학교 실습실 보관
	의사소통관리	메신저(Discord)를 통한 온라인 미팅, 매주 오프라인 개발 진행
	기타사항	매달 멘토-멘티 간 온라인 미팅

5. 장비(기자재/재료) 활용

번호	품명	작품에서의 주요기능
1	Raspberry Pi5	- 영상 데이터 처리 및 분석, 모터 제어, 외부 통신 등의 제어 허브
2	Arduino UNO	- 모터 구동을 위한, PWM 신호를 받아 차량을 제어하는 하드웨어 인터페이스
3	Logitech C270	- 프레임 단위로 실시간 영상을 수집하여, 영상 처리 알고리즘에 입력
4	Cooling Fan	- Raspberry Pi 성능 향상을 목적으로 부착
5	6량 건전지	- RC Car에 동력 공급
6	보조 배터리	- Raspberry Pi에 전력 공급
7	PVC 골판지	- 카메라 높이 조절을 위해 설치
8	흰색 테이프	- 도로 차선 구현

6. 프로젝트 작동 동영상

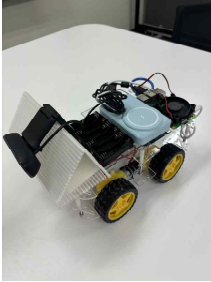
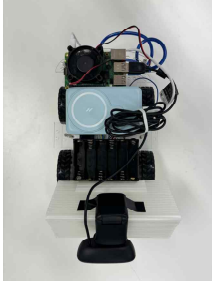
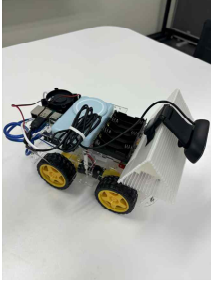
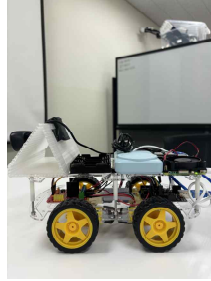
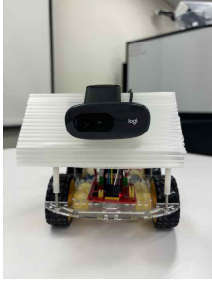
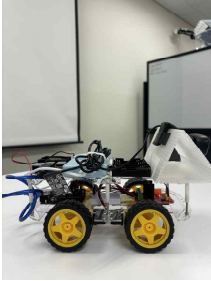
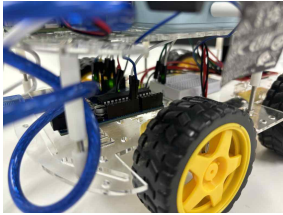

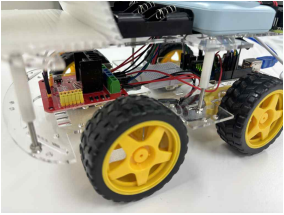
[1] 2024 한이음 비전 기반 지능형 주행 차선 인식 및 전방 차량 충돌 회피 기술 개발 (Line Detection)

>“https://www.youtube.com/watch?v=Ap_QmALgXLc”

[2] 2024 한이음 비전 기반 지능형 주행 차선 인식 및 전방 차량 충돌 회피 기술 개발 (Object Detection)

>“<https://www.youtube.com/watch?v=2-qXVjHOrG8>”

7. 결과물 상세 이미지

Left Front View	Top View	Right Front View
		
Left View	Front View	Right View
		
Inside 1	Rear View	Inside 2
		

8. 달성 성과

<input type="checkbox"/>	논문게재 및 포스터발표	게재(발표)자명	논문(포스터)명	게재(발표)처	게재(발표)일자
					2024. 00. 00.
<input type="checkbox"/>	앱(APP) 등록	등록자명	앱(APP)명	등록처	등록일자
					2024. 00. 00.
<input type="checkbox"/>	프로그램 등록	등록자명	프로그램명	등록처	등록일자
					2024. 00. 00.
<input type="checkbox"/>	특허/실용신안 출원	출원자명	특허/실용신안명	출원번호	출원일자
					2024. 00. 00.
<input type="checkbox"/>	기술이전	기술이전기업명	기술명	금액	이전일자
					2024. 00. 00.
<input type="checkbox"/>	공모전	구분(교내/대외)	공모전명	수상여부(출품/수상)	상격
<input type="checkbox"/>	실용화				
<input checked="" type="checkbox"/>	기타	기존 목표로 하였던 공모전에는 참가하지 못하였으나, 프로젝트 완성에는 성공			

III. 프로젝트 수행 내용

1. 업무 분담

번호	성명	역할	담당업무
1	김신형	멘 토	온라인 미팅 일정 조율, Q&A답변
2		지도교수	
3	임재혁	팀 장	Weather_API 기능 구현, Raspberry Pi 코딩
4	장민혁	팀 원2	차선 탐지 및 객체 탐지 알고리즘 구현, Raspberry Pi 코딩
5	조수연	팀 원3	차선 탐지 및 객체 탐지 알고리즘 구현, Arduino 코딩
6	진우열	팀 원4	차선 탐지 및 객체 탐지 알고리즘 구현, Arduino 코딩
공통업무			
YOLO 모델 DATASET 제작 및 학습, RC카 제작, 프로젝트 시나리오 구현			

2. 프로젝트 수행 일정

구분	추진내용	수행일정									
		3월	4월	5월	6월	7월	8월	9월	10월	11월	
계획	프로젝트 수행 계획										
분석	Canny Edge Detection 및 Hough Transform 알고리즘 분석										
	LDWS 이론 분석 및										
	카메라를 이용한 거리 계산 방식 분석										
설계	LDWS 이론 설계 및										
	YOLO 객체 인식 모델 설계										
	LDWS 성능 향상을 위한 세부 설계 및										
개발	카메라를 이용한 거리 계산 설계										
	예상 시나리오 구체화 및										
	LDWS 알고리즘 개발(OpenCV 영상처리)										
	RC카 Kit 조립 및 YOLO 모델 개발										
	+카메라를 이용한 거리 계산 알고리즘 개발										
	Croling API 알고리즘 개발										
	+메인 코드 개발 및 라즈베리 포팅										
테스트	시나리오를 기반으로 하는 모의 주행 테스트										
	+성능 디버깅										
	실제 주행 테스트										
종료	작품 보완										
온·오프라인 미팅	프로젝트 진행도 점검 및 질의응답										

3. 프로젝트 추진 과정에서의 문제점 및 해결방안

1) 프로젝트 관리 측면

- [1] 우리 팀은 매달 멘토님과 온라인 미팅을 진행하고, 일주일에 최소 한번은 학교나, 학교 앞 카페에서 만나 오프라인으로 개발을 진행하였다.
다만, 각자의 일정이 모두 상이하여, 매주 같은 날-시간에 만나지는 못하였고, 서로의 일정을 고려함에 따라, 매번 다른 날에 만나거나, 팀원 모두가 다 같이 만나지 못하는 경우가 여러 번 생겼었다.

때문에, 직전 주에 그다음 주 일정을 미리 계획하고, 해야 할 일을 정리하여, 역할을 분배해, 온라인으로도 여러 번 미팅하는 등의 방법을 마련하여, 초기 프로젝트를 진행할 때 비해, 안정적으로 진행할 수 있었다.
무엇보다 역할을 분배할 때는, 특정 작업을 선호하는 팀원에게 그 역할을 미리 배정함으로써, 작업 효율을 높일 수 있었다.

2) 프로젝트 개발 측면

- [1] 초기 프로젝트 목표는, RC카가 차선 탐지와 객체 탐지를 동시에 수행하도록 하는 것이었다. 하지만, 차선 탐지 알고리즘과 객체 탐지 알고리즘은 생각보다 매우 복잡하였고, 둘 중 하나의 기능조차 RC카가 수행하는 데 있어서 매우 버거웠다.

이를 해결하기 위해, 차선 탐지와 객체 탐지라는 두 개의 큰 목표를 분리해서 개발하기로 하였고, 실시간 영상 정보도 매 몇 FRAME을 건너뛰는 식의 방법으로 카메라(Raspberry Pi)의 부담을 최대한 줄여보았다.

- [2] RC카 양쪽 모터로 향하는 핀에, 상황에 맞는 PWM을 주더라도, 모터 문제 및 구조적 문제(바퀴가 두 개뿐)를 이유로, RC카가 해당 PWM에 맞게 주행하지 않는 것은, 이번 프로젝트의 가장 큰 문제였다. Software 쪽에서 정확한 차선 탐지를 하여 PWM을 전송하더라도, 모터가 그 신호를 제대로 받지 못하면, 차량이 잘못된 방향으로 주행했기 때문이다.

결국, 많은 노력에도 불구하고, Software쪽 에서 이를 해결하기에는 불가능하다는 결론을 내려, 초기 바퀴가 2개인 RC카를 버리고, 바퀴가 4개인 RC카와 새로운 모터를 만듦으로써, 주행 안정성을 높일 수 있었다.

- [3] Arduino용 RC카 특성상, 바퀴의 방향을 조절할 수 없고, 바퀴가 굴러가는 힘만 조절 가능하다는 점은, RC카의 주행 방향을 바꾸는 데 큰 어려움을 주었다. 일반적인 차량이 방향을 바꾸는 방법인, 바퀴의 방향을 조절할 수 없었기에 다른 방법을 마련해야 했다.

이는 RC카의 주행 방향을 바꿔야 하는 경우, 양 바퀴에 극단적으로 큰 차이의 PWM을 전송함으로써, 차량이 드리프트 하듯이 주행 방향을 바꾸도록 할 수 있었다.

4. 프로젝트를 통해 배우거나 느낀 점

- 이번 프로젝트에서는 현재 4차 산업혁명의 주축이 된 인공지능에 대하여 이해하고, 이 기술을 응용한 자율주행을 구현해보았다. 인공지능은 거의 모든 분야에 있어서 영향을 끼치는, 미래를 위하여 반드시 준비해야 할 중요한 기술로 떠오르고 있으며, 자율주행 기술 또한 수많은 기업과 나라에서 지원을 아끼지 않을 만큼 중요한 기술이다.

우리 팀은 이러한 시대 흐름에 따라, 이번 프로젝트에 지원하였다. 진행 과정에서 Canny Edge Detection, Hough Transform과 같은 다양한 기법을 익히고 응용해 보았으며, Canny Edge Detection에서 경계를 인식하기 위하여 이미지를 처리하는 방법, Hough Transform이 쓰이는 이유와 경계를 인식하기 위해 이미지에서 선형 특징을 추출하는 방법을 배우게 되었다. 이 기법들은 자율주행의 핵심 요소인 도로와 차선을 인식하는 데 매우 유용하게 사용되었다.

특히, Canny Edge Detection은 이미지에서 노이즈를 제거하고 유의미한 경계를 강조하는 데 탁월한 성능을 보였으며, Hough Transform은 이러한 경계를 직선이나 곡선으로 변환하여 도로와 차선의 위치를 파악하는 데 중요한 역할을 하였다.

또한, 우리는 딥러닝을 활용한 객체 인식 기술도 프로젝트에 적용하였다. 이를 통해 자율주행 차량이 보행자, 다른 차량, 신호등 등 다양한 객체를 인식하고 반응할 수 있도록 하였다.

프로젝트를 진행하면서 팀워크의 중요성도 크게 느낄 수 있었다. 자율주행 기술은 매우 복잡하고 다양한 전문 지식과 기술이 요구된다. 따라서 팀원들 각자가 자신이 잘하는 분야에서 최선을 다하며 협력하는 과정을 가졌고, 여기서 많은 것을 배우고 성장할 수 있었다고 생각한다. 문제를 해결하기 위해 서로의 의견을 존중하고 적극적으로 소통하는 과정이, 프로젝트의 성공에 큰 기여를 할 수 있었다고 생각한다.

더불어, 데이터 수집 및 전처리의 중요성도 실감했다. 자율주행 시스템은 대량의 데이터를 필요로 하며, 이러한 데이터가 얼마나 정제되고 정확하게 준비되었는지가 시스템의 성능에 큰 영향을 미친다. 실제 도로 환경에서 수집한 데이터를 통해 모델을 학습시키고 평가하는 과정에서, 데이터의 품질이 성능 향상에 얼마나 중요한지 알게 되었다.

마지막으로, 이번 프로젝트를 통해 지속적인 학습의 필요성을 느낄 수 있었다. 인공지능과 자율주행 기술은 빠르게 발전하고 있으며, 최신 기술 동향을 파악하고 이를 프로젝트에 적용하는 것이 매우 중요하다. 앞으로도 끊임없이 학습하고 새로운 기술을 습득하여 변화하는 시대에 발맞추어 나가야겠다는 다짐을 하게 되었다.

종합적으로, 이번 프로젝트는 인공지능과 자율주행 기술에 대한 깊이 있는 이해를 제공했을 뿐만 아니라, 실질적인 경험을 통해 팀워크와 문제 해결 능력을 향상시킬 수 있는 귀중한 기회였다. 이러한 경험들은 앞으로의 학업과 커리어에 큰 도움이 될 것이라 확신한다.

IV. 기대효과 및 활용 분야

1. 프로젝트의 기대효과

- 기존 대중적인 ADAS와 달리, 센서 등을 사용하지 않고, 비전 기반 영상 소스만을 사용하는 함에 있어, 경제성 확보
- 객체 검출 및 차선 인식을 통해 운전자에게 실시간 경고를 제공하여, 사고 예방에 기여
- 전방 충돌 경고 알림 시스템은 위험 상황을 사전에 감지하여, 운전자의 대응 시간을 확보
- 차선이탈 경보 장치는 차량 충돌 사고의 주요 원인인 차선 이탈상황을 감지하여, 사고 위험 감소
- 속도 제한 및 정지 신호와 같은 교통 표지판을 준수하여, 운전자 간의 충돌 회피 및 성숙한 운전문화 형성
- 실시간으로 제공되는 교통 정보와 날씨 예보는 운전자에게 편의성을 제공
- ADAS의 기능 향상은 운전자들의 스트레스를 감소시키고 운전보다 집중할 수 있도록 도움
- 운전 중 제공되는 정보들과 서비스는 연비 향상과 연결되어, 자동차의 연료 소비를 감소시키고 환경보호에 도움이 됨
- 효율적인 주행으로 인한 연료 소비 감소는 대기 오염과 탄소 배출을 줄여 지속 가능한 도로 환경을 조성

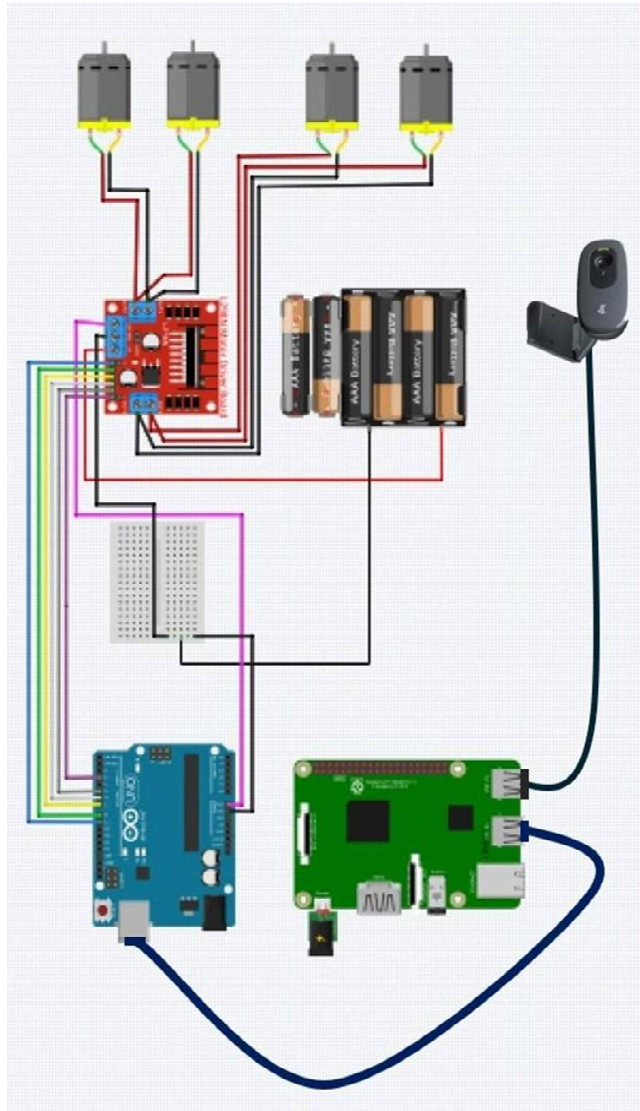
2. 프로젝트의 활용 분야

- 자율주행과 마찬가지로 다양한 이미지를 다루는 항공, 철도, 선박 분야
- 자율주행을 통한 자동화 운송 산업
- 교통통제와 교통사고가 없는 스마트시티 구현

V. 참고자료

□ 불임파일

○ 회로도



○ 소스 코드

-Line Detection

```

1  import cv2
2  import numpy as np
3  import serial
4  import time
5  import signal
6  import sys
7  import requests
8  import xml.etree.ElementTree as ET
9
10 port = '/dev/ttyACM0'
11 baud_rate = 9600
12 ser = None # 시리얼 포트 객체 초기화
13
14 # 날씨 API로부터 데이터 가져오기
15 def get_weather():
16     url = 'https://api.weatherapi.com/v1/current.xml?key=95eeb4d4c8bde42d2b6450219232708&q=Incheon&q=yes'
17     resp = requests.get(url)
18     xml_string = resp.content # API 응답 XML 데이터
19
20     root = ET.fromstring(xml_string)
21     location = root.find("./name").text
22     time = root.find("./localtime").text
23     update = root.find("./last_updated").text
24     temperature_c = root.find("./temp_c").text
25     humidity = root.find("./humidity").text
26     mm = float(root.find("./precip_mm").text) # mm 값은 실수로 변환
27     cloud = root.find("./cloud").text
28
29     # 날씨 상태 결정
30     if cloud == '1':
31         sky = "흐림"
32     elif mm != 0:
33         sky = "눈/비 내림"
34     else:
35         sky = "맑음"
36
37     # 데이터에 날씨 출력
38     print(f"현재 {location}은 (time)이며, 기온은 {temperature_c}°C, 습도는 {humidity}%, 강우/적설량은 {mm}mm입니다. 안전 운전하세요.")
39
40     # 강우/적설량이 있을 때 조립할 값 반환
41     return 10 if mm != 0 else 0 # 강우가 있을 경우 PWM 감소량 10 반환
42
43 # 신호 종료 처리
44 def signal_handler(sig, frame):
45     print('Exiting gracefully...')
46     if ser is not None:
47         send_pwm(ser, 0, 0) # 모터 정지
48         ser.close()
49     cv2.destroyAllWindows()
50     sys.exit(0)
51
52 # 시리얼 포트 연결
53 def connect_serial():
54     global ser
55     try:
56         ser = serial.Serial(port, baud_rate, timeout=1)
57         print(f"Connected to {port}.")
58     except serial.SerialException as e:
59         print(f"Serial connection error: {e}")
60
61 # PWM 값 전송
62 def send_pwm(ser, left_pwm, right_pwm):
63     try:
64         command = f"{left_pwm},{right_pwm}\n"
65         ser.write(command.encode())
66         print(f"Sent PWM: L:{left_pwm}, R:{right_pwm}")
67     except serial.SerialException as e:
68         print(f"PWM send error: {e}")
69
70 # RC기 제어
71 def control_robot(distance, side, pwm_offset):
72     if side == 'right':
73         print(f"Right side distance: {distance}")
74         if distance < 80:
75             send_pwm(ser, 10, 220)
76             print("Moving Forward sharply to the right...")
77         else:
78             send_pwm(ser, 70 + pwm_offset, 70 - pwm_offset)
79             print("Moving Forward gently...")
80
81     elif side == 'left':
82         print(f"Left side distance: {distance}")
83         if distance < 80:
84             send_pwm(ser, 220, 10)
85             print("Moving Forward sharply to the left...")
86         else:
87             send_pwm(ser, 70 - pwm_offset, 70 + pwm_offset)
88             print("Moving Forward gently...")
89
90 # 로봇 제어 루프
91 def robot_control_loop(cap, roi_points, pwm_offset):
92     frame_count = 0
93     while True:
94         ret, frame = cap.read()
95         if not ret:
96             break
97
98         frame_count += 1
99
100     if frame_count % 6 == 0: # 매 6번째 프레임마다 처리

```

```

101     frame_with_box, distance, side = detect_lanes(frame, roi_points)
102
103     if side is not None:
104         print(f"Detected line on the {side} before moving.")
105         control_robot(distance, side, pwm_offset)
106
107     cv2.imshow('Original Frame', frame_with_box)
108
109     if cv2.waitKey(1) & 0xFF == ord('q'):
110         break
111
112 # ROI 마스크 생성
113 def create_roi_mask(image, roi_points):
114     mask = np.zeros(image.shape[:2], dtype=np.uint8)
115     cv2.fillPoly(mask, [roi_points], 255)
116     return mask
117
118 # 라인 감지
119 def detect_lanes(image, roi_points):
120     height, width = image.shape[:2]
121     roi_mask = create_roi_mask(image, roi_points)
122
123     cv2.polylines(image, [roi_points], isClosed=True, color=(0, 255, 0), thickness=2)
124
125     hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
126
127     lower_white = np.array([0, 0, 200])
128     upper_white = np.array([180, 25, 255])
129     white_mask = cv2.inRange(hsv, lower_white, upper_white)
130
131     lower_yellow = np.array([20, 100, 100])
132     upper_yellow = np.array([30, 255, 255])
133     yellow_mask = cv2.inRange(hsv, lower_yellow, upper_yellow)
134
135     combined_mask = cv2.bitwise_or(white_mask, yellow_mask)
136     masked = cv2.bitwise_and(combined_mask, roi_mask)
137
138     blurred = cv2.GaussianBlur(masked, (5, 5), 0)
139     edges = cv2.Canny(blurred, 50, 150)
140
141     lines = cv2.HoughLines(edges, 1, np.pi / 180, 50, minLineLength=50, maxLineGap=150)
142
143     detected_line = None
144     if lines is not None:
145         slopes = []
146         for line in lines:
147             x1, y1, x2, y2 = line[0]
148             if x2 - x1 == 0:
149                 continue
150             slope = (y2 - y1) / (x2 - x1)
151             slopes.append(slope)
152
153         if slopes:
154             detected_line = lines[0][0]
155
156     result = image.copy()
157     distance = 0
158
159     center_x = (roi_points[0][0] + roi_points[1][0]) // 2
160     center_y = (roi_points[0][1] + roi_points[2][1]) // 2
161     center_point = (center_x, center_y)
162
163     cv2.circle(result, center_point, 5, (255, 255, 0), -1)
164
165     if detected_line is not None:
166         x1, y1, x2, y2 = detected_line
167         avg_x = (x1 + x2) // 2
168
169         error = avg_x - center_x
170         distance = abs(error)
171
172         return result, distance, 'right' if avg_x > center_x else 'left'
173
174     return result, distance, None
175
176 def main():
177     global ser
178     signal.signal(signal.SIGINT, signal_handler)
179     connect_serial()
180
181     pwm_offset = get_weather() # 강우에 따른 PWM 요령 설정
182
183     cap = cv2.VideoCapture(0)
184     cv2.namedWindow('Original Frame', cv2.WINDOW_NORMAL)
185     cv2.resizeWindow('Original Frame', 640, 480)
186
187     roi_points = np.array([[80, 440], [560, 440], [560, 220], [80, 220]])
188
189     try:
190         robot_control_loop(cap, roi_points, pwm_offset)
191     finally:
192         cap.release()
193         if ser is not None:
194             ser.close()
195         cv2.destroyAllWindows()
196
197 if __name__ == "__main__":
198     main()
199

```

-Object Detection

```

1 import cv2
2 import numpy as np
3 import torch
4 import serial
5 import time
6 import platform
7 import pathlib
8 import warnings
9
10 warnings.filterwarnings("ignore", category=FutureWarning)
11
12 # Serial Port Configuration
13 port = '/dev/ttyACM0'
14 baud_rate = 9600
15
16 # Camera Parameters
17 camera_matrix = np.array([[616.0, 0.0, 320.0], [0.0, 616.0, 240.0], [0.0, 0.0, 1.0]])
18
19 # OS-Specific Path Handling
20 if platform.system() == 'Windows':
21     pathlib.PosixPath = pathlib.WindowsPath
22 else:
23     pathlib.WindowsPath = pathlib.PosixPath
24
25 # Load Model
26 model_path = '/home/pi4/Desktop/yolov5/zhaniun_1009_n.pt'
27 model = torch.hub.load('ultralytics/yolov5', 'custom', path=model_path, force_reload=True)
28
29 confidence_threshold = 0.3
30
31 # Class Permanent Size (meters)
32 class_heights = {
33     0: 0.01, # green
34     1: 0.01, # red
35     2: 0.01, # yellow
36     3: 0.02, # person
37     4: 0.04, # car
38     5: 0.02, # 30
39     6: 0.02, # 50
40     7: 0.03, # crosswalk
41     8: 0.03 # childrenzone
42 }
43
44 # Serial Connection
45 def connect_serial():
46     try:
47         ser = serial.Serial(port, baud_rate, timeout=1)
48         print(f"Connected to {port}.")
49         return ser
50     except serial.SerialException as e:
51         print(f"Serial connection error: {e}")
52         return None
53
54 # Store the last detected class and its frame count
55 last_detected = {'class': None, 'count': 0}
56 frame_threshold = 3 # Minimum number of consecutive frames for a valid detection
57
58 # Send PWM Command
59 def send_pwm(ser, left_pwm, right_pwm, class_name):
60     try:
61         command = f"({left_pwm},{right_pwm}\n"
62         ser.write(command.encode())
63         print(f"({class_name}) detected.. trip by ({left_pwm}, {right_pwm})")
64     except serial.SerialException as e:
65         print(f"PWM send error: {e}")
66
67 # Process each frame
68 def process_frame(frame, ser):
69     global last_detected
70     frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
71     results = model(frame_rgb)
72
73     closest_distance = float('inf')
74     closest_class = None
75
76 # Assign unique colors for each class
77 colors = {
78     0: (0, 255, 0), # green
79     1: (0, 0, 255), # red
80     2: (0, 255, 255), # yellow
81     3: (255, 0, 0), # person
82     4: (255, 255, 0), # car
83     5: (128, 128, 128), # 30
84     6: (0, 128, 255), # 50
85     7: (255, 0, 255), # crosswalk
86     8: (255, 128, 0) # childrenzone
87 }
88
89 for det in results.xyxy[0]:
90     x1, y1, x2, y2, conf, cls = det
91     if conf > confidence_threshold:
92         x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)
93         h = y2 - y1
94         distance = (class_heights[int(cls)] * camera_matrix[1, 1]) / h
95
96 # Draw bounding box and display class/distance info
97 label = model.names[int(cls)]
98 cv2.rectangle(frame, (x1, y1), (x2, y2), colors[int(cls)], 2)
99 cv2.putText(frame, f"({label}) ({distance:.2f}m)", (x1, y1 - 10),
100             cv2.FONT_HERSHEY_SIMPLEX, 0.5, colors[int(cls)], 2)

```

```

101
102     # Track the closest object
103     if distance < closest_distance:
104         closest_distance = distance
105         closest_class = int(cls)
106
107     # Check if the closest class is detected consecutively
108     if closest_class == last_detected["class"]:
109         last_detected["count"] += 1
110     else:
111         last_detected["class"] = closest_class
112         last_detected["count"] = 1
113
114     # Execute PWM command only if the class is stable for 'frame_threshold' frames
115     if last_detected["count"] >= frame_threshold:
116         execute_pwm_command(scr, closest_class, closest_distance)
117
118     return frame
119
120 # Execute PWM Command based on class and distance
121 def execute_pwm_command(scr, cls, distance):
122     if cls == 0: # green
123         send_pwm(scr, 200, 200, "green")
124
125     elif cls == 1 and distance <= 0.3: # red
126         send_pwm(scr, 0, 0, "red")
127     elif cls == 1: # red (not too close)
128         send_pwm(scr, 140, 140, "red")
129
130     elif cls == 2 and distance <= 0.3: # yellow (close)
131         send_pwm(scr, 140, 140, "yellow")
132     elif cls == 2: # yellow (far)
133         send_pwm(scr, 200, 200, "yellow")
134
135     elif cls == 3 and distance <= 0.4: # person (very close)
136         send_pwm(scr, 0, 0, "person")
137     elif cls == 3 and distance <= 0.6: # person (moderately close)
138         send_pwm(scr, 140, 140, "person")
139     elif cls == 3: # person (far)
140         send_pwm(scr, 200, 200, "person")
141
142     elif cls == 4 and distance <= 0.6: # car (close)
143         send_pwm(scr, 0, 0, "car")
144     elif cls == 4 and distance <= 0.8: # car (moderately close)
145         send_pwm(scr, 140, 140, "car")
146     elif cls == 4: # car (far)
147         send_pwm(scr, 200, 200, "car")
148
149     elif cls == 5: # 30
150         send_pwm(scr, 140, 140, "30")
151
152     elif cls == 6: # 50
153         send_pwm(scr, 200, 200, "50")
154
155     elif cls == 7 and distance <= 0.3: # crosswalk
156         send_pwm(scr, 140, 140, "crosswalk")
157     elif cls == 7:
158         send_pwm(scr, 200, 200, "crosswalk")
159
160     elif cls == 8 and distance <= 0.3: # childrenzone
161         send_pwm(scr, 140, 140, "childrenzone")
162     elif cls == 8:
163         send_pwm(scr, 200, 200, "childrenzone")
164
165 # Main Function
166 def main():
167     cap = cv2.VideoCapture(1)
168     ser = connect_serial()
169     cv2.namedWindow('Frame', cv2.WINDOW_NORMAL)
170
171     try:
172         while True:
173             ret, frame = cap.read()
174             if not ret:
175                 print("Failed to grab frame")
176                 break
177
178             frame = cv2.resize(frame, (320, 240))
179             processed_frame = process_frame(frame.copy(), ser)
180             cv2.imshow('Frame', processed_frame)
181
182             if cv2.waitKey(1) & 0xFF == ord('q'):
183                 break
184
185     except KeyboardInterrupt:
186         print("Program interrupted. Stopping the robot.")
187
188     finally:
189         if ser:
190             send_pwm(scr, 0, 0, "stop")
191             ser.close()
192             cap.release()
193             cv2.destroyAllWindows()
194
195 if __name__ == '__main__':
196     main()
197

```

-Arduino

```

const int ENA = 9; // 오른쪽 모터 (MOTORA)
const int IN1 = 8;
const int IN2 = 7;
const int IN3 = 6;
const int IN4 = 5;
const int ENB = 3; // 왼쪽 모터 (MOTORB)

```

```

void setup()
{
    pinMode(ENA, OUTPUT);
    pinMode(ENB, OUTPUT);
    pinMode(IN1, OUTPUT);
    pinMode(IN2, OUTPUT);
    pinMode(IN3, OUTPUT);
    pinMode(IN4, OUTPUT);

    Serial.begin(9600);
}

```

```

void setMotors(int leftSpeed, int rightSpeed)
{
    // 왼쪽 모터 방향 설정
    if (leftSpeed >= 0)
    {
        digitalWrite(IN3, HIGH);
        digitalWrite(IN4, LOW);
    }
    else
    {
        digitalWrite(IN3, LOW);
        digitalWrite(IN4, HIGH);
        leftSpeed = -leftSpeed;
    }
}

```

```

// 오른쪽 모터 방향 설정
if (rightSpeed >= 0)
{
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
}
else
{
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    rightSpeed = -rightSpeed;
}

```

```

// 모터 속도 설정
analogWrite(ENB, leftSpeed);
analogWrite(ENA, rightSpeed);
}

```

```

void loop()
{
    if (Serial.available() > 0)
    {
        String input = Serial.readStringUntil('\n');
        int commaIndex = input.indexOf(',');

        if (commaIndex != -1)
        {
            int leftPWM = input.substring(0, commaIndex).toInt();
            int rightPWM = input.substring(commaIndex + 1).toInt();

            setMotors(leftPWM, rightPWM);

            Serial.print("Left PWM: ");
            Serial.print(leftPWM);
            Serial.print(", Right PWM: ");
            Serial.println(rightPWM);
        }
    }
}

```