차선 인지/신호등 인지

1. 차선인지 & Binarization
2. Region of Interest & Bird’s eyes view
3. Curve fitting & 신호등 인지

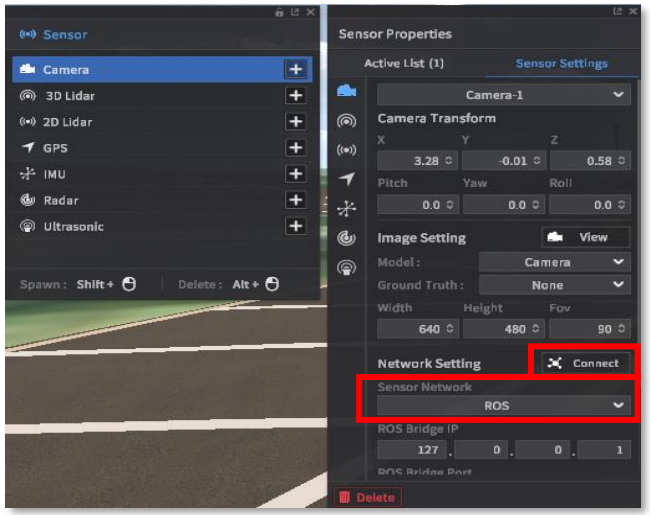
1

차선인지 & Binarization

카메라 센서 이미지 받기

1 시뮬레이터 카메라 센서 사용법

- Edit – Sensor –Sensor edit mode로 이동 후
  Sensor List에 Camera 클릭 –마우스 커서 이동 - Shift + 마우스 좌 클릭
- 네트워크 세팅 탭에 ROS 선택
- 이전의 다른 센서들처럼 Rosbridge를 실행하고, Camera Setting에 connect 클릭 시 client connected라고 메시지가 뜨면 연결 완료




```
setting /run_id to 1e56f596-e50e-11eb-b5c3-08002743d974
process[rosout-1]: started with pid [25613]
started core service [/rosout]
process[rosbridge_websocket-2]: started with pid [25616]
process[rosapi-3]: started with pid [25617]
registered capabilities (classes):
 - rosbridge_library.capabilities.call_service.CallService
 - rosbridge_library.capabilities.advertise.Advertise
 - rosbridge_library.capabilities.publish.Publish
 - rosbridge_library.capabilities.subscribe.Subscribe
 - <class 'rosbridge_library.capabilities.defragmentation.Defragment'>
 - rosbridge_library.capabilities.advertise_service.AdvertiseService
 - rosbridge_library.capabilities.service_response.ServiceResponse
 - rosbridge_library.capabilities.unadvertise_service.UnadvertiseService
[INFO] [1626313438.535502]: Rosbridge WebSocket server started on port 9090
[INFO] [1626313759.217993]: Client connected.  1 clients total.
```

카메라 센서 이미지 받기

1 시뮬레이터 카메라 센서 사용법

- [x] Rviz 실행 영상

카메라 센서 이미지 받기

2 Image parser node 만들기

- 이미 만들어 놓은 패키지 안에 image_parser.py 생성
- 아래와 같이 IMGParser로 클래스를 정의하고 노드 작성

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

import rospy
import cv2
import numpy as np
import os, rospkg

from sensor_msgs.msg import CompressedImage
from cv_bridge import CvBridgeError

class IMGParser:
    def __init__(self):

        self.image_sub = rospy.Subscriber("/image_jpeg/compressed", CompressedImage, self.callback)

    def callback(self, msg):
        try:
            np_arr = np.fromstring(msg.data, np.uint8)
            img_bgr = cv2.imdecode(np_arr, cv2.IMREAD_COLOR)
        except CvBridgeError as e:
            print(e)

        cv2.imshow("Image window", img_bgr)
        cv2.waitKey(1)


if __name__ == '__main__':

    rospy.init_node('image_parser', anonymous=True)

    image_parser = IMGParser()

    rospy.spin()
```

카메라 센서 이미지 받기

2 Image parser node 만들기

- 노드에 대한 설명
- 4:opencv 모듈 가져오기
- 8: 시뮬레이터의 compressed images를 받을 때 쓰는 sensor message
- 14: 시뮬레이터의 카메라 ros topic이름과 일치해야 함

```
1  #!/usr/bin/env python
2
3  import rospy
4  import cv2
5  import numpy as np
6  import os, rospkg
7
8  from sensor_msgs.msg import CompressedImage
9  from cv_bridge import CvBridgeError
```

```
11 class IMGParser:
12     def __init__(self):
13
14         self.image_sub = rospy.Subscriber("/image_jpeg/compressed", CompressedImage, self.callback)
15
```

카메라 센서 이미지 받기

2 Image parser node 만들기

- Image parser node 만들기
- 노드에 대한 설명
- 18: np.fromstring으로 bytes를 uint8 array로 변환
- 19: 다시 [세로, 가로, 채널]의 이미지 array로 변환
- 23: 토픽에서 받은 이미지를 띄움

```
16      def callback(self, msg):
17          try:
18              np_arr = np.fromstring(msg.data, np.uint8)
19              img_bgr = cv2.imdecode(np_arr, cv2.IMREAD_COLOR)
20          except CvBridgeError as e:
21              print(e)
22
23          cv2.imshow("Image window", img_bgr)
24          cv2.waitKey(1)
25
```

카메라 센서 이미지 받기

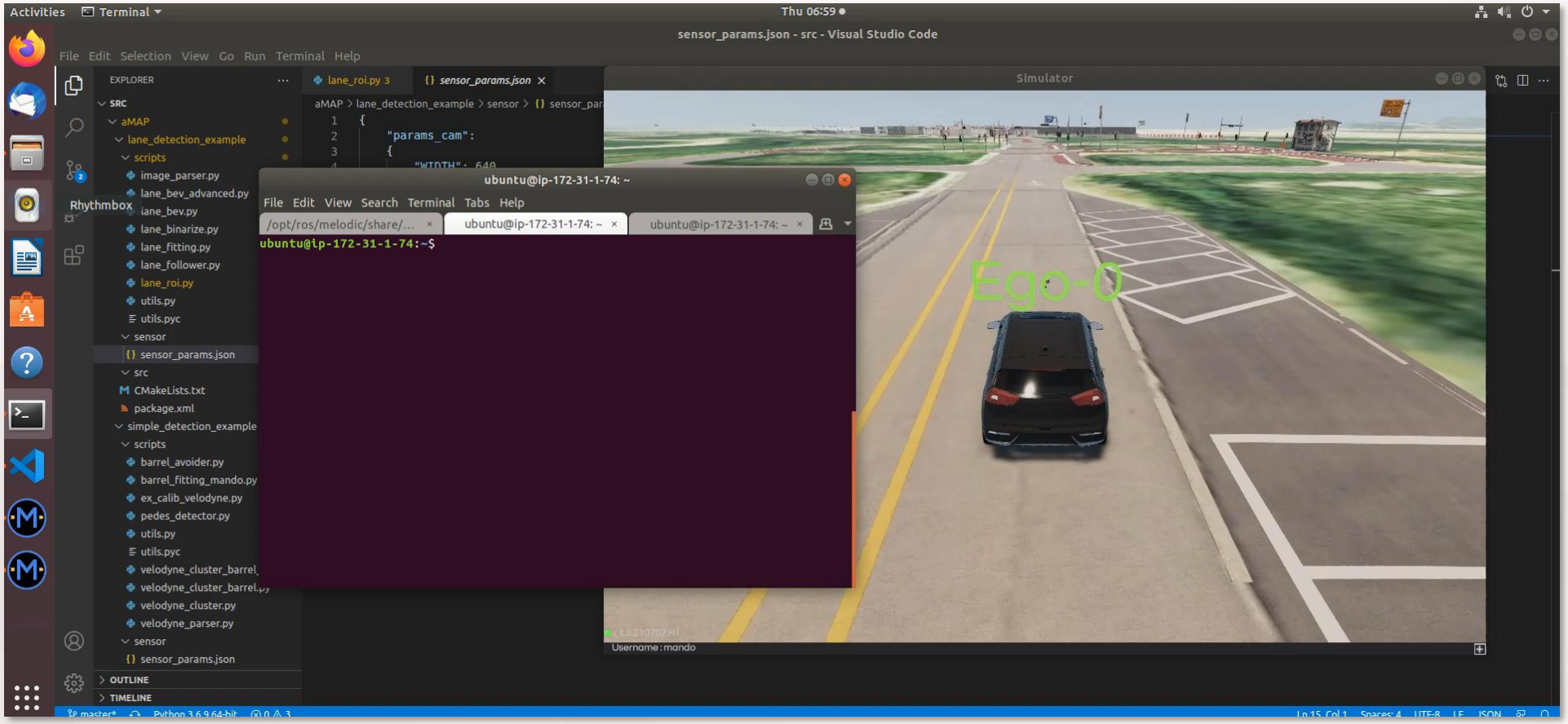
2 Image parser node 만들기

```
25
26
27  if __name__ == '__main__':
28
29      rospy.init_node('image_parser', anonymous=True)
30
31      image_parser = IMGParser()
32
33      rospy.spin()
```

카메라 센서 이미지 받기

3 시뮬레이터 카메라 센서 사용법

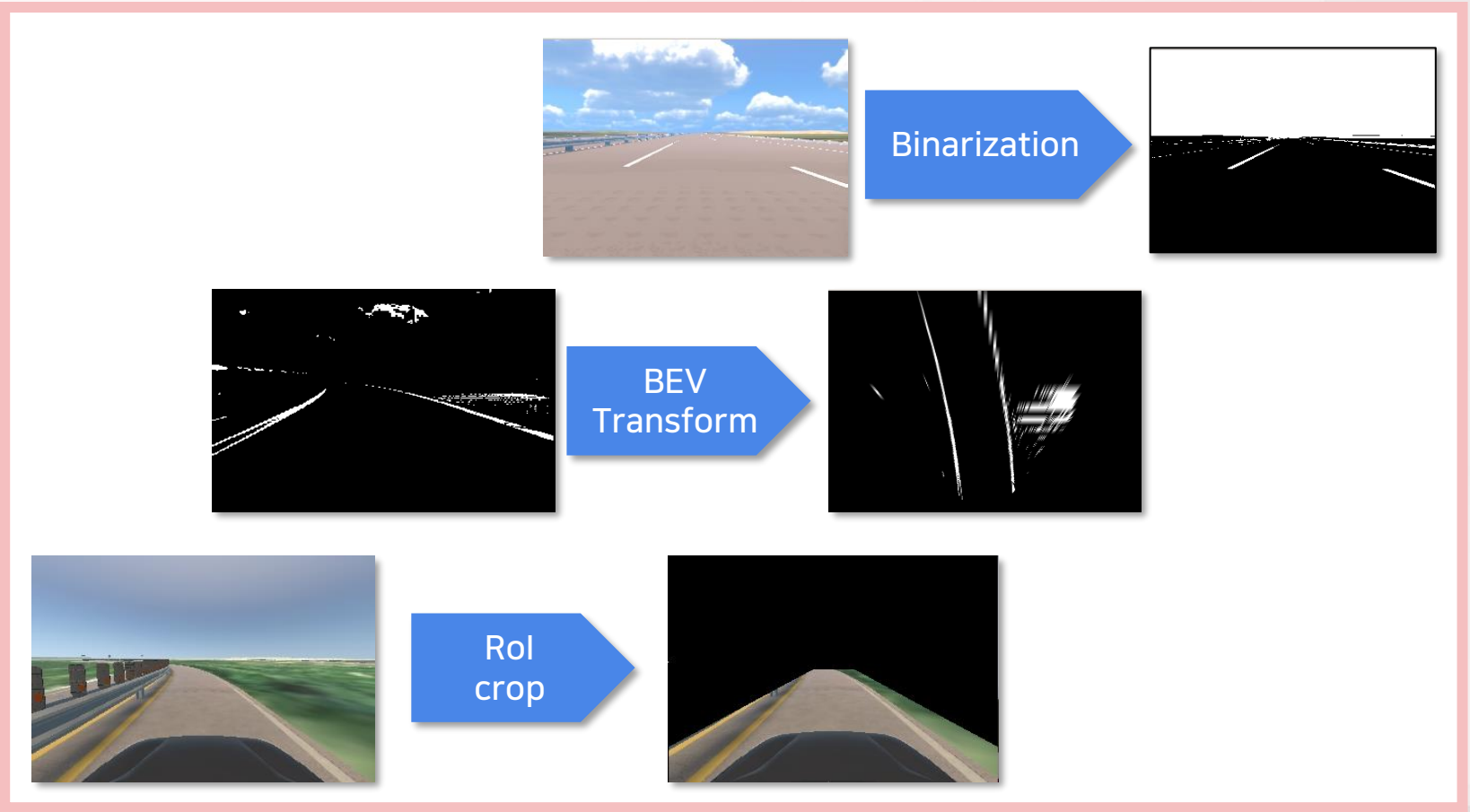
실행 영상



차선 인지 과정

1 차선 인지 과정 단계들

- 이진화(binarization)
- Bird's eye view transform
- Region of Interest
- Curve fitting

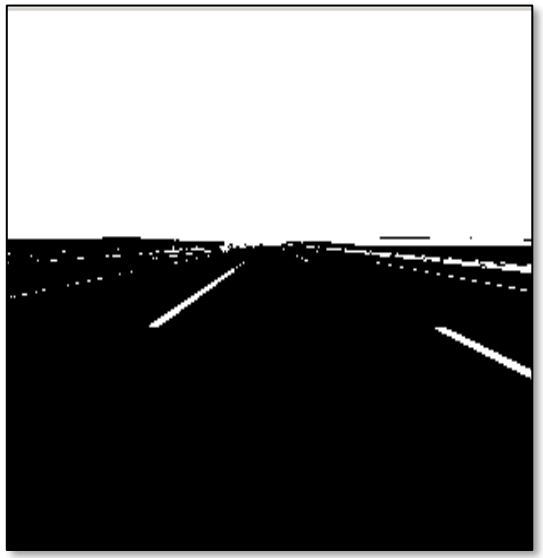


차선 인지 과정

2 이진화 (binarization)

- 이미지의 색상, 명도 등의 특성을 가지고 차선의 픽셀만 255, 나머지 픽셀 값은 0으로 반환

Binarization



차선 인지 과정

3 Bird's eye view transform

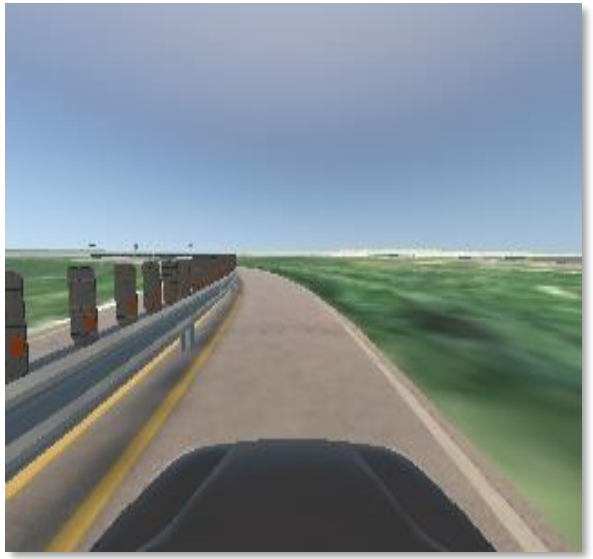
- 하늘에서 내려다보는 시점으로 이미지를 펼쳐놓는 변환 과정
- 차선의 곡률을 더 명확히 보기 위해 사용

BEV Transform

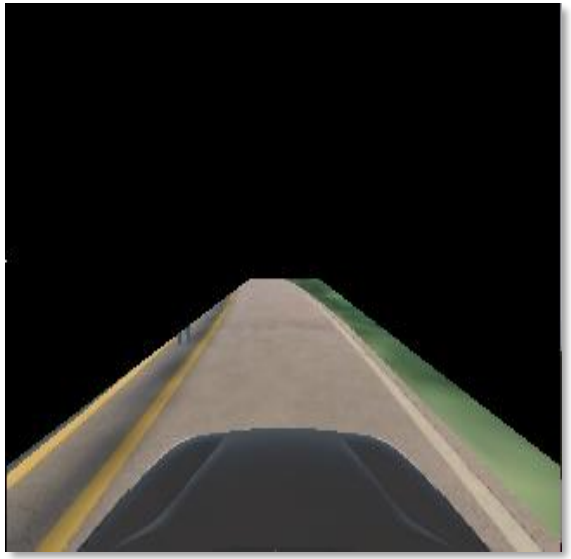
차선 인지 과정

4 Region of Interest

- 하늘이나 도로 바깥 영역들을 잘라내고 차선이 존재하는 부분만 편집해서 보는 과정
- 지나치게 잘라내면 곡률이 큰 구간에서 차선을 놓칠 수도 있음



RoI
Crop



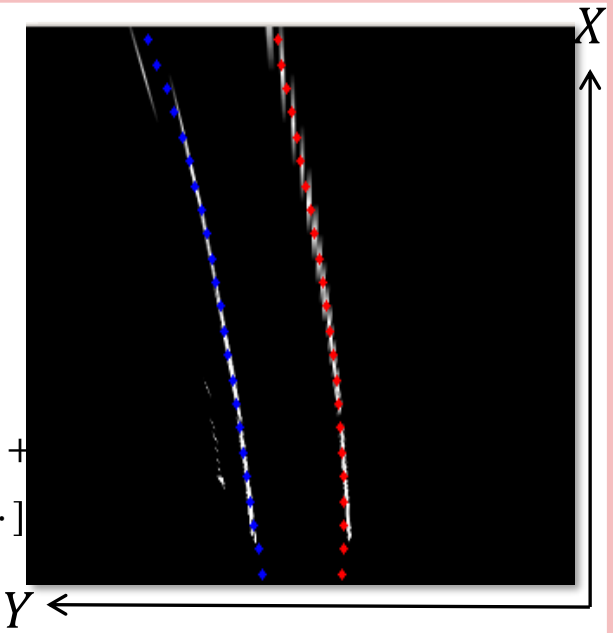
차선 인지 과정

5 Curve fitting

- 차선의 픽셀 정보들을 가지고 다항식 형태로 regression(회귀)

Fitting

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \cdots +$$
$$= [\beta_0, \beta_1, \beta_2, \cdots][1, x_1, x_1^2, \cdots]$$



이진화

1 Binarization

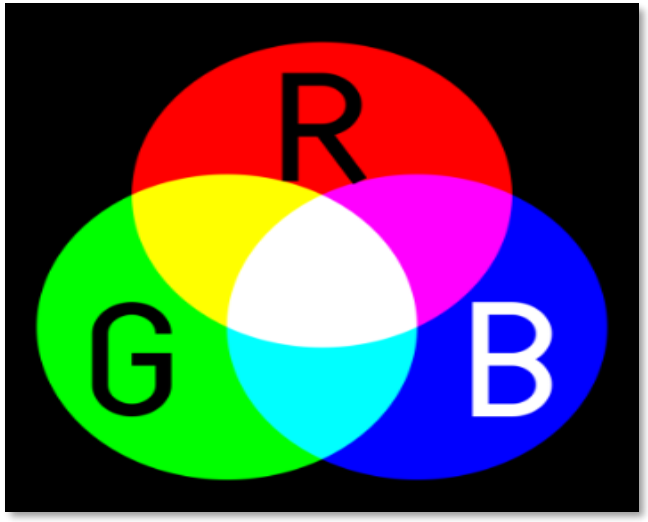
- 색상 이미지를 특정 조건에 따라 흑백으로 전환
- 명도로 구분하기 좋은 객체에 쓰기 좋음

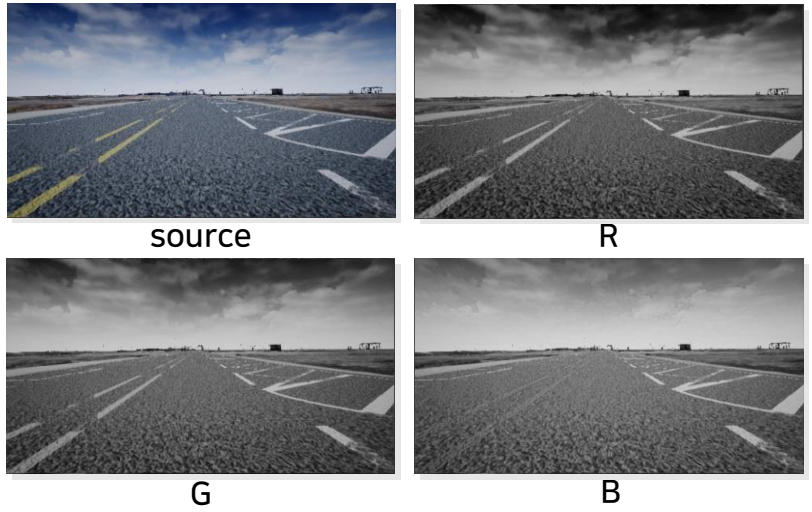
[출처] http://joe-schueller.github.io/2016/04/03/lane-detection-systems.html

이진화

1 Binarization

- 대부분의 이미지 파일은 [세로, 가로, 채널]의 array 형태로
- RGB: 빛의 3원색을 기반으로, 채널이 (red, green, blue) 3개의 벡터 형태로 구성
- 여러 색이 혼합되어 표현되기에 특정 색만 골라서 이진화 하긴 힘듦

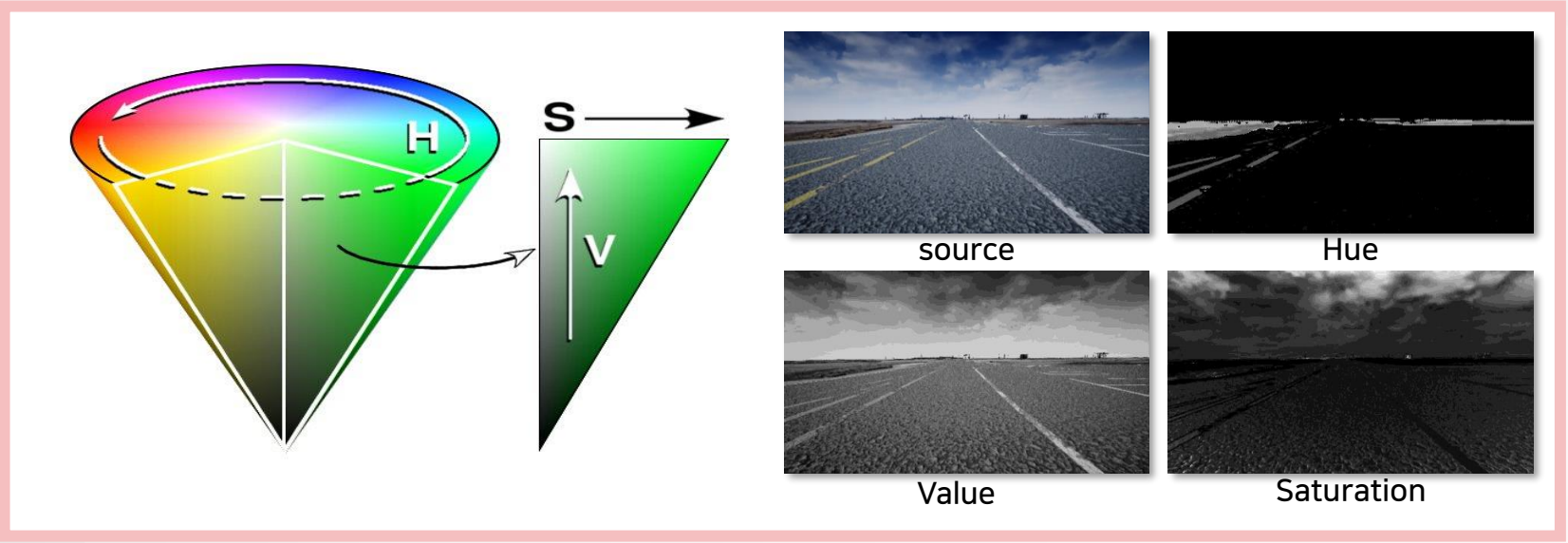




이진화

2 HSV 채널

- Hue (색조): 색의 종류, 0º~360º의 범위를 가짐
- Saturation (채도): 색의 선명도, 진함의 정도 (가장 진한 상태를 100%로 함)
- Value (밝기):색의 밝기, 밝은 정도 (가장 밝은 상태를 100%로 함)



[출처] https://ko.wikipedia.org/wiki/HSV_%EC%83%89_%EA%B3%B5%EA%B0%84#/media/File:HSV_cone.jpg

이진화

3 HSV로 이미지 변환하기

- 이전에 만들었던 image_parser.py 안에 채널 변환 추가
- Opencv(cv2)의 함수들을 사용

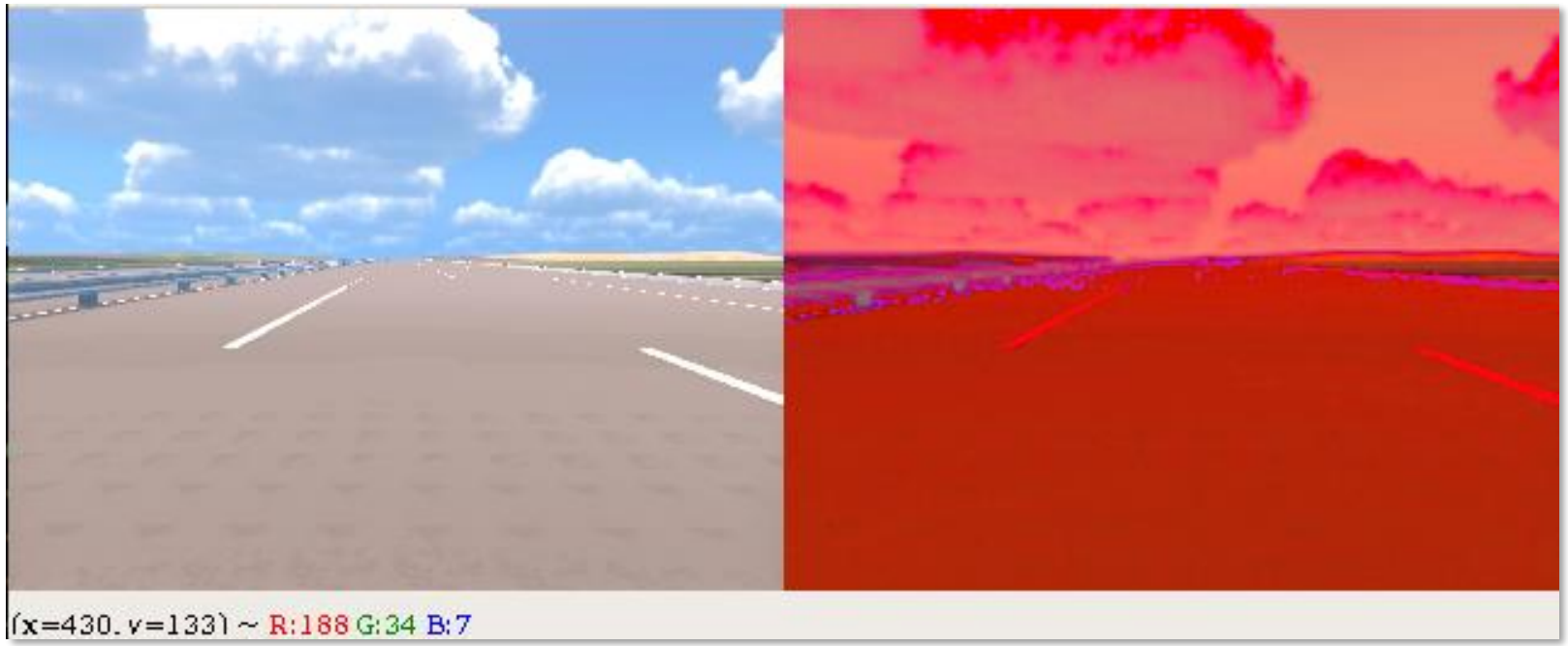
- Image_parser.py 에서 cv2로 색상 convert
- 23 : cv2.cvtColor : BGR 대신 다른 색상채널로 convert 하는 함수
- 23 : cv2.COLOR_BGR2HSV : HSV 로 바꾸기 위한 옵션
- 25 : np.concatenate : img_bgr과 img_hsv를 가로 방향으로 붙이는 함수
- 이미지는 [세로, 가로, 채널]로 array가 구성되어 있기 때문에, 가로로 concat 하려면, axis는 1로 주어야 함

```
16      def callback(self, msg):
17          try:
18              np_arr = np.fromstring(msg.data, np.uint8)
19              img_bgr = cv2.imdecode(np_arr, cv2.IMREAD_COLOR)
20          except CvBridgeError as e:
21              print(e)
22
23          img_hsv = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2HSV)
24
25          img_concat = np.concatenate([img_bgr, img_hsv], axis=1)
26
27          cv2.imshow("Image window", img_concat)
28          cv2.waitKey(1)
29
```

이진화

3 HSV로 이미지 변환하기

- RGB와 HSV의 비교
- H-B, S-G, V-R로 매치되서 cv2.imshow에 표현



이진화

4 Binarization

- 색상 채널 변환 후 특정 hsv 값만 255, 나머지는 0으로 binarize
- cv2.inRange() : hsv array에서 lower, upper 사이에 있는 값들만 255, 나머지는 0으로 바꿔주는 함수

```
def callback(self, msg):
    try:
        np_arr = np.fromstring(msg.data, np.uint8)
        self.img_bgr = cv2.imdecode(np_arr, cv2.IMREAD_COLOR)
    except CvBridgeError as e:
        print(e)

    img_hsv = cv2.cvtColor(self.img_bgr, cv2.COLOR_BGR2HSV)

    lower_wlane = np.array([redacted]
    upper_wlane = np.array([30,60,255])

    img_wlane = cv2.inRange(img_hsv, lower_wlane, upper_wlane)

    img_wlane = cv2.cvtColor(img_wlane, cv2.COLOR_GRAY2BGR)

    img_concat = np.concatenate([self.img_bgr, img_hsv, img_wlane], axis=1)

    cv2.imshow("Image window", img_concat)
    cv2.waitKey(1)
```

이진화

4 Binarization

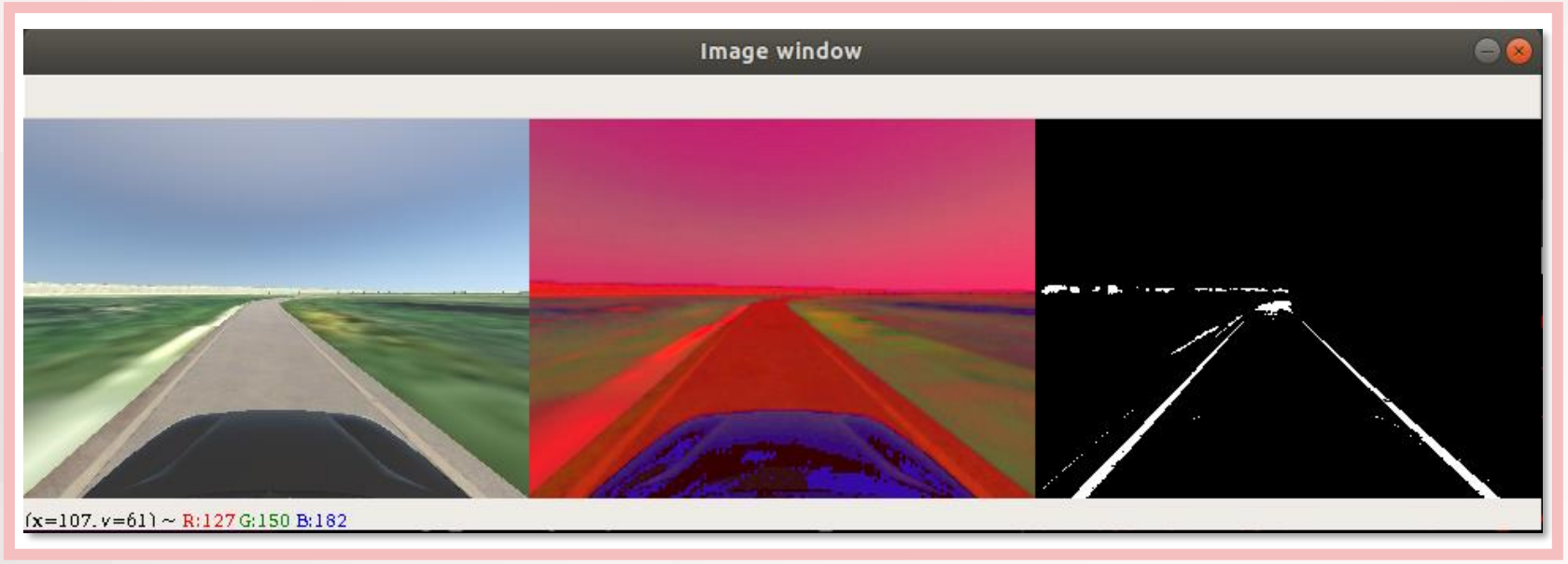
차선에 해당되는 hsv 값을 포함하는 lower, upper bound를 알아서 찾아보기

```
def callback(self, msg):
    try:
        np_arr = np.fromstring(msg.data, np.uint8)
        self.img_bgr = cv2.imdecode(np_arr, cv2.IMREAD_COLOR)
    except CvBridgeError as e:
        print(e)

    img_hsv = cv2.cvtColor(self.img_bgr, cv2.COLOR_BGR2HSV)

    lower_wlane = np.array([redacted]
    upper_wlane = np.array([30,60,255])

    img_wlane = cv2.inRange(img_hsv, lower_wlane, upper_wlane)

    img_wlane = cv2.cvtColor(img_wlane, cv2.COLOR_GRAY2BGR)

    img_concat = np.concatenate([self.img_bgr, img_hsv, img_wlane], axis=1)

    cv2.imshow("Image window", img_concat)
    cv2.waitKey(1)
```

이진화

4 Binarization

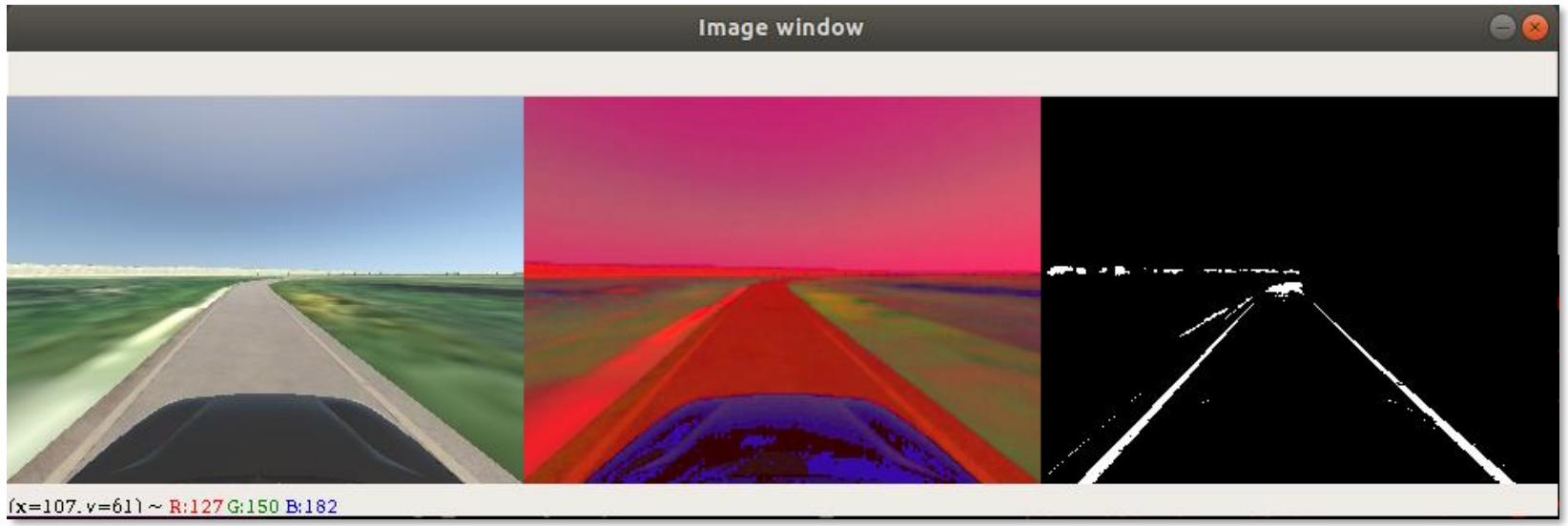
- 색상 채널 변환 후 특정 hsv 값만 255, 나머지는 0으로 binarize
- cv2.inRange() : hsv array에서 lower, upper 사이에 있는 값들만 255, 나머지는 0으로 바꿔주는 함수



이진화

4 Binarization

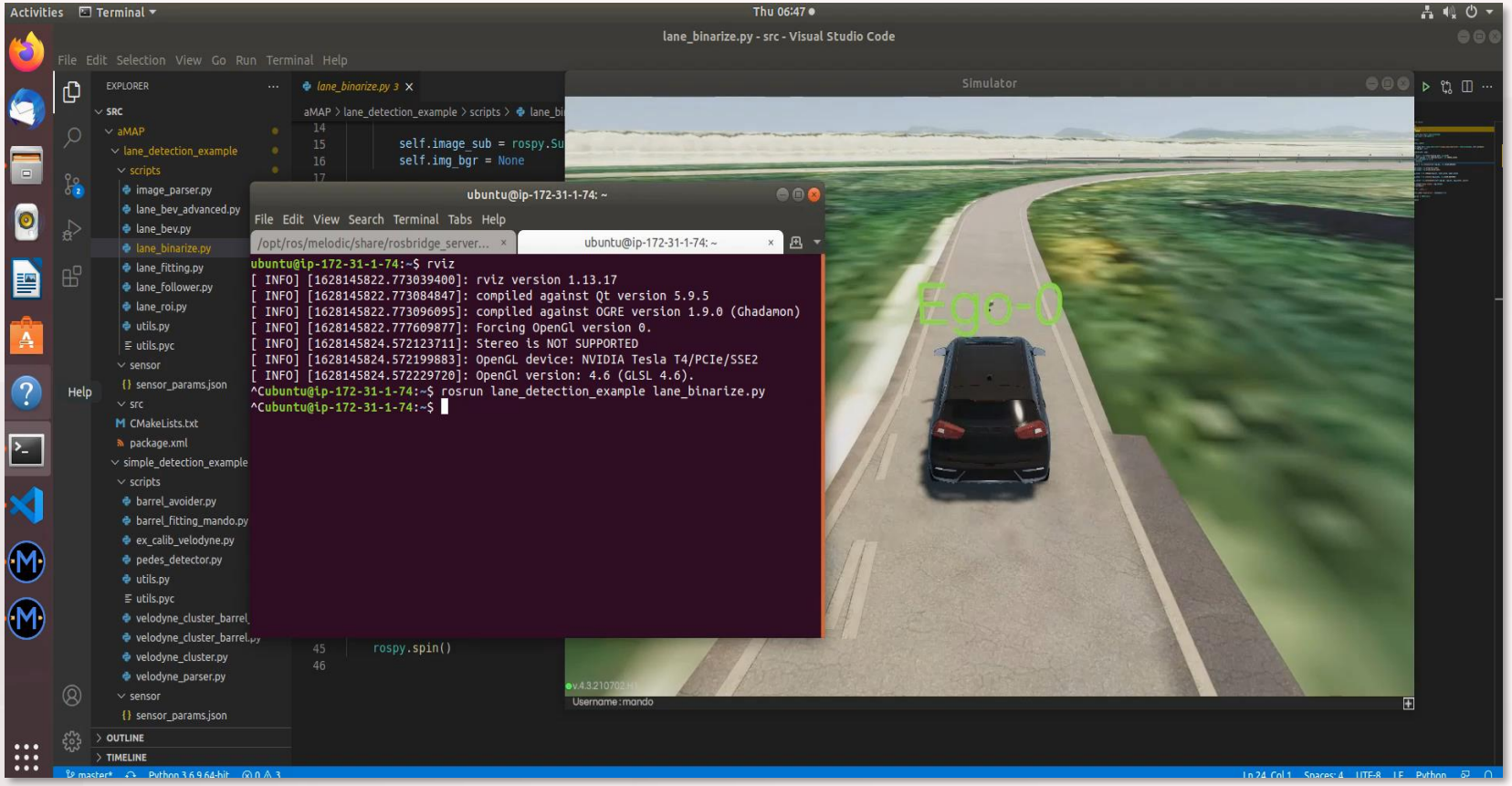
차선에 해당되는 hsv 값을 포함하는 lower, upper bound를 알아서 찾아보기



이진화

4 Binarization

실행 영상



References

- http://joe-schueller.github.io/2016/04/03/lane-detection-systems.html
- https://ko.wikipedia.org/wiki/HSV_%EC%83%89_%EA%B3%B5%EA%B0%84#/media/File:HSV_cone.jpg

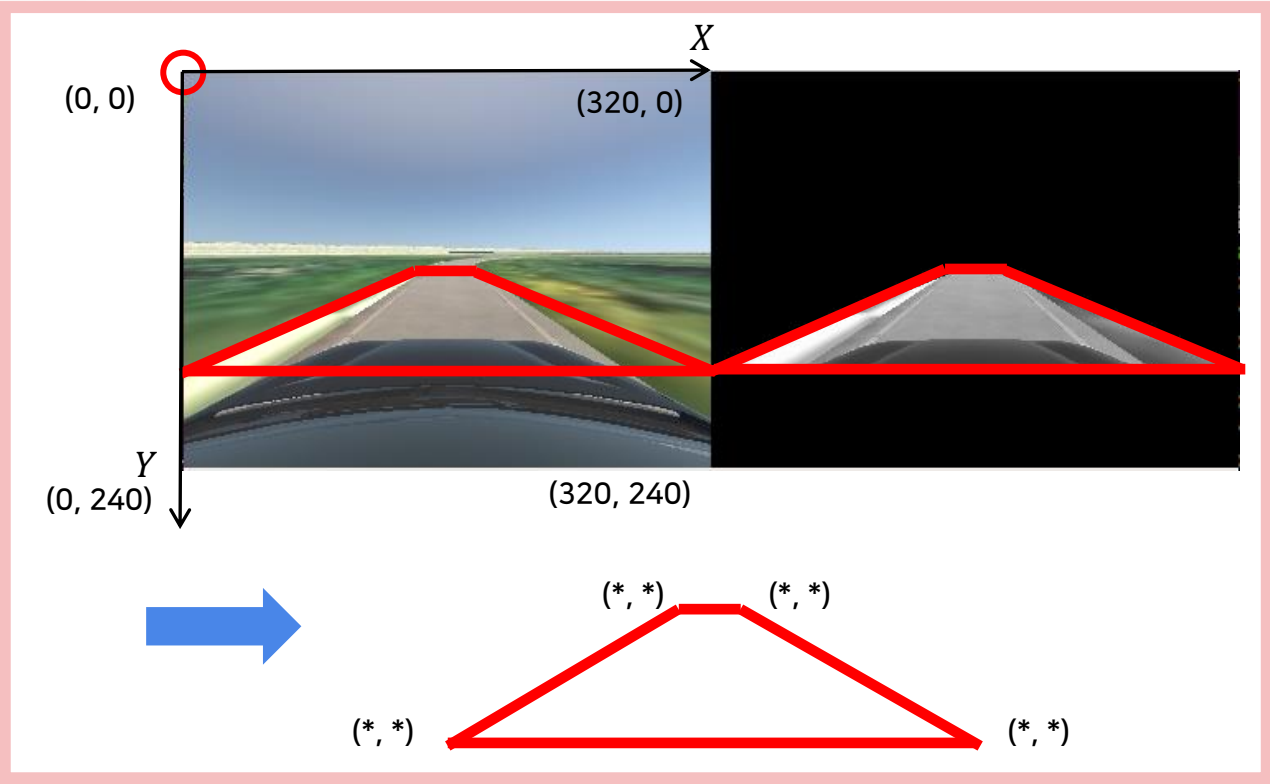
2

Region of Interest & Bird’s eyes view

Region of Interest

1 RoI 이미지 설정

- 아래와 같이 320x240 사이즈의 이미지가 주어져 있음
- 원하는 영역의 [x, y] 좌표들을 정의해서 RoI영역 정의



Region of Interest

1 RoI 이미지 설정

- Image parser 노드를 복사해서 새로운 노드를 정의 (lane_roi.py)
- IMGParser안에 mask_roi()를 추가
- RoI의 포인트들을 클래스 안에 정의

```
class IMGParser:

    def __init__(self):

        self.image_sub = rospy.Subscriber("/image_jpeg/compressed", CompressedImage, self.callback)

        self.crop_pts = np.array(
            [[
                [
                [          ],
                [          ],
                [          ]
            ]]
        )
```

Region of Interest

1 RoI 이미지 설정

- Image parser 노드를 복사해서 새로운 노드를 정의 (lane_roi.py)
- IMGParser안에 mask_roi()를 추가
- RoI의 포인트들을 클래스 안에 정의

```
class IMGParser:

    def __init__(self):

        self.image_sub = rospy.Subscriber("/image_jpeg/compressed", CompressedImage, self.callback)

        self.crop_pts = np.array(
            [[
                [0,180],
                [140,120],
                [180,120],
                [320,180]
            ]]
        )
```

Region of Interest

1 RoI 이미지 설정

- IMGParser안에 mask_roi()를 추가

```
def mask_roi(self, img):

    h = img.shape[0]
    w = img.shape[1]

    if len(img.shape)==3:

        # image shape : [h, w, 3]

        c = img.shape[2]
        mask = np.zeros((h, w, c), dtype=np.uint8)

        mask_value = (255, 255, 255)

    else:

        # binarized image or grayscale image : [h, w]

        mask = np.zeros((h, w), dtype=np.uint8)

        mask_value = (255)

    cv2.fillPoly(mask, self.crop_pts, mask_value)

    mask = cv2.bitwise_and(mask, img)

    return mask
```

Region of Interest

1 RoI 이미지 설정

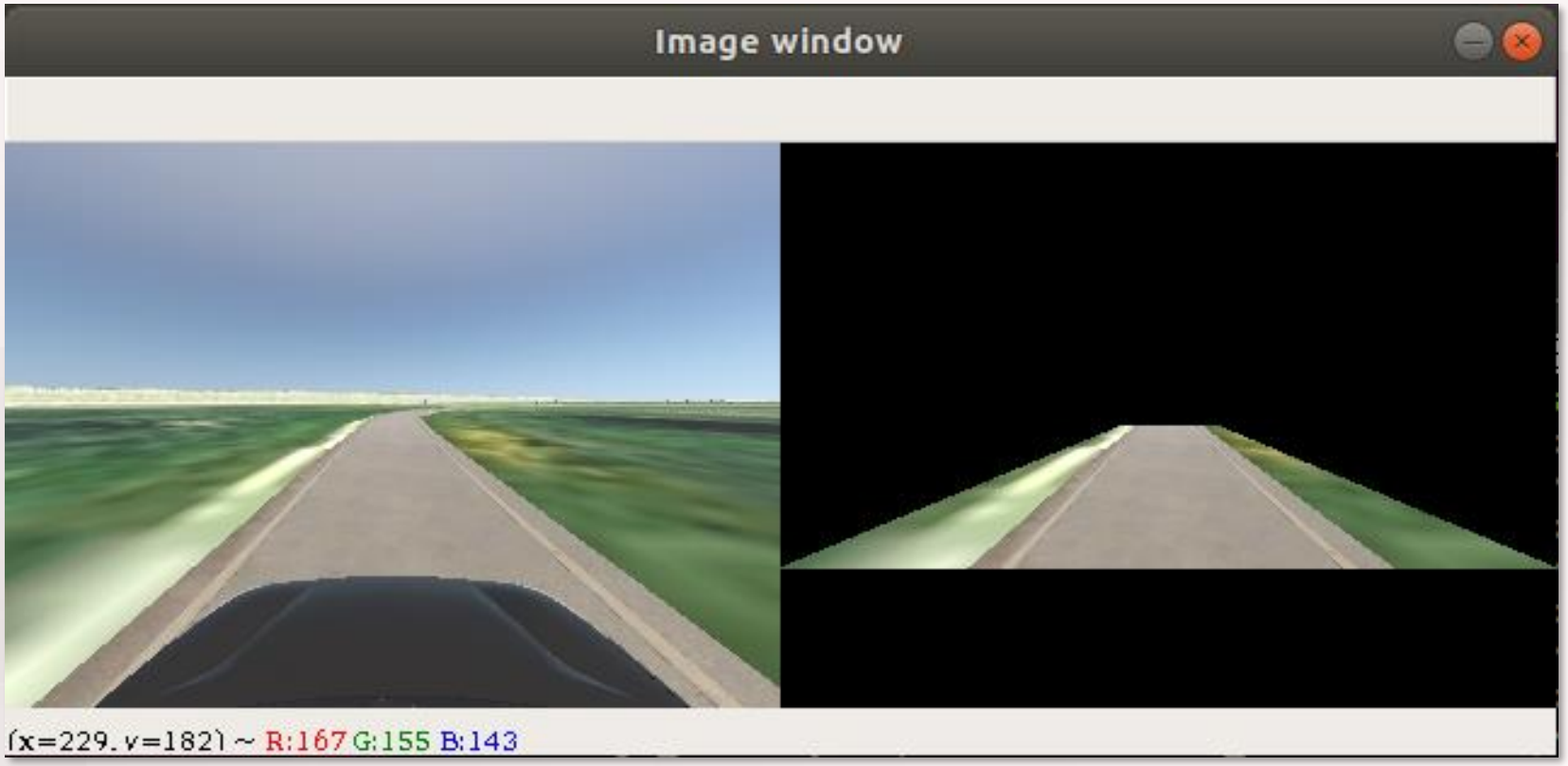
- IMGParser의 callback 함수 안에 mask_roi()를 실행

```
def callback(self, msg):
    try:
        np_arr = np.fromstring(msg.data, np.uint8)
        img_bgr = cv2.imdecode(np_arr, cv2.IMREAD_COLOR)
    except CvBridgeError as e:
        print(e)

    self.mask = self.mask_roi(img_bgr)

    if len(self.mask.shape)==3:
        img_concat = np.concatenate([img_bgr, self.mask], axis=1)

    else:
        img_concat = np.concatenate([img_bgr, cv2.cvtColor(self.mask, cv2.COLOR_GRAY2BGR)], axis=1)

    cv2.imshow("Image window", img_concat)
    cv2.waitKey(1)
```

Region of Interest

1 RoI 이미지 설정

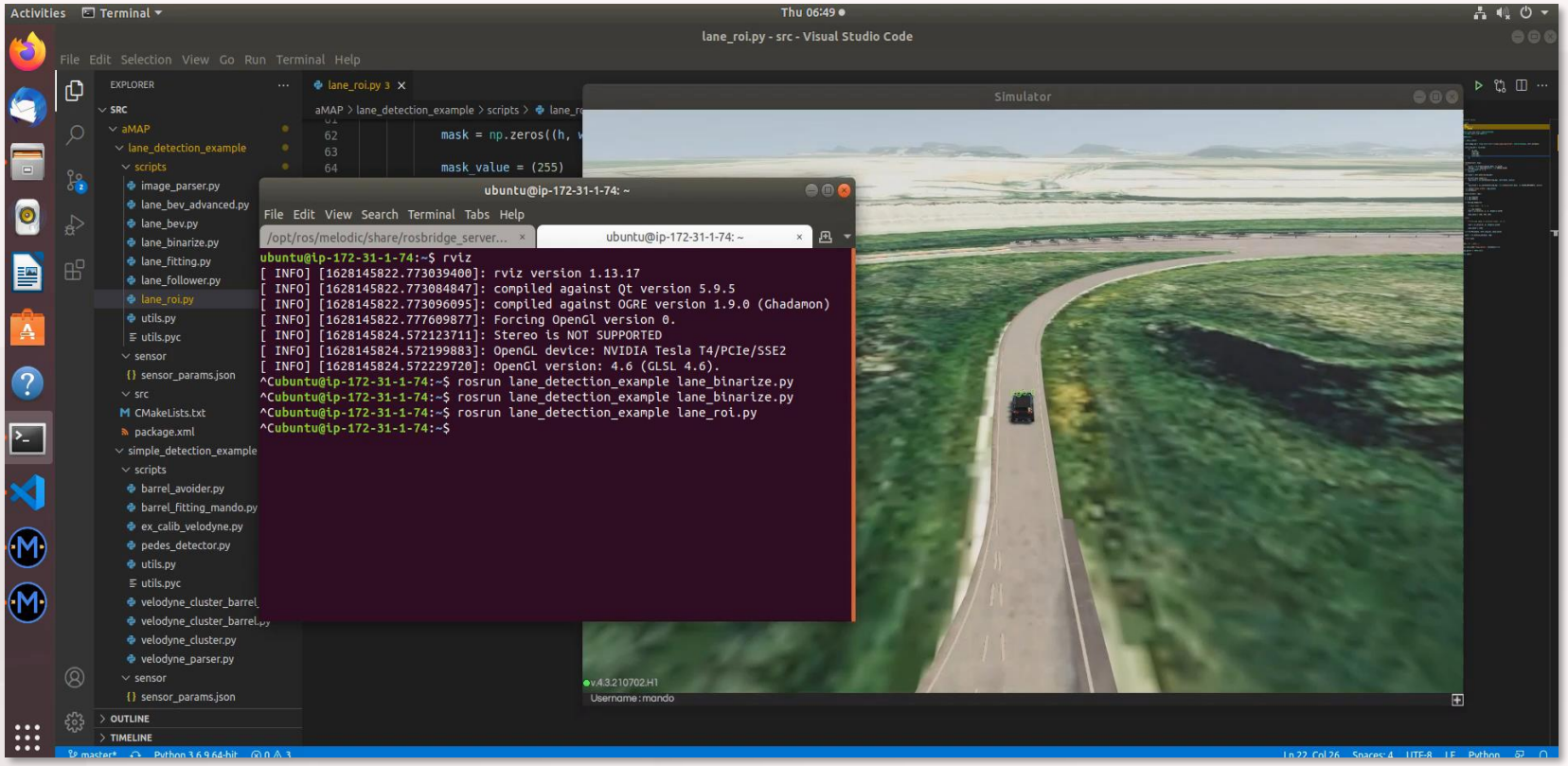
☑ 노드 실행 결과



Region of Interest

1 RoI 이미지 설정

☑ 실행 영상



Bird's eye view

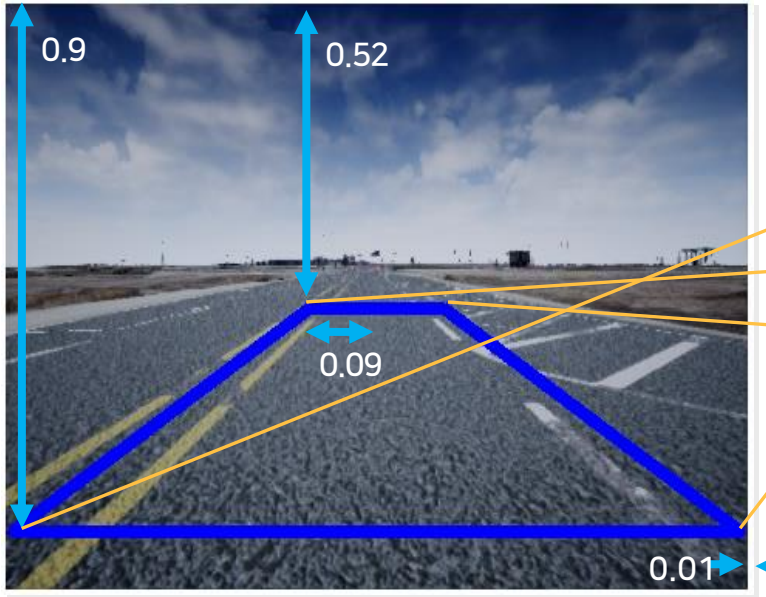
1 Image warping

- 특정 영역을 위에서 내려다 본 이미지처럼 변환하기 위해 warping
- 본래 3d 공간에 대한 정보 없이 늘렸기 때문에 정확한 좌표는 알기 어려움

Bird's eye view

1 Image warping

- 원본 이미지 내 warping할 영역 정의




```
source_points = np.float32([
    [0.01 * x, 0.9 * y],
    [(0.5 * x) - (x*0.09), (0.52)*y],
    [(0.5 * x) + (x*0.09), (0.52)*y],
    [x - (0.01 * x), 0.9 * y]
    ])
```

Bird's eye view

1 Image warping

- 이미지 warping 메소드 정의
- python cv2의 getPerspectiveTransform와 warpPerspective 사용

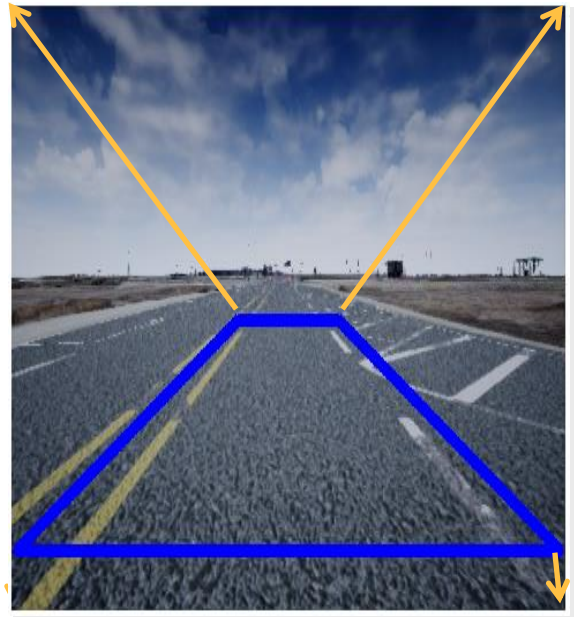
```
12
13  def warp_image(img, source_prop):
14
15      image_size = (img.shape[1], img.shape[0])
16
17      x = img.shape[1]
18      y = img.shape[0]
19
20      destination_points = np.float32([
21      [0, y],
22      [0, 0],
23      [x, 0],
24      [x, y]
25      ])
26
27      source_points = source_prop * np.float32([[x, y]]* 4)
28
29      perspective_transform = cv2.getPerspectiveTransform(source_points, destination_points)
30
31      warped_img = cv2.warpPerspective(img, perspective_transform, image_size, flags=cv2.INTER_LINEAR)
32
33      return warped_img
34
```

Bird's eye view

1 Image warping

- 이미지 warping 메소드 정의
- 29cv2.getPerspectiveTransform(source_points, destination_points) : source pts와 destination pts를 대응시키는 warping transform 오퍼레이터 생성
- 31: cv2.warpPerspective : 이미지를 cv2.getPerspectiveTransform로 생성된 오퍼레이터로 warping

```
29    perspective_transform = cv2.getPerspectiveTransform(source_points, destination_points)
30
31    warped_img = cv2.warpPerspective(img, perspective_transform, image_size, flags=cv2.INTER_LINEAR)
32
33    return warped_img
34
```



Bird's eye view

1 Image warping

- 이전에 만들어 둔 image parser 노드에 image_warp()를 추가

```
class IMGParser:
    def __init__(self):

        self.image_sub = rospy.Subscriber("/image_jpeg/compressed", CompressedImage, self.callback)
        self.img_bgr = None

        self.source_prop = np.float32([[0.01, 0.80],
                                       [0.5 - 0.14, 0.52],
                                       [0.5 + 0.14, 0.52],
                                       [1 - 0.01, 0.80]
                                       ])

    def callback(self, msg):
        try:
            np_arr = np.fromstring(msg.data, np.uint8)
            self.img_bgr = cv2.imdecode(np_arr, cv2.IMREAD_COLOR)
        except CvBridgeError as e:
            print(e)

        img_warp = warp_image(self.img_bgr, self.source_prop)

        img_concat = np.concatenate([self.img_bgr, img_warp], axis=1)

        cv2.imshow("Image window", img_concat)
        cv2.waitKey(1)

```

Bird's eye view

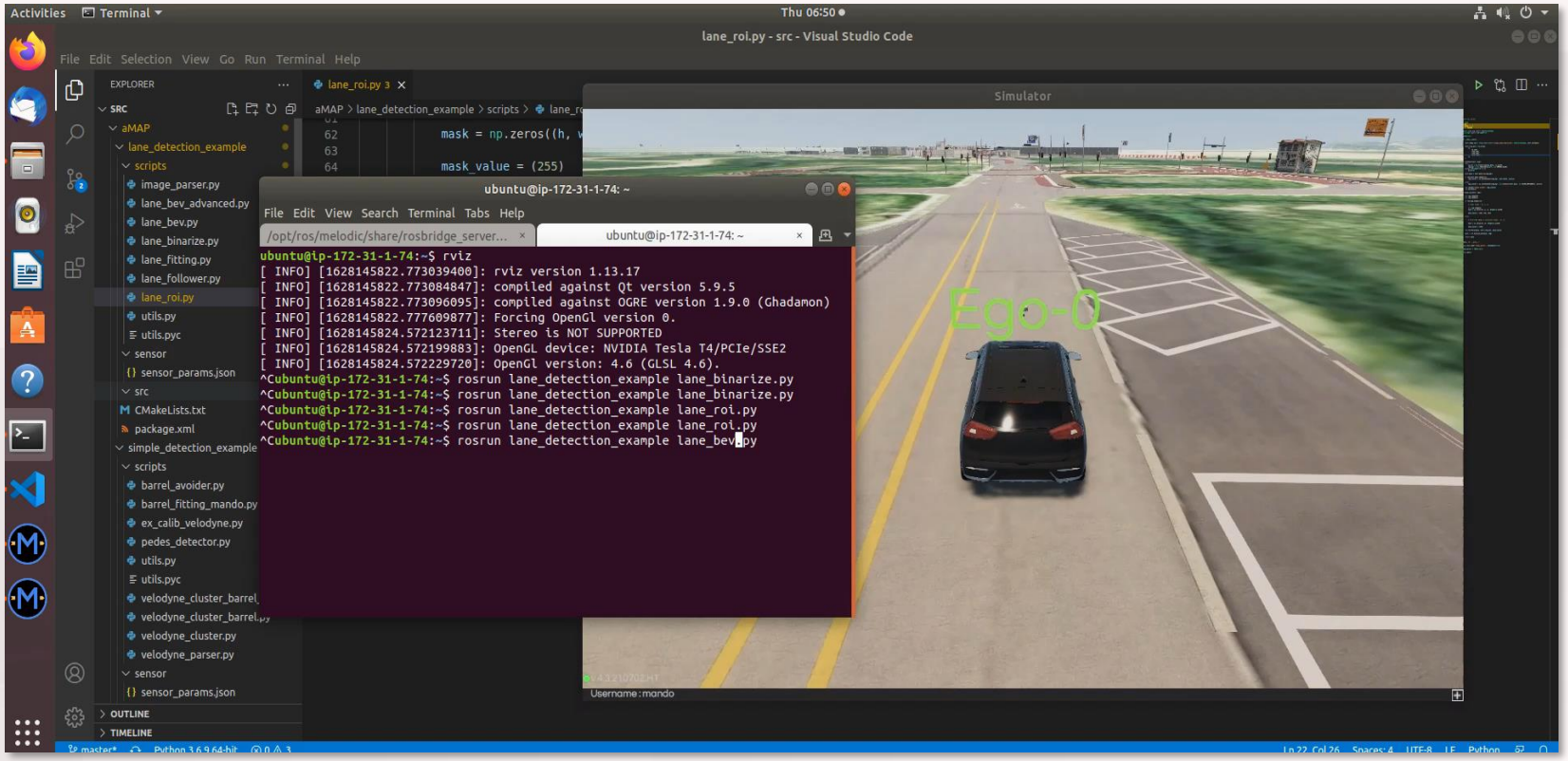
1 Image warping

☑ 노드 실행 결과

Bird's eye view

1 Image warping

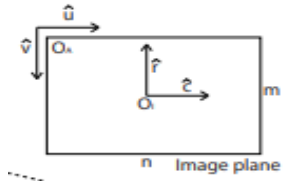
☑ 실행 영상

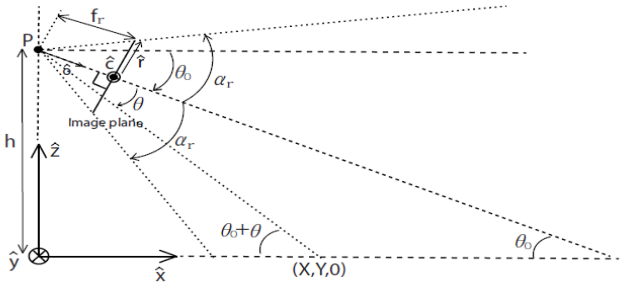


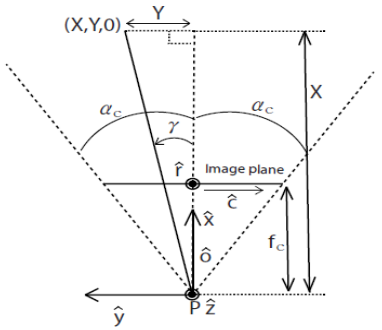
Bird's eye view

2 Adaptive Inverse Perspective Module

- 2d 이미지 내의 도로 마크들을 3d로 reconstruct
- https://irap.kaist.ac.kr/publications/jjeong-2016-urai.pdf







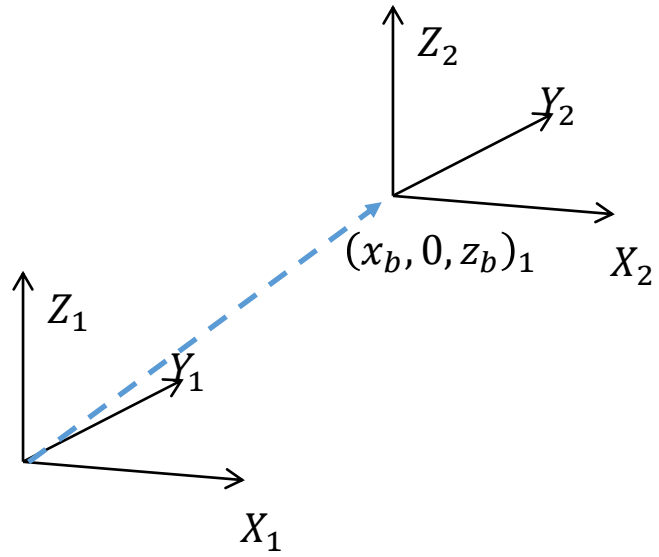
$$\mathbf{X}(\mathbf{v}) = \mathbf{h}\frac{\tan(\theta_0)\left(1-2\frac{\mathbf{v}-1}{\mathbf{m}-1}\right)\tan(\alpha_r)-1}{\tan(\theta_0)+\left(1-2\frac{\mathbf{v}-1}{\mathbf{m}-1}\right)\tan(\alpha_r)} \qquad \mathbf{Y}(\mathbf{u},\mathbf{v}) = \left(1-2\frac{\mathbf{u}-1}{\mathbf{n}-1}\right)\tan(\alpha_c)\mathbf{X}(\mathbf{v})$$

Bird's eye view

2 Adaptive Inverse Perspective Module

- 아래와 같이 scripts/utils.py에 bev_transfrom class 추가

```
self.RT_b2g = np.matmul(np.matmul(traslationMtx(xb, 0, zb), rotationMtx(np.deg2rad(-90), 0, 0)),
                        rotationMtx(0, 0, np.deg2rad(180)))
```



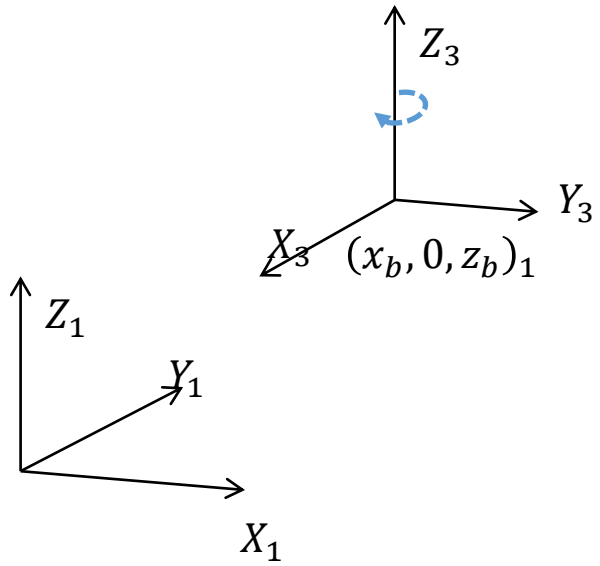
$$\begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x_b \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & z_b \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \\ 1 \end{bmatrix}$$

Bird's eye view

2 Adaptive Inverse Perspective Module

- 아래와 같이 scripts/utils.py에 bev_transfrom class 추가

```
self.RT_b2g = np.matmul(np.matmul(traslationMtx(xb, 0, zb), rotationMtx(np.deg2rad(-90), 0, 0)),
                        rotationMtx(0, 0, np.deg2rad(180)))
```



$$\begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x_b \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & z_b \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 & x_b \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & z_b \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos\psi & -sin\psi & 0 & 0 \\ sin\psi & cos\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_3 \\ Y_3 \\ Z_3 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 & x_b \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & z_b \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(-90°) & -sin(-90°) & 0 & 0 \\ sin(-90°) & cos(-90°) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_3 \\ Y_3 \\ Z_3 \\ 1 \end{bmatrix}$$

Bird's eye view

3 BEV 노드 만들기

- 현재 작성중인 패키지 안에 utils.py를 만들고 그 안에 BEVTransform을 생성

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

import rospy
import cv2
import numpy as np
import os, rospkg
import json

from sensor_msgs.msg import CompressedImage
from cv_bridge import CvBridgeError

from utils import BEVTransform

class IMGParser:
    def __init__(self):

        self.image_sub = rospy.Subscriber("/image_jpeg/compressed", CompressedImage, self.callback)
        self.img_bgr = None

    def callback(self, msg):
        try:
            np_arr = np.fromstring(msg.data, np.uint8)
            self.img_bgr = cv2.imdecode(np_arr, cv2.IMREAD_COLOR)
        except CvBridgeError as e:
            print(e)
```

Bird's eye view

3 BEV 노드 만들기

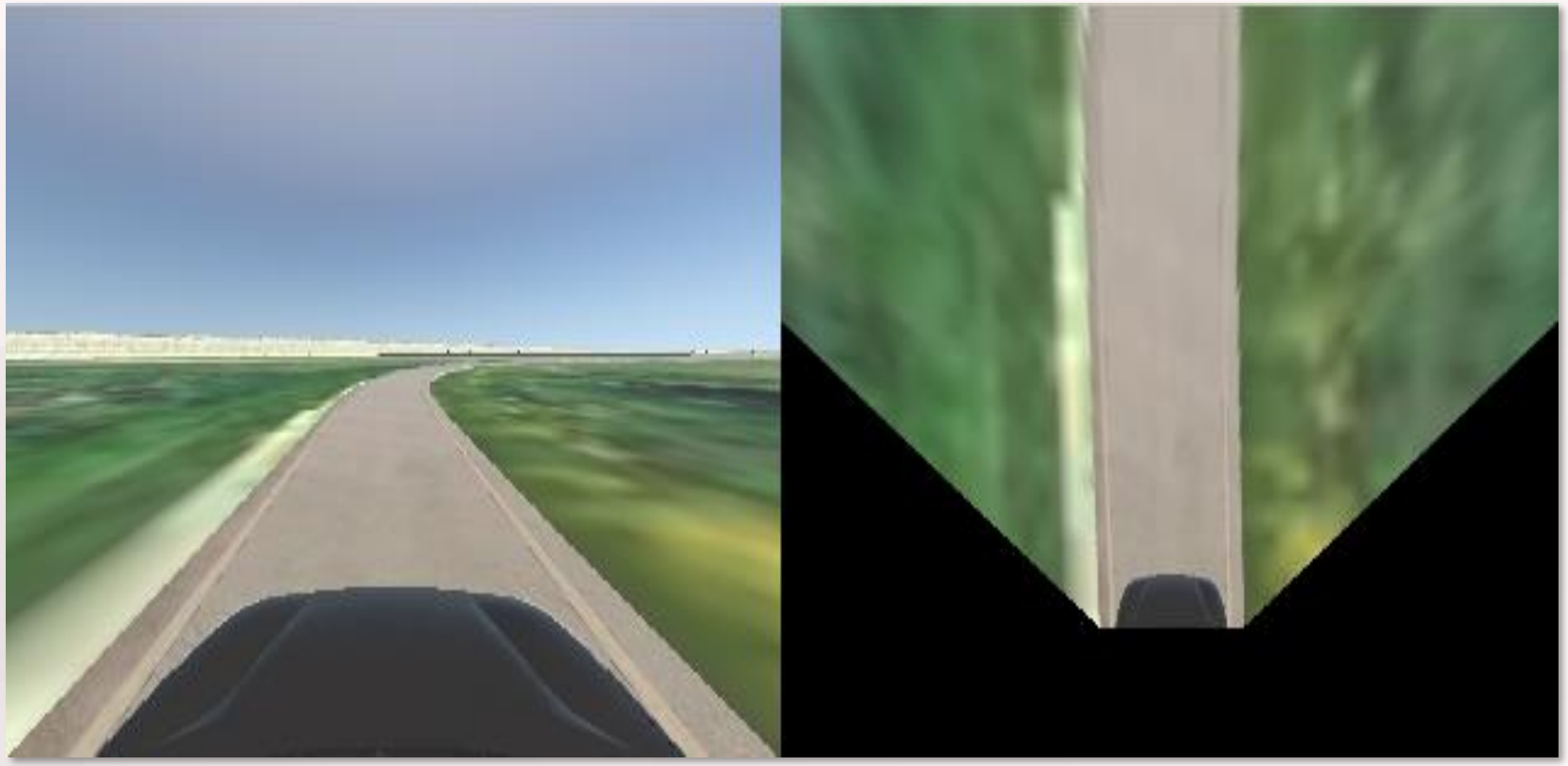
- 생성된 lane_birdview.py 의 메인 루프
- 30: Rospack() : ros 패키지의 Path를 찾아낼 때 쓸수 있음
- 32: .get_path("package의 이름")
- 34: Package의 sensor_params.json 파일을 찾아서 load
- 37: 이전에 저장해둔 params_cam load
- 41, 42: Image parser와 bev module 정의
- 44: While loop를 30hz로 실행
- 50: 이미지 BEV 변환

```
27
28  def main():
29
30      rp = rospkg.RosPack()
31
32      currentPath = rp.get_path("lane_detection_example")
33
34      with open(os.path.join(currentPath, 'sensor/sensor_params.json'), 'r') as fp:
35          sensor_params = json.load(fp)
36
37      params_cam = sensor_params["params_cam"]
38
39      rospy.init_node('lane_birdview', anonymous=True)
40
41      image_parser = IMGParser()
42      bev_op = BEVTransform(params_cam=params_cam)
43
44      rate = rospy.Rate(30)
45
46      while not rospy.is_shutdown():
47
48          if image_parser.img_bgr is not None:
49
50              img_warp = bev_op.warp_bev_img(image_parser.img_bgr)
51
52              img_concat = np.concatenate([image_parser.img_bgr, img_warp], axis=1)
53
54              cv2.imshow("birdview", img_concat)
55
56              cv2.waitKey(1)
57
58              rate.sleep()
59
60
61  if __name__ == '__main__':
62
63      main()
64
```

Bird's eye view

4 Adaptive Inverse Perspective Module

☑ 노드 실행 결과



차선 인지/신호등 인지

1. 차선인지 & Binarization
2. Region of Interest & Bird’s eyes view
3. Curve fitting & 신호등 인지

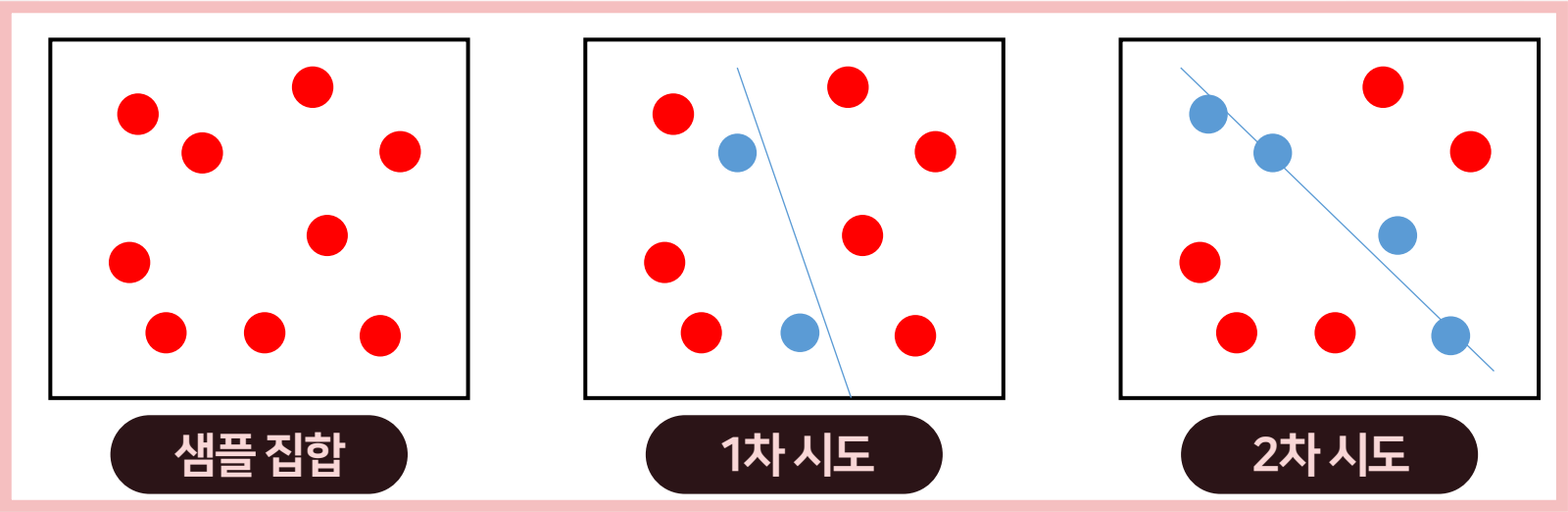
3

Curve fitting & 신호등 인지

Curve fitting

1 RANSAC

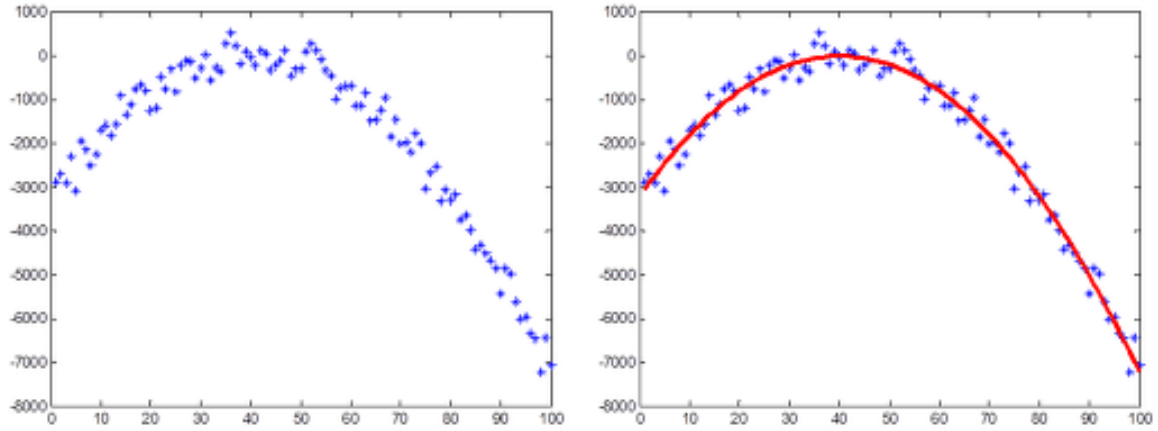
- RANSAC (RANdom Sample Consenssus)
  - 정상 분포에 속해 있는 데이터 집합으로 회귀 분석을 수행할 수 있게 해주는 기법
    말 그대로 랜덤으로 얻은 무작위 샘플을 가지고 데이터를 비교하는 시스템
  - 데이터들을 fitting(근사) 할 때 사용함
  - 하나의 경향성을 기준으로 하여 데이터를 추정



Curve fitting

1 RANSAC

- RANSAC (RANdom Sample Consenssus)
  - 데이터들을 fitting할 때 사용할 수 있는 방법 중 하나는 최소제곱법
  - 최소제곱법은 데이터가 많이 흩어져 있지 않다면 훌륭한 fitting 결과를 나타낼 수 있음

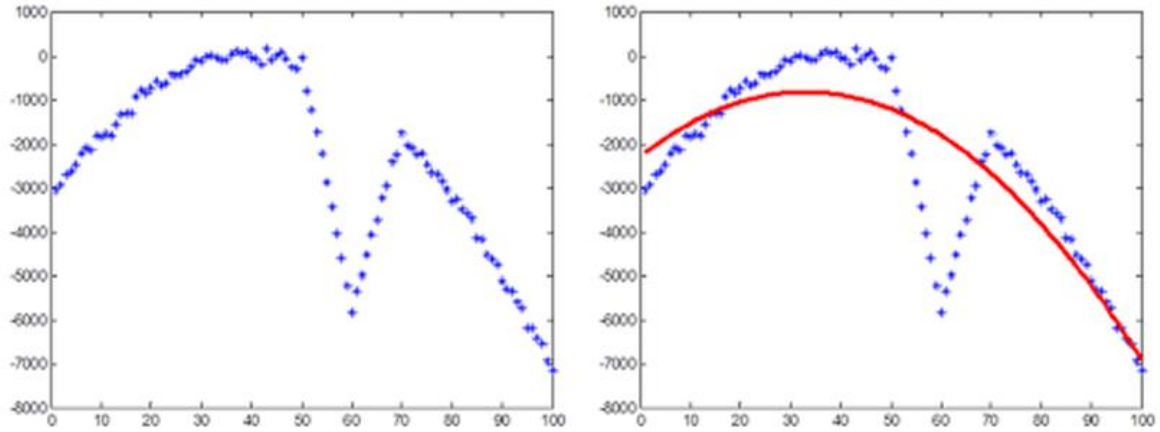


최소제곱법을 사용한 fitting

Curve fitting

1 RANSAC

- RANSAC (RANdom Sample Consenssus)
  - 데이터들을 fitting할 때 사용할 수 있는 방법 중 하나는 최소제곱법
  - 노이즈가 데이터들과 경향이 많이 다르게 나타난다면 최소제곱법으로는 fitting가 잘 이루어 지지 않음

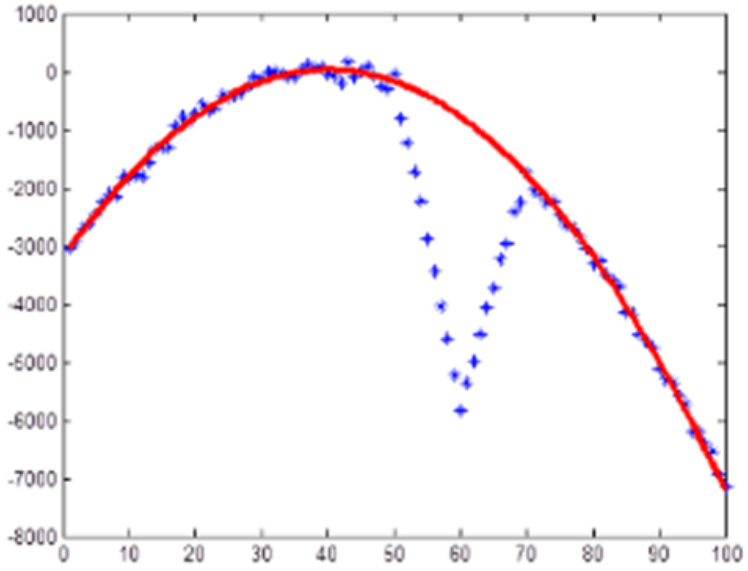


최소제곱법을 사용한 fitting

Curve fitting

1 RANSAC

- RANSAC (RANdom Sample Consenssus)
  - RANSAC을 사용할 경우 노이즈에 강건한 모습을 볼 수 있음



RANSAC을 사용한 fitting

Curve fitting

1 RANSAC

- RANSAC (RANdom Sample Consenssus) 의 절차

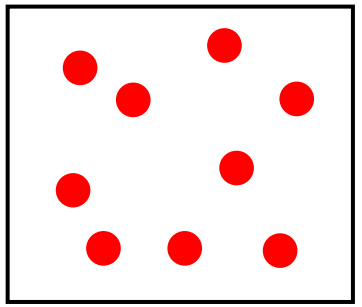
1. 데이터에서 임의의 개수를 선택하여 이를 inlier로 가정하고 회귀 모델을 구함
2. 나머지 데이터들을 회귀 모델과 비교하여 사용자가 지정한 허용 오차내에 있는 데이터들을 inlier로 포함
3. 재구성된 inlier를 이용해 다시 회귀 모델을 구함
4. 회귀 모델과 inlier의 오차를 측정
5. 이 오차가 사용자가 지정한 값 내에 있거나 지정한 알고리즘 구동 반복회수에 도달했으면 종료
6. 만약 이 조건을 만족하지 못하면 1단계부터 다시 시작

Curve fitting

1 RANSAC

RANSACRegressor()

```
>>> ransac = RANSACregressor (LinearRegression(),
                              max_trials=100,
                              min_samples=50,
                              residual_metric=lambda x: np.sum(sp.abs(x),
                              axis=1,
                              residual_threshold=5.0,
                              random_state=0)
```



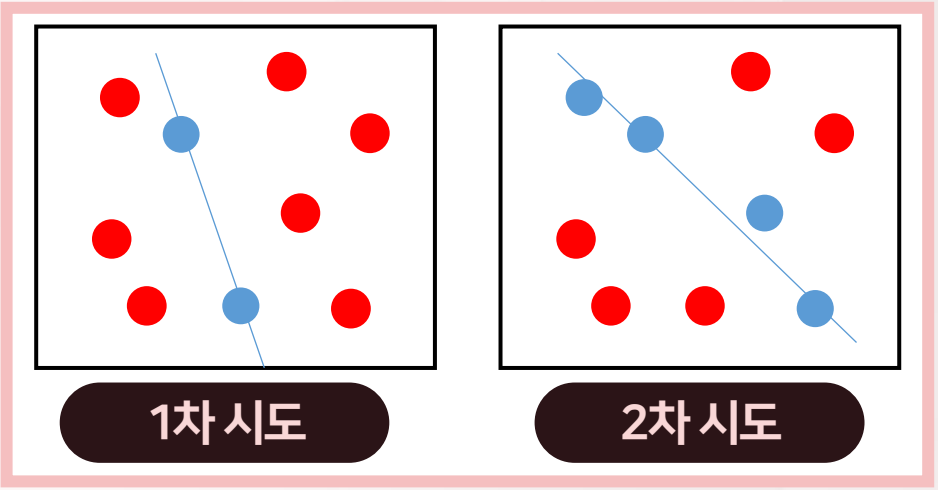
샘플 집합

Curve fitting

1 RANSAC

RANSACRegressor의 인자

- max_trials : 알고리즘의 최대 반복 횟수
- min_samples : inlier 값으로 무작위로 선택할 샘플의 최소 개수
- residual_metric : 회귀 모델과 데이터의 오차 측정 함수 지정
- residual_threshold : inlier로 포함시키기 위한 허용 오차



Curve fitting

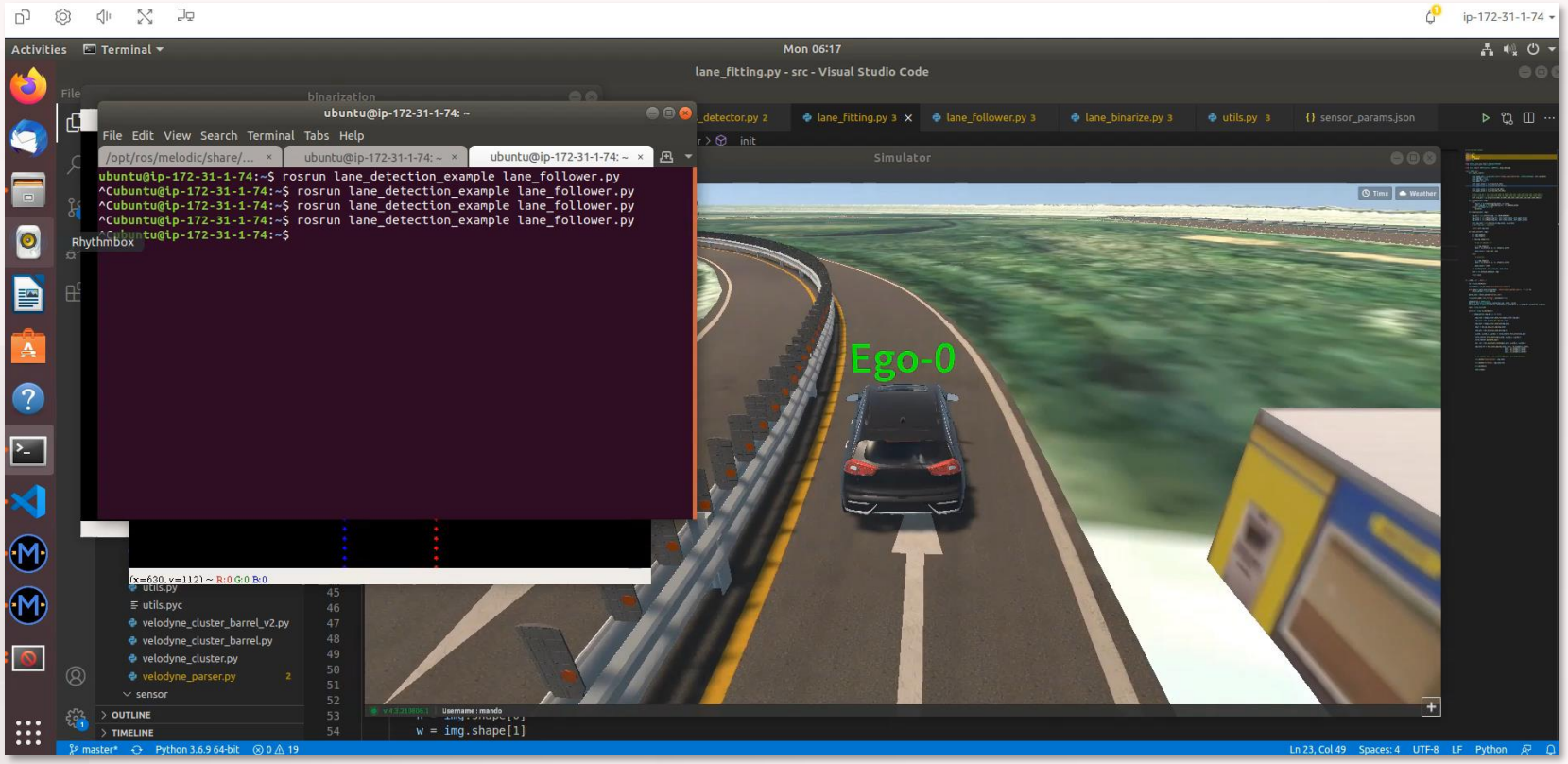
1 Adaptive Inverse Perspective Module

- Binarized image를 입력 후 3차식으로 fitting을 시도한 결과

Curve fitting

1 Adaptive Inverse Perspective Module

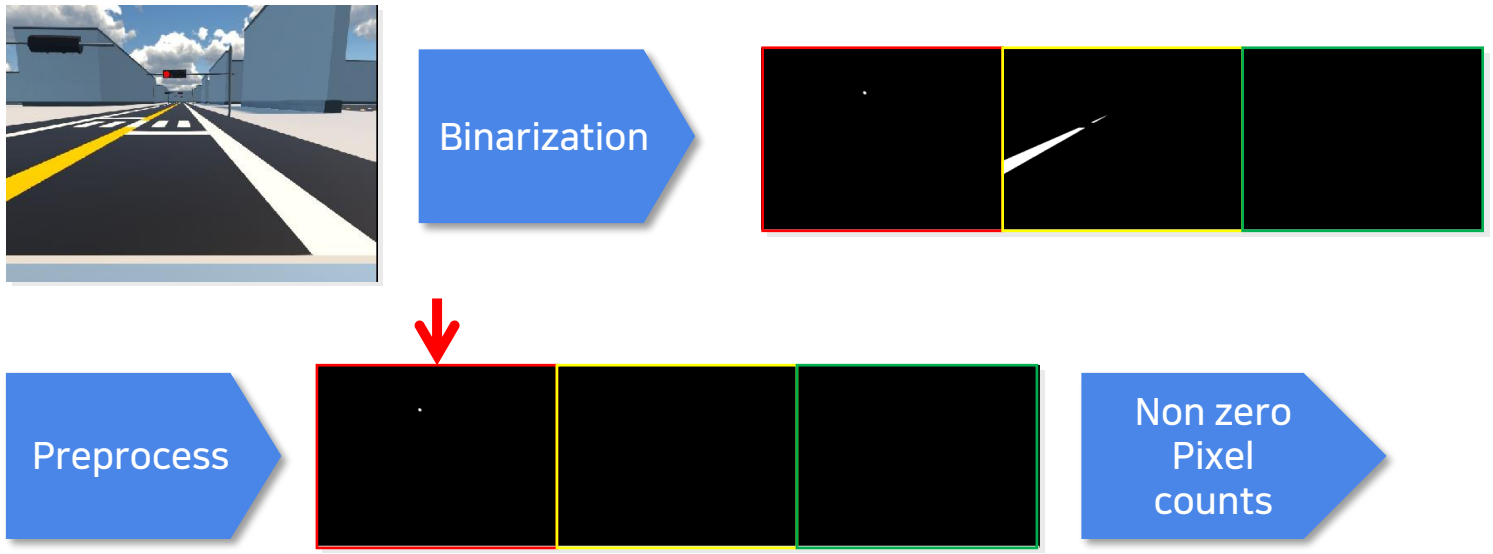
- Lane detection and control 실행 결과



신호등 인지

1 신호등 인지 과정 단계들

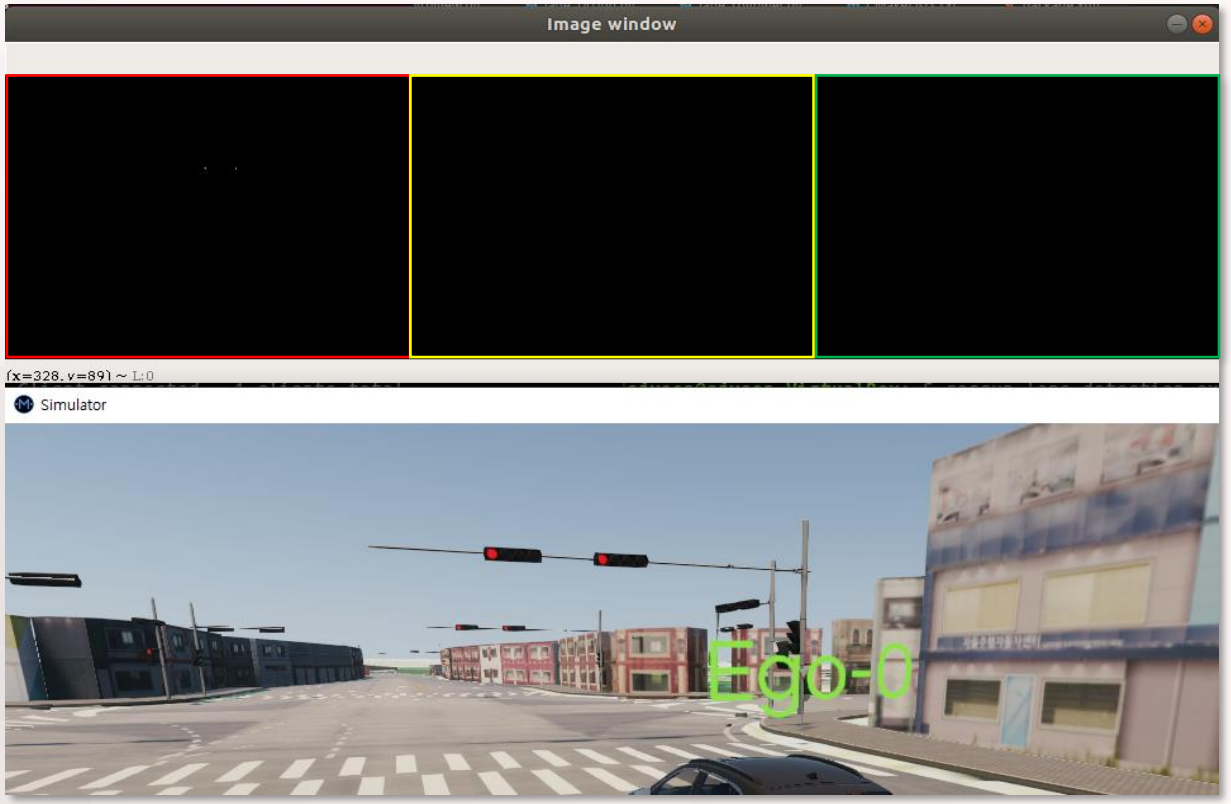
1. 차선 인지에 쓰던 이진화 과정으로 3가지 binarized image를 추출
2. 픽셀 개수를 세서, 가장 높은 수를 가진것으로 판단
3. 노란 차선과 빨간색 아스팔트 영역도 들어갈 수 있으므로 RoI 를 정의해서 컷



신호등 인지

1 신호등 인지 과정 단계들

- HSV 채널 중 hue를 가지고 이진화 시도한 결과



신호등 인지

2 신호등 색상 이진화 예시

- HSV 채널 중 hue를 가지고 이진화 시도한 결과