OpenCV와 rosbag을 이용한 차선인식

프로그래머스 자율주행 코스 1기 조정민

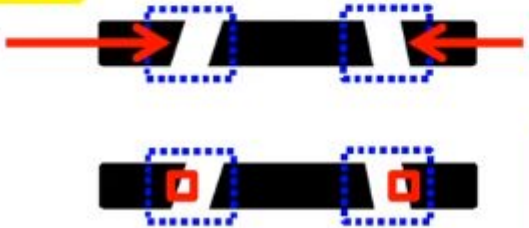
차선을 인식하여 운전

- 차선 추종 주행
  - 좌우 차선을 찾아내어 차선을 벗어나지 않게끔 주행하면 된다

카메라 입력으로
취득한 영상에서
적절한 영역을 잘라냄

어디를 잘라서 볼까???
→ 이걸 잘 정해야 한다.

이진화



우선 바깥에서 중앙으로 가면서 흰색 점을 찾는다.

그 점 주위에 사각형을 쳐서 사각형 안에 있는
흰색점의 개수를 구한다. 기준 개수 이상이면
바로 거기가 차선이다!

01:54

/usb_cam/image_raw 토픽을 구독 - Subscribe

토픽 데이터를 OpenCV 이미지 데이터로 변환

OpenCV 영상처리

ROI	관심영역 잘라내기
GrayScale	흑백 이미지로 변환
Gaussian Blur	노이즈 제거
HSV - Binary	HSV 기반으로 이진화 처리

차선 위치 찾고 화면 중앙에서 어느쪽으로 치우쳤는지 파악

핸들을 얼마나 꺾을지 결정 (조향각 설정 각도 계산)

모터 제어를 위한 토픽 발행 – Publish

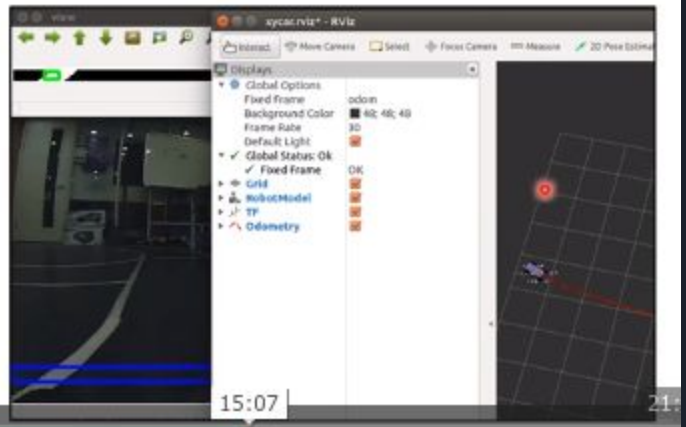
결과물 제출 방법

- 동영상 제출
  - RVIZ에서 자동차가 움직이는 것을 휴대폰으로 촬영해

```
.py  track1.avi
c$
c$ rosbag play -l cam_topic.bag
```

- 파일 제출
  - line_follow.py 파일 제출

- 문서 제출 (PPT)
  - line_follow.py 파일 소스코드를
    상세히 설명하는 내용을 담아 제출



RIVZ 차선 인식 주행 launch 파일

```
<launch>
    <param name="robot_description" textfile="$(find rviz_xycar)/urdf/xycar_3d.urdf"/>
    <param name="use_gui" value="true"/>

    <!-- rviz display -->
    <node name="rviz_visualizer" pkg="rviz" type="rviz" required="true"
                args="-d $(find rviz_xycar)/rviz/rviz_drive.rviz"/>

    <node name="robot_state_publisher" pkg="robot_state_publisher"
                type="state_publisher"/>


    <node name="converter" pkg="rviz_xycar" type="converter.py" />
    <node name="rviz_odom" pkg="rviz_xycar" type="rviz_odom.py" />
    <node name="driver" pkg="line_drive" type="line_follow_obo.py" output="screen"/>
</launch>
```

RIVZ 차선 인식 주행 with rosbag launch 파일

```
<launch>
   <param name="robot_description" textfile="$(find rviz_xycar)/urdf/xycar_3d.urdf"/>
   <param name="use_gui" value="true"/>

   <!-- rviz display -->
   <node name="rviz_visualizer" pkg="rviz" type="rviz" required="true"
               args="-d $(find rviz_xycar)/rviz/rviz_drive.rviz"/>

   <node name="robot_state_publisher" pkg="robot_state_publisher"
               type="state_publisher"/>

<node name="rosbag_play" pkg="rosbag" type="play" output="screen" required="true" args="$(find
line_drive)/src/cam_topic.bag"/>
   <node name="converter" pkg="rviz_xycar" type="converter.py" />
   <node name="rviz_odom" pkg="rviz_xycar" type="rviz_odom.py" />
   <node name="driver" pkg="line_drive" type="line_follow_obo.py" output="screen"/>
</launch>
```

line_follow 파일 - 콜백함수에서 가공 가능한 이미지 변수로 변환

```
#!/usr/bin/env python

import cv2, time
import rospy
import numpy as np
from xycar_motor.msg import xycar_motor
from std_msgs.msg import Header
from sensor_msgs.msg import Image
from cv_bridge import CvBridge

bridge = CvBridge()
cv_image = np.empty(shape=[0])
motor_msg = xycar_motor()

def img_callback(data):
    global cv_image
    cv_image = bridge.imgmsg_to_cv2(data, "bgr8")
```

line_follow 파일- left와 right의 각도 만큼 angle 각도를 부여한다. left와 right가 음수라면 이전 angle 과 speed를 사용한다.

```
rospy.init_node('driver')
motor_pub = rospy.Publisher('xycar_motor', xycar_motor, queue_size=1)
rospy.Subscriber("/usb_cam/image_raw/", Image, img_callback)

def drive(left,right):
    global width_640, motor_msg ,motor_pub
    motor_msg.header = Header()
    motor_msg.header.stamp = rospy.Time.now()
    if( left >= 0 and right > 0):
        diff = width_640 - (left + right)
        motor_msg.speed = 10
        motor_msg.angle = diff * 0.4
    motor_pub.publish(motor_msg)
```

line_follow 파일 - 이미지 처리에 사용할 값 초기화

```
threshold_100 = 100
width_640 = 640
scan_width_200, scan_height_20 = 200, 20
lmid_200, rmid_440 = scan_width_200, width_640 - scan_width_200
area_width_20, area_height_10 = 20, 10
vertical_430 = 430
row_begin_5 = (scan_height_20 - area_height_10) // 2
row_end_15 = row_begin_5 + area_height_10
pixel_threshold_160 = 0.8 * area_width_20 * area_height_10
```

line_follow 파일 - 하얀 픽셀 개수가 threshold 값보다 큰지 확인

```
    for l in range(area_width_20, lmid_200):
        area = bin[row_begin_5:row_end_15, l - area_width_20:l]
        if cv2.countNonZero(area) > pixel_threshold_160:
            left = l
            break
    for r in range(width_640 - area_width_20, rmid_440, -1):
        area = bin[row_begin_5:row_end_15, r:r + area_width_20]
        if cv2.countNonZero(area) > pixel_threshold_160:
            right = r
            break
    if left != -1:
        lsquare = cv2.rectangle(view,
                                (left - area_width_20, row_begin_5),
                                (left, row_end_15),
                                (0, 255, 0), 3)
    else:
        print("Lost left line")
    else:    if right != -1:
        rsquare = cv2.rectangle(view,(right, row_begin_5),
                                (right + area_width_20, row_end_15),(0, 255, 0), 3)
        print("Lost right line")
```

line_follow 파일 - 주행 함수 호출 및 영상 송출

```
    if(left != -1 and right != -1):
        drive(left,right)
    cv2.imshow("origin", cv_image)
    cv2.imshow("view", view)

    hsv = cv2.cvtColor(cv_image, cv2.COLOR_BGR2HSV)
    lbound = np.array([0, 0, threshold_100], dtype=np.uint8)
    ubound = np.array([131, 255, 255], dtype=np.uint8)
    hsv = cv2.inRange(hsv, lbound, ubound)
    cv2.imshow("hsv", hsv)
    cv2.waitKey(1)
cap.release()
cv2.destroyAllWindows()
```

RIVZ 차선 인식 주행 실행

```
# rosbag
# terminal A

rosbag play -l camtopic.bag

# terminal B
roslaunch line_drive line_follow.launch
```

RIVZ 차선 인식 주행 실행

```
roslaunch line_drive line_follow_rosbag.launch
```

명도차 기반 차선 인식 주행

- 한쪽 차선만을 확인하면서 주행하도록 만들었다.

```
def drive(left,right):
    global width_640, motor_msg ,motor_pub
    motor_msg.header = Header()
    motor_msg.header.stamp = rospy.Time.now()
    if(left > 15):
        motor_msg.speed = 30
        motor_msg.angle = min(left / 2 , 50)
        motor_pub.publish(motor_msg)
    elif(left < 0):
        motor_pub.publish(motor_msg)
    else:
        motor_msg.speed = 30
        motor_msg.angle = 0
        motor_pub.publish(motor_msg)
```

```
roslaunch auto_drive line_follow_obo.launch
```