

Project 제안서

Lane Keeper with OPENCV



- 작성자 : 27기 180이기진
- 작성일 : 2021. 09. 17

목 차

1. Project 소개

- (1) 선정 동기
- (2) 목표

2. Project 내용

- (1) Hardware Architecture
- (2) System Architecture 및 회로도
- (3) 목표 구현 내용
- (4) Project 시나리오

3. Project 진행 일정

4. 용어 정리

5. 참고 자료

1. Project 소개

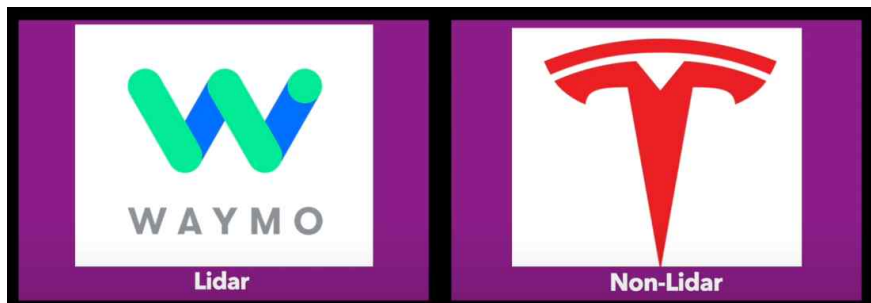
(1) 선정 동기

교통수단은 인류의 역사가 비약적으로 발전하는데 큰 역할을 하였다. 자동차가 있었기에 우리는 먼 곳까지 이동할 수 있었고, 도시, 국가간의 교류 및 상호발전이 가능할 수 있었다. 하지만 인간은 항상 완벽한 존재가 아니었기에, 많은 교통사고가 있었고, 그에 따른 많은 희생을 감수해야만 했다. 이러한 실수를 조금 더 줄이고, 인간의 편의성을 증진하고자 프로그램이 이동수단을 스스로 운전하는 자율주행 기술의 필요성이 대두되었다.



2015~2020 연도별 사고건수, 사망자, 부상자 수 (출처-도로교통공단)

자율주행을 구현하는 방식에는 크게 2가지가 있는데, ‘라이다’를 사용하는가의 여부가 그 2가지이다. 라이다(Lidar)는 자율주행에 필요한 센서(카메라, 레이더, 라이다)중 한가지이며, 레이저를 이용하여 물체로부터 반사된 거리를 계산해서 선명한 3d 이미지를 생성해준다. 하지만 이 센서는 가격이 상당히 비싸 대량 생산이 어렵다는 단점이 있다. 그리고, 라이다 기반의 자율주행은 미리 만들어 놓은 초정밀 지도를 달리며 라이다로 GPS를 잡으며 딥러닝 알고리즘을 통해 예정된 경로를 달리는 식이다. 즉, 초정밀 지도가 없는 부분은 자율주행이 거의 불가하며 Edge case (빙판길 제어)에 대한 대처가 잘 이루어질 수 없다. 따라서, 가장 효율적인 방법은 초정밀지도와 라이다를 이용한 것이 아닌, 카메라 센서와 좋은 알고리즘이라고 할 수 있다.



이에 착안하여 나는 차선을 벗어나지 않으며, 노상 물체의 등장에 따라 유연하게 대처하는 자율주행 자동차를 만들어 보고자 하였다.

[2] 목표

가. 차선 인식 및 지키기

- OpenCV로 양쪽 차선을 인식, 좌표를 추출하여 곡률 및 이탈정보 계산
- 감지한 차선에 맞춰서 조향 및 주행

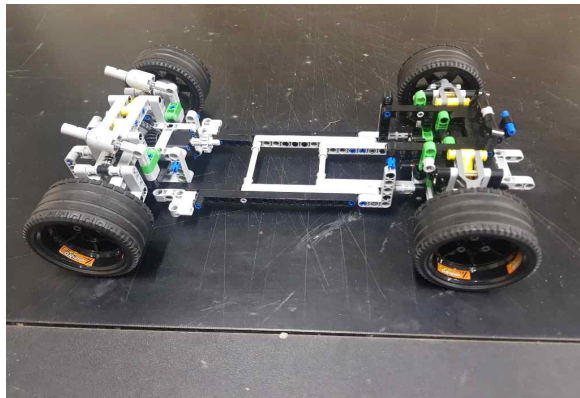
나. 전방에 갑자기 물체 출현시 정지

- 초음파 센서로 물체와의 거리를 측정해 돌발 출현하는 물체를 감지 및 차량 정지

2. Project 내용

[1] Hardware Architecture

자동차의 프레임 및 기어부품은 1학년 RC카를 만들 때 사용했던 레고 자동차 42039의 프레임을 활용한다. 위에 레고 부품을 추가하여 웹캠과 초음파 센서, Cortex M4와 RaspberryPi를 장착할 공간을 만든다. 조향은 서보모터로, 가감속은 dc 모터로 진행할 것이다.



※모터 선정기준

로봇 중량 W: 1kg

바퀴 지름 D: 6.8cm

모터의 등속 회전 속도 f: 3.18turn/s

가속 구간 t: 1s

부하 관성 모멘트 : $J = 1 * 6.8 * 6.8 / 8 = 5.78 \text{ [kg cm}^2\text{]}$

가속 토크 : $T_a = 5.78 / 980 * 2 * 3.14 * 3.18 / 0.5 = 0.24 \text{ [kg cm]}$

등속 토크 : $T_m = 0.1 * 1 * 6.8 / 4 = 0.17 \text{ [kg cm]}$

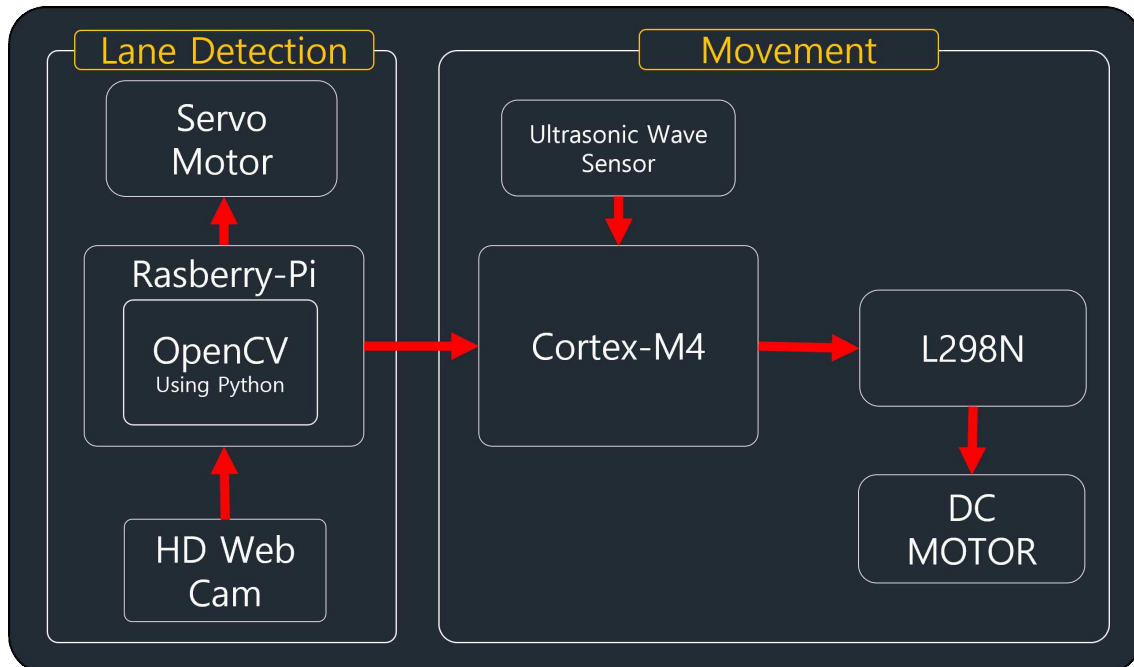
가속시 필요 토크 = $T_a + T_m = 0.24 + 0.17 = 0.41 \text{ [kg cm]}$

등속시 필요 토크 = $T_m = 0.17 \text{ [kg cm]}$

모터의 토크는 최소 0.41 kgcm 이상으로 선정

분류	부품	모델명	사진	스펙
모터	DC Motor	Lego 8883		모터 전압: 9V 토크: 0.4078 kg-cm RPM: 380(No Load)
	Servo Motor	HS-311		모터 전압: 4.8~6V 속도: 0.17sec/60deg 토크: 3~3.7kg/cm
	Motor Driver	L298		모터 전압: 5V
센서	Ultrasonic Sensor	HC-SR04 x3		작동 전압: 5V 감지 범위: 2cm~3m
	Camera	IRC70M		이미지센서: FHD1080p 영상 화소: 200만화소 프레임: 30fps
배터리	Battery	삼성 EB-U1200		용량: 10000mAh 출력: 9V->1.67A 12V- >1.25A
	Usb Tester	알리바바		변경가능 전압, 전류: 3-30V, 0-5A

[2] System Architecture 및 회로도



[3] 목표 구현 내용

1) 차선 인식(Lane Detection) 알고리즘

라즈베리파이에 웹캠을 연결하여 실시간 영상을 받아온다. 라즈베리파이의 성능 때문에, 받아온 영상의 프레임을 낮춰서 차선을 검출한다.

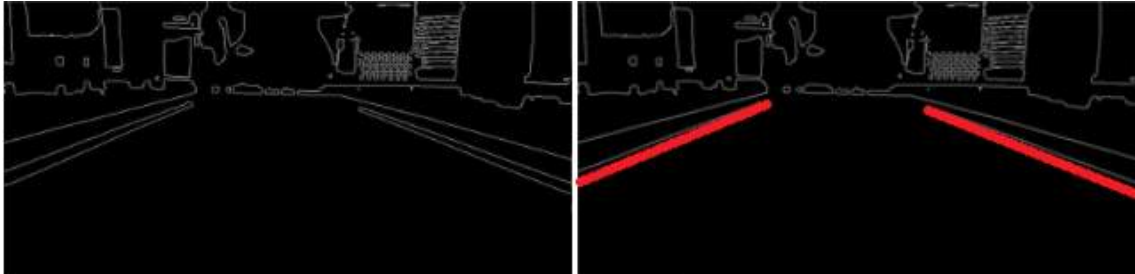
바닥에 양쪽 차선의 역할을 하는 검은 테이프를 붙여서 서킷을 제작하고, Python의 OpenCV를 통하여 차선을 검출한다. 순서는 다음과 같다.



- ① 원본 영상을 실시간으로 받아와, 영상의 특정 하단 부분만 분리 후, GrayScale로 변환
- ② 흑백으로 변환된 영상의 노이즈를 Gaussian Blur를 통해 제거
- ③ Canny 함수를 사용하여 변환된 영상의 엣지를 검출
Opencv의 Canny 함수를 사용하여 위의 사진과 같이 차선의 엣지를 검출해낸다.

④ hough 변환을 통해 양 차선의 중심을 구함

위의 Canny Edge Detection을 통해 구해진 엣지들 중에서 차선만을 골라내기 위하여 HoughLines() 함수를 사용한다. 이렇게 차선에 해당하는 직선을 검출하여 빨간색으로 표시하게 되면 아래와 같은 영상을 결과물로 얻을 수 있다.



이후, Hough Transform을 이용하여 빨간색으로 표시된 차선의 중심 직선을 구해서 영상의 가운데에 표시한다. 만약 굴곡이 심한 커브에서 한쪽 차선만 감지될 경우, 그 차선의 끝점과 영상의 반대쪽 끝에 y값이 같은 점을 잡고, 그 중심점을 끝으로 중심 직선을 구한다. 이 중심 직선의 끝점을 계속 업데이트하며 아래에 이어지는 알고리즘을 통해 매 순간마다의 조향각을 구한다.

2) 움직임(Movement) 알고리즘

1. 차선 인식에 따른 조향 및 구동

① 전륜 조향각

차량의 횡방향 제어 알고리즘은 상위 제어와 하위 제어로 나뉜다.

상위 제어기는 차량 속도, 위치 및 이동하고자 하는 미래 경로에 기반하여 목표 조향각을 결정한다. 하위 제어기는 장치의 조향을 담당한다. 상위 제어에는 Pure Pursuit 방법을 사용하고자 한다.

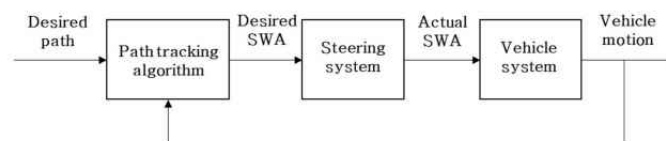


Fig. 1 Block diagram of lateral control structure including vehicle system

Hough Transform을 통해 얻은 차선 중심 끝 좌표를 $g(g_x, g_y)$ 라 하고, 이 g점을 계속 업데이트시키면서 그 값을 Pure Pursuit 알고리즘에 적용시킨다. 이를 통해 앞바퀴의 조향각을 연속적으로 얻을 수 있다.

② 서보모터 제어

원하는 조향각만큼 서보모터를 돌린다. 서보모터는 라즈베리파이에 바로 연결되어 있으므로, 영상처리 부분과 동일 코드 내에서 조향한다. 차의 조향축은 레크와(렉과) 피니언 기어로 이루어져 있다.

부품으로 사용할 피니언 기어의 pitch circle 반지름은 $r=0.6\text{cm}$ 이다.



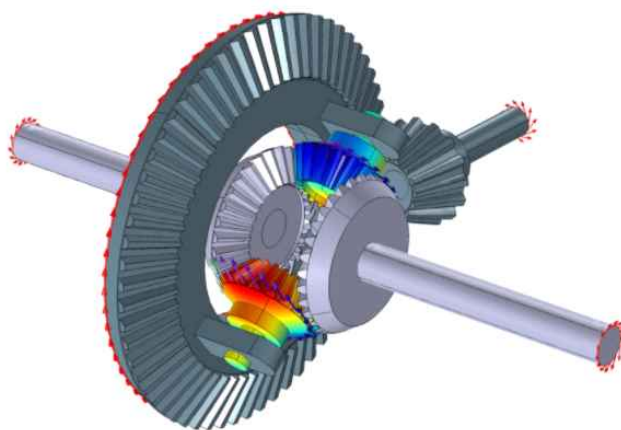
목표 조향각을 θ_1 , 피니언 기어의 회전 각을 θ_2 라고 하고(시초선과 각의 양의 방향은 위의 사진처럼 잡는다.), 레크 기어 력의 길이와 피니언 기어의 반지름을 고려해서 θ_1 과 θ_2 의 관계식을 구하면, 그 관계식은 아래와 같이 나온다.

$$\theta_2 = \frac{1152}{\pi} * \arctan(\theta_1)$$

이 관계식을 가지고 pwm 신호의 펄스 폭(duty)를 조절하여 원하는 각도로 서보 모터를 조절한다.

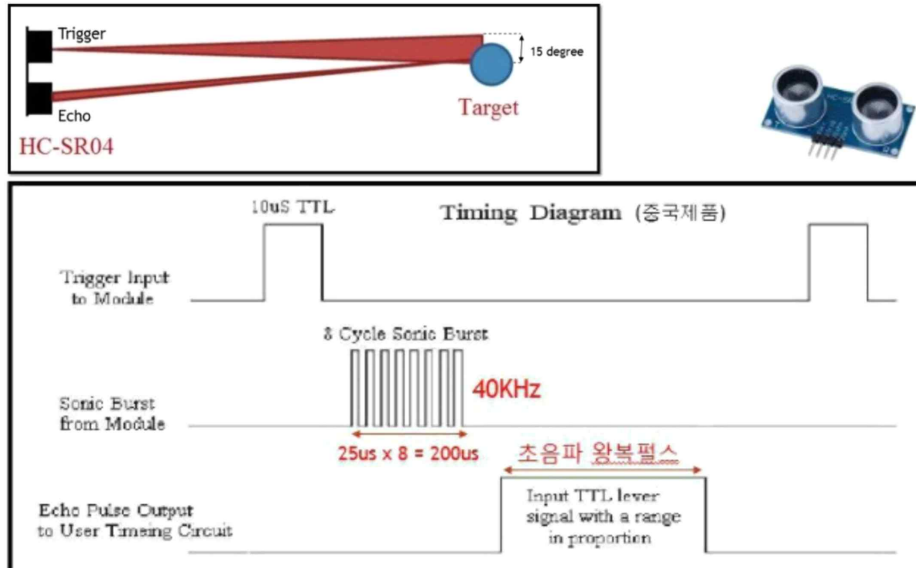
③ 후륜동력 전달을 위한 모터 구동

코어텍스에서 모터드라이버를 통해 일정한 속도로 dc모터를 돌린다. dc모터는 후륜축 동력 배분장치의 링 기어와 연결되어, 2개의 뒷바퀴를 돌린다. 차의 조향 시, 뒷바퀴는 양쪽이 각각 다른 속도로 돌아야 하므로, 이때 차동 기어[differential gear]를 사용하여 이를 가능케 한다. 차동기어는 레고의 부품을 사용할 것이다.



2. 물체 출현시 급정지

전후방에 위치한 초음파 센서를 통해 차와 물체사이의 거리를 측정한다. 사용할 HC-SR04센서는 전방 2~400cm의 15도까지 정밀 측정이 가능하다.



Timing Diagram을 보면, 우선 모듈의 Trigger Input에 10µs의 하이펄스를 입력하면, 40kHz의 8개 초음파 버스트가 발생한다. 송신부인 EC0는 초음파 발신 직후 HIGH가 되고, 반향을 감지하면 LOW가 된다. 따라서 센서로부터 물체와의 거리는 다음과 같이 측정된다.

$$\text{거리} = \text{EC0 HIGH PULSE 시간(왕복시간)} \times \text{소리의 속도}(340\text{m/s})/2$$

실제 측정에서는 차량의 10~20cm정도 전방에서 갑자기 센서값이 변한다면, 코어텍스로 신호를 전송하여 차량을 한시적으로 정지시키고자 한다.

[4] Project 시나리오

1. 정해진 트랙 위에 자동차를 놓는다.
2. 양쪽 차선을 감지하여 벗어나지 않고 자동주행을 한다.
3. 노상에 물체가 갑자기 출현하면 급정지한다.

3. Project 진행 일정


진행 목표	9월		10월					11월		
	4	5	1	2	3	4	5	1	2	3
라즈베리파이 환경 구축				중 간 고 사						
재료 구매 및 하드웨어 제작										
OpenCV로 차선 인식 코딩										
모터와 서보모터 제어 코딩										
차선 인식 알고리즘과 제어 코드 연동										
테스트 및 디버깅, 코드 개선										

4. 용어 정리


1. Sobel Edge Detection

Sobel 엣지 검출은 1차 미분을 기반으로 한 유명한 엣지 검출 방법이다. 정확히는 3x3크기의 행렬을 사용하여 연산을 하였을때 중심을 기준으로 각방향의 앞뒤의 값을 비교하여서 변화량을 검출하는 알고리즘이다. Sobel 엣지 검출을 통해서 우리는 수직, 수평, 대각 엣지를 각각 검출할 수 있다.


-1	0	1	-1	-2	-1	0	1	2	-2	-1	0
-2	0	2	0	0	0	-1	0	1	-1	0	1
-1	0	1	1	2	1	-2	-1	0	0	1	2




수직 엣지 검출



수평 엣지 검출



대각 엣지 검출1



대각 엣지 검출2

2. Canny Edge Detection

말 그대로 테두리를 찾아주는 알고리즘이다. 총 4단계의 알고리즘으로 구성되어 있다.

1단계: Noise Reduction

가우시안 필터를 이용하여 이미지 노이즈를 감소시킨다.

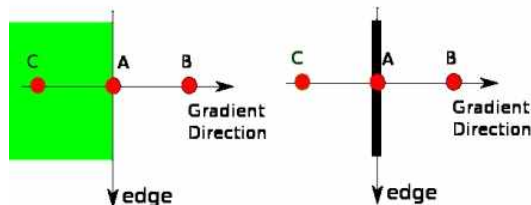
2단계: Edge Gradient Detection

Sobel Kernel을 수직, 수평 방향으로 적용하여 각 방향의 Gradient를 획득한다. 수평방향 기울기를 Gx , 수직방향 기울기를 Gy 라고 하면, (x,y) 픽셀에서 edge gradient는 다음 식으로 구할 수 있다. Gradient 방향은 edge와 수직인 방향이다.

$$Edge\ Gradient\ (G) = \sqrt{Gx^2 + Gy^2}$$
$$\angle\ (\theta) = \tan^{-1}\left(\frac{Gy}{Gx}\right)$$

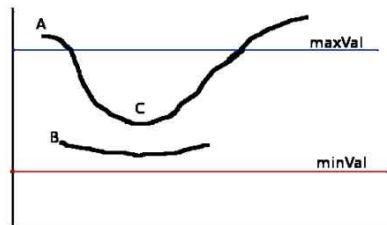
3단계: Non-maximum Supression

Edge에 기여하지 않은 픽셀을 제거하기 위해 이미지를 스캔한다. 이미지를 스캔하는 동안, gradient 방향으로 스캔구역에서 gradient가 최대인 픽셀을 찾아낸다. 밑의 그림에서, A는 edge위에, B, C는 gradient 방향 위에 놓여있다. A지점에서의 gradient 값이 B, C보다 큰지 확인한 후, 값이 크다면 그냥 다음 단계로, 그렇지 않다면 값을 0으로 만든다.



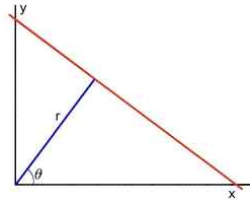
4단계: Hyteresis Thresholding

3단계를 거친 것들이 실제 Edge가 맞는지 판단한다. 최대, 최소의 문턱값을 설정하여, 최대 문턱값을 넘기면 edge, 최소 문턱값을 못넘기면 edge가 아니고, 사이에 있는 값들은 픽셀연결구조를 보고 edge여부를 판단함.



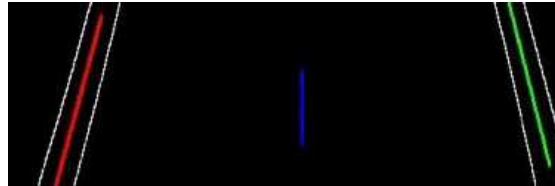
3. Hough Transform

Hough Transform(허프 변환)은 직선을 찾는 알고리즘이다. 허프 변환은 직선의 방정식을 활용하는데, 극 좌표계를 이용한 삼각함수로 표현하며, 식은 다음과 같다.



$$r = x \cos \theta + y \sin \theta$$

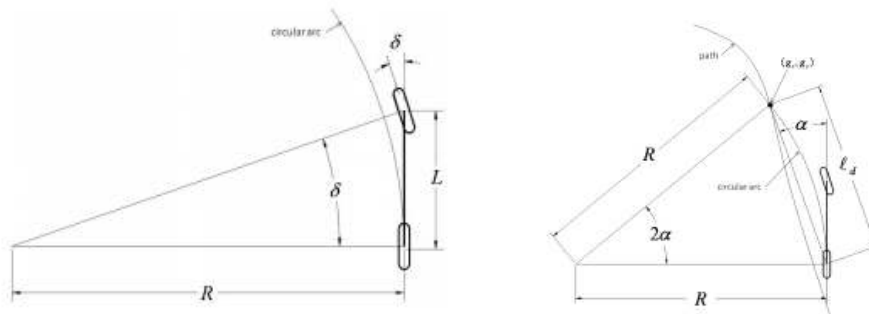
두 점은 $[x, y, x+w, y]$ 의 정보를 포함하는 리스트로 나타난다. 이때 x 값이 영상의 x 축 중심값보다 작다면 좌측 차선, $x+w$ 값이 영상의 x 축 중심값보다 크다면 우측 차선으로 분류한다. 좌측 차선의 왼쪽 점, 우측 차선의 오른쪽 점 x 값들의 평균을 각각 구하고 이 두 평균값을 합하여 2로 나누면 차선의 중심값을 구할 수 있다.



4. Pure Pursuit

Pure Pursuit는 차량의 현재 위치에서 도달하고자 하는 경로상의 지점을 순간순간 업데이트하여 추종해야하는 전륜 조향 각도를 계산하는 방법이다.

Pure Pursuit의 원리는 아래와 같이 나타낼 수 있다.



좌측의 그림은 ackerman의 steering bicycle model이고, 오른쪽이 pure pursuit의 기하 모델이다.

여기서 목표 지점의 좌표를 $g(g_x, g_y)$ 로 하면 차량 뒷바퀴의 중심점으로부터 점g까지의 거리를 ld (look ahead distance)라고 나타낸다. 이 거리와 각 식을 계산하면, 회전 반경 및 곡률은 아래와 같이 계산된다. $\delta(t)$ 는 곧 목표 지점으로 가기 위한 전륜 조향각이다.

$$\begin{aligned}\frac{\ell_d}{\sin(2\alpha)} &= \frac{R}{\sin\left(\frac{\pi}{2} - \alpha\right)} & \kappa &= \frac{2\sin(\alpha)}{\ell_d} & \kappa &= \frac{2}{\ell_d^2} e_{\ell_d}, \\ \frac{\ell_d}{2\sin(\alpha)\cos(\alpha)} &= \frac{R}{\cos(\alpha)} & \delta(t) &= \tan^{-1}\left(\frac{2L\sin(\alpha)}{kv_x(t)}\right) \\ \frac{\ell_d}{\sin(\alpha)} &= 2R\end{aligned}$$

5. 참고 자료

참고 링크

라이다 관련 자료

<https://www.youtube.com/watch?v=DuEaBRLgJ2M>

레고 모터 제어

<https://www.hackster.io/Notthemarsian/take-control-over-lego-power-functions-ee0bfa>

Canny Edge Detection

<https://m.blog.naver.com/samsjang/220507996391>

https://docs.opencv.org/3.4/da/d22/tutorial_py_canny.html

Hough Transform

<https://diy-project.tistory.com/115>

https://docs.opencv.org/3.4.0/d9/db0/tutorial_hough_lines.html

서보모터 제어

<https://blog.naver.com/chandong83/220858569356>

<https://m.blog.naver.com/chandong83/221850060160>

참고 논문

Jarrod M.Snyder, Automatic Steering Methods for Autonomous Automobile Path Tracking, 2009, 78p

R.Craig Coulter, Implementation of the Pure Pursuit Path Tracking Algorithm, 1992, 15p

김여정, 정윤서, 황소영, 라즈베리 파이와 아두이노 및 OpenCV를 활용한 공유형 자율주행차 모델 설계, 2020, 한국정보통신학회 여성 ICT 학술대회 논문집, 3p

김현식, 장재영, 김찬수, 전중남, PID 제어를 이용한 자율주행자동차의 차선 추적, 2019, 2019년 추계학술발표대회 논문집, 4p

김창희, 이동필, 이경수, 자율주행 경로 추종 성능 개선을 위한 차량 조향 시스템 특성 분석, 2020, 6p