

Valet Parking Robot 개발

객체 인식
기술지원팀

2019. 10. 11

ADAS
ONE

1-1 유아용 전동차 (Toy Car)

1-2 구조물 (Chassis)

1-3 조향각 계산기 (Steering Angle - PWM Converter)

2-1 Camera Calibration

2-2 AVM

3-1 객체인식 (Object Detection)

3-2 주차선인식 (Parking Lane Detection)

3-3 주차칸인식 (Parking Space Detection)

4-1 주차칸결정 (Parking Space Decision)

4-2 주차경로결정 (Parking Path Decision)

4-3 gym-gazebo

5-1 주차장지도제작 (Parking Lot Mapping)

추신 : 이번주에 진행된 부분만
검정색 표시, 미진행 부분은
회색 표시.

유아용 전동차 (Toy Car)

1) 시스템 정의

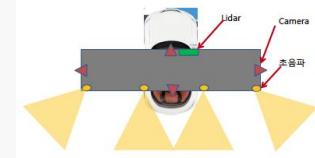
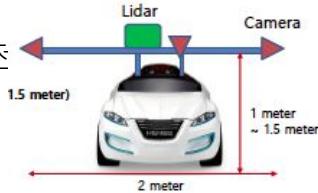
- a) 목적 및 기능 : 유아용 전동차 선정, HW를 장착할 구조물 제작
- b) 배경 및 필요성 : 발렛파킹로봇의 SW 모듈을 테스트할 프로토타입 제작 (제어, 센싱, 비전)
- c) 시스템 운영개념 : 내장배터리로 전원공급되어 통신명령에 따라 주행제어되는 전동차
- d) 운영 및 환경 제약사항 : 저가형 배터리로 인해 충전 7시간/작동 1시간 예상, 모터 해상도

2) 기능 요구사항

- a) 주행 : 12V 듀얼 모터
- b) 조향 : 스텝링 모터
- c) 제어기 : Xavier-Arduino ROS 통신 확인
- d) 모터 제어 : 임베디드 보드에서 제어해 계산된 경로를 주행, Arduino에서 모터 가동/방향전환/인코더 확인
- e) 모듈 지지 : 스텠봉을 체결하여 구조물을 설치, 카메라/라이더/센서류 위치 선정
- f) 서스펜션 : 카메라 진동 억제

3) 비기능 요구사항

- a) 전진/후진 추진속도 : 6~8km/h
- b) 예비/확장 배터리 : 배터리로 교체 가능
- c) 크기 제원 : 전고 1~1.5m, 전폭 1.5~2m
- d) 허용하중 : 20kg+

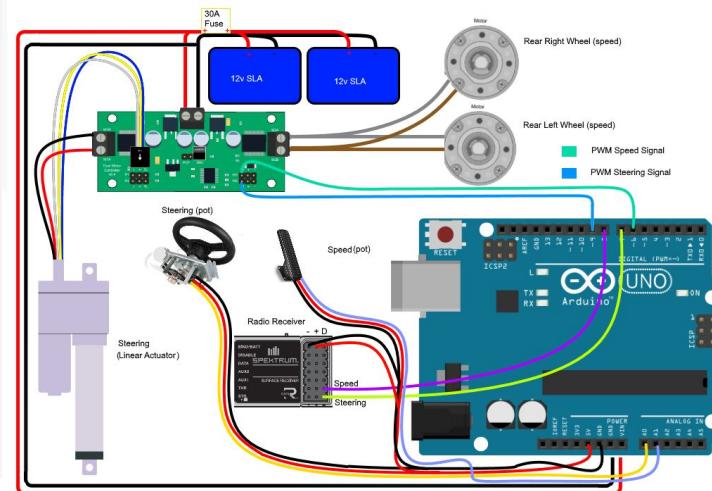


유아용 전동차 (Toy Car)

ADAS
ONE

전동차 구축 일정

- 1) 전동차 34만원 : 구매 진행 및 조립, 시운전 완료 (10/2주차)
- 2) 전장 : 제어기, 모터 드라이브 분석 및 제어기 교체 예정(10/3주차)
- 3) 제어 : 컴퓨터를 통해 전동차 기본 주행 제어 테스트 예정(10/3주차)
- 4) 구조물 : 자재 구매, 조립 및 차체 개조 예정 (10/4주차)
- 5) 자동제어 시스템 : 자비어 및 카메라 등 센서류 장착.



유아용 전동차 (Toy Car)



유아용 전동차 (Toy Car)

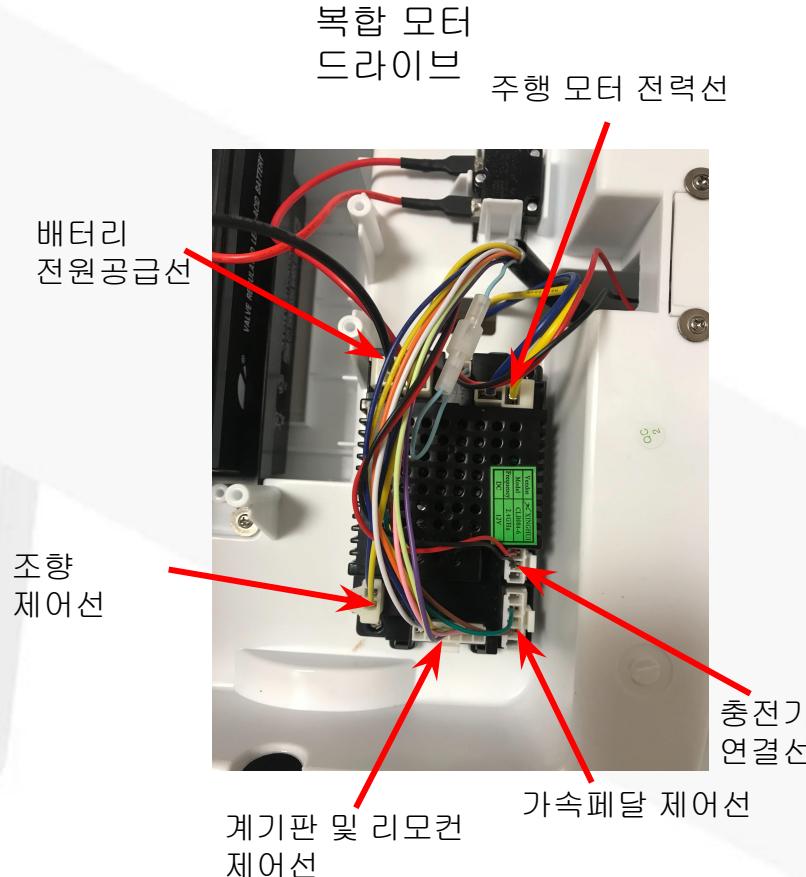
ADAS
ONE

전동차 분석 및 개조

- 1) 제어기와 모터 드라이브 신호선 전압계
검침 분석
- 2) 제어기 아두이노 교체 연결
- 3) 아두이노를 통해 조향&주행 모터 제어

현재 대략적인 배선 분석만 되어있는 상태.

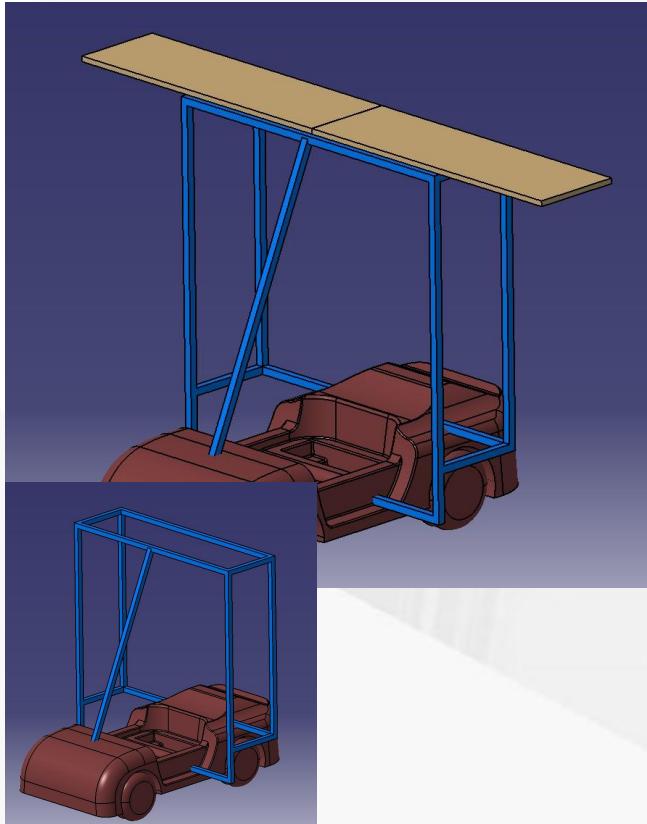
모터 제어를 위해 신호선의 상세 내역 분석과
모터 드라이브 제어 코드 테스트 예정.



유아용 전동차 (Toy Car)

ADAS
ONE

상부구조물 (Chassis)



절단 사이즈	가격
300mm 이하	1600원
600mm 이하	2000원
900mm 이하	2300원
1200mm 이하	2600원

* 절단 및 타공 비용 무료 (양 끝 2곳: 타공)
* CR 1.1T 20mm 각파이프(6미터)무게: 약 4kg입니다.
* 별도의 사이즈도 절단 가능합니다. (전화 문의 바랍니다.)



작업성이 좋은 미송집성목



미송 집성목은 색상이 밝고 깨끗하여 비교적 가볍고 부드러운 목재입니다.

단단하고 훨씬성이 격이 가구제작에 용이합니다

접착력과 도장성이 매우 뛰어난 목재입니다

- 1) 각파이프
20X20X1.1T
무게합: 6.7kg
택배비 6000원
합계 45400원
- 1000mm 5개
920mm 4개
260mm 4개
115mm 2개
50mm 4개(받침용)

- 2) 조인트
무게합: 1.9kg
단가 2500원
합계 25000원
- 상단 2개
중하단 6개
중간/하단 2개

- 3) 상판
1200X300X18T
무게합: 6.5kg
2EA = 25000원

무게합계 : 15.1kg+부자재
구조물 자재 합계 금액 :
95,400원

차량 허용 하중 : 25kg

1) 시스템 정의

- a) 목적 및 기능 : Path Planning에서 계산된 경로를 따라 로봇이 실제 주행할 수 있는 조향모터와 주행모터 회전량을 계산
- b) 시스템 운영개념 : Path Planning 데이터를 받아 정보 계산 후 모터 제어기로 데이터 전송
- c) 운영 및 환경 제약사항 : 노면상태 등 환경 요인에 의한 오차 발생 가능성 있음.

2) 기능 요구사항

- a) 입력부 : MAIN SYSTEM의 Path Planning에서 (다음 위치 좌표, 현재 위치 좌표) 입력.
- b) 회전량계산부 : 다음 위치 좌표에 도달하기 위한 θ 를 추정
- c) 출력부 : EMBEDDED의 Motor PWM에게 /steering_angle을 ROS pub.

3) 비기능 요구사항

- a) 데이터 측정
 - i) 차체의 최소 조향반경
 - ii) 조향모터 각도에 따른 차체 회전 반경
 - iii) 최대 조향시 속도에 따른 구조물 안정성

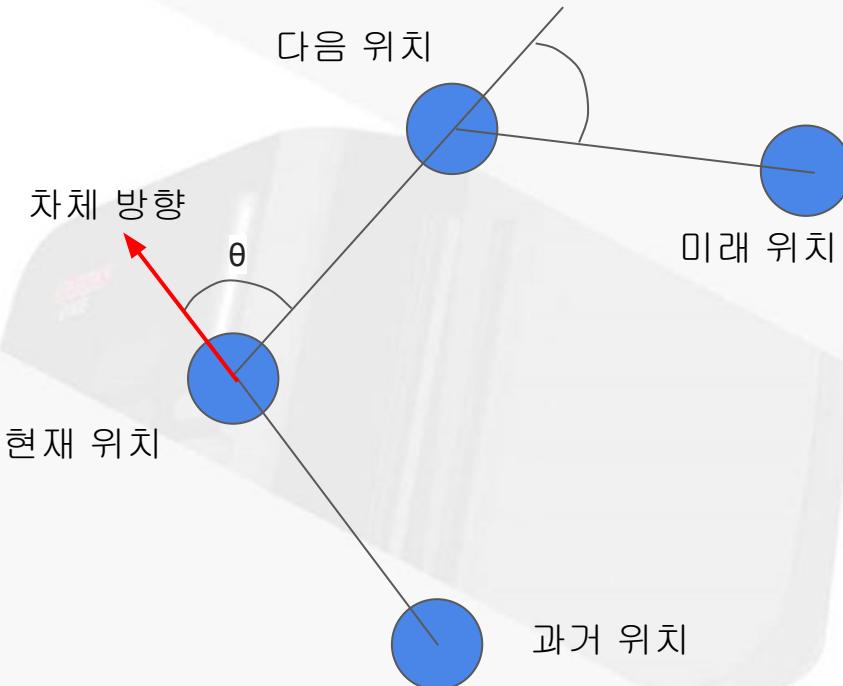
```
analogWrite(enA, motorSpeedA); // Send PWM signal to motor A  
analogWrite(enB, motorSpeedB); // Send PWM signal to motor B
```

전동차에 최적화가 필요하여 Path planning과 전동차 기본제어가 완료된 후 작성.

조향각 계산기 (Steering Angle - PWM Converter)

조향각 계산모델

Path Planning에서 찍어주는 포인트



- 1) 조향 모터 회전 각도
 $=A^*B^*(\text{현재 바퀴 조향각} - \theta)$

A(모터 바퀴 회전비)
B(바퀴 조향각과 차체 회전각 비)

- 2) 주행중 속도 모터 제어

직진: $\theta = 0$ -> 차량 속도 최대(8km/h)
회전: $\theta = \text{MAX}$ -> 구조물이 안정한 속도
(테스트 필요)

- 3) 차체의 조향 반응속도가 느리면 미래 위치를 미리 계산하여 조향 시간을 단축할 수 있다.

Camera Calibration

1) Camera Calibration

- a) 목적 및 기능 : 카메라 내부/외부 파라미터 추정 및 영상왜곡 보정
- b) 배경 및 필요성 : libxcam에 입력하여 **bird-eye view**의 왜곡율을 보정
- c) 시스템 운영개념 : 카메라보정 포팅 & 체스보드 보정

2) 기능 요구사항

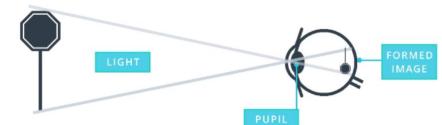
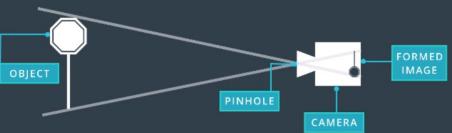
- a) 카메라 보정
 - i) 입력 : BGGR/fisheye 1920x1080x16bit x20
 - ii) 출력 : Fx, Fy, Cx, Cy, K1, K2, K3, Rv, Tv
- b) 영상 왜곡 보정
 - i) 입력 : RGB 영상, CamMat, Dist
 - ii) 출력 : 왜곡이 보정된 RGB 영상

3) 비기능 요구사항

- a) Camera Calibration Porting : OpenCV Library 이용
- b) 4 Fish-eye Camera & Xavier 인터페이스: 전동차의 전방, 좌측, 우측, 후방에 설치, 출력확인

Pinhole
Camera
Model

The
Human
Eye



Return values:
ret, mtx, dist, rvecs, tvecs

1) 경과

- a) OpenCV Xavier 포팅: 설치스크립트 깃허브 env 브랜치에 공유
- b) 내부파라미터 : 초점거리, 주점거리, 왜곡계수 구하여 보정 성공
- c) 외부파라미터 : 회전량, 이동량 구하여 3축 동기화 성공
- d) 디렉토리 관리 : 디렉토리 생성, 취소시 제거, 변수 텍스트 저장

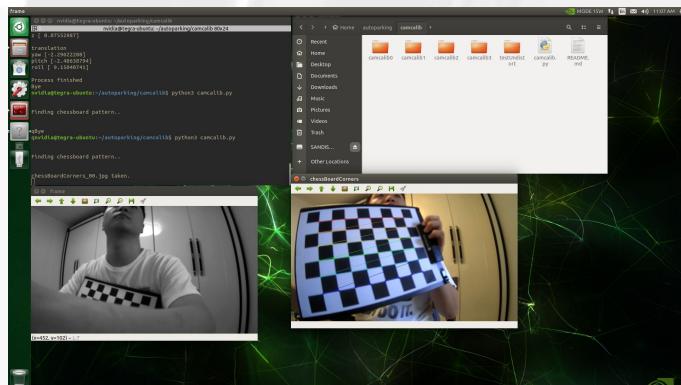
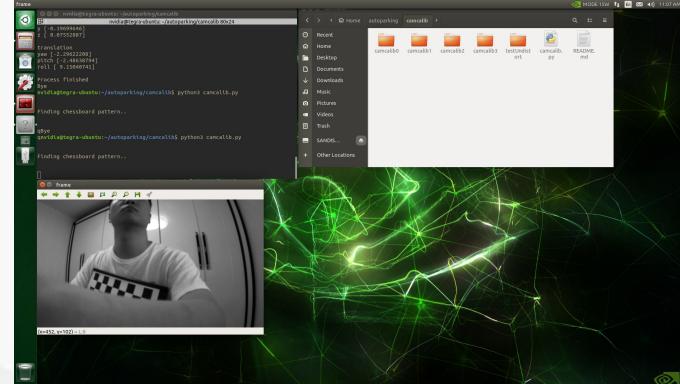
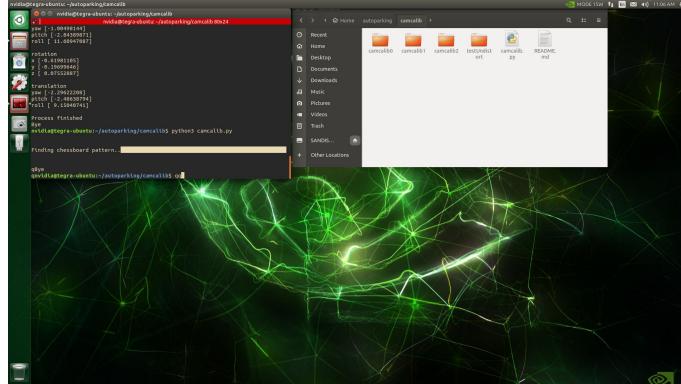
2) 향후 방향

- a) 모듈화 및 문서화
 - i) 변수값로 가변화 & 주석 / 리드미 / 커맨드파서 작성
- b) 4채널 카메라 보정 + JP4.2
 - i) 입력 : fisheye 1280x720x32bit
 - ii) 출력 : Fx, Fy, Cx, Cy, K1, K2, K3, Rv, Tv
- c) libxcam 지원
 - i) 입력 : 4Ch calibrated camera image
 - ii) 출력 : Bird-eye view

3) 비기능적 요구사항

- a) 구동체 설치 : 전동차의 전방, 좌측, 우측, 후방에 설치

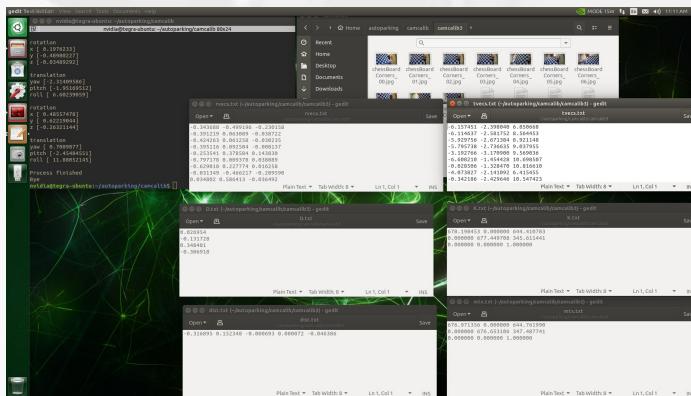
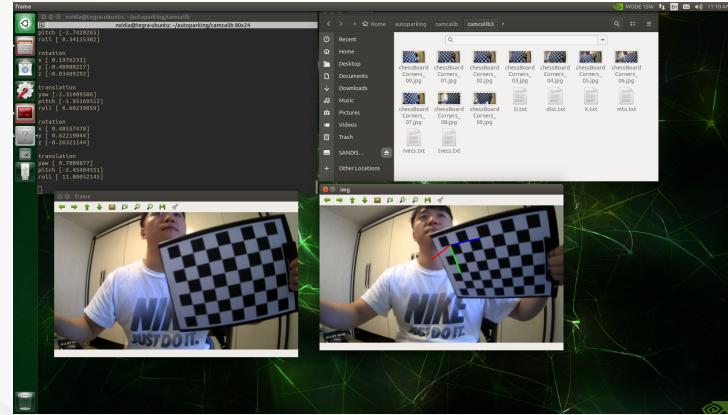
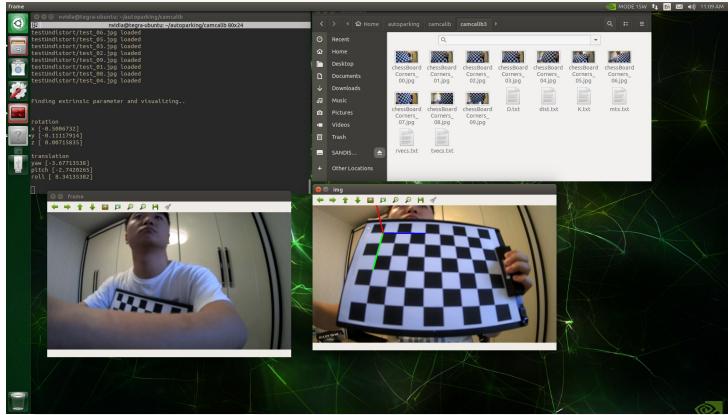
Camera Calibration



카메라
캘리브레이
션
디렉토리
자동 관리

체스보드
패턴인식

Camera Calibration



외부파라미터 검증 → 좌표축
추적

내외부 파라미터 텍스트 저장

1) Around View Monitor 시스템 정의

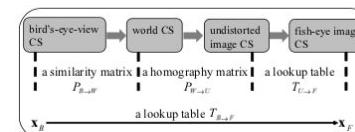
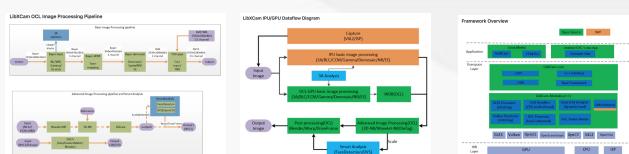
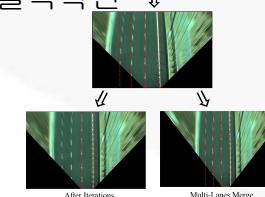
- a) 목적 및 기능 : 차량의 360도 scene bird's-eye view로 제공
- b) 배경 및 필요성 : 주차칸인식과 주차선인식의 입력
- c) 시스템 운영개념 : AVM이 4개 카메라에서 1장의 주변버드뷰영상을 생성
- d) 시스템 아키텍쳐 : Xavier JP4.2에 Intel 기반 libxcam_app 포팅

2) 기능 요구사항

- a) 입력 : BGGR/fisheye 1920x1080x16bit x4, Cam Calib Prams
- b) Image Stitching : 파노라마와 같은 surround view 생성
- c) Fisheye Distortion Correction : world의 bowl-view를 평탄화
- d) Inverse Perspective Mapping : 원근 제거 & 2D로의 매핑 → bird's-eye view
- e) 형변환 : C++ NV12 1280x720 → python BGR (320 x 160)
- f) 출력 : python BGR 변환

3) 비기능 요구사항

- a) 4 fish-eye camera & Xavier 인터페이스: 전동차의 전방, 좌측, 우측, 후방에 설치, 출력확인



객체인식 (Object Detection)

1) 시스템 정의

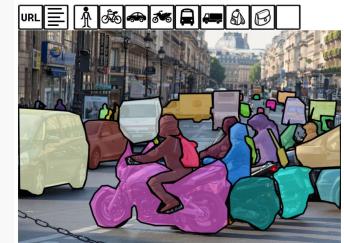
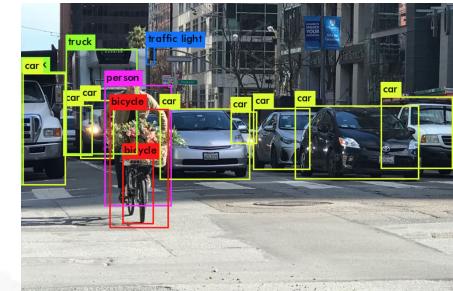
- a) 목적 및 기능 : 전진/후진 경로에 장애물을 식별
- b) 배경 및 필요성 : 주행방향에 나타난 장애물에 따라 다른 행동을 하기 위함
- c) 초기요구사항
 - i) 4-클래스 검출 (차량, 보행자, 자전거, 오토바이)
 - ii) 주행속도 10~30km → 정지시간 2~3초 → 정지거리 16~24m → 25m 이내 객체 검출
 - iii) 이미지크기 300x300일시 25m내 사람 25x25
 - iv) 근접시 오검출로 인하여 시맨틱세그멘테이션 접근
 - v) 데이터셋 및 목표객체 확장성

2) 기능 요구사항

- a) 훈련 : COCO에서 (640x480, JSON) 입력, 가중치 최적화
- b) 입력부 : /dev/video에서 python 모듈로 스트리밍, gstreamer vs uvccapture
- c) 추론 : 최적화된 파라미터에 의한 forward-pass
- d) 출력부 : 메인시스템으로 (python list of bounding boxes & class predictions) 반환

3) 비기능 요구사항

- a) 계산가속 : CUDA, CuDNN, Half-precision, TensorRT
- b) 환경 구축 : PyTorch 1.1
- c) 감독학습 : COCO Dataset (car, truck, bus, motorcycle, bicycle)
- d) 오픈소스 : maskrcnn-benchmark (YOLO, CenterNet, DarkNet, Faster R-CNN)



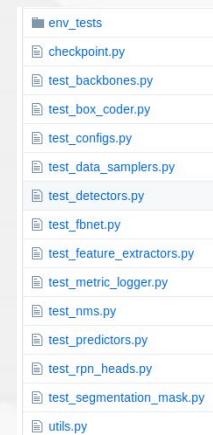
라이브러리 설정

Faster R-CNN and Mask R-CNN in PyTorch 1.0

페이스북 리서치팀에서 개발하고 제공하는 오브젝트 디텍션과 시멘틱 세그멘테이션이 함께 가능한 라이브러리

- 1) 오브젝트 디텍션과 시멘틱 세그멘테이션을 함께 처리할 수 있음
- 2) R-CNN 기반 pretrained 모델 제공
- 3) 500MB의 적은 GPU 메모리 사용
- 4) 싱글 및 멀티 GPU 트레이닝
- 5) Mixed precision training
- 6) Dataset 추가 용이, 목표 객체 확장성

- 1) pytorch 1.0
- 2) CUDA 9.0+



라이브러리 분석

Facebook research mask r-cnn

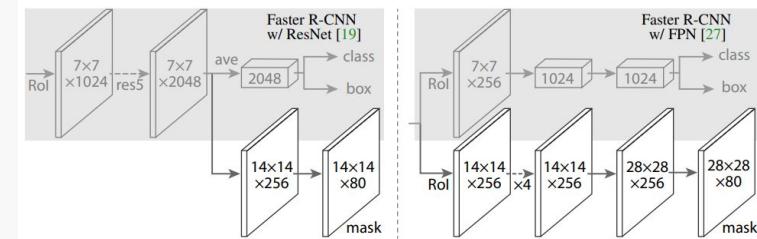
객체 인스턴스를 차별화하지 않고 각 픽셀을 고정된 범주로 분류하는 것이 목표로 개발

라이브러리의 핵심 강점

- 1) Mask R-CNN은 구현이 간단하고 Faster R-CNN 보다 속도가 빠름.
- 2) 계산 오버헤드가 적음.
- 3) 복잡한 다단계 캐스트케이드 방법을 통해 마스크의 병렬 예측을 기반으로 함.

Mask R-CNN의 핵심 요소

- 1) Faster R-CNN : 2 단계로 구성됨.
 - a) RPN을 통해 경계박스 후보 산출,
 - b) RoIPool을 사용하여 최종 경계박스 추출.
- 2) Mask R-CNN : 마찬가지로 2단계 구성.
 - a) RPN을 통해 클래스와 박스를 예측,
 - b) 첫번째 단계와 동시에 오프셋, 마스크 R-CNN이 각각에 대한 이진 마스크를 출력.



기존의 Faster RCNN과의 Head Architecture 차이

- 기존의 Faster RCNN 헤드 2개를 확장
- FPN 백본 마스크 분기 추가
- 단순성을 위해 14X14X 대신에 7X7 RoI로 작동하도록 변경

주차선인식 (Parking Lane Detection)

1) 시스템 정의

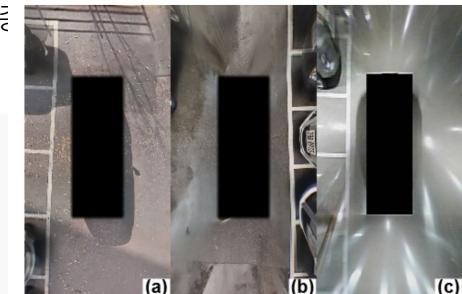
- a) 목적 및 기능 : 탑뷰영상에서 차선을 찾아냄
- b) 배경 및 필요성 : 자동주차경로계산에 카메라 영상을 추상화함으로써 정확성 향상
- c) ~~초기요구사항 : 4 클래스라벨 (0 배경, 1 주차선, 2 차량, 3 장애물)~~
- d) 요구사항 : 2-클래스라벨 (0 배경, 1 주차선)
- e) 시스템 운영개념 : 딥뉴럴 시맨틱 세그멘테이션 모델 추론
- f) 운영 및 환경 제약사항 : Xavier에 libxcam_app 포팅

2) 기능 요구사항

- a) 주차선 훈련 : AVM (320x160, GT) 입력, 가중치 최적화
- b) 입력부 : python 모듈로 영상입력 (320x160 RGB)
- c) 주차선 추론 : 최적화된 파라미터에 의한 forward-pass
- d) 주차선 추출 : 주차선을 제외한 차량, 장애물의 출력을 배제
- e) 출력부 : 주차칸인식기에게 주차선영상 (200x100 Numpy.array) 반환

3) 비기능 요구사항

- a) 오픈소스 : 기존모델 개량
- b) 계산가속 : CUDA, CuDNN, Half-precision, TensorRT
- c) 환경 구축 : PyTorch 1.1.0
- d) 감독학습 : AVM Dataset - Semantic Segmentation (outdoor/indoor)
- e) 모델 : DeepLabV3+ (Xception65)



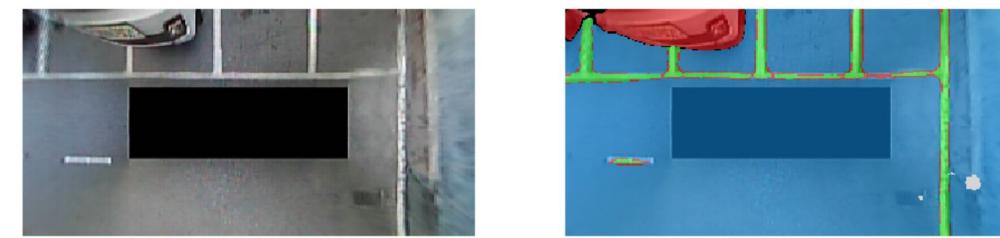
(a) outdoor-day, (b) outdoor-rainy, (c) indoor

주차칸인식 (Parking Lane Detection)

1) 기존에 만든 시맨틱 세그멘테이션



2) 텐서플로를 이용한 시맨틱 세그멘테이션 포팅



주차칸인식 (Parking Space Detection)

ADAS
ONE



Figure 9. Directions of the marking-point patterns.

1) 시스템 정의

- a) 목적 및 기능 : 로봇 주변에 있는 주차칸을 찾기
- b) 배경 및 필요성 : 주변에 모든 주차칸을 찾아낸 후, 자율주차의 목적지가 되는 하나의 주차칸을
- c) 시스템 운영개념 : 주차코너를 찾고, 그 칸의 입구선을 찾고, 입구점을 분석해, 주차칸을 찾음
- d) 운영 및 환경 제약사항 : Xavier에 libxcam_app 포팅

2) 기능 요구사항

- a) 입력 : libxcam_app (C++) → 카메라 NV12영상 (1280x720 NV12) 입력
- b) 주변 주차칸 추론 (Python) : 입력받은 영상에 있는 주차칸 주변 주차칸입구점
- c) 출력 : 주변 주차칸 (양 입구점) → 주차칸 결정부 (Python)

3) 비기능 요구사항

- a) 환경 구축 : Matlab
- b) 오픈소스 포팅 : ~~Parking Slot Detection L Algo., Tongji Parking Slot Dataset (600x600)~~
- c) AVM Dataset - Parking Space
- d) 오픈소스 가공 : maskrcnn-benchmark

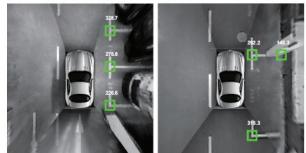
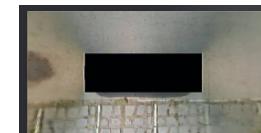
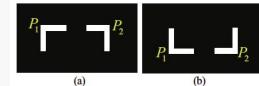


Figure 10. Marking point detection results on two typical surround-view images.



(a) (b)



(a) (b)

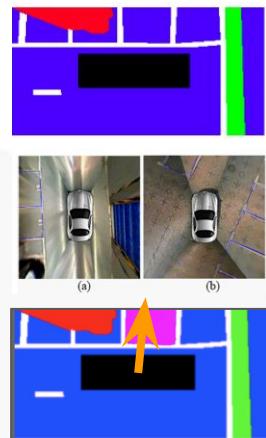
```
%YAML:1.0
empty:
  - { type:0, condition:0, slot:0, tr:[ 302, 119 ], tl:[ 239, 119 ] }
  - { type:0, condition:0, slot:0, tr:[ 235, 121 ], tl:[ 171, 121 ] }
  - { type:0, condition:0, slot:0, tr:[ 166, 122 ], tl:[ 96, 122 ] }
  - { type:0, condition:0, slot:0, tr:[ 92, 123 ], tl:[ 21, 124 ] }
occupied:
  []
```

1) 시스템 정의

- a) 목적 및 기능 : 주변 주차칸이 인식한 후 목표주차칸을 결정
 - b) 배경 및 필요성 : 주차칸인식은 Matlab, 주차칸결정은 Python에서 수행이 편리
 - c) 시스템 운영개념 : 주변 주차칸들 중에서 목표주차칸과 가장 가까운 것의 좌표를 출력

2) 기능 요구

- a) 입력 1 : 주차선인식 → 주차선영상 (320x160 Numpy.array) 입력
 - b) 입력 2 : 주차칸인식 → 주변 주차칸 (양 입구점) 입력
 - c) 입력 3 : 서버 → 목표 (상대위치) 입력
 - d) 입력 4 : EMBEDDED →(localization + odometry) → /position ROS sub.
 - e) 주차칸결정 (python) : 목표위치과 로봇위치를 대조해 목표 주차칸 결정 & 마킹
 - f) 출력 : 목표주차칸영상 (320x160 Numpy.array) → 주차경로추정



주차경로결정 (Parking Path Decision)

1) 시스템 정의

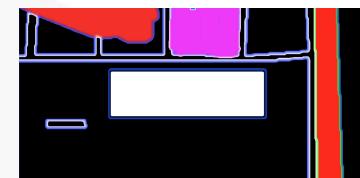
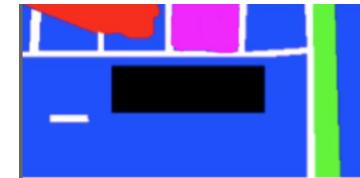
- a) 목적 및 기능 : 목표 주차칸에 후방으로 주차하는 경로 추정
- b) 배경 및 필요성 : 이차원 평면 상에 장애물을 피해 도착하는 게임을 푸는 것
- c) 시스템 운영개념 : 로봇의 환경 모방, 강화학습, 모델을 환경에서 풀이
- d) 운영 및 제약사항 : 현실의 파라미터 정규화 필요

2) 기능 요구사항

- a) 주차경로 훈련 + 시뮬레이터 : gym-gazebo
 - i) 환경 : 차선 or 배경, 시야범위 제한
 - ii) 보상 : 목표 주차칸과의 평균거리 반비례, 목표주차칸과 평행, 충돌 시 페널티
 - iii) 행동 : 조향각, 추진속도 (양/음), 페이즈 분리 (전진, 후진)
- b) 입력 : 목표주차칸영상 (320x160 Numpy.array) 입력
- c) 주차칸 추적 : 프레임간 트래킹, 목표영역 추가마킹 (connected component)
- d) 주차경로 추론 : 최적화된 파라미터에 의한 forward-pass
- e) 출력 : 조향각, 속도 (양/음)

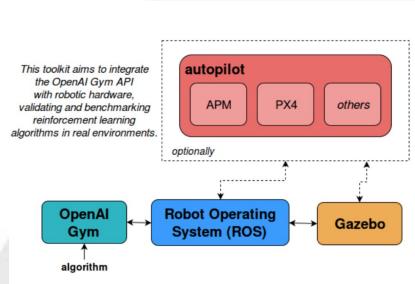
3) 비기능 요구사항

- a) 프레임워크 포팅 : gym-gazebo
- b) 오픈소스 : OpenAI Gym의 튜토리얼
 - i) duckie town (강화학습-로봇제어), deep-drive (3차원 주행)

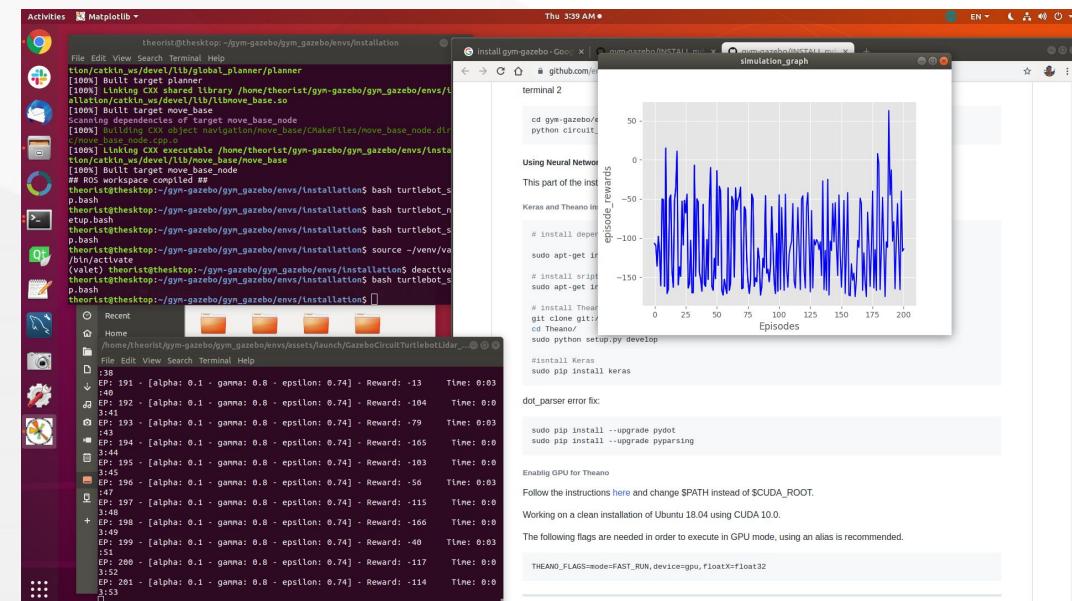


주차경로결정 (Parking Path Decision)

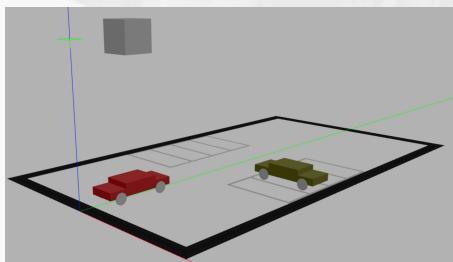
gym-gazebo 포팅



TurtleBot 훈련



Gazebo 시각화 & 응용중



주차장지도제작 (Parking Lot Mapping)

1) 시스템 정의

- a) 목적 및 기능 : 야외주차장의 상공사진에서 주차장의 구조를 추출
- b) 배경 및 필요성 : 로봇의 localization, 서버의 Visualization, 클라이언트의 UI에 주차장의 정적구조 필요
- c) 시스템 운영개념 : 시연주차장의 차선정보 구하기
- d) 운영 및 환경 제약사항 : 자동추출 한계, 동적 객체 출력 안함 (보행자, 차량)

2) 비기능 요구사항

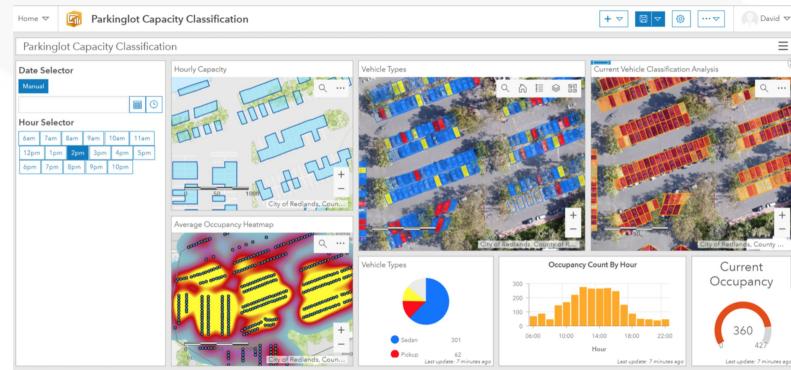
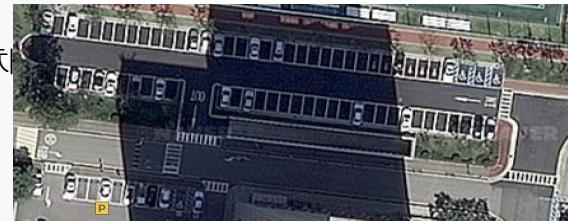
- a) 상공영상 확보 : 구글보다 네이버
- b) 지도 생성 : ArcGIS Free Trial을 이용한 차선따라 그리기
- c) GPS 좌표 확보 : RTK GPS 활용
- d) 주차칸 번호 할당 : JSON {slot {id, latitude, longitude}}
- e) 상대좌표 부여 : LT(0, 0), RB(1,1)로 상대좌표공간 생성
- f) 영상 생성 : 0 배경, 1 차선, 2 차량

3) ~~자동화된 주차장 구조 추출 (2016년 논문)~~

- a) ~~빌딩제거, 차량검출, 주차공간검출, grouping~~
- b) ~~낮은 검출율 : 차량 66%대, 주차공간 20~40%대~~

3. Localization : 위치

- 주차장 Mapping
- GPS + odometry
- 서버 Visualization : 차량 현재 위치 on the Map(google map with GPS)



9월	1주차	팀빌딩 (청소기로봇)
	2주차	계획수립 (발렛파킹로봇)
	3주차	요구사항 정의서, 테스트 시나리오
	4주차	시스템 설계 및 Pseudo Code
10월	a.	시스템 설계 및 Pseudo Code
	5주차	단위 시스템 개발 1 - 주차선 인식
	6주차	단위 시스템 개발 2 - 주차칸 인식
	7주차	단위 시스템 개발 3 (중간고사) - 객체인식
11월	8주차	단위 시스템 개발 4 - 주차장 Mapping
	9주차	단위 시스템 테스트
	10주차	시스템 통합 - 비전통합, 서버 Visualization
	11주차	시스템 통합 - ROS 통합, 조향각계산기
12월	12주차	시스템 통합 - 서비스통합
	13주차	테스트 및 보고서
	14주차	테스트 및 보고서
	15주차	테스트 및 보고서 (기말고사)

- 1) 단기 방향성
 - a) 카메라 인수 시
 - i) Camera Calibration
 - ii) AVM
 - b) 서버 인수 시
 - i) Object Detection
 - ii) Parking Lane Detection
 - iii) Parking Space Detection
- 2) 장기 방향성
 - a) 강화학습
 - i) gym-gazebo 미로탈출 포팅
 - ii) gym-gazebo 주차경로 포팅
 - b) 주차장 매핑
 - i) GeoAI를 이용한 툴 활용 시도

연구 개발 추진 일정 (14주)

1. 계획 수립 및 요구사항 정의서, 테스트 시나리오 : 1주
2. 시스템 설계 및 Sudoj Code : 2주
3. 단위 시스템 개발1 : 3주
4. 단위 시스템 개발2 : 2주
5. 단위 시스템 테스트 : 1주
6. 시스템 통합 : 3주
7. 통합 테스트 및 보고서 : 2주

소요

소요물

1. 차체
 - a. 유아용 전동차 x 1
2. 구조물
 - a. 스탠 봉 x 00
 - b. 설치 잡자재
3. 보드
 - a. NVIDIA Jetson Xavier x 1
 - b. SD Card 128Gb+ x 3
 - c. USB C-A Cable
4. 하드웨어
 - a. 12V 배터리, 충전기
 - b. 4-CH Camera HW interface
5. 개발환경
 - a. GPU 서버 → 외부공개 IP

1. 시스템소프트웨어
 - a. JetPack 4.2
2. 미들웨어
 - a. PIP
 - b. virtualenv
3. 프레임워크
 - a. ROS Melodic
 - b. OpenCV 3.4
 - c. PyTorch 1.1
 - d. TensorFlow 1.4
 - e. Matlab ?
4. 어플리케이션
 - a. libxcam_app
 - b. Camera driver for Xavier

연구 개발 추진 일정 (14주)

1. 계획 수립 및 요구사항 정의서, 테스트 사나리오 : 1주
2. 시스템 설계 및 Sudoc Code : 2주
3. 단위 시스템 개발1 : 3주
4. 단위 시스템 개발2 : 2주
5. 단위 시스템 테스트 : 1주
6. 시스템 통합 : 3주
7. 통합 테스트 및 보고서 : 2주