창연공 1조 AD 프로젝트

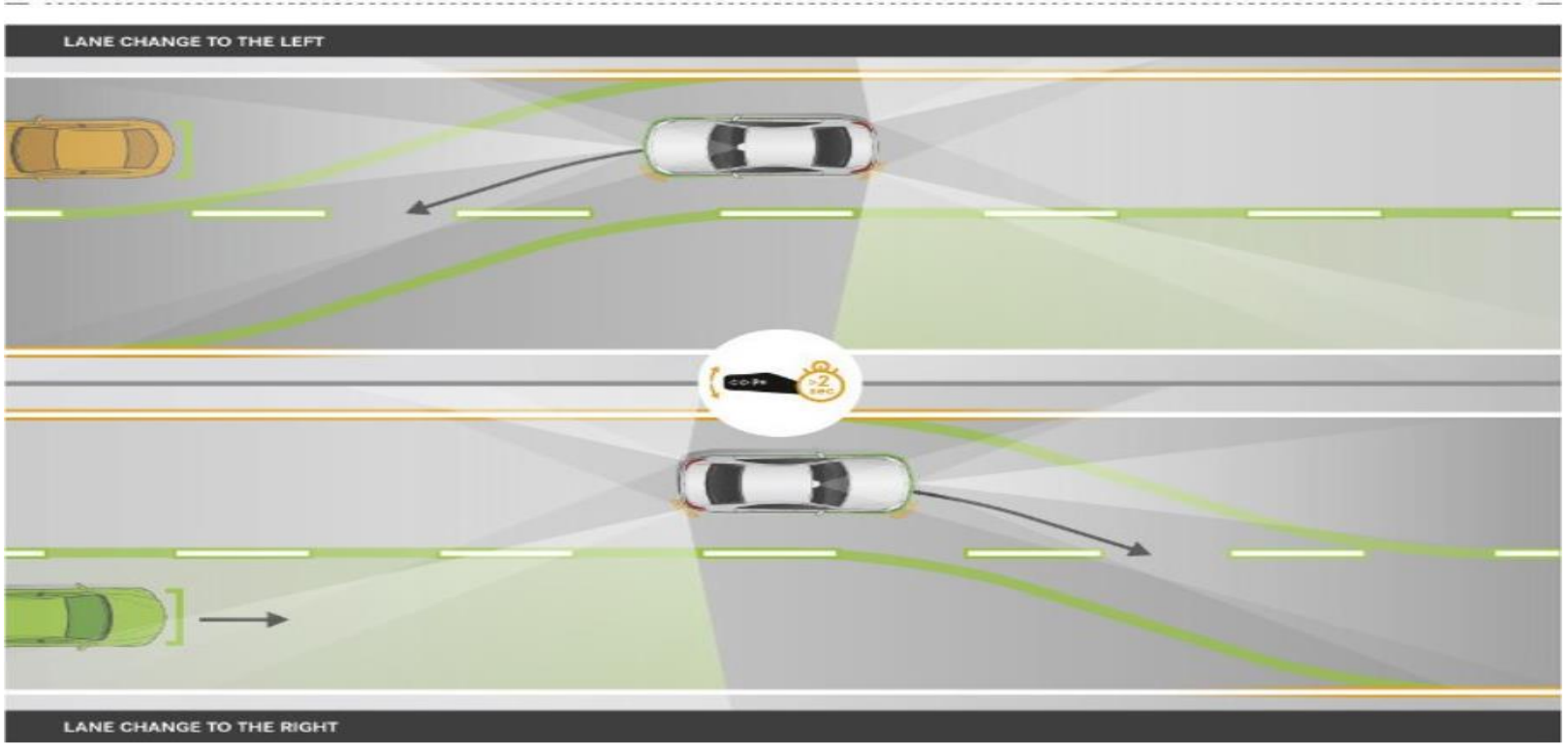
안금장 20181637 오홍석 20191627
윤현승 20191634 강영환 2014302
이성연 20191640

INDEX

1. 주제
2. 동기, 목표
3. 알고리즘
4. 결과
5. 개선할 점

[차선변경]

인접한 차선의 차량 유무에 따른 안전한 차선 변경



[차선변경]

동기 : 자율주행자동차가 실생활에 적용되기 위해 필요한 것

목표 : 장애물 또는 차량을 인지하여 차선 변경을 판단하고 점선인 차선

에서 차선 변경을 수행할 수 있게 하는 것

[점선 판단]

```
def conv_image(self, data):
    self.cam_img = self.bridge.imgmsg_to_cv2(data, 'bgr8')
    self.gray = cv2.cvtColor(self.cam_img, cv2.COLOR_BGR2GRAY)

    self.Blur = cv2.GaussianBlur(self.gray, (5, 5), 0)
    hsv = cv2.cvtColor(self.cam_img, cv2.COLOR_BGR2GRAY)
    lower_white = np.array([0, 0, 70])
    upper_white = np.array([131, 255, 255])
    self.range = cv2.inRange(hsv, lower_white, upper_white)
```

기존의 차선 인식 코드는 유지

Blur처리를 한 이미지를 HSV로 변환시켜준 뒤 inRange()함수를 사용하여

이진화한 이미지를 변수에 저장

목차 3. 알고리즘

```
r_dotarea = self.range[240: 340, 440: 640]
l_dotarea = self.range[240: 340, 80: 280]
if 50 < cv2.countNonZero(r_dotarea) < 300:
    self.l_dot = True
elif 50 < cv2.countNonZero(l_dotarea) < 300:
    self.r_dot = True

# Return positions of left and right lines detected.
return self.left, self.right, self.l_dot, self.r_dot
```

이진화된 이미지를 왼쪽 차선과 오른쪽 차선만큼 ROI설정

ROI구역의 흰색의 픽셀 값들의 개수가 범위에 포함된다면 해당

차선의 점선유무를 True로 변경

리턴 값 또한 차선의 점선 유무 추가

[장애물 탐지]

```python
class ObstacleDetector:

    def __init__(self, topic):
        self.left = []
        self.mid = -1
        self.right = []
        self.back = -1
        rospy.Subscriber(topic, Int32MultiArray, self.read_distance)

    def read_distance(self, data):
        self.left.extend([data.data[0], data.data[7], data.data[3]])
        self.mid = data.data[1]
        self.back = data.data[4]
        self.right.extend([data.data[2], data.data[6], data.data[5]])

    def get_distance(self):
        return self.left, self.mid, self.right, self.back
```

초음파 센서에서 얻은 정보를 방향에 맞게 저장

왼쪽과 오른쪽 거리 정보는 리스트로 변경, 후방을 감지하는 정보도 추가

목차 3. 알고리즘

```python
def steer(self, left, right, l_dot, r_dot, obs_l, obs_r):

    if left == -40 or right == 680:
        mid = (left + right) // 2
        angle = (mid - 320) // 1.8
    else:
        angle = 0

    if min(obs_l) > 40 and l_dot == True:
        angle = -25
    elif min(obs_r) > 40 and r_dot == True:
        angle = 25
    print("angle:", angle)
    return angle
```

왼쪽 또는 오른쪽의 점선 확인이 True이고 해당 방향의 초음파 센서 최솟값이 안전 거리를 충족할 시에 조향각 설정

[속도 설정]

```
def accelerate(self, obs_m, otherspeed, obs_back):
    if obs_m < 50:
        speed = 0
    elif obs_back < 60:
        speed = otherspeed
    else:
        speed = 20

    print("speed : ", speed)
    return speed
```

전방의 초음파 센서 값이 안전 거리보다 작으면 속도는 0으로 설정

후방의 초음파 센서 값이 안전 거리보다 작으면 속도는 입력된 다른 차량의 속도로 설정

목차 4. 결과

점선판단 결과 : 이진화한 이미지 속에서 하얀색의 픽셀 수를 검사하여서 정상적으로 차선이 점선임을 판단

초음파센서를 통한 안전거리 판단, 속도 조절 :

장애물이 존재할 경우에는 차선이 점선일지라도 차선 변경을 하지 않음.

차선 변경 후 뒤의 물체가 일정한 속도로 다가올 때 안전거리보다 가까워지면 물체의 속도를 차량에 적용되어 주행함.

목차 5 . 개선할 점

1. 뒤에 오는 차량이 항상 같은 속도로 주행하는 것이 아니므로 주어진 속도를 적용시키는 것은 문제가 있다.

   어떠한 방식을 적용할 수 있을까?

감사합니다