Valet Parking Robot 개발

개발 계획

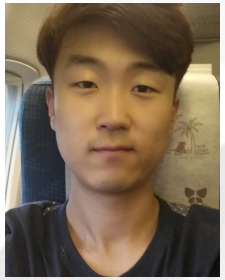
영상처리2팀

2019. 9. 20

ADAS ONE

목차

팀 소개



이종희

자동차부품사 선행개발연구원, 물리학과,
기계공학과

기구설계, 통계, 딥러닝, CAD

이홍인

소프트웨어학과

영상처리, 딥러닝, 강화학습, 임베디드

요구사항 분석

연구 개발 추진 담당 업무

1. HW 개발자 : 제어 및 주차
   - HW 개발 구축 및 플랫폼 개발
   - 제어 시스템 개발 및 차량 제어 알고리즘
   - 초음파 인식 및 통신
   - Driving Path + Birdeye view => 후진 주차 및 주행 제어

2. 시스템 SW 및 응용 SW : 통합
   - Linux SW platform
   - Lidar 인식
   - Calibration, Lidar + Vision => Fusion SW
   - 시스템 통합

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

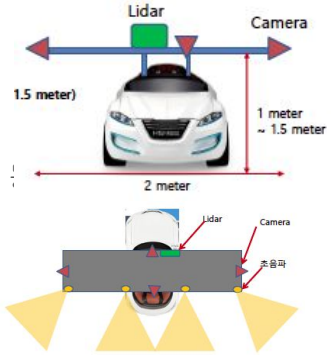
4. 영상처리 SW : 인식
   - Deep Learning 객체 인식 : 차량, 보행자, 자전거, 오토바이,
   - Deep Learning : Free space
   - 주차선 인식 SW : 주차선 코너 인식

	A	B	C	D	E	F	G	H
5	분류	모듈명	입력의 주체	입력자료형	출력대상	출력자료형	주	부
6			서버	주차슬롯번호			유혜리	박성훈
7		Path Planning	SLAM(장애물탐지	지도영상	내비게이션	경로 좌표들	유혜리	박성훈
8			Lidar 해석	Lidar 영상	서버 지도	지도영상	유혜리	박성훈
9			초음파 해석	초음파 영상	내비게이션	로봇 좌표	유혜리	박성훈
10			객체인식	1:사람 2:차	내비게이션	대기신호	유혜리	박성훈
11		SLAM(장애물탐지기)	객체인식	0: 배경	내비게이션	정지신호	유혜리	박성훈
12					조향모터 드라이브	조향각	유혜리	박성훈
13		내비게이션	Path Planning	경로 좌표들	추진모터 드라이브	속도	유혜리	박성훈
14			조향각 계산기	조향각			박성훈	유혜리
15			Odometry	이동거리			박성훈	유혜리
16			나침반 해석	방위각			박성훈	유혜리
17			GPS 해석	위도 경도			박성훈	유혜리
18	시스템	위치보정기	주차슬롯인식	0:배경 1:슬롯 2:차량	SLAM(장애물탐지	로봇 좌표	박성훈	유혜리
19			추진모터 인코더	전기신호	Odometry	속도	박성훈	이종희
20		속도 계산기	내비게이션	속도	모터 PWM	PWM값	박성훈	이종희
21		추진모터 드라이브	속도 계산기	PWM값	추진모터	전력	박성훈	이종희
22			조향모터 인코더	전기신호	위치보정기	조향각	박성훈	이종희
23		조향각 계산기	내비게이션	조향각	조향모터 드라이브	PWM값	박성훈	이종희
24		조향모터 드라이브	조향각 계산기	PWM값	조향모터	전력	박성훈	이종희
25		Lidar 해석	Lidar	Lidar 데이터	SLAM(장애물탐지	Lidar 영상	유혜리	박성훈
26		초음파 해석 (LIN)	초음파	초음파 데이터	SLAM(장애물탐지	초음파 영상	유혜리	박성훈
27					주차코너인식	탑뷰영상	이홍인	이종희
28					세그멘테이션	탑뷰영상	이홍인	이종희
29		AVM	카메라	전후좌우영상	강화학습 파킹패스	탑뷰영상	이홍인	이종희
30		GPS 해석	GPS	GPS 데이터	위치보정기	위도 경도	박성훈	유혜리
31		Odometry	속도 계산기	속도	위치보정기	이동거리	박성훈	유혜리
32	제어	Compass 해석	나침반	나침반 데이터	위치보정기	방위각	박성훈	유혜리
33			전방카메라	전방영상			이종희	이홍인
34		객체인식	후방카메라	후방영상	SLAM(장애물탐지	0:배경 1:사람 2:차량	이종희	이홍인
35		세그멘테이션	AVM	탑뷰영상	주차슬롯인식	0:배경 1:차선 2:차량	이종희	이홍인
36		주차코너인식	AVM	탑뷰영상	주차슬롯인식	0:배경 1:코너 2:차량	송장영	강인한
37			세그/주차코너인식	0:배경 1:코너	강화학습 파킹패스		송장영	강인한
38		주차슬롯인식	SLAM(장애물탐지	1:차선 2:차량 3: 사람	위치보정기	0:배경 1:차선 2:차량 3:슬롯	송장영	강인한
39	비전	강화학습 파킹패스	AVM	탑뷰영상	내비게이션	경로 좌표들	강인한	이홍인, 안도현
40		서버 지도	SLAM(장애물탐지	지도영상	주차현황	주차슬롯번호	이홍인	송장영
41	서버	주차 명령	어플	주차슬롯번호	Path Planning	주차슬롯번호	강인한	송장영
42		주차 요청	사용자	주차슬롯번호	서버	주차슬롯번호	강인한	송장영
43	클라이언트 어플	주차 현황 출력	서버 지도	지도슬롯번호	사용자	지도영상	강인한	송장영

질문 & 요구사항 분석

작업 분해 → 역할 분담

전동차 & 구조물



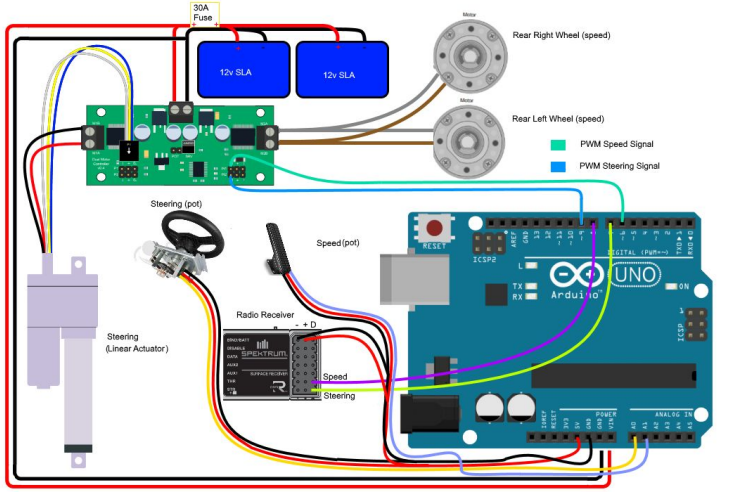
1) 시스템 정의
   a) 목적 및 기능 : 유아용 전동차 선정, HW를 장착할 구조물 제작
   b) 배경 및 필요성 : 발렛파킹로봇의 SW 모듈을 테스트할 프로토타입 제작 (제어, 센싱, 비전
   c) 시스템 운영개념 : 내장배터리로 전원공급되어 통신명령에 따라 주행제어되는 전동차
   d) 운영 및 환경 제약사항 : 저가형 배터리로 인해 충전 10시간/작동 1시간 예상, 모터 해상도
2) 기능 요구사항
   a) 주행 : 12V 듀얼 모터
   b) 조향 : 스테핑 모터 (추정)
   c) 제어기 : 제어기 정보 비공개, Arduino 보드 이용계획, Xavier-Arduino ROS 통신
   d) 모터 제어 : 임베디드 보드에서 제어해 계산된 경로를 주행, Arduino에서 모터 가동/방향전환/인코더 확인
   e) 모듈 지지 : 삼각대와 선반을 이용한 구조물을 설치, 카메라/라이더/센서류 위치 선정
   f) 서스펜션 : 카메라 진동 억제
3) 비기능 요구사항
   a) 전진/후진 추진속도 : 6~8km/h
   b) 배터리 : 예비배터리, 확장된 배터리로 교체 가능
   c) 크기 제원 : 전고 1~1.5m, 전폭 1.5~2m
   d) 허용하중 : 20kg+

전동차 제품별 비교 자료

제품명	제조사	가격	서스펜션 (카메라흔들림)	제어기	속도	통신	차폭 (구조물 안정성)	특징
BMW M6 GT3	대호토이즈	35만원	4바퀴	공개거부	2~6km/h	블루투스	69.1cm	동가격 대비 주행성능 좋음.(속도)
폭스바겐 뉴비틀	대호토이즈	33만원	있음	회신없음	3~5km/h	블루투스	71.6cm	동가격대비 차폭이 넓음.
BMW GT	JIAJIA/씨투엠뉴에너지	30만원	있음	회신없음	3~6km/h	블루투스	64cm	동가격 대비 주행성능 좋음.(속도)
벤츠 GLA 12V	태성토이즈	46만원	4바퀴	공개거부	2~6km/h	블루투스	70cm	고급 서스펜션
F8 sports	헤네스	68만원	가스식(고급)	ARM Cortex-M3 32bit	2~8km/h	블루투스	72cm	정밀 전자 조향기, 고급 서스펜션
T870	헤네스	96만원	가스식(고급)	ARM Cortex-M3 32bit	2~8km/h	블루투스	77.5cm	태블릿 PC 장착, 정밀 전자 조향기, 고급 서스펜션

전동차 & 구조물

내부에 자비어, 배터리팩 장착



1) 선반 1.9만원 : 가격을 고려하여 2m에 가깝게 연장할 수 있는 제품을 선택.
2) 삼각대 5300원 : 차체 높이와 합계가 1.5m에 가깝게 연장될 수 있는 제품 중 저렴한 것 선택.
3) 배터리팩 9.8만원 : 필요시 상위 배터리로 교체. 4시간 충전 1시간 사용. (기존 10 충전 1시간 운용)
4) 전장 : 모터관련 전장의 경우 작동을 확인함. 전동차도 유사하리라 예상.

가변 선반과 삼각대를 이용하여 베이스 제작.
테이프, 케이블타이, 조인트를 이용하여 모듈의 분해조립이 가능하도록 체결.

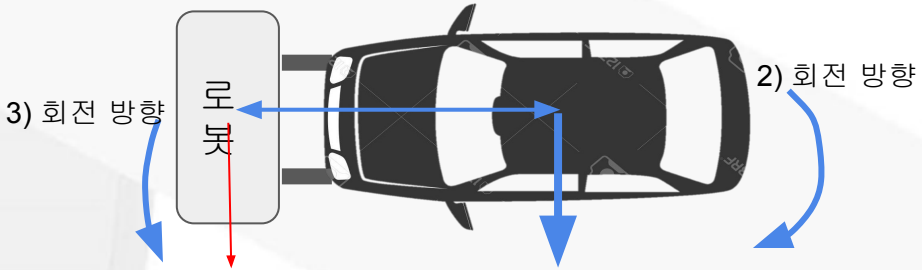
전동차 & 구조물

타사 로봇 구조 검토

1) 상기 로봇 구조는 360도 회전하는 바퀴 운용
2) 차량의 밑에서 주행하기 때문에 주변 정보를 파악하여 장애물을 회피하는데 제한이 있음
3) 장애물이 없는 실내 주차장에 적합

로봇의 자전 기능의 효용성 검토

1) 로봇 주행 토크 : 정지에서 3초에 걸쳐 8km/h까지 가속
추진력 = 질량X가속도 = 1900 kg X 0.75 m/s^2 = 1400 kgf
모터 토크로 환산 : 로봇 바퀴 지름이 0.3m일 경우.
1400 kgf X 0.15 m = 210 kgf·m
기어 및 바퀴 저항 보정치 40% 적용 -> 최소 294 kgf·m



2) 차량 도킹 후 자전시 회전속도
각가속도 = (토크-마찰) / 회전관성모멘트
= (294-160)kgf·m / (1600kg X (3m)^2)
= 0.009 rad/s^2 = 0.53 deg/s^2
90도 회전에 약 78초 소요.

3) 결론 : 로봇이 중심이아니고 차량을 중심으로 로봇이 회전할 경우 더 5배 더 빠르게 회전하고 안정성있음.

조향각&속도 계산기

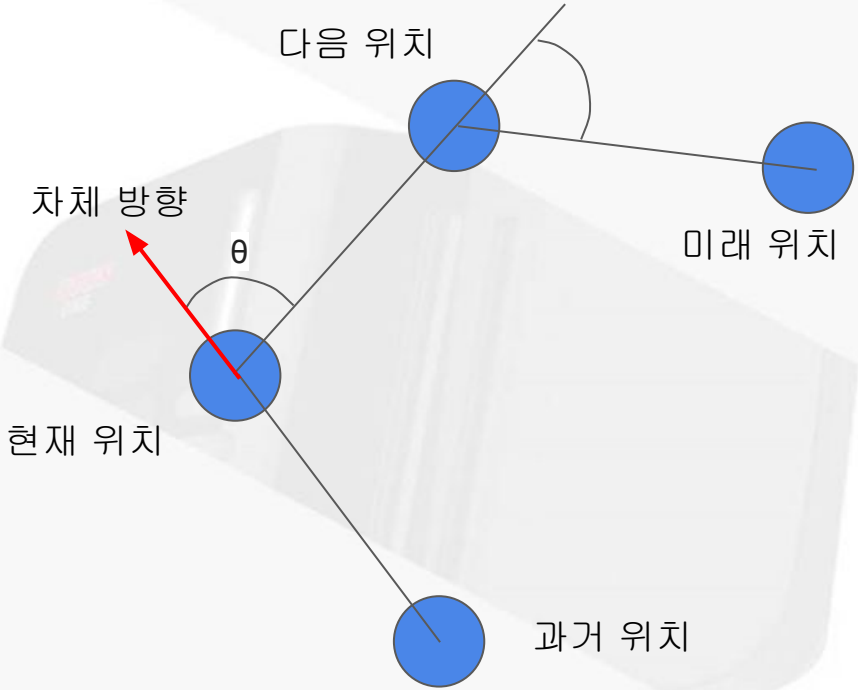
1) 시스템 정의
   a) 목적 및 기능 : Path Planning에서 계산된 경로를 따라 로봇이 실제 주행할 수 있는 조향모터와 주행모터 회전량을 계산
   b) 시스템 운영개념 : Path Planning 데이터를 받아 정보 계산 후 모터 제어기로 데이터 전송
   c) 운영 및 환경 제약사항 : 노면상태 등 환경 요인에 의한 오차 발생 가능성 있음.
2) 기능 요구사항
   a) 입력부 : MAIN SYSTEM의 Path Planning에서 (다음 위치 좌표, 현재 위치 좌표) 입력.
   b) 회전량계산부 : 다음 위치 좌표에 도달 하기 위한 θ를 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
```

조향각&속도 계산기

조향각 계산모델

Path Planning에서 찍어주는 포인트



1) 조향 모터 회전 각도
=A*B*(현재 바퀴 조향각 - θ)

A(모터 바퀴 회전비)
B(바퀴 조향각과 차체 회전각 비)

2) 주행중 속도 모터 제어

직진: θ = 0 -> 차량 속도 최대(8km/h)
회전: θ = MAX -> 구조물이 안정한 속도
(테스트 필요)

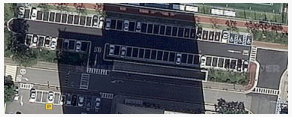
3) 자체의 조향 반응속도가 느리면 미래 위치를 미리 계산하여 조향 시간을 단축할 수 있다.

주차장 Mapping

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

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 등)

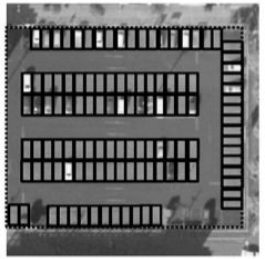
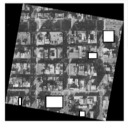
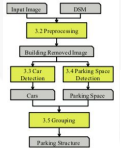





2) 비기능 요구사항
   a) 상공영상 확보 : 구글보다 네이버
   b) Hough 변환 : 직선성분이 강한 픽셀을 추출
   c) 엣지마스크 연산 : 직교방향에 변화량이 높은 픽셀을 추출
   d) Data annotation : 그림판 등에서 차선따라 그리기
   e) GPS 좌표 확보 : 네이버지도 API 이용해 확보, GPS센서값과 대조 필요
   f) 주차칸 번호 할당 : JSON {slot {id, latitude, longitude}}
   g) 상대좌표 부여 : LT(0, 0), RB(1,1)로 상대좌표공간 생성
   h) 영상 생성 : 0 배경, 1 차선, 2 차량

37.544028, 127.0769505

3) 자동화된 주차장 구조 추출 (2016년 논문)
   a) 빌딩제거, 차량검출, 주차공간검출, grouping
   b) 낮은 검출율 : 차량 66%대, 주차공간 20~40%대

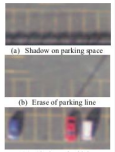
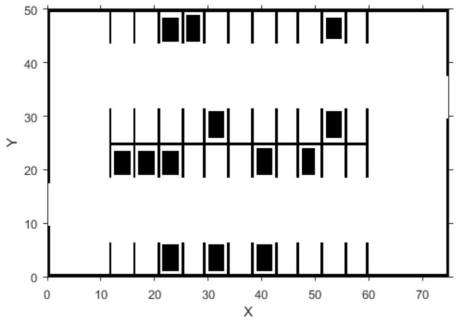


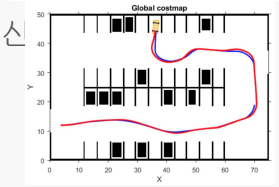
TABLE 2. Results of parking space detection

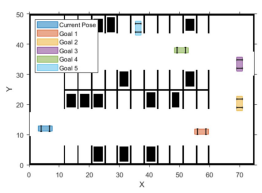
	#correct	#all space	detection [%]
Urban1	44	184	23.9
Urban2	24	90	26.7
Urban3	77	157	48.7
Urban4	17	104	16.3
Total	162	536	30.2

서버 Visualization

1) 시스템 정의
   a) 목적 및 기능 : 주차된 차량의 현황을 제시
   b) 초기 요구사항 : 차량 현재 위치 on the Map
   c) 시스템 운영개념 :
   d) 운영 및 환경 제약사항
2) 기능 요구사항
   a) 입력 : MAIN SYSTEM에서 로봇의 차량 주차완료 시그널 수신
   b) DB관리: 주차된 차량의 점유정보 갱신
   c) DB GUI : 서버단 차량 현재위치 출력
   d) 클라이언트 갱신 : 신규 주차정보 갱신

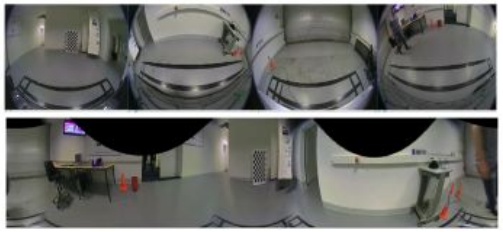


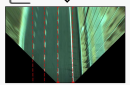


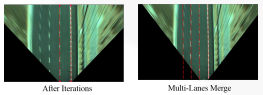


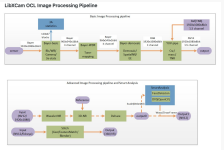
AVM

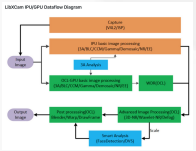
1) Around View Monitor 시스템 정의
   a) 목적 및 기능 : 차량의 360도 scene bird's-eye view로 제공
   b) 배경 및 필요성 : 주차칸인식과 주차선인식의 입력
   c) 시스템 운영개념 : AVM이 4개 카메라에서 1장의 주변버드뷰영상을 생성
   d) 시스템 아키텍쳐 : Xavier JP4.2에 Intel 기반 libxcam_app 포팅
2) 기능 요구사항
   a) 입력 : BGGR/fisheye 1920x1080x16bit x4
   b) Camera Calibration : 영상의 경계에 체스보드를 두어 인접한 카메라 쌍을 동시보정
   c) Image Stitching : 파노라마와 같은 surround view 생성
   d) Fisheye Distortion Correction : world의 bowl-view를 평탄화
   e) Inverse Perspective Mapping : 원근 제거 & 2D로의 매핑 → bird's-eye view
   f) 출력 : NV12 1280x720 영상
3) 비기능 요구사항
   a) 4 fish-eye camera & Xavier 인터페이스: 전동차의 전방, 좌측, 우측, 후방에 설치, 출력확인

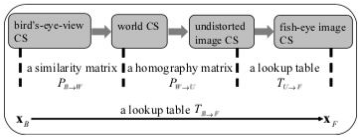










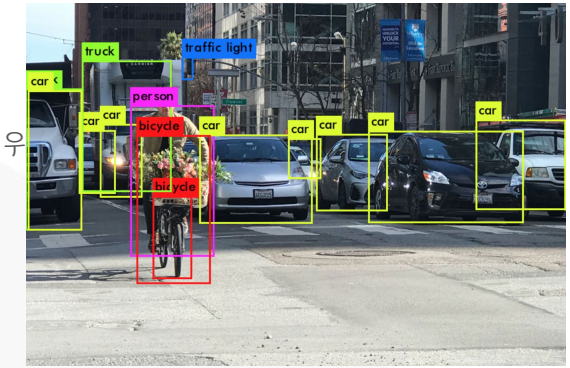


주차선인식

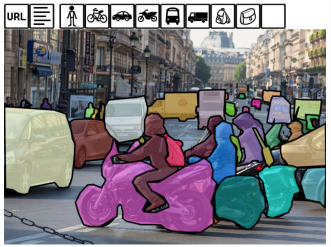
1) 시스템 정의
   a) 목적 및 기능 : 탑뷰영상에서 차선을 찾아냄
   b) 배경 및 필요성 : 자동주차경로계산에 카메라 영상을 추상화함으로써 노이즈를 제거하고 안정성 부여
   c) 초기요구사항 : 4-클래스라벨 (0 배경, 1 주차선, 2 차량, 3 장애물)
   d) 시스템 운영개념 : 딥뉴럴 시맨틱 세그멘테이션 모델 추론
   e) 운영 및 환경 제약사항 : Xavier에 libxcam_app 포팅

2) 기능 요구사항
   a) 주차선 훈련 : AVM (320x160, GT) 입력, 가중치 최적화
   b) 입력부 : libxcam_app (c++) 에서 python 모듈로 (1280x720 NV12) 입력
   c) 주차선 추론 : 최적화된 파라미터에 의한 forward-pass
   d) 출력부 : 주차칸인식기에게 주차선영상 (200x100 Numpy.array) 반환
3) 비기능 요구사항
   a) 계산가속 : CUDA, CuDNN, Half-precision, TensorRT
   b) 환경 구축 : PyTorch 1.1.0
   c) 감독학습 : AVM Dataset
   d) 모델 : DeepLabV3+ (Xception65)

객체인식

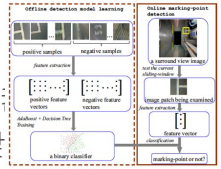


1) 시스템 정의
   a) 목적 및 기능 : 전진/후진 경로에 장애물을 식별
   b) 배경 및 필요성 : 주행방향에 나타난 장애물에 따라 다른 행동을 하기 위
   c) 초기요구사항 : 4-클래스 검출 (차량, 보행자, 자전거, 오토바이)
   d) 시스템 운영개념 : 신경망 기반 객체검출 추론
   e) 운영 및 환경 제약사항 : 후진시 후방 견인 차량으로 인한 영상 제한
2) 기능 요구사항
   a) 주차선 훈련 : COCO에서 (640x480, JSON) 입력, 가중치 최적화
   b) 입력부 : /dev/video에서 python 모듈로 스트리밍, gstreamer vs uvccapture
   c) 주차선 추론 : 최적화된 파라미터에 의한 forward-pass
   d) 출력부 : 주차칸인식기에게 (python list of bounding boxes & class predictions) 반환
3) 비기능 요구사항
   a) 계산가속 : CUDA, CuDNN, Half-precision, TensorRT
   b) 환경 구축 : PyTorch 0.4.1
   c) 감독학습 : COCO Dataset (car, truck, bus, motorcycle, bicycle)
   d) 모델 : CenterNet, YOLOv3



주차칸인식

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 주차
   c) 시스템 운영개념 : 주차코너를 찾고, 그 칸의 입구선을 찾고, 입구점을 분석해, 주차칸을 찾
   d) 운영 및 환경 제약사항 : Xavier에 libxcam_app 포팅

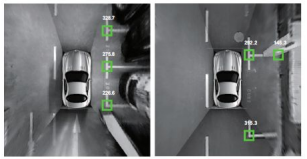
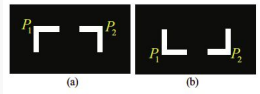
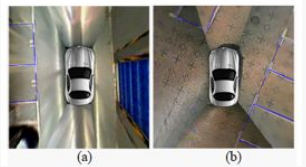


Figure 10. Marking-point detection results on two typical surround-view images.

2) 기능 요구사항
   a) 입력 : libxcam_app (C++) → 카메라 NV12영상 (1280x720 NV12) 입력
   b) 주변 주차칸 추론 (Matlab) : 입력받은 영상에 있는 주차칸
      i) 주차코너 추론부 : 특징 & 앙상블 모델, 이진영상 (0 : 코너아님 1: 코너)
      ii) 주차칸 추론부 : 입구선 분석 → 주변 주차칸 추론 (양 입구점 & 기울기)
   c) 출력 : 주변 주차칸 (양 입구점, 기울기) → 주차칸 결정부 (Python)

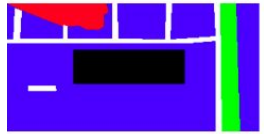


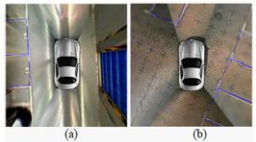
3) 비기능 요구사항
   a) 환경 구축 : Matlab
   b) 오픈소스 포팅 : Parking Slot Detection L Algo., Tongji Parking Slot Dataset (600x600)
   c) 처리율 : 20~25 FPS (2.4GHZ Intel i5, 4G RAM)



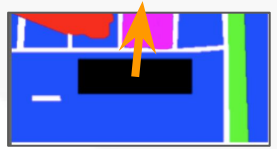
주차칸결정

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 목표GPS와 가장 가까운 접한 것을 목표로 설정
2) 기능 요구사항
   a) 입력 1 : 주차선인식 → 주차선영상 (200x100 Numpy.array) 입력
   b) 입력 2 : 주차칸인식(Matlab) → 주변 주차칸 (양 입구점, 기울기) 입력
   c) 입력 3 : 서버 → 목표GPS (위도, 경도) 입력
   d) 입력 4 : EMBEDDED → /gps (위도, 경도) ROS sub.
   e) 주차칸결정 (python) : 목표GPS과 로봇GPS를 대조해 목표 주차칸 결정 & 마킹
   f) 출력 : 목표주차칸영상 (200x100 Numpy.array) → 주차경로추정



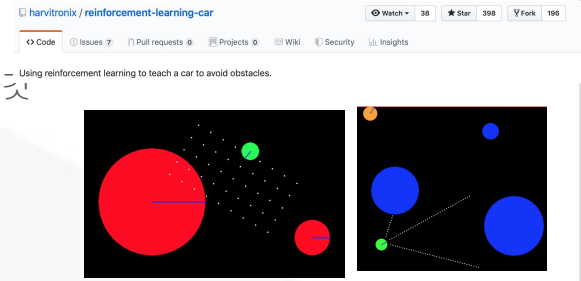


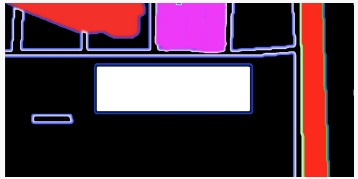
37.544028, 127.0769505
37.5439784, 127.0769508

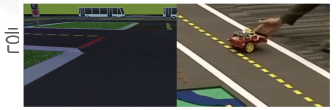


주차경로추정

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 : 강화학습 적용
      i) 환경 : 같은 클래스 라벨, 유사한 환경 구성, 시야범위 제한
      ii) 보상 : 목표 주차칸과의 중심거리 반비례, 목표주차칸과 평행, 충돌 시 페널티
      iii) 행동 : 조향각, 추진속도 (양/음), 페이즈 분리 (전진, 후진)
   b) 입력 : 목표주차칸영상 (200x100 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 구축 : OpenAI Gym, Gym-gazebo, Keras
   b) 오픈소스 : harvitronix, 2D sim., Keras, 원형장애물, 전부 프리스페이스, 초음파센서 구현
      i) duckie town (강화학습-로봇제어), deep-drive (3차원 주행)

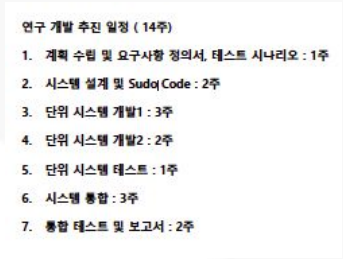






개발 자원

9월	1주차	팀빌딩 (청소기로봇)
	2주차	계획수립 (발렛파킹로봇)
	3주차	요구사항 정의서, 테스트 시나리오
	4주차	시스템 설계 및 Pseudo Code
10월	5주차	시스템 설계 및 Pseudo Code
	6주차	단위 시스템 개발 1 - 주차장 Mapping, 서버 Visualization, 주차선 인식
	7주차	단위 시스템 개발 2
	8주차	**단위 시스템 개발 3 (중간고사) - 주차칸 인식**
	9주차	단위 시스템 개발 4 -객체인식,주차경로추정
11월	10주차	단위 시스템 테스트
	11주차	시스템 통합 - 비전통합
	12주차	시스템 통합 - ROS 통합
	13주차	시스템 통합 - 서비스통합
12월	14주차	테스트 및 보고서
	15주차	테스트 및 보고서
	16주차	**테스트 및 보고서 (기말고사)**



소요

소요물

1. 차체
   a. 유아용 전동차 x 1
2. 구조물
   a. 신축 선반 x 1
   b. 삼각대 x 1
   c. 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 : AVM
   c. GPS : Localization
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 1.2?
   e. Matlab ?
4. 어플리케이션
   a. libxcam_app
   b. SurDes Camera Porting