Lecture 6. 카메라

Outline

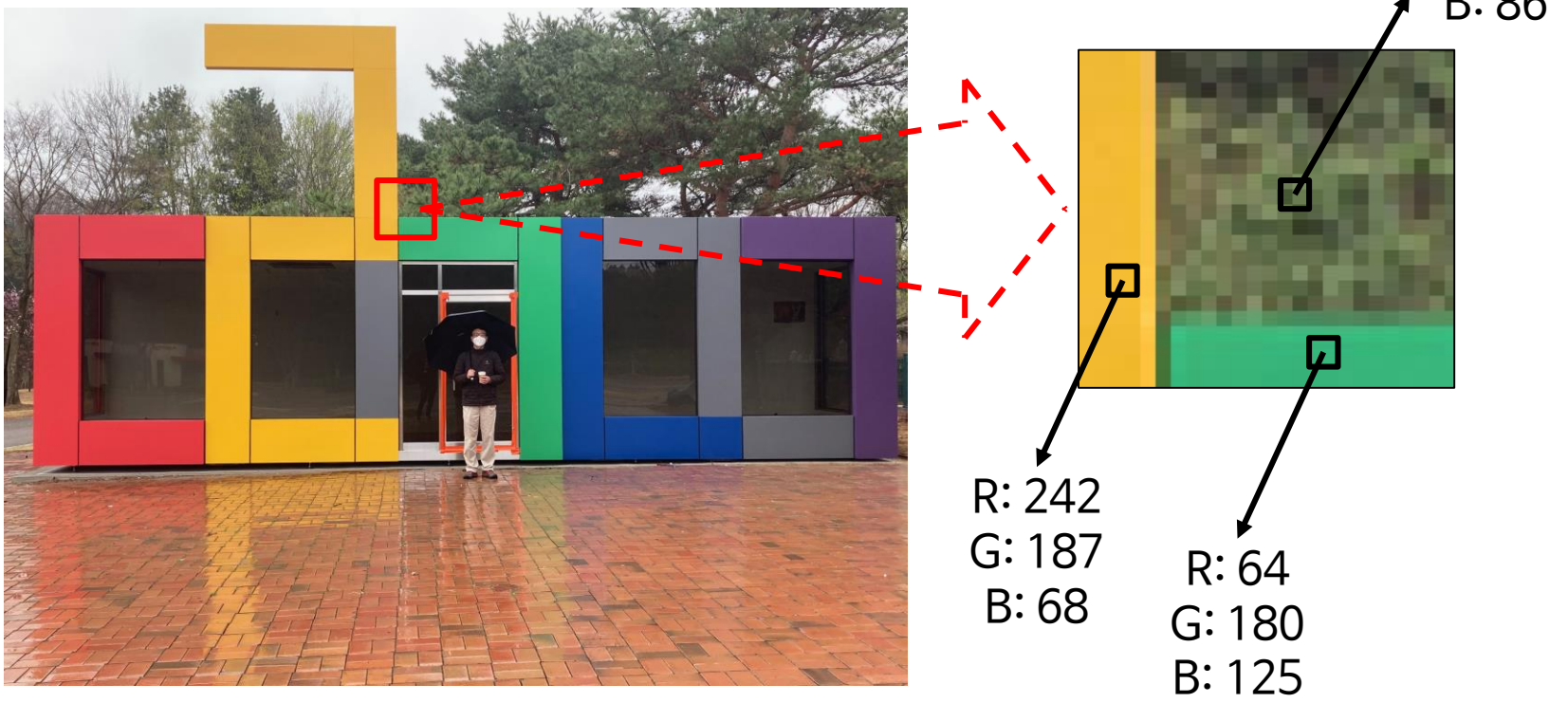
- 카메라와 영상처리
- 카메라를 이용한 차선 추종

카메라

- 카메라(camera, 영상 센서, ...)
  - ROS에서는 다양한 종류의 카메라를 지원함
- 카메라의 출력: 영상 (image), 30fps or 60fps
  - pixel들의 집합

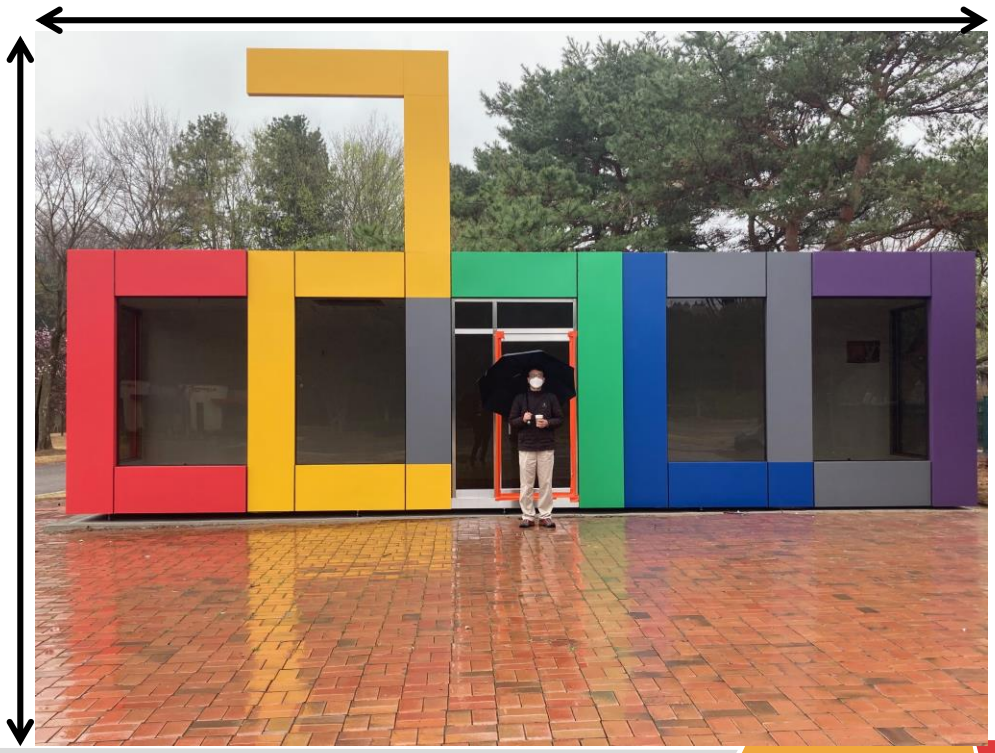
카메라

- 카메라의 출력: 영상 (image)
  - pixel들의 집합
  - 한 pixel은 RGB 세 개의 값으로 구성됨



카메라

- 카메라의 출력: 영상 (image)
  - pixel들의 집합
  - 한 pixel은 RGB 세 개의 값으로 구성됨
  - 해상도: 가로 세로 pixel의 개수



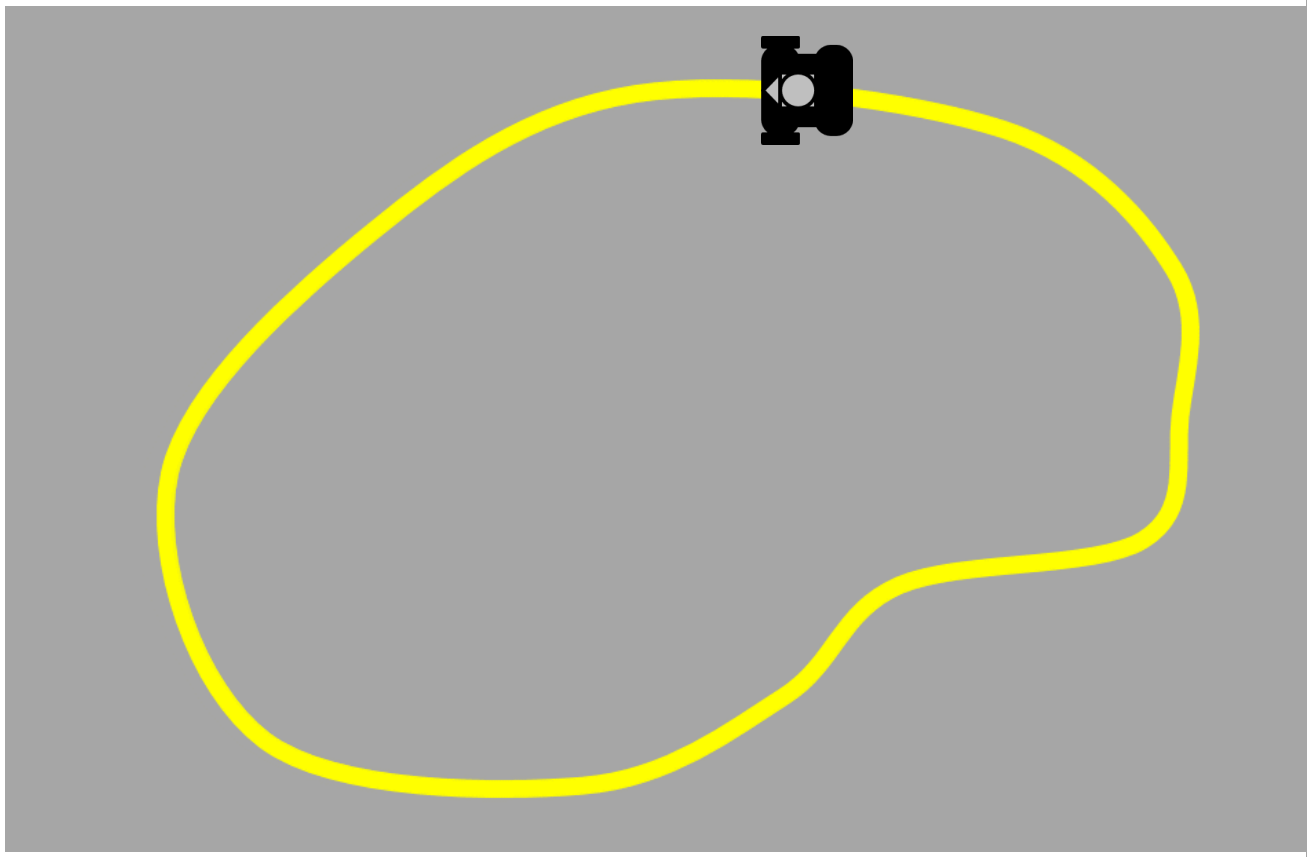
Full HD: 1920 x 1080
4K: 3840 x 2160

영상 처리

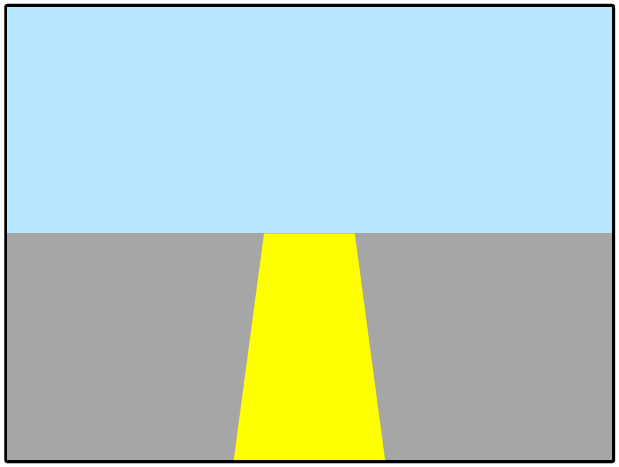
- 영상 처리
  - 목적에 맞게 영상을 가공
  - 예) 파란색 부분만 추출

예제 목표

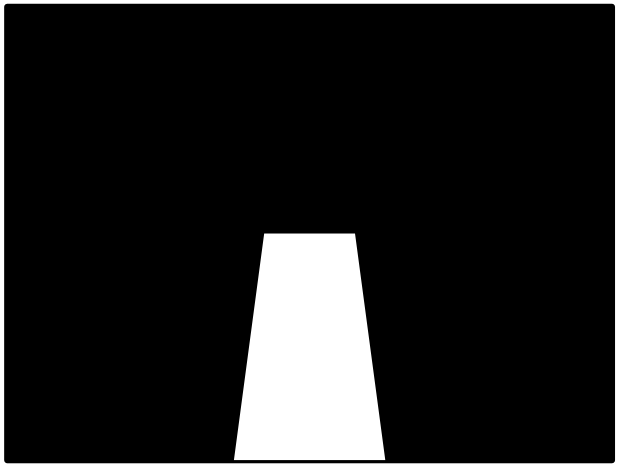
- 노란색 차선을 따라서 주행해 보자.



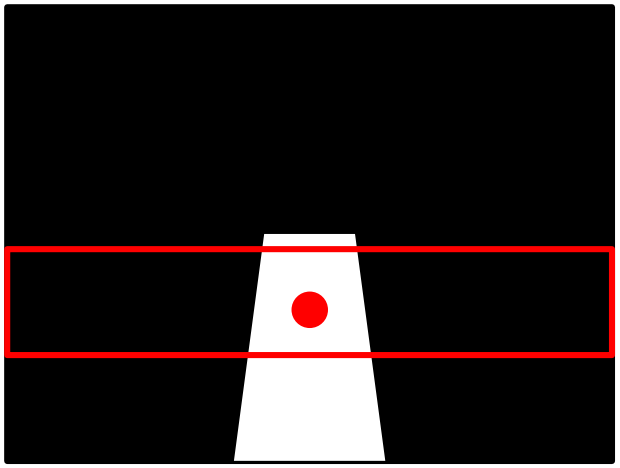
제어 절차



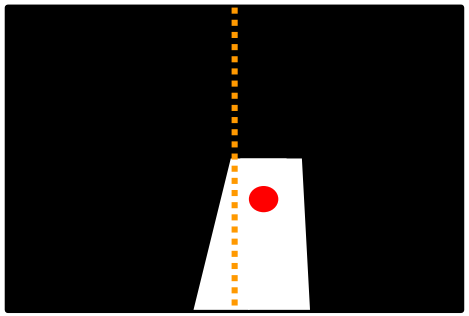
카메라 화면



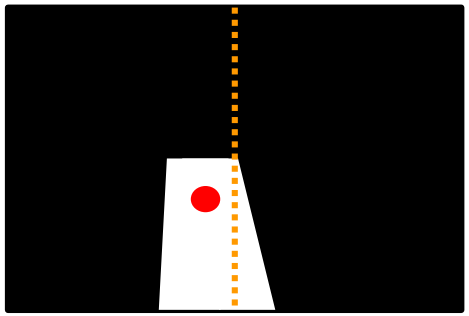
노란색 영역 추출



관심 있는 부분에서 노란색 영역의 중심점 계산



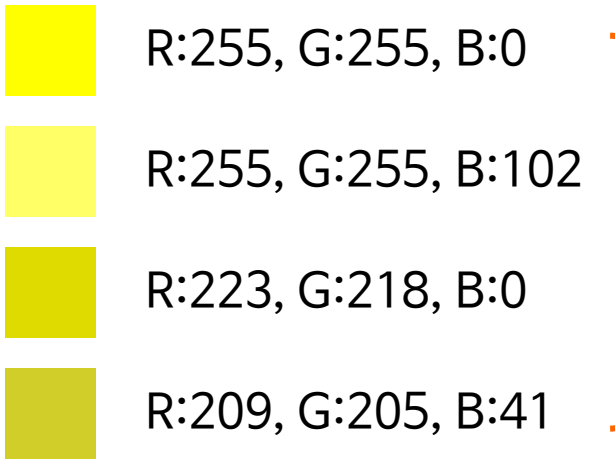
노란색 영역의 중심이 오른쪽에 있으면?
우회전



노란색 영역의 중심이 왼쪽에 있으면?
좌회전

특정 색 영역 추출

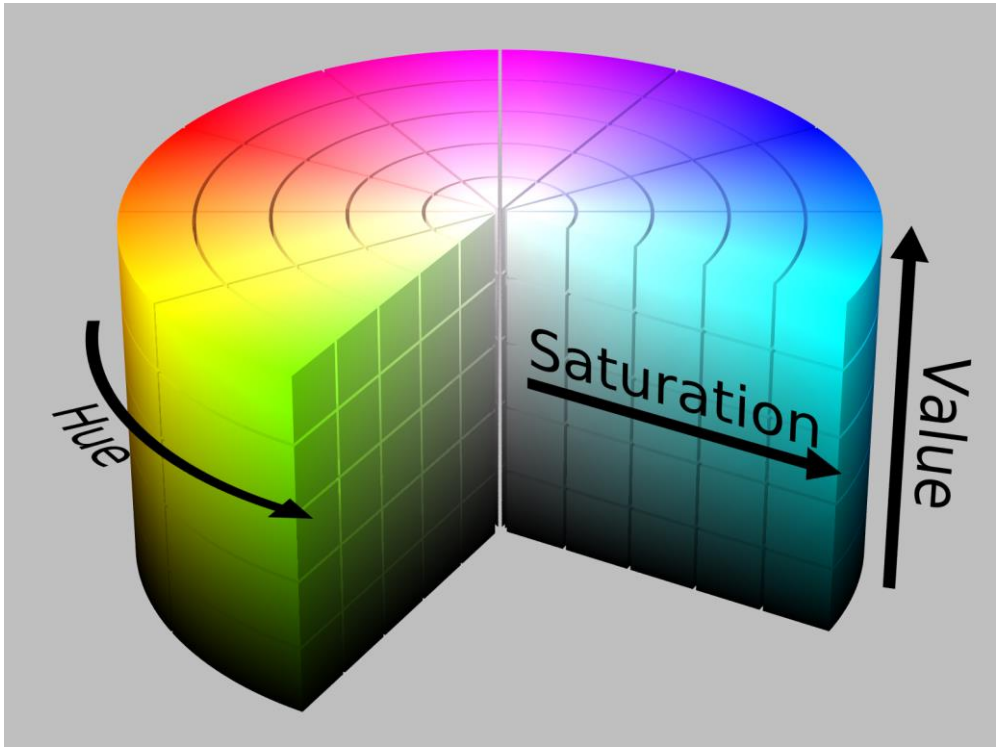
- RGB로는 특정 색 추출이 쉽지 않음

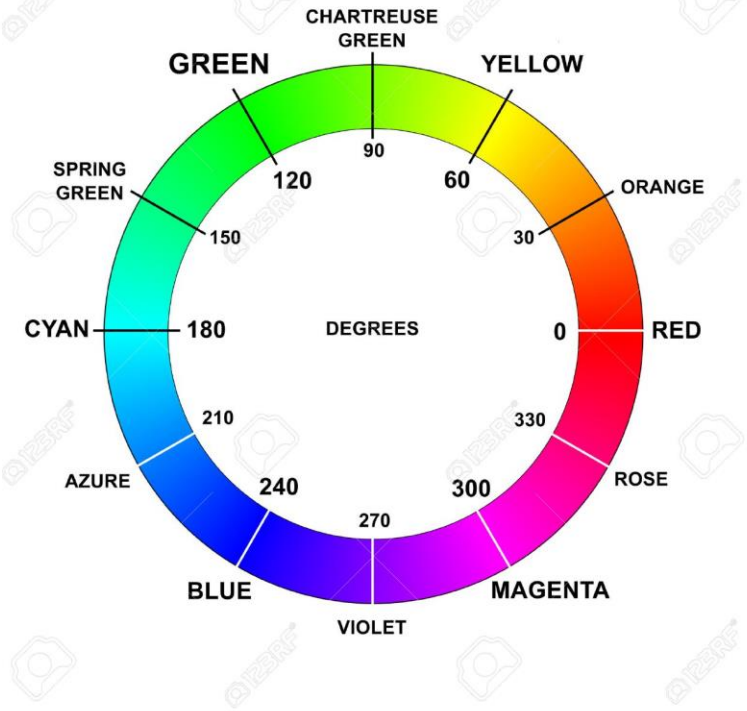



```
if(... R ... G ... B ...) {
    Yellow
}
```

특정 색 영역 추출

- HSV 색 공간
  - Hue (색상), Saturation (채도), Value (명도)





특정 색 영역 추출

- RGB → HSV

R:255, G:255, B:0	H: 60, S: 255, V: 255
R:255, G:255, B:102	H: 60, S: 154, V: 255
R:223, G:218, B:0	H: 59, S: 255, V: 224
R:209, G:205, B:41	H: 59, S: 206, V: 210

H가 60 근처이면 노란색
채도는? 대략 높은 값
명도는? 대략 높은 값

```
if((40<=H<=80) && (S>=100) && (V>=100)) {
   Yellow
}
```

OpenCV

- 필요한 작업들
  - RGB to HSV 변환
  - 노란색 영역 추출
  - 중심 구하기

- OpenCV에서 라이브러리로 제공
  - 영상 처리에서 폭넓게 사용됨

Outline

- 카메라와 영상처리
- 카메라를 이용한 차선 추종

노란색 차선을 추종

```
#!/usr/bin/env python3

import rospy, cv2, cv_bridge, numpy
from sensor_msgs.msg import Image
from geometry_msgs.msg import Twist

class Follower:
    def __init__(self):
        self.bridge = cv_bridge.CvBridge()
        cv2.namedWindow("window", 1)

        self.image_sub = rospy.Subscriber('/camera/image',
                                          Image, self.image_callback)

        self.err = 0
```

영상 관련 토픽을 구독하고 처리하기 위해 필요한 패키지들

OpenCV bridge를 이용

카메라 영상을 화면에서 보기 위해

카메라 영상 토픽을 구독

토픽 이름

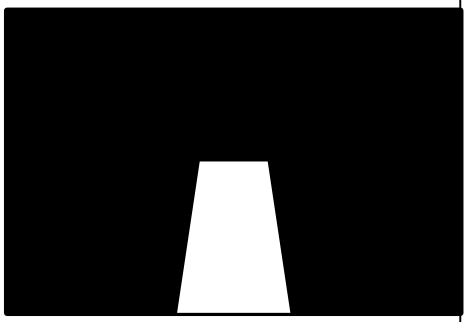
노란색 차선을 추종

```
def image_callback(self, msg):
    image = self.bridge.imgmsg_to_cv2(msg,desired_encoding='bgr8')
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

    lower_yellow = numpy.array([ 20, 100, 100])
    upper_yellow = numpy.array([ 40, 255, 255])
    mask = cv2.inRange(hsv, lower_yellow, upper_yellow)

    h, w, d = image.shape
    search_top = 3*h/4
    search_bot = 3*h/4 + 20
    mask[0:search_top, 0:w] = 0
    mask[search_bot:h, 0:w] = 0

    M = cv2.moments(mask)
    if M['m00'] > 0:
        cx = int(M['m10']/M['m00'])
        cy = int(M['m01']/M['m00'])
        cv2.circle(image, (cx, cy), 20, (0,0,255), -1)
```

OpenCV에서 처리하는 구조로

HSV로 변환

지정된 범위의 HSV 값을 가지는 영역 추출
H: 20~40, S: 100~255, V: 100~255
주의: OpenCV에서 반으로 나뉜 H를 사용 (0~180)



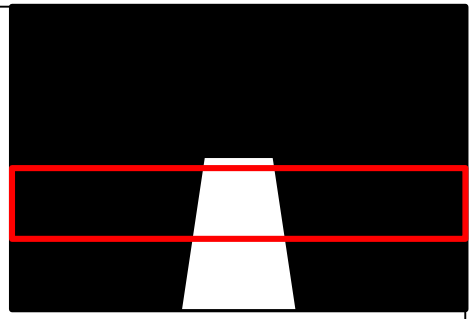
노란색 차선을 추종

```
h, w, d = image.shape
search_top = 3*h/4
search_bot = 3*h/4 + 20
mask[0:search_top, 0:w] = 0
mask[search_bot:h, 0:w] = 0
```

가로 세로 크기 구하기

이 영역에 대해

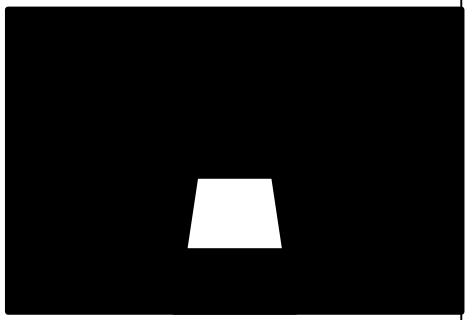
나머지 부분 제거



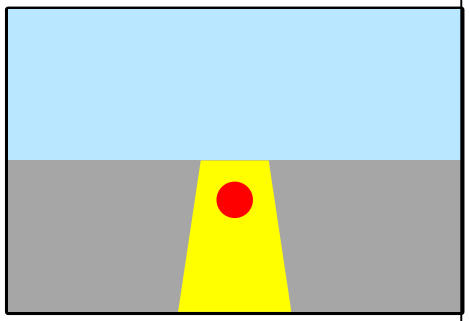
```
M = cv2.moments(mask)
if M['m00'] > 0:
    cx = int(M['m10']/M['m00'])
    cy = int(M['m01']/M['m00'])
    cv2.circle(image, (cx, cy), 20, (0,0,255), -1)
    self.err = cx - w/2

cv2.imshow("window", image)
cv2.waitKey(3)
```

M['m00']이 0이면
추출된 영역이 없다는 의미

무게 중심의 x, y
좌표 구하기



시각적 효과를 위해 무게 중심에
빨간 원이 그려진 영상을 화면에
보여 줌



노란색 차선을 추종

```
    def image_callback(self, msg):
        ...
            self.err = cx - w/2
        ...
```

노란색 영역의 중심이 화면의
가운데에서 얼마나 떨어져 있는가?
+: 화면의 가운데 보다 오른쪽에
-: 화면의 가운데 보다 왼쪽에

```
rospy.init_node('follower')
follower = Follower()
cmd_vel_pub = rospy.Publisher('cmd_vel', Twist, queue_size=1)

rate = rospy.Rate(30)
cmd = Twist()
cmd.linear.x = 0.5
```

기본적으로는 전진

```
while not rospy.is_shutdown():
    cmd.angular.z = -float(follower.err)/100.;
    cmd_vel_pub.publish(cmd)
    rate.sleep()
```

err 값에 따라 왼쪽, 오른쪽으로
err가 양수 → - 각속도, 우회전
err가 음수 → + 각속도, 좌회전
err의 양에 비례해서 더 큰 각속도

OpenCV 설치 및 패키지 생성

- OpenCV 설치

```
user@hostname$  sudo apt install libopencv-dev
user@hostname$  sudo apt install ros-noetic-opencv-apps
user@hostname$  sudo apt install ros-noetic-cv-*
```

- 패키지 생성

```
user@hostname$ cd ~/my_ws/src
user@hostname$ catkin_create_pkg followbot rospy roscpp sensor_msgs \
                geometry_msgs image_transport cv_bridge OpenCV
```

OpenCV 설치 및 패키지 생성

- CMakeLists.txt 수정

```
user@hostname$ cd followbot
user@hostname$ gedit CMakeLists.txt
```

  - 다음을 추가

```
set(OpenCV_DIR /usr/share/OpenCV)
```

- 컴파일

```
user@hostname$ cd ../../
user@hostname$ catkin_make
user@hostname$ source devel/setup.bash
```

주행 트랙 생성

- 배포 파일 들을 다운 받은 뒤

```
user@hostname$ cd src/followbot
user@hostname$ mkdir launch
user@hostname$ cp ~/Downloads/course.launch launch/
user@hostname$ mkdir worlds
user@hostname$ cp ~/Downloads/course.world worlds/
user@hostname$ cp ~/Downloads/course.material ./
user@hostname$ cp ~/Downloads/course.png ./
```

- package.xml 수정

```
user@hostname$ gedit package.xml
```

  - 다음을 추가

```
    <exec_depend>gazebo_ros</exec_depend>
    <export>
        <gazebo_ros gazebo_media_path="${prefix}" />
    </export>
```

주행 트랙 생성

- 주행 트랙 실행

```
user@hostname$ cd ../../
user@hostname$ catkin_make
user@hostname$ roslaunch followbot course.launch
```

노드 만들기 및 실행

- 다른 창에서
- 노드 소스 파일 작성

```
user@hostname$ cd ~/my_ws
user@hostname$ source devel/setup.bash
user@hostname$ cd src/followbot
user@hostname$ mkdir scripts
user@hostname$ cd scripts
user@hostname$ gedit followbot.py
```

- 실행 파일 지정

```
user@hostname$ chmod +x followbot.py
```

- 실행

```
user@hostname$ rosrun followbot followbot.py
```