자율주행 자동차

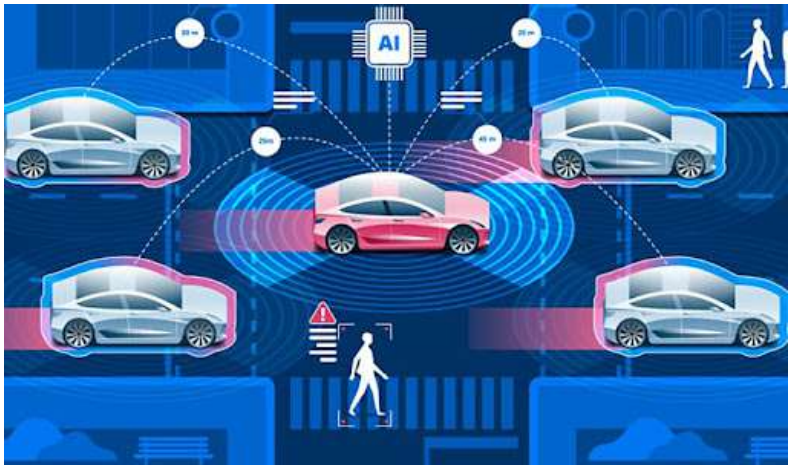
과목 : 영상처리시스템

담당교수님 : 박희재 교수님

17100052 김동중

17100179 이진우

개요

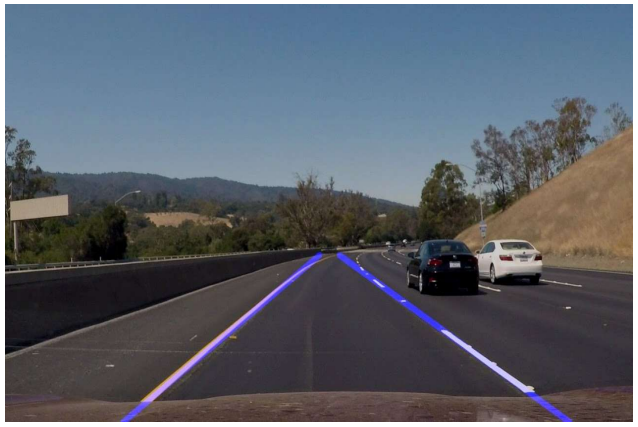


현대 사회에서 자율주행 기술은 단순히 '나 대신 운전해주는 것'이라 보기엔 너무나 많은 영역과 분야에서 사용되고 발전되고 있습니다.

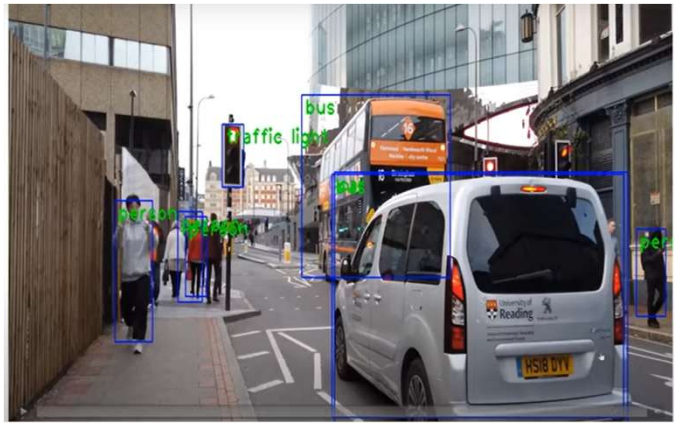
자율주행자동차가 활성화 된다면 자동차는 소유가 아닌 공유의 개념이 형성 될 것이고, 운전하는 시간을 새로 사용하면서 전혀 다른 라이프스타일을 만들어줄 것입니다.
우리는 이 시대의 흐름을 바꿀 자율주행 자동차가 어떠한 원리인지 알아보고자 하였습니다.

시스템 설명

차선 인식



신호등 인식 및 물체종류구별



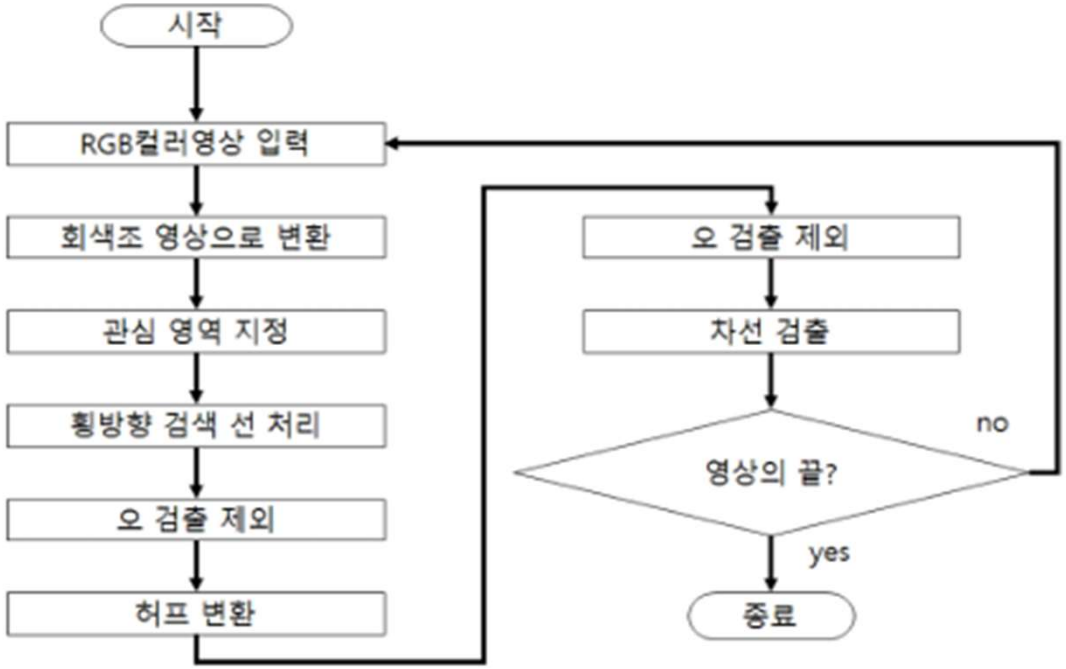
자율주행 영상처리 기법에는 무엇이 들어갔을까요?

먼저 차선을 인지할 수 있어야 합니다. 차량이 양 차선 사이에서 올바르게 제어할 수 있어야 합니다.

둘째, 주변의 사물을 인지할 수 있어야 합니다. 인식 대상이 사람인지 자동차인지 신호등인지에 따라 다른 판단을 내려야 하기 때문에 사물을 정확히 인식하는 것은 중요합니다.

다행히도, 이러한 기법들에 대한 이론들은 사람들이 많이 연구해 왔습니다.

사용 이론 및 Open CV 메서드



차선 인식의 알고리즘으로 Hough Transform을 사용하였습니다.

허프 변환은 이미지에서 선과 원 같은 단순한 형태를 찾는 방식입니다. 이진화 된 이미지에서 직선을 찾을 때 상대적으로 빠른 검출을 보여줍니다.

사용 이론 및 Open CV 메서드

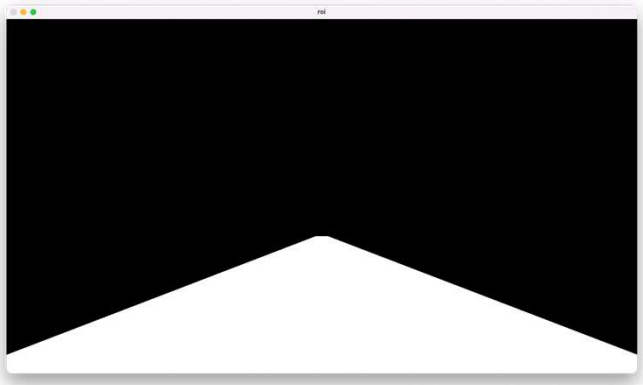
img

Grayscale + Gaussblur

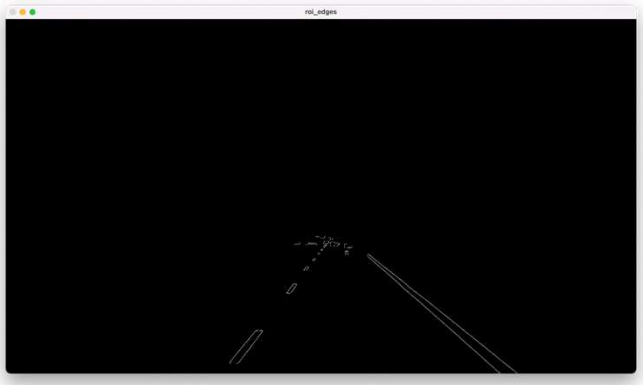
저희는 우선 OpenCV에서 제공하는 메서드를 통해 차선인식을 실시해보았습니다. 가장 먼저 동영상을 불러와 캡쳐를 실시하고 캡쳐된 이미지를 불러왔습니다. 그 후 이미지를 gray scale로 변환해준 후 노이즈를 제거하기 위해서 gaussblur를 적용시켰습니다.

사용 이론 및 Open CV 메서드

ROI



ROI + canny-edge detector



다음으로 모서리 감지를 위해 canny-edge detector을 적용시킨 후, 미리 관심 영역을 지정하여 둔 roi와 bitwise and 연산을 실시하였고 마지막으로 Houghline을 사용하여 직선들을 감지해내었습니다. 이때 감지된 직선들을 단순히 이미지에 표시하게 되면 직선들이 중간에 끊기는 현상이 발생하여 깔끔하게 차선이 표시되지 않았기 때문에 houghline에서 반환 받은 직선의 양 끝점 좌표를 회귀분석하여 차선 하나당 하나의 근사적인 직선을 긋게 하였습니다.

사용 이론 및 Open CV 메서드

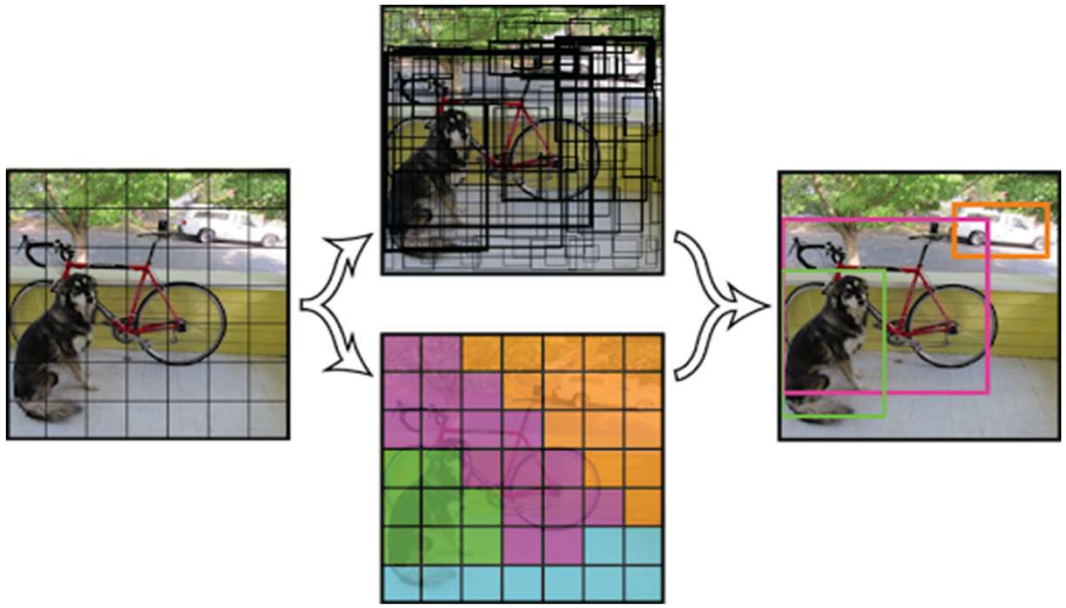
```
for line in H_lines:
    for x1, y1, x2, y2 in line:
        slope = (y2 - y1) / (x2 - x1)
        if math.fabs(slope) < 1/2: # 기울기가 1/2 이하이면 무시한다
            continue
        if slope > 0: # 기울기가 양수이면
            x_right.extend([x1, x2])
            y_right.extend([y1, y2])
        else:
            x_left.extend([x1, x2])
            y_left.extend([y1, y2])


left_eq = np.poly1d(np.polyfit(y_left,x_left,1)) # 1차 함수로 근사
right_eq = np.poly1d(np.polyfit(y_right,x_right,1))

left_x_0 = int(left_eq(y_end))
left_x_end = int(left_eq(y_start))
right_x_0 = int(right_eq(y_end))
right_x_end = int(right_eq(y_start))
```

호프변환에서 얻은 선분들을 그대로 이으면 상당히 지저분한 여러 선분들이 생깁니다. 하나의 정확도 있는 선분을 그리기 위해, 호프변환으로 얻은 선분의 끝점들을 리스트 안에 저장하고, np.poltfit을 이용하여 새로운 직선의 방정식을 회귀분석을 통해 구하였습니다.

사용 이론 및 Open CV 메서드



YOLO 는 you Only Look Once, 즉 이미지를 한번 보는 것만으로 Object의 종류와 위치를 추측하는 딥러닝 기반의 물체인식 알고리즘을 뜻합니다.

간단한 처리과정으로 속도가 매우 빠릅니다. 다른 system들에 비해, 2배 정도 높은 mAP를 보입니다.
Image 전체를 한 번에 바라보는 방식으로 class에 대한 맥락적 이해도가 높습니다.
Object에 대한 좀 더 일반화된 특징을 학습합니다. 다만 작은 물체에 대해서는 정확도가 낮을 수 있습니다.

사용 이론 및 Open CV 메서드

#Yolo 로드

```
net = cv2.dnn.readNet("yolov3.weights", "yolov3.cfg")
classes = []
with open("coco.names", "r") as f:
    classes = [line.strip() for line in f.readlines()]
layer_names = net.getLayerNames()
output_layers = [layer_names[i-1] for i in net.getUnconnectedOutLayers()]
colors = np.random.uniform(0, 255, size=(len(classes), 3))
```

.

#물체감지

```
blob = cv2.dnn.blobFromImage(img, 0.00392, (416, 416), (0, 0, 0), True, crop=False)
net.setInput(blob)
outs = net.forward(output_layers)
```

...

결과

https://www.youtube.com/watch?v=o-lD5uRsBHU

1. YOLO 알고리즘이 적용된 영상을 불러와서 차선을 표시하는 버전
2. 원본 영상에 YOLO 알고리즘과 차선알고리즘을 동시에 적용하는 버전
작동은 하지만 컴퓨터의 성능때문에 많이 끊김

해당 링크는 알고리즘을 따로 적용한 1번 버전입니다.
첨부파일은 두 알고리즘을 합친 버전으로 올렸습니다.

느낀 점

- 동영상에서 차선을 인식하는 것과 같은 임무는, 혼자서 구현해내지도 상당히 어렵습니다. 하지만 Open CV는 이런 고차원의 자료들을 누구나 이용 할 수 있게 해줌으로써 직접 손으로 구현해 볼 수 있게 해줍니다. OpenCV의 장점은 무궁무진한 것 같습니다.

- 이러한 OpenCV가 아무리 좋더라도, 그 존재성과 사용방법을 모르면 사용할 수 없습니다. 이번 강의를 통해 저희가 이용하고 직접 사용해 볼 수 있도록 능력을 길러주신 박희재 교수님께 감사드립니다.

참고 자료

- https://www.youtube.com/watch?v=RFqvTmEFtOE
- https://webnautes.tistory.com/1244
- https://www.sciencetimes.co.kr/news/%EC%9E%90%EC%9C%A8%EC%A3%BC%ED%96%89%EC%9E%90%EB%8F%99%EC%B0%A8-%ED%95%B5%EC%8B%AC%EC%9D%80-%EC%98%81%EC%83%81%EC%9D%B8%EC%8B%9D/
- 자동차의 자기 주행 차선 검출을 위한 시각 센싱(Vision Sensing for the Ego-Lane Detection of a Vehicle) - 김동욱, 도용태