Open CV

Open Source Computer Vision

Department of Electrical Engineering, Incheon National University
Hwasu Lee, Gihoon Song

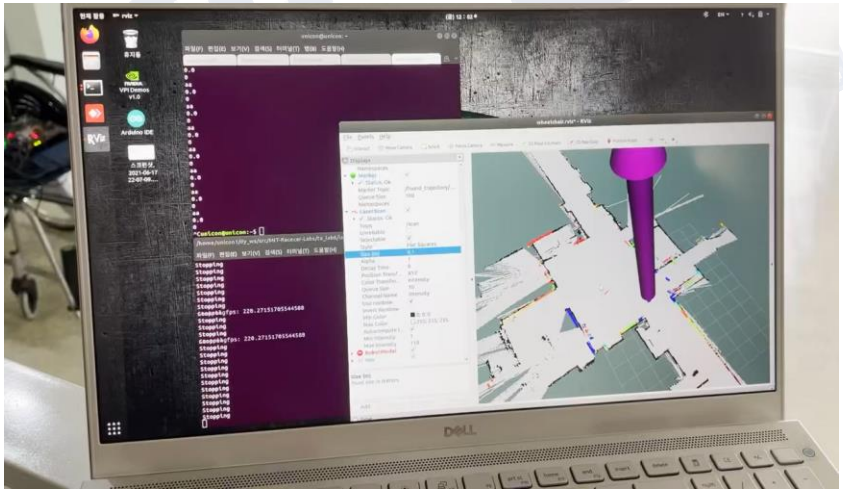
2024.07.15~17

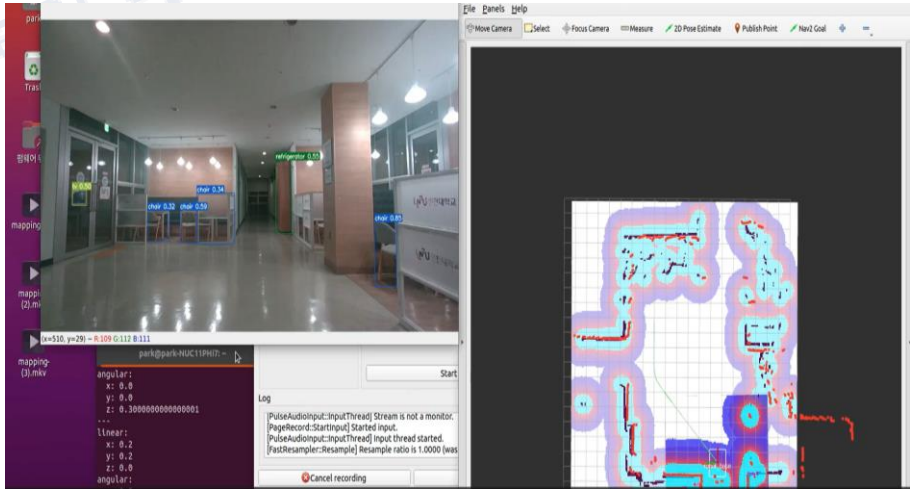
강사 소개

- 이화수 (Hwasu Lee)
- 2015.03 ~ 2022.02: 인천대학교 전기공학과 학사
  2022 ~ 현재: 인천대학교 전기공학과 석사 과정 재학 중
- 무인지능 시스템제어 연구실 소속
  (지도교수: 강창묵 교수, https://uniconlab.wixsite.com/main)
- 관심 분야: 예측 제어, 경로 탐색, 심층 강화학습,
  ROS 1 & ROS 2 기반 SLAM 및 Navigation
- 적용 플랫폼: 자율주행 차량, 모바일 로봇 등
- Projects
  - 자율주행 전동휠체어 핵심 기술 개발
  - 순찰로봇의 도심지 적용을 위한 자율주행 기술 개발
  - 지능형 건물 바닥 청소 로봇 플랫폼 개발
  - 인천공항 터미널 폐기물 수집 및 이송로봇 개발

연구실 수행 프로젝트


자율주행 실증운영




File Panels Help
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
Publish Point
Nav2 Goal
Cancel recording

Contents

1. Open CV 개요

1. Open CV 개요

1) Open CV(Open Source Computer Vision)란?

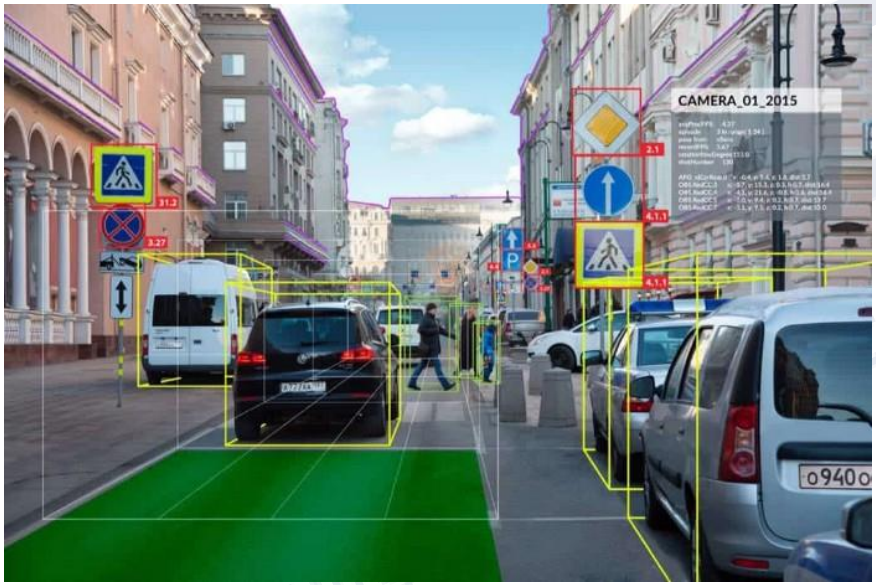
- Open CV는 “**Open Source Computer Vision Library**”의 약자로, 이미지 및 동영상 처리에 사용되는 라이브러리

- C++, Python, Java 등 다양한 프로그래밍 언어를 지원하며, 이미지 프로세싱, 컴퓨터 비전 및 머신러닝 알고리즘에 대한 다양한 함수와 라이브러리 제공

- Open CV는 무료로 제공되며, 비즈니스 및 개인 프로젝트 모두에서 사용 가능

1. Open CV 개요

1) Open CV(Open Source Computer Vision)란?

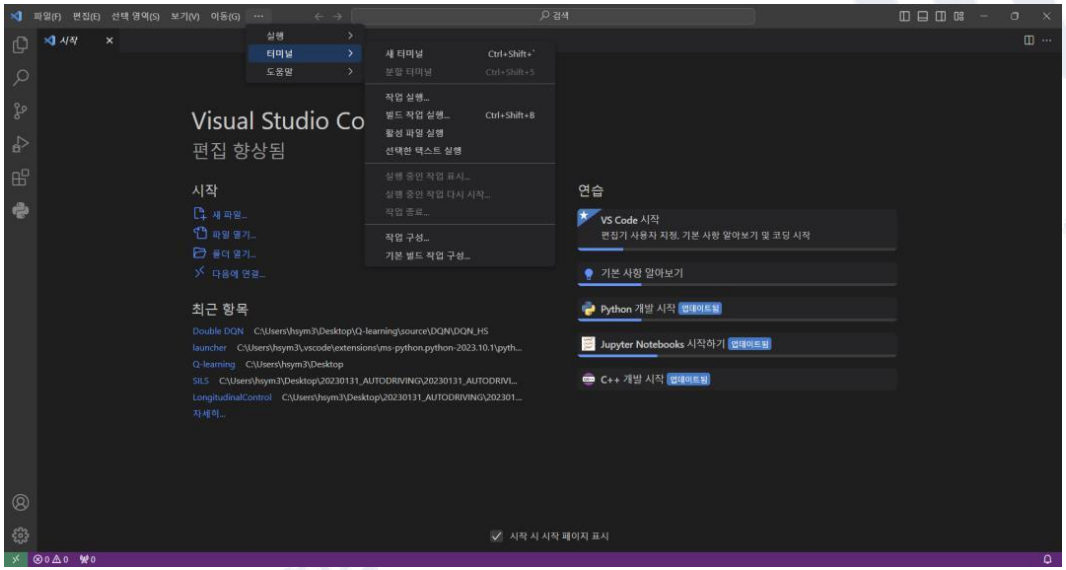


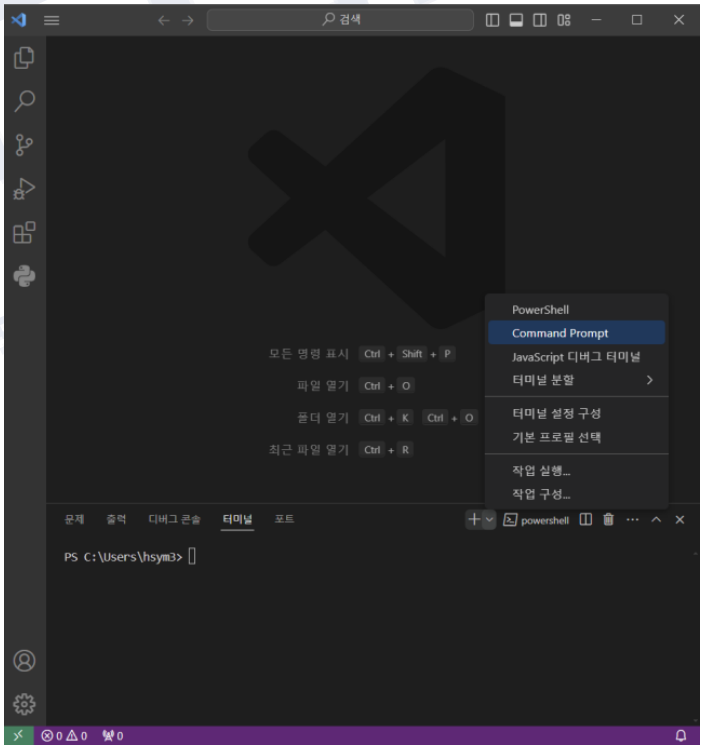
- **Open CV**를 통해 이미지나 영상에서 **객체 검출**, **추적**, **특징 추출**, **패턴 인식** 등의 작업 수행 가능
- Open CV에서는 이미지 필터링, 색 공간 변환, 이미지 모핑, 카메라 보정 등 다양한 기능을 제공
- 최근 **Open CV**와 **Deep Learning**을 결합하여 이미지 및 영상 데이터를 쉽게 처리할 수 있게 되었으며, 이를 기반으로 객체 인식 및 추적 등의 작업을 빠르게 수행할 수 있게 되었음

1. Open CV 개요

2) Open CV 설치 (Python 기반)

➔ VS Code 및 Python3는 사전에 설치되어 있다고 가정





- VS Code에서 터미널을 실행 후, 명령 프롬프트(Command Prompt) 클릭

1. Open CV 개요

2) Open CV 설치 (Python 기반)

$ pip install opencv-python

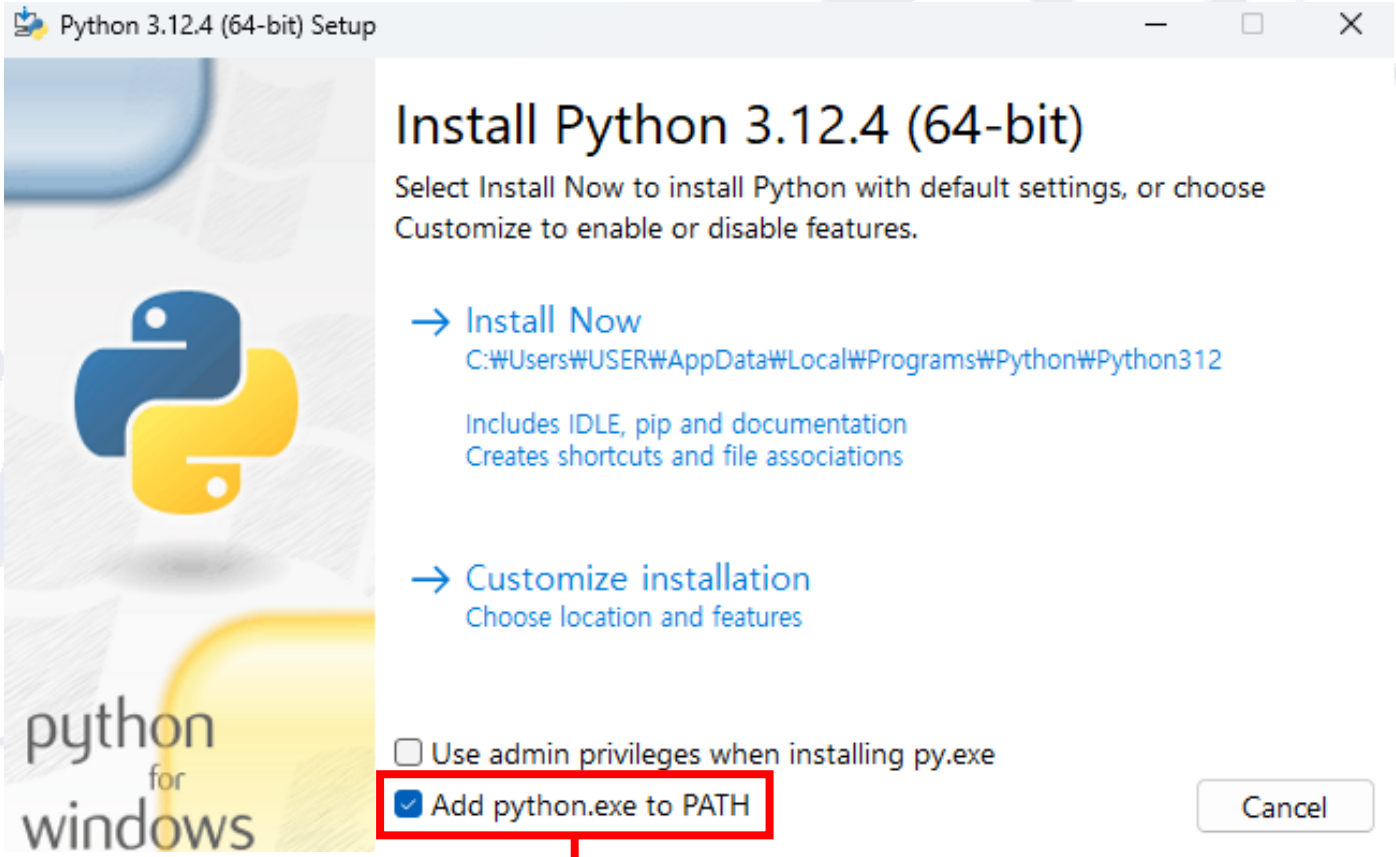
$ pip install numpy

```
PS C:\Users\USER> pip install opencv-python
Collecting opencv-python
  Downloading opencv_python-4.10.0.84-cp37-abi3-win_amd64.whl.metadata (20 kB)
Collecting numpy>=1.21.2 (from opencv-python)
  Downloading numpy-2.0.0-cp312-cp312-win_amd64.whl.metadata (60 kB)
     ---------------------------------------- 60.9/60.9 kB 3.2 MB/s eta 0:00:00
Downloading opencv_python-4.10.0.84-cp37-abi3-win_amd64.whl (38.8 MB)
   ---------------------------------------- 38.8/38.8 MB 11.3 MB/s eta 0:00:00
Downloading numpy-2.0.0-cp312-cp312-win_amd64.whl (16.2 MB)
   ---------------------------------------- 16.2/16.2 MB 11.7 MB/s eta 0:00:00
Installing collected packages: numpy, opencv-python
Successfully installed numpy-2.0.0 opencv-python-4.10.0.84

[notice] A new release of pip is available: 24.0 -> 24.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip
PS C:\Users\USER> pip install numpy
Requirement already satisfied: numpy in c:\users\user\appdata\local\programs\python\python312\lib\site-packages (2.0
.0)

[notice] A new release of pip is available: 24.0 -> 24.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

1. Open CV 개요

2) Open CV 설치 (Python 기반)

➔ 만약 설치 과정에서 pip 관련 오류가 발생한다면, Python 재설치 수행



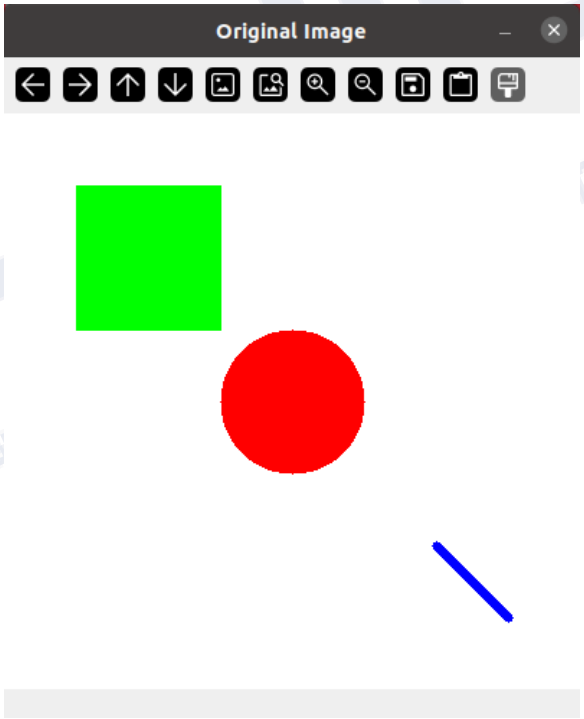
➔ Python 재설치 시, 반드시 해당 부분 체크 !

1. Open CV 개요

3) Open CV 주요 기능

- **이미지 읽기 및 쓰기**

  → 이미지를 파일에서 읽어오고, 처리된 이미지를 파일로 저장 가능



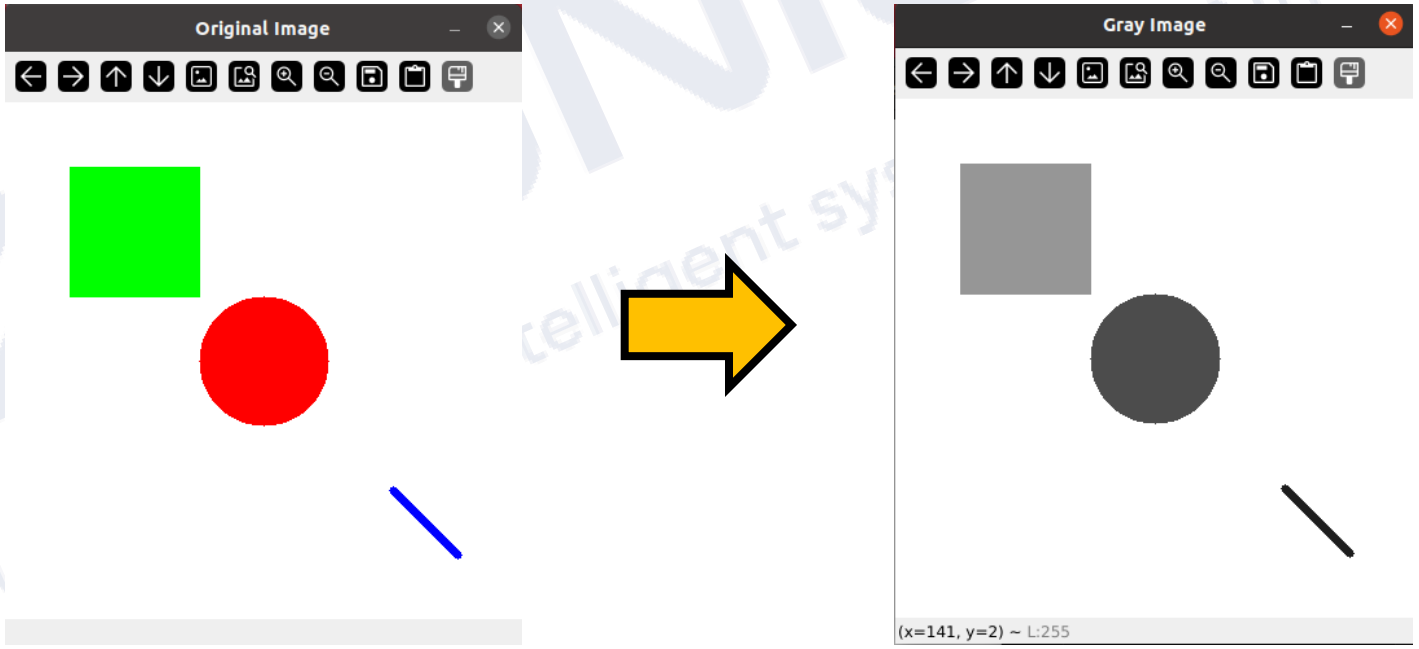
1. Open CV 개요

3) Open CV 주요 기능

- **이미지 변환**

  → 이미지를 다양한 색상 공간으로 변환 가능

  → 다음 예시와 같이, RGB 이미지를 Gray scale 또는 HSV 색상 공간으로 변환 가능



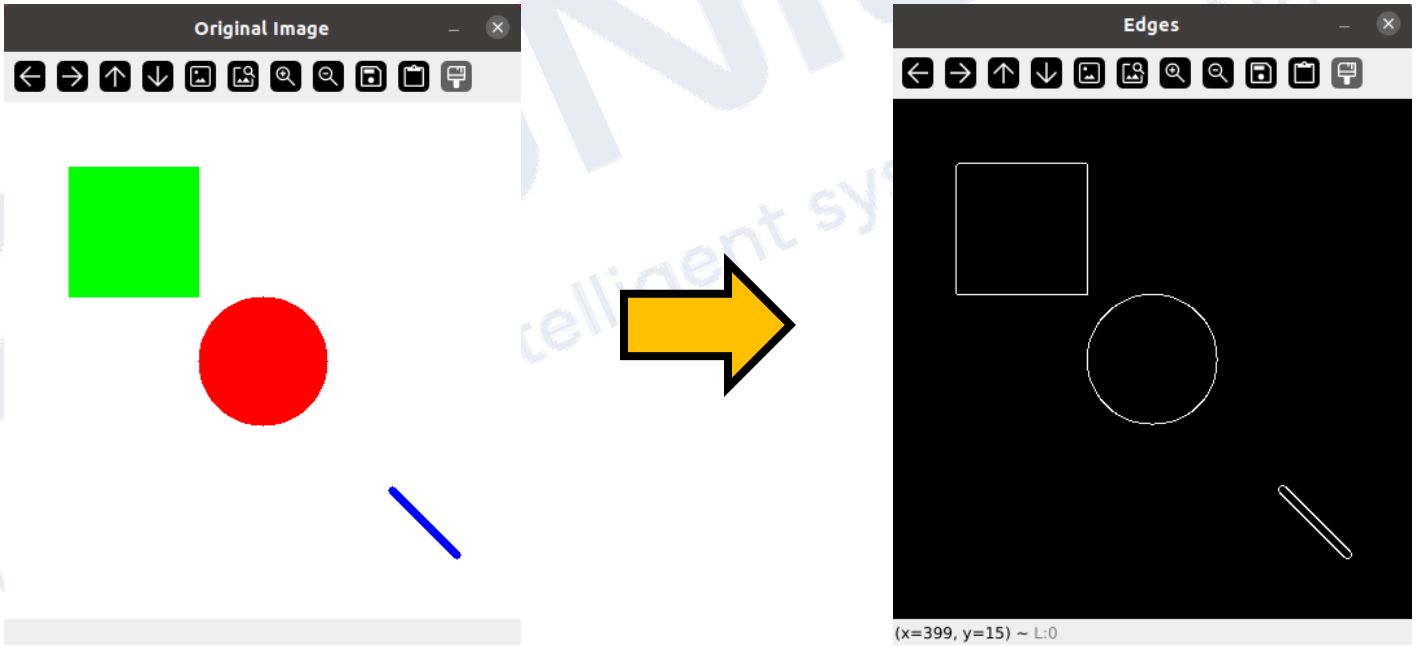
1. Open CV 개요

3) Open CV 주요 기능

- **Edge 검출**

  → 이미지에서 객체의 경계를 찾기 위해 edge를 검출

  → 대표적인 edge 검출 기법으로 'Canny edge' 검출 기법이 존재

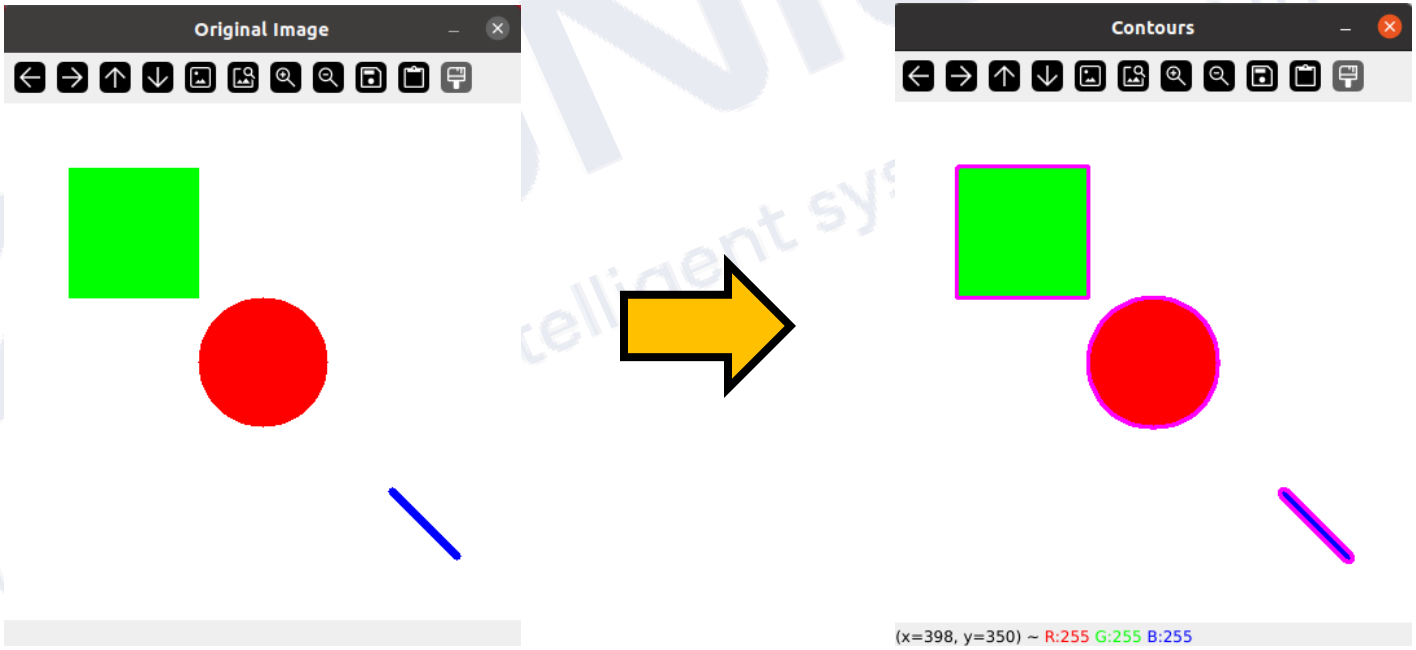


1. Open CV 개요

3) Open CV 주요 기능

- **윤곽선 검출**

  → 이미지에서 객체의 윤곽선을 검출



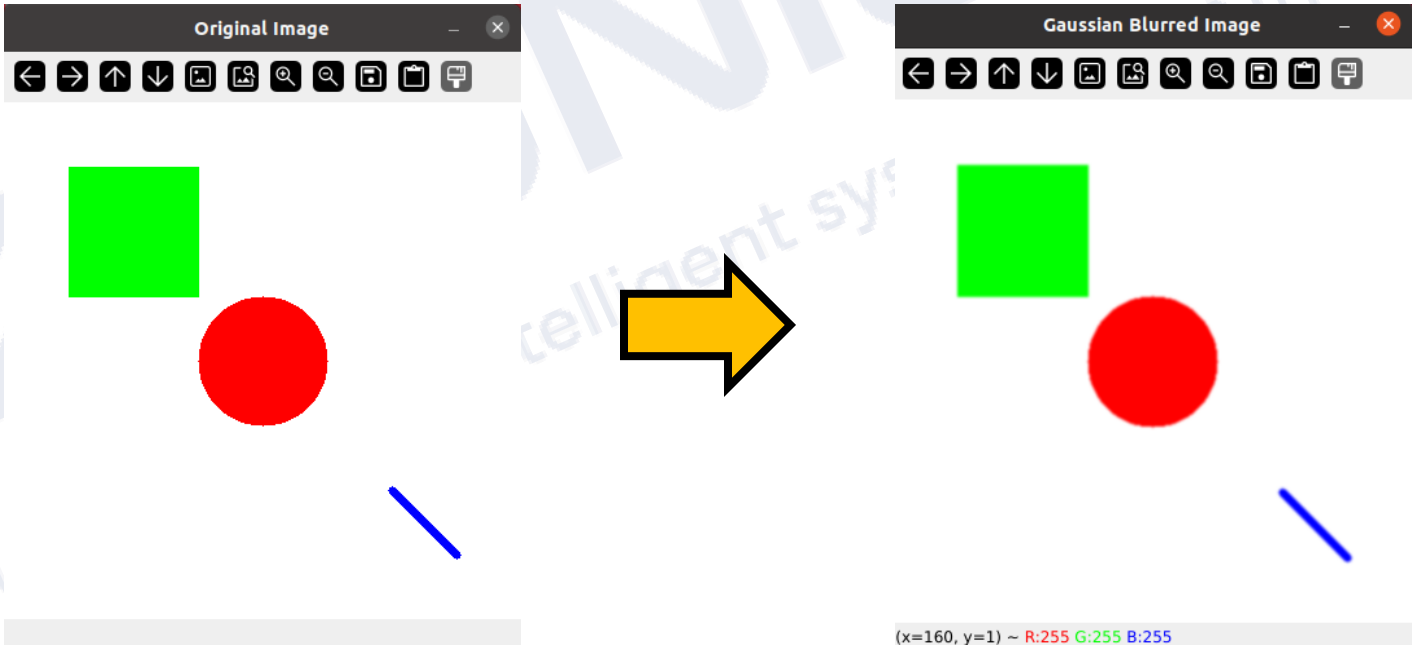
1. Open CV 개요

3) Open CV 주요 기능

- **이미지 필터링**

  → 이미지를 필터링하여 노이즈를 제거하거나 부드럽게 변환 가능

  → 대표적인 필터링 기법으로 'Gaussian Blur', 'Median Blur' 등이 있음

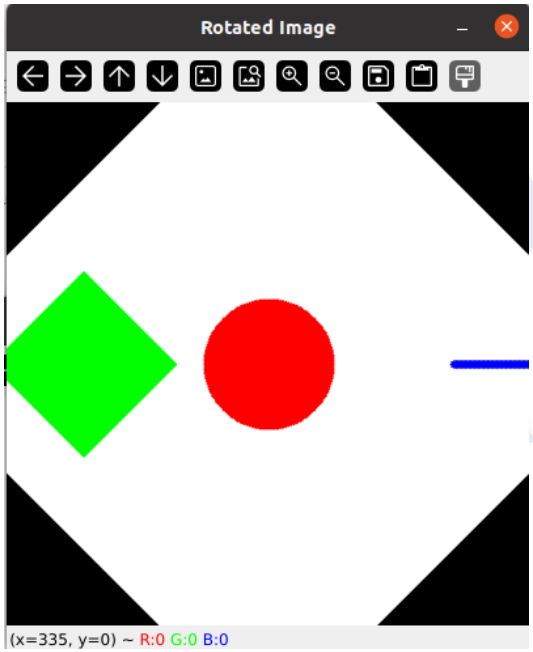


1. Open CV 개요

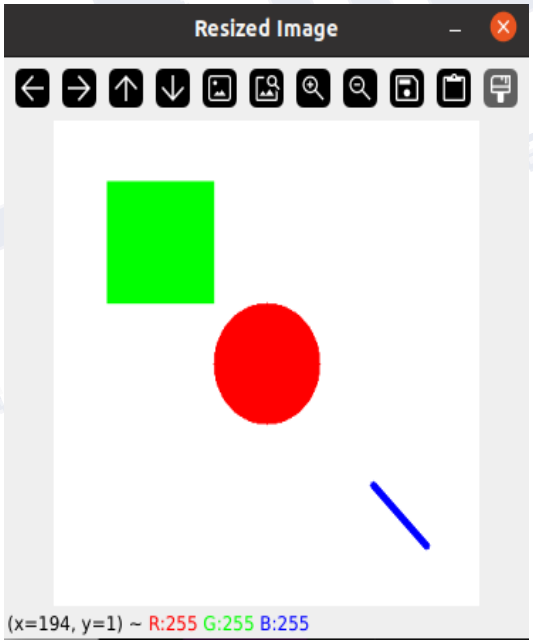
3) Open CV 주요 기능

- **이미지 조정**

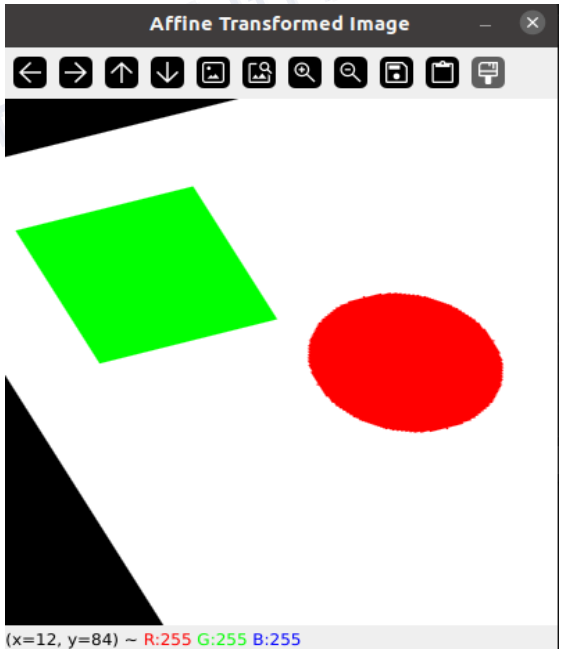
  → 이미지 회전, 크기 조절, 기울이기 등의 기능을 통해 이미지 데이터를 변환 가능



이미지 회전



이미지 크기 조절



이미지 기울이기

1. Open CV 개요

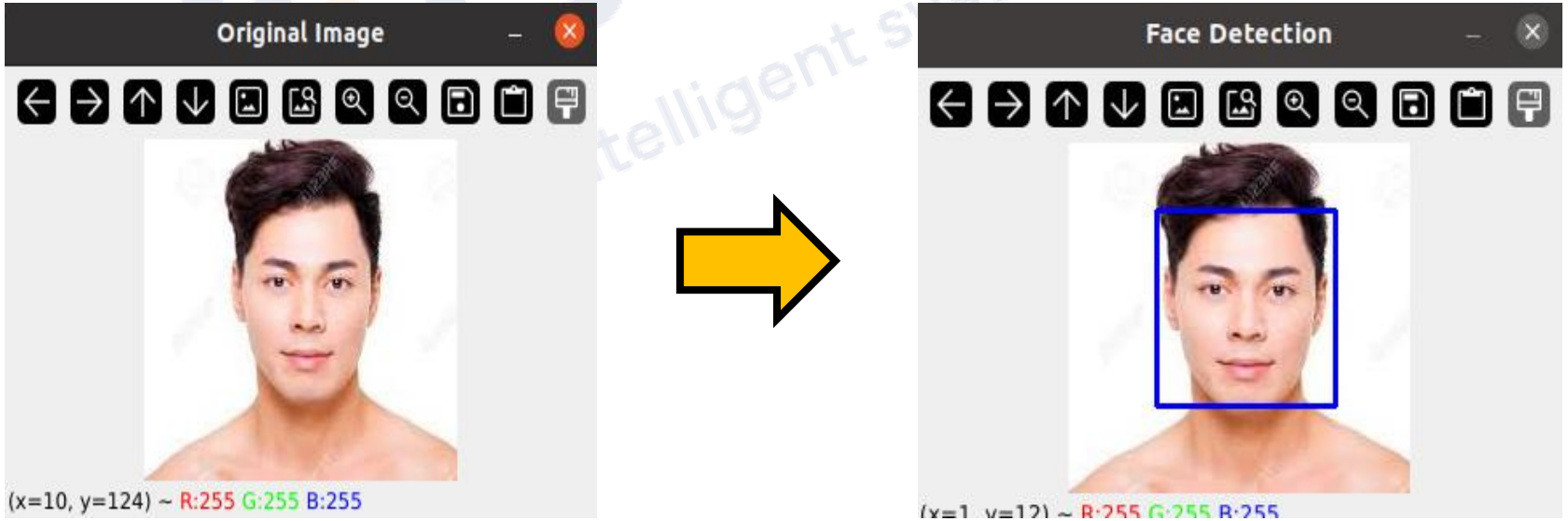
3) Open CV 주요 기능

- **객체 인식**

  → 특정 객체를 인식하고 위치를 파악 가능

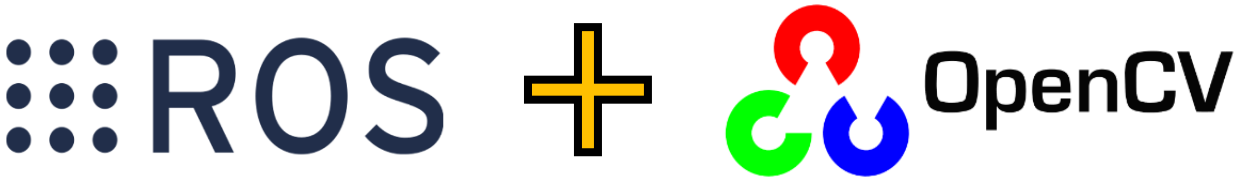
  → 객체 인식을 위한 대표적인 기법으로는 'Haar Cascade' 기반 객체 인식과, Deep Learning 기반의 객체 인식 등이 존재

  → 'Haar Cascade'란, Open CV에 내장된 기능으로 얼굴 또는 객체 등을 검출 시 이용되는 머신러닝 기반의 기술을 의미



1. Open CV 개요

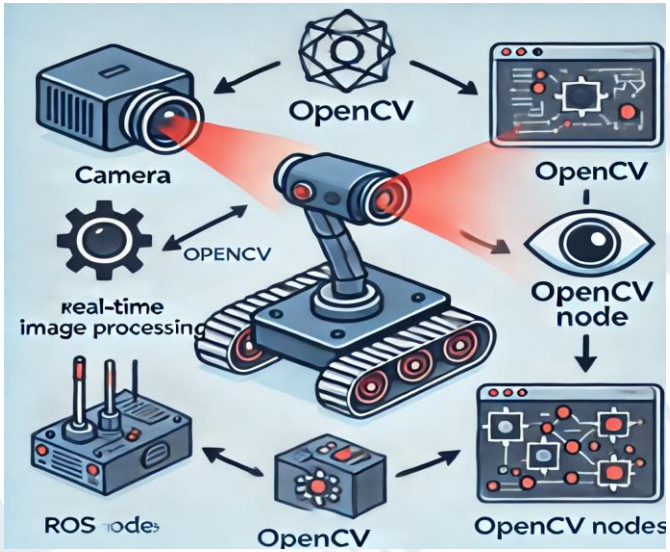
4) CV bridge 기반 ROS 환경 및 Open CV 연동



- **실시간 이미지 처리**
  → 로봇에 부착된 Camera 센서를 통해 수집된 이미지를 실시간으로 처리하여 장애물 감지, 객체 추종 등 다양한 작업을 수행 가능
- **모듈성 및 재사용성**
  → ROS(Robot Operating System)의 Module(모듈)식 구조를 활용하여 Open CV 기반 이미지 처리 기능을 독립적인 노드로 구현하고, 다양한 프로젝트에서 재사용 가능
- **커뮤니티 및 오픈 소스 지원**
  → ROS와 Open CV는 모두 오픈 소스 프로젝트로, 광범위한 커뮤니티 지원과 수 많은 자료를 제공
  → 최신 기술에서 발생할 수 있는 문제에 대한 해결이 용이하며, 이를 바탕으로 다양한 최신 기술을 편하게 사용 가능

1. Open CV 개요

4) CV bridge 기반 ROS 환경 및 Open CV 연동



- **다양한 센서 데이터 통합**

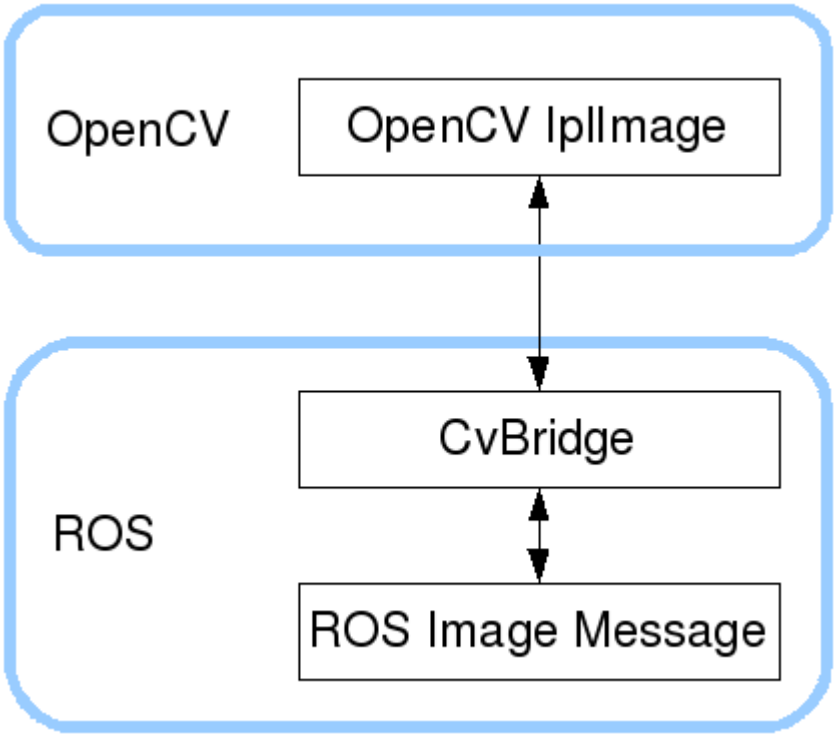
  → Open CV를 통해 처리한 이미지 데이터와 더불어, 다양한 센서들을 결합하여 도출한 데이터를 바탕으로 더욱 정밀하게 환경 인식 수행 가능

- **시뮬레이션 및 로봇 적용**

  → Gazebo 시뮬레이터를 통해 가상 환경에서 로봇을 테스트한 뒤, 실제 로봇에 적용함으로써 테스트 및 개발 시간을 단축 가능

1. Open CV 개요

4) CV bridge 기반 ROS 환경 및 Open CV 연동



- **cv_bridge란?**

→ ROS에서 제공하는 패키지로, ROS 이미지 메시지와 Open CV 이미지 간의 원활한 변환을 돕는 기능

→ cv_bridge를 통해 ROS 환경의 노드가 수신한 이미지를 Open CV 형식으로 변환하여, 다양한 이미지 처리 작업을 수행 가능

1. Open CV 개요

4) CV bridge 기반 ROS 환경 및 Open CV 연동

- ROS noetic 환경 기반 Open CV 설치

  → $ pip install opencv-python (또는 $ pip3 install opencv-python)

```
gihoon@gihoon-B650M-HDV-M-2:~$ pip install opencv-python
Collecting opencv-python
  Downloading opencv_python-4.10.0.84-cp37-abi3-manylinux_2_17_x86_64.manylinux2
014_x86_64.whl (62.5 MB)
     |                                        | 62.5 MB 11.8 MB/s
Requirement already satisfied: numpy>=1.17.3; python_version >= "3.8" in ./.loca
l/lib/python3.8/site-packages (from opencv-python) (1.21.5)
Installing collected packages: opencv-python
Successfully installed opencv-python-4.10.0.84
```

- ROS noetic 환경에 Open CV 설치가 완료되었는지 확인

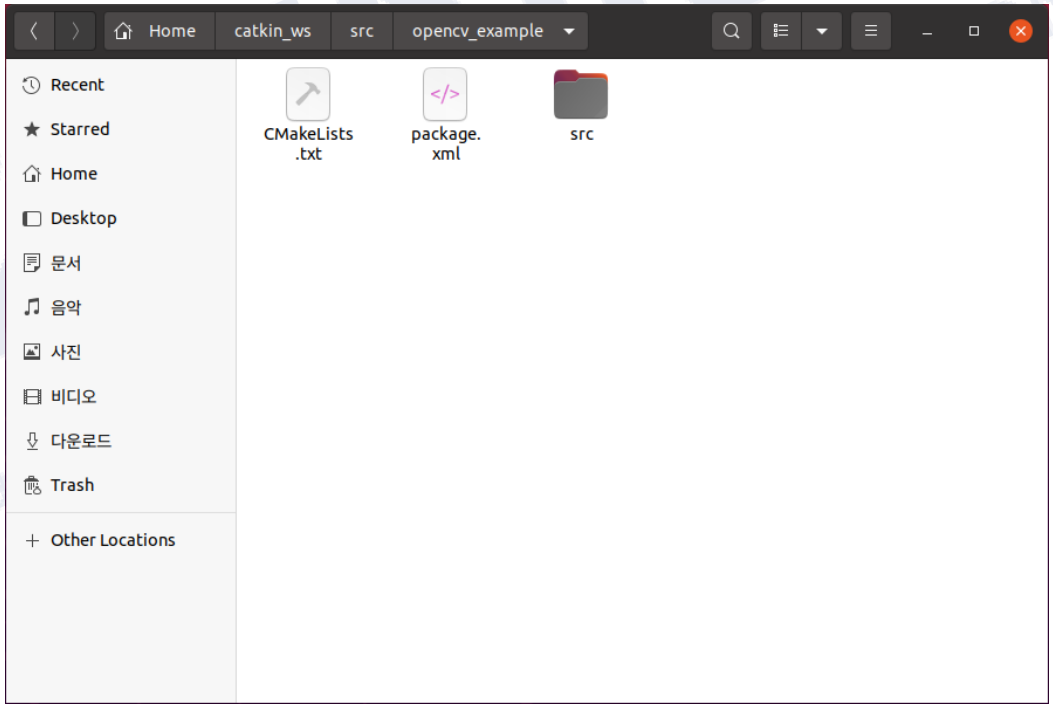
  → $ dpkg -l | grep ros-noetic-cv-bridge

```
gihoon@gihoon-B650M-HDV-M-2:~$ dpkg -l | grep libopencv
ii  libopencv-calib3d-dev:amd64          4.2.0+dfsg-5                amd64
ii  libopencv-calib3d4.2:amd64           4.2.0+dfsg-5                amd64
ii  libopencv-contrib-dev:amd64          4.2.0+dfsg-5                amd64
ii  libopencv-contrib4.2:amd64           4.2.0+dfsg-5                amd64
ii  libopencv-core-dev:amd64             4.2.0+dfsg-5                amd64
ii  libopencv-core4.2:amd64              4.2.0+dfsg-5                amd64
ii  libopencv-dev                        4.2.0+dfsg-5                amd64
ii  libopencv-dnn-dev:amd64              4.2.0+dfsg-5                amd64
ii  libopencv-dnn4.2:amd64               4.2.0+dfsg-5                amd64
ii  libopencv-features2d-dev:amd64       4.2.0+dfsg-5                amd64
ii  libopencv-features2d4.2:amd64        4.2.0+dfsg-5                amd64
```

1. Open CV 개요

4) CV bridge 기반 ROS 환경 및 Open CV 연동

- ROS noetic 환경에서 Camera 이미지 수신 및 Open CV 기반 처리 실습 (1/7)
  - → $ cd ~/catkin_ws/src
  - → $ catkin_create_pkg opencv_example rospy std_msgs sensor_msgs cv_bridge



1. Open CV 개요

4) CV bridge 기반 ROS 환경 및 Open CV 연동

- ROS noetic 환경에서 Camera 이미지 수신 및 Open CV 기반 처리 실습 (2/7)

  → $ cd opencv_example (새롭게 생성한 패키지로 이동)

  → $ gedit CMakeLists.txt (CMakeLists 파일을 수정 → cv_bridge와 Open CV 의존성 추가)

Open | CMakeLists.txt ~/catkin_ws/src/opencv_example | Save

```
1 cmake_minimum_required(VERSION 3.0.2)
2 project(opencv_example)
3
4 ## Compile as C++11, supported in ROS Kinetic and newer
5 # add_compile_options(-std=c++11)
6
7 ## Find catkin macros and libraries
8 ## if COMPONENTS list like find_package(catkin REQUIRED COMPONENTS xyz)
9 ## is used, also find other catkin packages
10 find_package(catkin REQUIRED COMPONENTS
11   rospy
12   std_msgs
13   sensor_msgs
14   cv_bridge
15 )
16
17 find_package(OpenCV REQUIRED)
18
19 catkin_package()
20
21 include_directories(
22   ${catkin_INCLUDE_DIRS}
23   ${OpenCV_INCLUDE_DIRS}
24 )
```

1. Open CV 개요

4) CV bridge 기반 ROS 환경 및 Open CV 연동

- ROS noetic 환경에서 Camera 이미지 수신 및 Open CV 기반 처리 실습 (3/7)

  → $ cd opencv_example (새롭게 생성한 패키지로 이동)

  → $ gedit package.xml (package.xml 파일을 수정 → cv_bridge와 Open CV 의존성 추가)

```
51  <buildtool_depend>catkin</buildtool_depend>
52  <build_depend>cv_bridge</build_depend>
53  <build_depend>rospy</build_depend>
54  <build_depend>sensor_msgs</build_depend>
55  <build_depend>std_msgs</build_depend>
56  <build_export_depend>cv_bridge</build_export_depend>
57  <build_export_depend>rospy</build_export_depend>
58  <build_export_depend>sensor_msgs</build_export_depend>
59  <build_export_depend>std_msgs</build_export_depend>
60  <exec_depend>cv_bridge</exec_depend>
61  <exec_depend>rospy</exec_depend>
62  <exec_depend>sensor_msgs</exec_depend>
63  <exec_depend>std_msgs</exec_depend>
64
```

```
<build_depend>cv_bridge</build_depend>
<exec_depend>cv_bridge</exec_depend>

<build_depend>opencv</build_depend>
<exec_depend>opencv</exec_depend>
```

1. Open CV 개요

4) CV bridge 기반 ROS 환경 및 Open CV 연동

- ROS noetic 환경에서 Camera 이미지 수신 및 Open CV 기반 처리 실습 (4/7)

  → $ cd ~/catkin_ws/src/opencv_example/ (새롭게 생성한 패키지 내 src 폴더로 이동)

  → $ gedit test.py (src 폴더 내 python 파일 생성 후 저장)

  → $ chmod +x test.py (새롭게 생성한 python 파일에 실행 권한 부여)

```
gihoon@gihoon-B650M-HDV-M-2:~$ cd catkin_ws/src/opencv_example/src/
gihoon@gihoon-B650M-HDV-M-2:~/catkin_ws/src/opencv_example/src$ chmod +x test.py
```

  → $ cd ~/catkin_ws/ && catkin_make (catkin_ws 빌드)

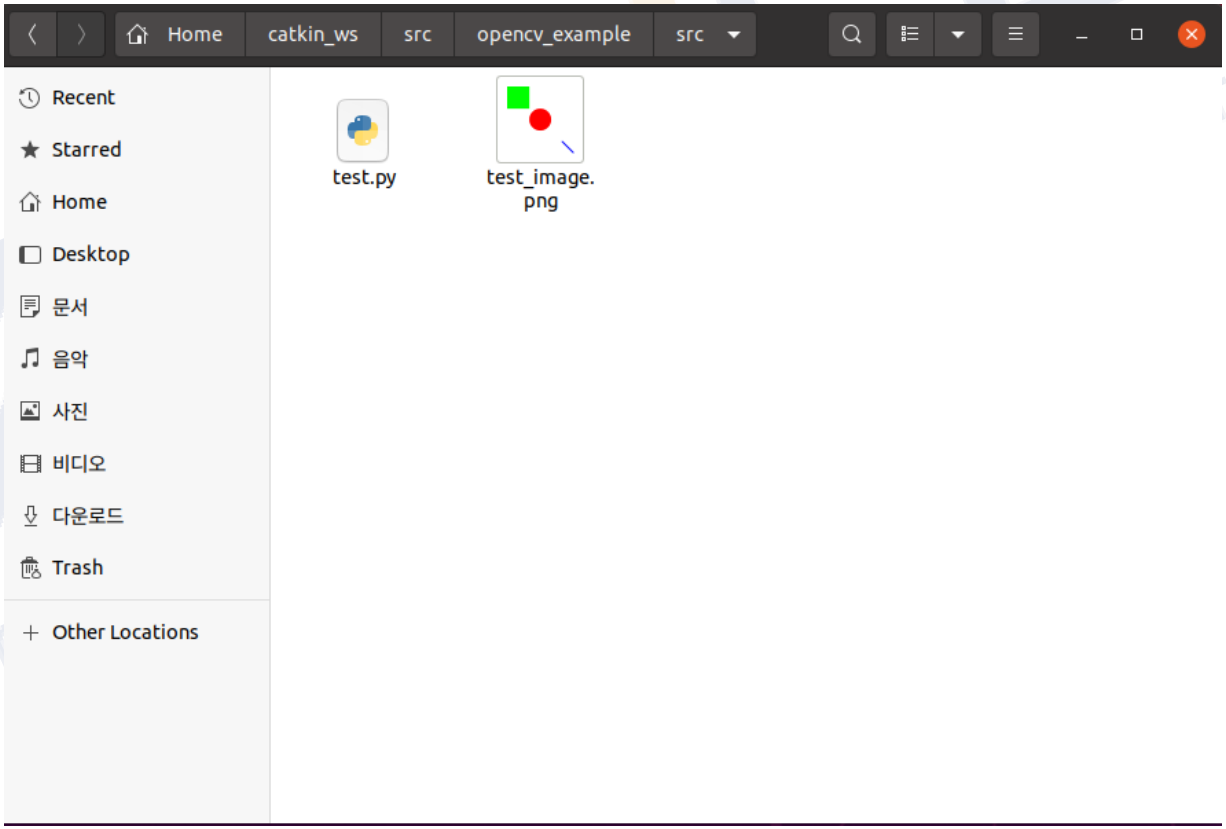
  → $ source devel/setup.bash (빌드 후, 환경 변수 재 설정)

1. Open CV 개요

4) CV bridge 기반 ROS 환경 및 Open CV 연동

- ROS noetic 환경에서 Camera 이미지 수신 및 Open CV 기반 처리 실습 (5/7)

  → Open CV 실습에 사용할 이미지를 다운로드 후, 저장



1. Open CV 개요

4) CV bridge 기반 ROS 환경 및 Open CV 연동

- ROS noetic 환경에서 Camera 이미지 수신 및 Open CV 기반 처리 실습 (6/7)

  → $ cd ~/catkin_ws/src/opencv_example/src/ (새롭게 생성한 패키지 내 src 폴더 진입)

  → $ gedit test.py (새롭게 생성한 test.py 파일에 아래와 같이 작성 후, 저장)

```python
#!/usr/bin/env python3
import cv2
import rospy


image_path = '/home/user/catkin_ws/src/opencv_example/src/test_image.png' # 이미지 경로를 설정하세요
image = cv2.imread(image_path)

# 1. 원본 이미지 표시
cv2.imshow('Original Image', image)

cv2.waitKey(0)
cv2.destroyAllWindows()

```

→ 본인이 저장한 이미지 파일의 경로를 작성 !!

1. Open CV 개요

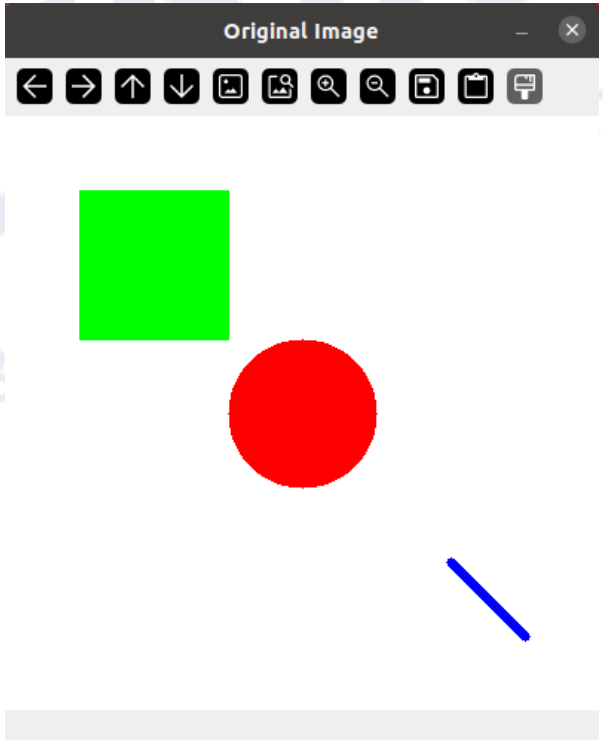
4) CV bridge 기반 ROS 환경 및 Open CV 연동

- ROS noetic 환경에서 Camera 이미지 수신 및 Open CV 기반 처리 실습 (7/7)

  → $ roscore (첫 번째 터미널)

  → $ cd ~/catkin_ws/src/opencv_example/src/ (두 번째 터미널)

  $ python3 test.py (두 번째 터미널)



2. Open CV 실습

2. Open CV 실습

1) 이미지 출력

- Python 환경에서 Open CV를 통해 다운받은 원본 이미지를 화면에 출력하기 (실습)

  → 주의 사항 1 : 이미지 파일의 확장자는 반드시 “jpg” 또는 “png” 파일을 사용할 것.

  → 주의 사항 2 : Python code 상에 작성하는 이미지의 경로를 정확하게 작성할 것.

test.py

scripts > test.py

```python
#!/usr/bin/env python3
import cv2
import numpy as np

image_path = '/home/gihoon/catkin_ws/src/lane_detection/scripts/download.jpg' # 이미지 경로를 설정하세요
image = cv2.imread(image_path)

# 1. 원본 이미지 표시
cv2.imshow('Original Image', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

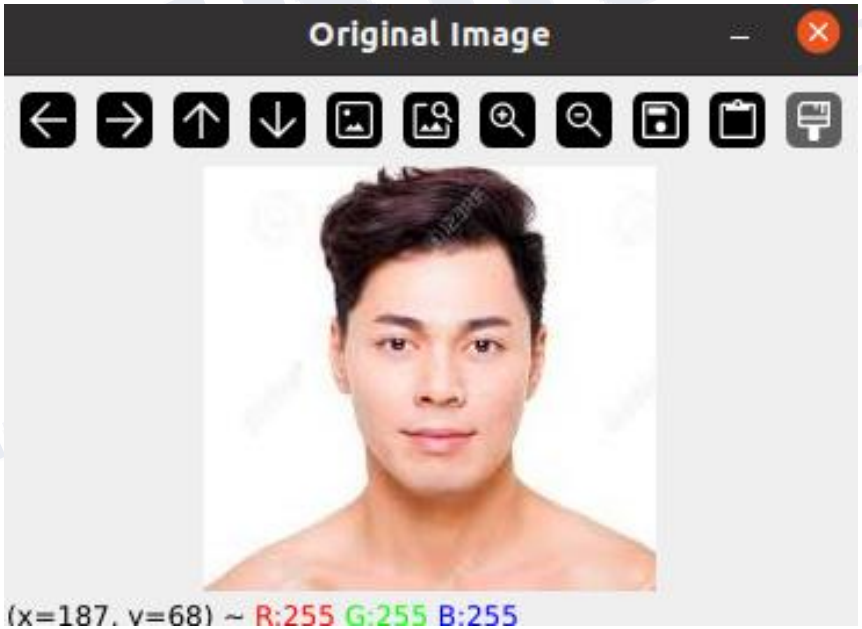
2. Open CV 실습

1) 이미지 출력

- Python 환경에서 Open CV를 통해 다운받은 원본 이미지를 화면에 출력하기 (실습)

  → 주의 사항 1 : 이미지 파일의 확장자는 반드시 “jpg” 또는 “png” 파일을 사용할 것.

  → 주의 사항 2 : Python code 상에 작성하는 이미지의 경로를 정확하게 작성할 것.



2. Open CV 실습

2) 이미지 gray scale

- Python 환경에서 Open CV를 통해 원본 이미지를 gray scale화하여 화면에 출력하기 (실습)

  → 주의 사항 1 : 이미지 파일의 확장자는 반드시 “jpg” 또는 “png” 파일을 사용할 것.

  → 주의 사항 2 : Python code 상에 작성하는 이미지의 경로를 정확하게 작성할 것.

test.py

scripts > test.py

```python
#!/usr/bin/env python3
import cv2
import numpy as np

image_path = '/home/gihoon/catkin_ws/src/lane_detection/scripts/download.jpg' # 이미지 경로를 설정하세요
image = cv2.imread(image_path)

gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2.imshow('Gray Image', gray_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

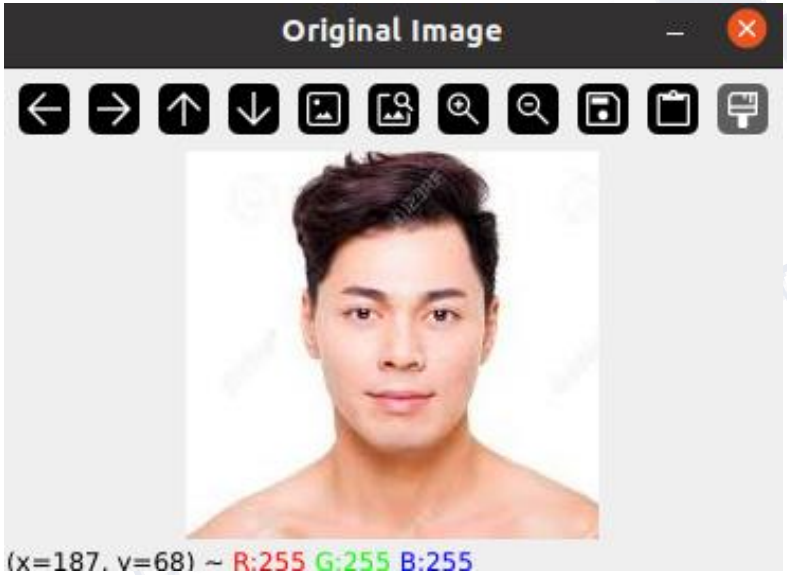
2. Open CV 실습

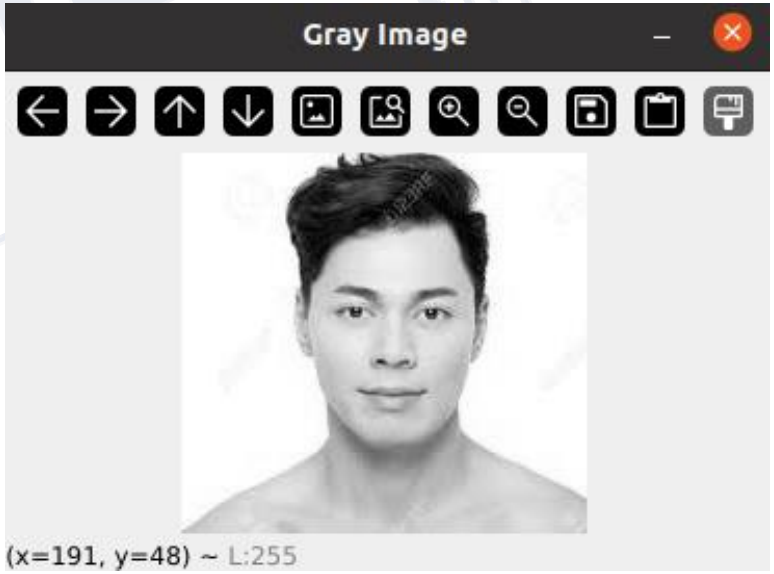
2) 이미지 gray scale

- Python 환경에서 Open CV를 통해 원본 이미지를 gray scale화하여 화면에 출력하기 (실습)

  → 주의 사항 1 : 이미지 파일의 확장자는 반드시 “jpg” 또는 “png” 파일을 사용할 것.

  → 주의 사항 2 : Python code 상에 작성하는 이미지의 경로를 정확하게 작성할 것.





2. Open CV 실습

3) 이미지 edge 검출

- Python 환경에서 Open CV를 통해 원본 이미지의 Edge를 검출하여 화면에 출력하기 (실습)

  → 주의 사항 1 : 이미지 파일의 확장자는 반드시 “jpg” 또는 “png” 파일을 사용할 것.

  → 주의 사항 2 : Python code 상에 작성하는 이미지의 경로를 정확하게 작성할 것.

test.py

scripts > test.py

```python
#!/usr/bin/env python3
import cv2
import numpy as np

image_path = '/home/gihoon/catkin_ws/src/lane_detection/scripts/download.jpg' # 이미지 경로를 설정하세요
image = cv2.imread(image_path)

edges = cv2.Canny(image, 50, 150)
cv2.imshow('Edges', edges)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

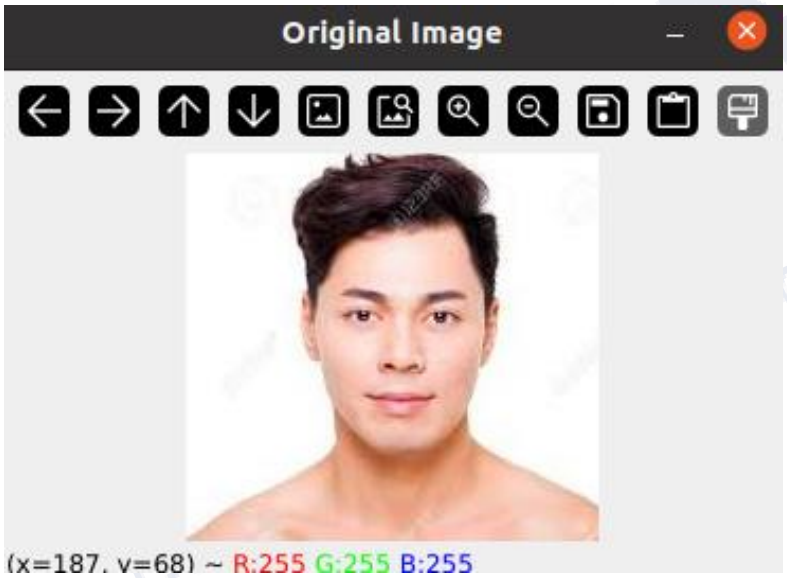
2. Open CV 실습

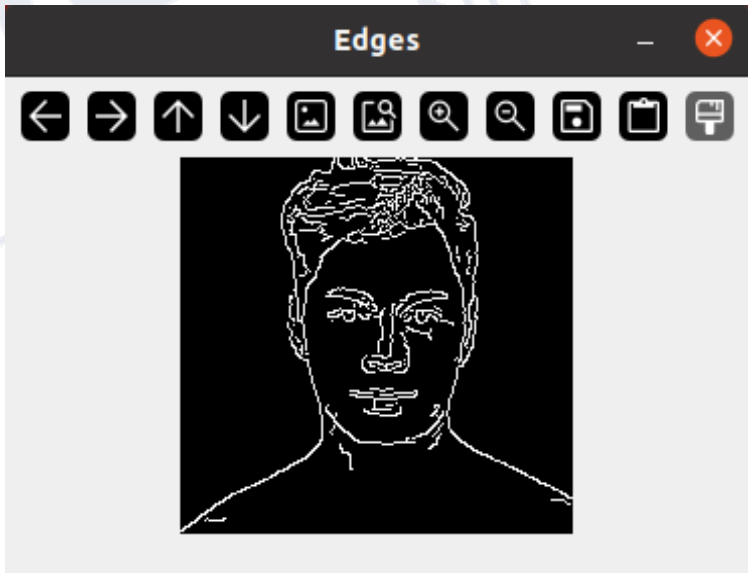
3) 이미지 edge 검출

- Python 환경에서 Open CV를 통해 원본 이미지의 Edge를 검출하여 화면에 출력하기 (실습)

  → 주의 사항 1 : 이미지 파일의 확장자는 반드시 “jpg” 또는 “png” 파일을 사용할 것.

  → 주의 사항 2 : Python code 상에 작성하는 이미지의 경로를 정확하게 작성할 것.





2. Open CV 실습

4) 이미지 윤곽선 검출

- Python 환경에서 Open CV를 통해 원본 이미지의 윤곽선을 검출하여 화면에 출력하기 (실습)

  → 주의 사항 1 : 이미지 파일의 확장자는 반드시 “jpg” 또는 “png” 파일을 사용할 것.

  → 주의 사항 2 : Python code 상에 작성하는 이미지의 경로를 정확하게 작성할 것.

test.py

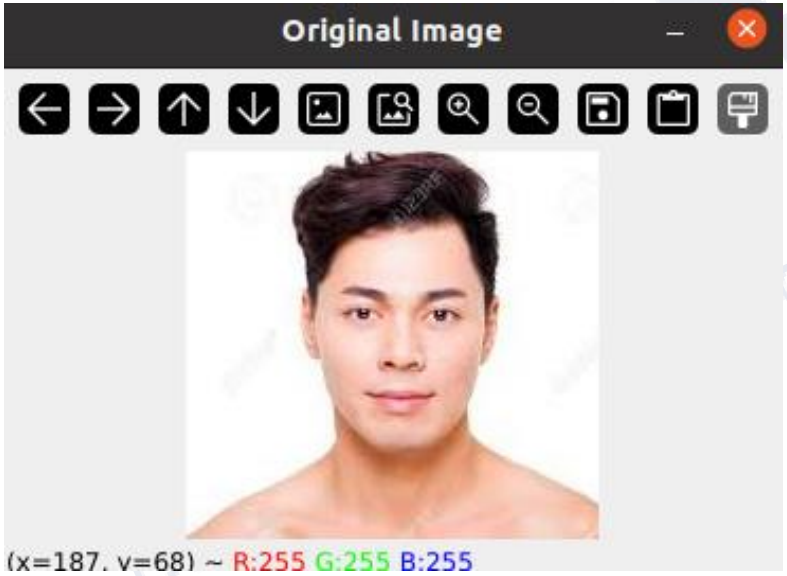
scripts > test.py

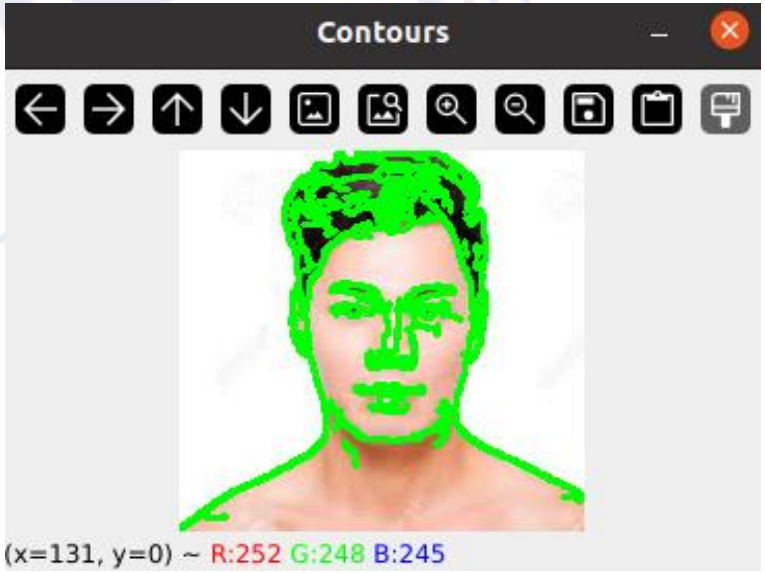
```python
#!/usr/bin/env python3
import cv2
import numpy as np

image_path = '/home/gihoon/catkin_ws/src/lane_detection/scripts/download.jpg' # 이미지 경로를 설정하세요
image = cv2.imread(image_path)

edges = cv2.Canny(image, 50, 150)

contours, hierarchy = cv2.findContours(edges, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
contour_image = image.copy()
cv2.drawContours(contour_image, contours, -1, (0, 255, 0), 2)
cv2.imshow('Contours', contour_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

2. Open CV 실습

4) 이미지 윤곽선 검출

- Python 환경에서 Open CV를 통해 원본 이미지의 윤곽선을 검출하여 화면에 출력하기 (실습)
  → 주의 사항 1 : 이미지 파일의 확장자는 반드시 “jpg” 또는 “png” 파일을 사용할 것.
  → 주의 사항 2 : Python code 상에 작성하는 이미지의 경로를 정확하게 작성할 것.





2. Open CV 실습

5) 이미지 회전 / 크기 조절 / 기울이기

- Python 환경에서 Open CV를 원본 이미지를 변환(회전, 크기 조절, 기울이기) 후 출력하기 (실습)

  → 주의 사항 1 : 이미지 파일의 확장자는 반드시 “jpg” 또는 “png” 파일을 사용할 것.

  → 주의 사항 2 : Python code 상에 작성하는 이미지의 경로를 정확하게 작성할 것.

test.py

scripts > test.py

```python
#!/usr/bin/env python3
import cv2
import numpy as np

image_path = '/home/gihoon/catkin_ws/src/lane_detection/scripts/download.jpg' # 이미지 경로를 설정하세요
image = cv2.imread(image_path)

# 이미지 회전
(h, w) = image.shape[:2]
center = (w / 2, h / 2)
M = cv2.getRotationMatrix2D(center, 45, 1.0)
rotated_image = cv2.warpAffine(image, M, (w, h))
cv2.imshow('Rotated Image', rotated_image)

# 이미지 크기 조정
resized_image = cv2.resize(image, (300, 300))
cv2.imshow('Resized Image', resized_image)

# 이미지 기울이기 (Affine Transform)
pts1 = np.float32([[50, 50], [200, 50], [50, 200]])
pts2 = np.float32([[10, 100], [200, 50], [100, 250]])
M = cv2.getAffineTransform(pts1, pts2)
affine_image = cv2.warpAffine(image, M, (w, h))
cv2.imshow('Affine Transformed Image', affine_image)

# 모든 창을 동시에 닫기 위해 waitKey와 destroyAllWindows를 한 번만 호출
cv2.waitKey(0)
cv2.destroyAllWindows()
```

2. Open CV 실습

5) 이미지 회전 / 크기 조절 / 기울이기

- Python 환경에서 Open CV를 원본 이미지를 변환(회전, 크기 조절, 기울이기) 후 출력하기 (실습)

  → 주의 사항 1 : 이미지 파일의 확장자는 반드시 “jpg” 또는 “png” 파일을 사용할 것.

  → 주의 사항 2 : Python code 상에 작성하는 이미지의 경로를 정확하게 작성할 것.

2. Open CV 실습

6) 이미지 filtering

- Python 환경에서 Open CV를 통해 원본 이미지를 Filtering(Gaussian blur)하여 출력하기 (실습)

  → 주의 사항 1 : 이미지 파일의 확장자는 반드시 “jpg” 또는 “png” 파일을 사용할 것.

  → 주의 사항 2 : Python code 상에 작성하는 이미지의 경로를 정확하게 작성할 것.

test.py

scripts > test.py

```python
#!/usr/bin/env python3
import cv2
import numpy as np

image_path = '/home/gihoon/catkin_ws/src/lane_detection/scripts/download.jpg' # 이미지 경로를 설정하세요
image = cv2.imread(image_path)

blurred_image = cv2.GaussianBlur(image, (5, 5), 0)
cv2.imshow('Gaussian Blurred Image', blurred_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

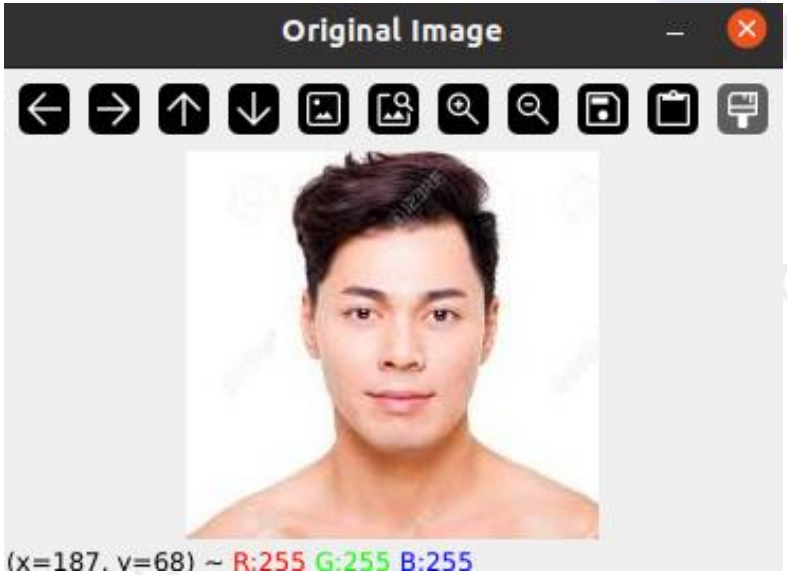
2. Open CV 실습

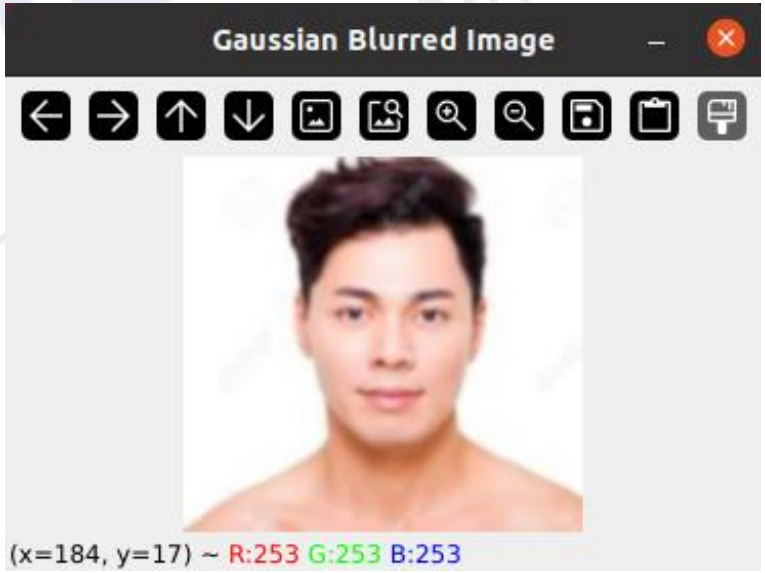
6) 이미지 filtering

- Python 환경에서 Open CV를 통해 원본 이미지를 Filtering(Gaussian blur)하여 출력하기 (실습)

  → 주의 사항 1 : 이미지 파일의 확장자는 반드시 “jpg” 또는 “png” 파일을 사용할 것.

  → 주의 사항 2 : Python code 상에 작성하는 이미지의 경로를 정확하게 작성할 것.





2. Open CV 실습

7) 이미지 속 객체 검출

- Python 환경에서 Open CV를 이용하여, 원본 이미지 속 객체를 인식하도록 하기 (실습)

  → 주의 사항 1 : 이미지 파일의 확장자는 반드시 “jpg” 또는 “png” 파일을 사용할 것.

  → 주의 사항 2 : Python code 상에 작성하는 이미지의 경로를 정확하게 작성할 것.

test.py

scripts > test.py

```python
#!/usr/bin/env python3
import cv2
import numpy as np

image_path = '/home/gihoon/catkin_ws/src/lane_detection/scripts/download.jpg' # 이미지 경로를 설정하세요
image = cv2.imread(image_path)

face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
faces = face_cascade.detectMultiScale(image, 1.3, 5)
for (x, y, w, h) in faces:
    cv2.rectangle(image, (x, y), (x+w, y+h), (255, 0, 0), 2)

cv2.imshow('Face Detection', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

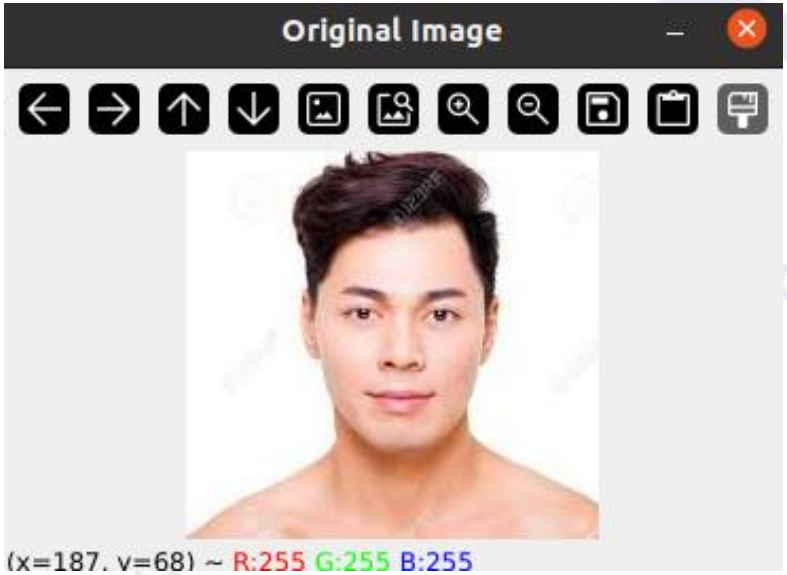
2. Open CV 실습

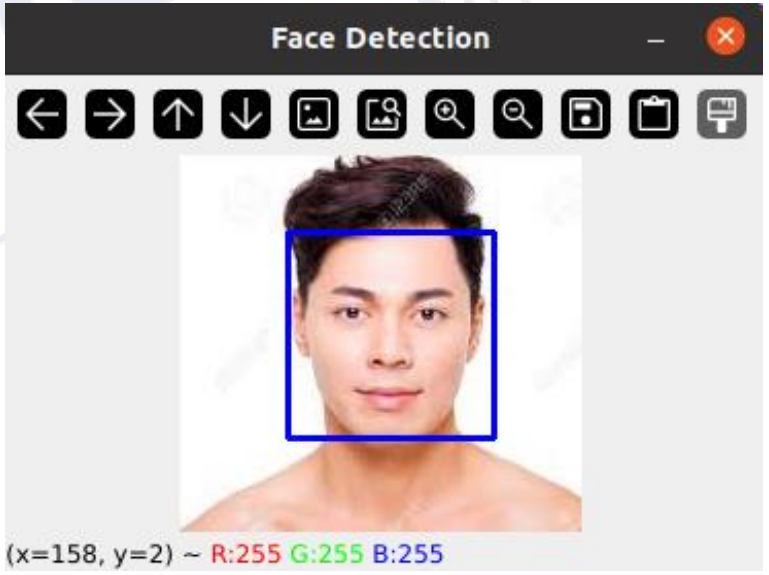
7) 이미지 속 객체 검출

- Python 환경에서 Open CV를 통해 원본 이미지를 Filtering(Gaussian blur)하여 출력하기 (실습)

  → 주의 사항 1 : 이미지 파일의 확장자는 반드시 “jpg” 또는 “png” 파일을 사용할 것.

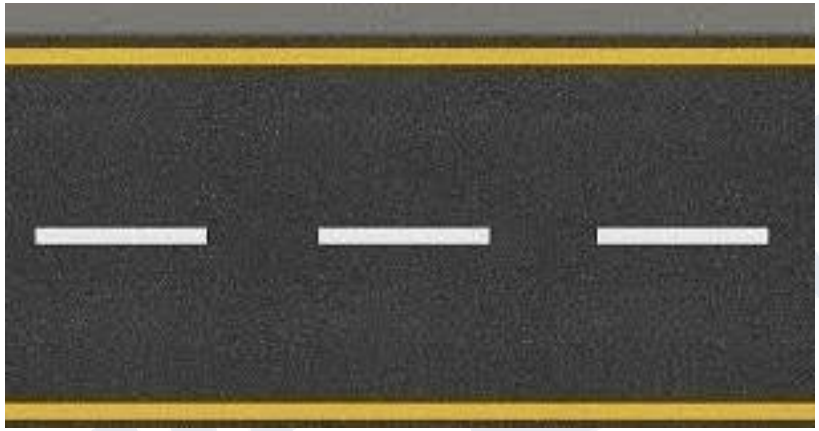
  → 주의 사항 2 : Python code 상에 작성하는 이미지의 경로를 정확하게 작성할 것.





2. Open CV 실습

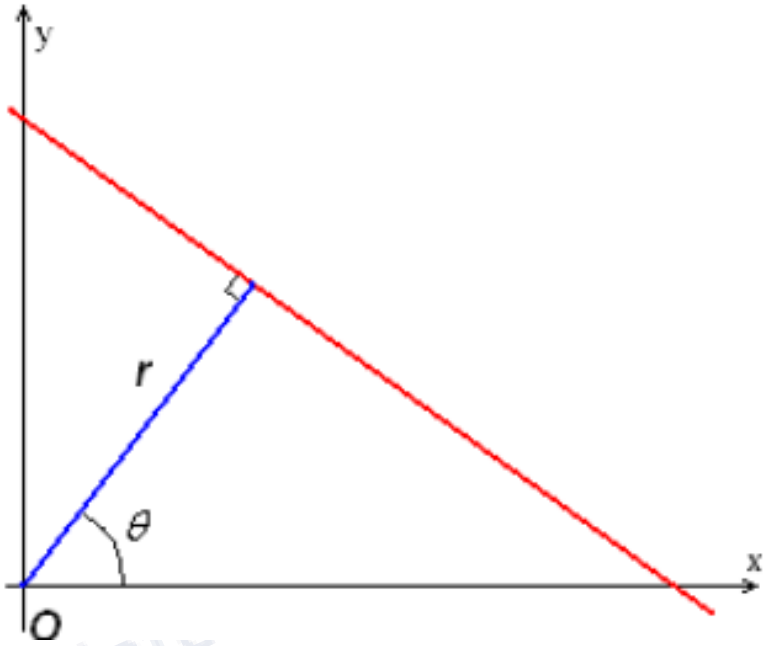
8) 차선 인식 (Hough / HSV / Hough + HSV)



- 위 도로 이미지 속 차선을 Hough transform, HSV(Hue + Saturation + Value) 색 공간 등의 Open CV 기반의 기법을 사용하여 검출

2. Open CV 실습

8) 차선 인식 (Hough / HSV / Hough + HSV)



- Hough transform 기법은 Computer Vision과 Image Process에서 선, 원, 사각형 등의 도형을 검출하는 데 사용되는 기법
- 직선 검출 과정

  : 이미지 gray scale → 이미지 필터링(Gaussian Blur) → 이미지 Edge 검출 → Hough transform

2. Open CV 실습

8) 차선 인식 (Hough / HSV / Hough + HSV)

- Python 환경에서 Open CV의 내장 함수를 이용하여 도로 이미지 속 차선 검출 수행하기 (실습)

  → 주의 사항 1 : Gray scale, Gaussian blur(filtering), Edge detection, Hough transform 사용

  → 주의 사항 2 : 도로 이미지에 따라 검출 결과의 정확도가 다를 수 있음.

```python
def process_image(image_path):
    # 이미지 읽기
    cv_image = cv2.imread(image_path)

    # 1. 이미지를 회색조로 변환
    gray_image = cv2.cvtColor(cv_image, cv2.COLOR_BGR2GRAY)

    # 2. 가우시안 블러 필터 적용
    blurred_image = cv2.GaussianBlur(gray_image, (5, 5), 0)

    # 3. 에지 검출
    edges = cv2.Canny(blurred_image, 50, 150)

    # 4. Hough 변환을 사용한 선 검출
    lines = cv2.HoughLinesP(edges, 1, np.pi / 180, 20, minLineLength=30, maxLineGap=200)
    line_image = np.zeros_like(cv_image)
    if lines is not None:
        for line in lines:
            x1, y1, x2, y2 = line[0]
            cv2.line(line_image, (x1, y1), (x2, y2), (0, 255, 0), 5)

    # 5. 원본 이미지와 검출된 선을 합성
    combined_image = cv2.addWeighted(cv_image, 0.8, line_image, 1, 1)

    # 결과 이미지 표시
    cv2.imshow('Lane Detection', combined_image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

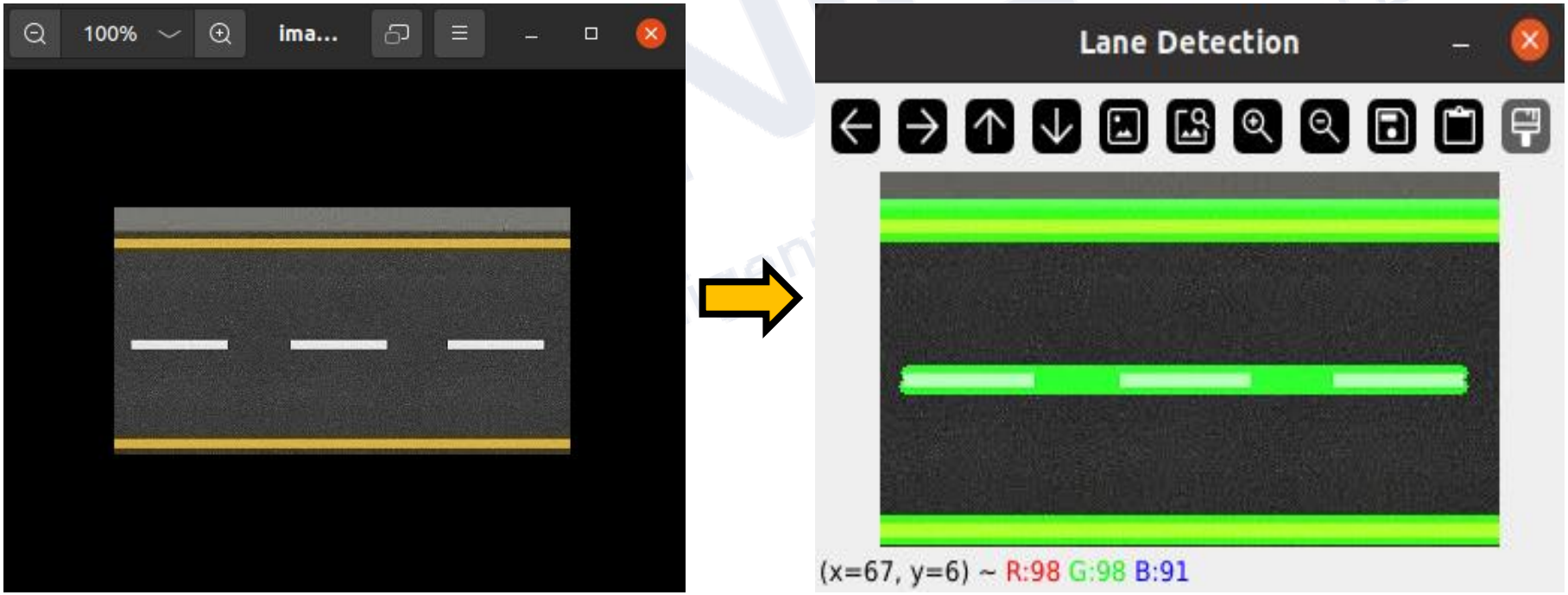
2. Open CV 실습

8) 차선 인식 (Hough / HSV / Hough + HSV)

- Python 환경에서 Open CV의 내장 함수를 이용하여 도로 이미지 속 차선 검출 수행하기 (실습)

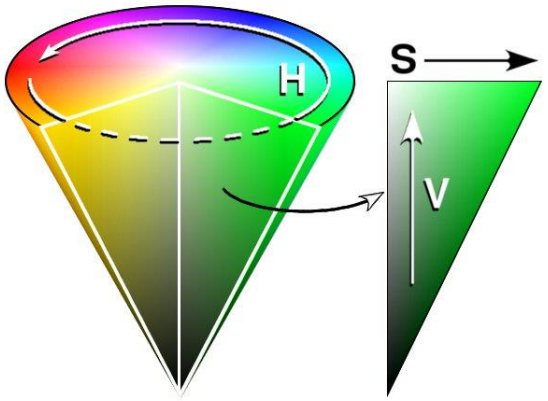
  → 주의 사항 1 : Gray scale, Gaussian blur(filtering), Edge detection, Hough transform 사용

  → 주의 사항 2 : 도로 이미지에 따라 검출 결과의 정확도가 다를 수 있음.



2. Open CV 실습

8) 차선 인식 (Hough / HSV / Hough + HSV)



- HSV(Hue + Saturation + Value) 모델은 인간의 색 인지에 기반을 둔 색상 모델
- HSV는 Hue(색상), Saturation(채도), Value(명도)를 기반으로 하는 색 공간
- 특정 색상 범위를 쉽게 지정하여 필터링(filtering)할 수 있다는 이점 존재

2. Open CV 실습

8) 차선 인식 (Hough / HSV / Hough + HSV)

- Python 환경에서 Open CV의 내장 함수를 이용하여 도로 이미지 속 차선 검출 수행하기 (실습)

  → 주의 사항 1 : HSV 색상 공간을 사용하여 도로 이미지 속 차선(노란색 차선, 흰색 차선)을 검출할 것.

  → 주의 사항 2 : 도로 이미지에 따라 검출 결과의 정확도가 다를 수 있음.

```python
def process_image(image_path):
    # 이미지 읽기
    cv_image = cv2.imread(image_path)

    # 1. 이미지를 HSV 색 공간으로 변환
    hsv_image = cv2.cvtColor(cv_image, cv2.COLOR_BGR2HSV)

    # 2. 노란색 차선 필터링 (HSV 범위 설정)
    yellow_lower = np.array([20, 100, 100])
    yellow_upper = np.array([30, 255, 255])
    yellow_mask = cv2.inRange(hsv_image, yellow_lower, yellow_upper)

    # 3. 흰색 차선 필터링 (HSV 범위 설정)
    white_lower = np.array([0, 0, 200])
    white_upper = np.array([180, 25, 255])
    white_mask = cv2.inRange(hsv_image, white_lower, white_upper)

    # 4. 노란색과 흰색 마스크 결합
    combined_mask = cv2.bitwise_or(yellow_mask, white_mask)

    # 5. 결과 이미지
    combined_image = cv2.bitwise_and(cv_image, cv_image, mask=combined_mask)

    # 결과 이미지 표시
    cv2.imshow('Lane Detection', combined_image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

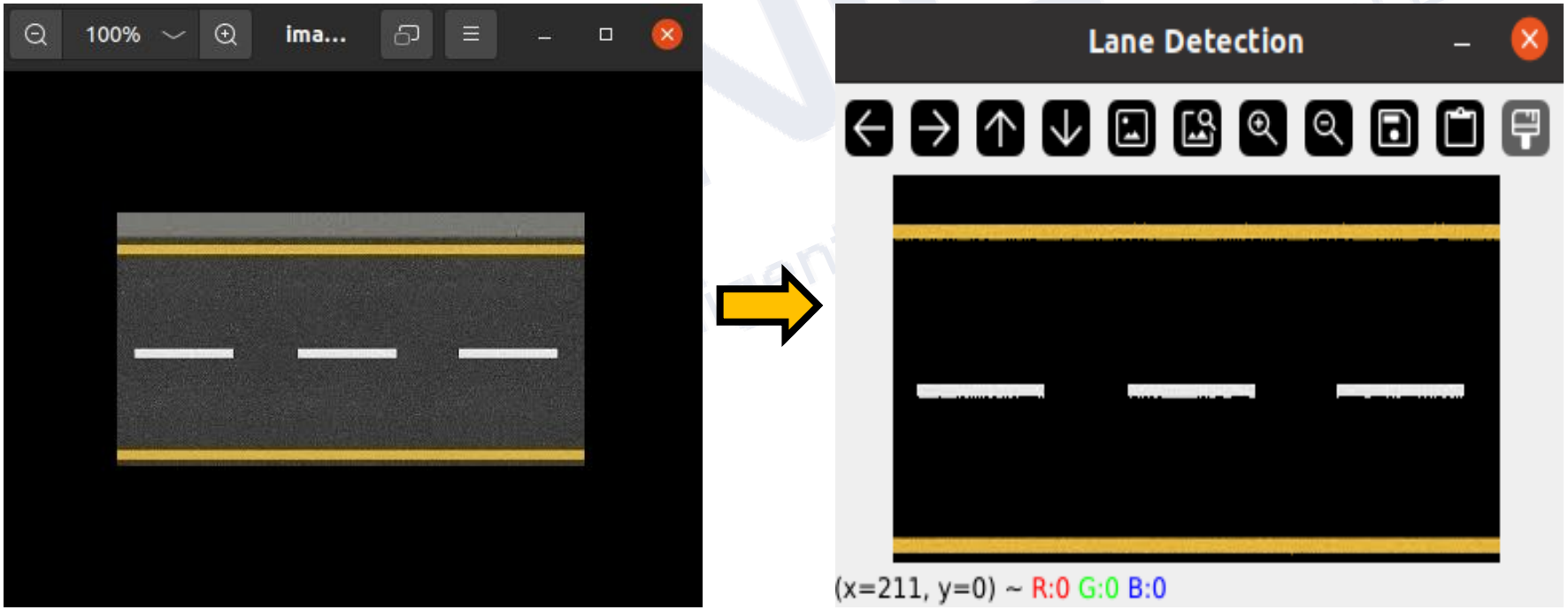
2. Open CV 실습

8) 차선 인식 (Hough / HSV / Hough + HSV)

- Python 환경에서 Open CV의 내장 함수를 이용하여 도로 이미지 속 차선 검출 수행하기 (실습)

  → 주의 사항 1 : HSV 색상 공간을 사용하여 도로 이미지 속 차선(노란색 차선, 흰색 차선)을 검출할 것.

  → 주의 사항 2 : 도로 이미지에 따라 검출 결과의 정확도가 다를 수 있음.



2. Open CV 실습

8) 차선 인식 (Hough / HSV / Hough + HSV)

Method	Characteristic
Hough	▪ Gray scale로 인한 색상 정보 손실 ▪ Edge 검출 수행 시, noise에 민감
HSV	▪ 조명 변화에 민감 ▪ 차선 검출 시, HSV 범위 설정에 민감
Hough + HSV	▪ Noise 및 조명 변화에 대한 강건성 증가 ▪ 다른 객체와의 혼동 감소

2. Open CV 실습

8) 차선 인식 (Hough / HSV / Hough + HSV)

- Python 환경에서 Open CV의 내장 함수를 이용하여 도로 이미지 속 차선 검출 수행하기 (실습)

  → 주의 사항 1 : Hough transform과 HSV 색상 공간을 모두 사용하여 이미지 속 차선을 검출할 것.

  → 주의 사항 2 : 도로 이미지에 따라 검출 결과의 정확도가 다를 수 있음.

```python
# 1. 관심 영역(ROI) 설정 (하단 절반)
roi_vertices = [
    (0, height),
    (0, height // 2),
    (width, height // 2),
    (width, height)
]
mask = np.zeros_like(cv_image)
match_mask_color = (255,) * cv_image.shape[2]
cv2.fillPoly(mask, np.array([roi_vertices], np.int32), match_mask_color)
masked_image = cv2.bitwise_and(cv_image, mask)

# 2. 이미지를 HSV 색 공간으로 변환
hsv_image = cv2.cvtColor(masked_image, cv2.COLOR_BGR2HSV)

# 3. 노란색 차선 필터링 (HSV 범위 설정)
yellow_lower = np.array([20, 100, 100])
yellow_upper = np.array([30, 255, 255])
yellow_mask = cv2.inRange(hsv_image, yellow_lower, yellow_upper)

# 4. 흰색 차선 필터링 (HSV 범위 설정)
white_lower = np.array([0, 0, 200])
white_upper = np.array([180, 25, 255])
white_mask = cv2.inRange(hsv_image, white_lower, white_upper)

# 5. 노란색과 흰색 마스크 결합
combined_mask = cv2.bitwise_or(yellow_mask, white_mask)

# 6. 에지 검출 (HSV 필터링 후)
edges = cv2.Canny(combined_mask, 50, 150)

# 7. Hough 변환을 사용한 선 검출
lines = cv2.HoughLinesP(edges, 1, np.pi / 180, 20, minLineLength=30, maxLineGap=200)
if lines is not None:
    for line in lines:
        x1, y1, x2, y2 = line[0]
        cv2.line(cv_image, (x1, y1), (x2, y2), (0, 255, 0), 5)

# 결과 이미지 표시
cv_image = cv2.resize(cv_image, (960, 540))  # 이미지 크기 조정
cv2.imshow('Lane Detection', cv_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

ROI(Region of Interest)를 통해 이미지 속 필요한 영역만을 선택 후 처리 가능

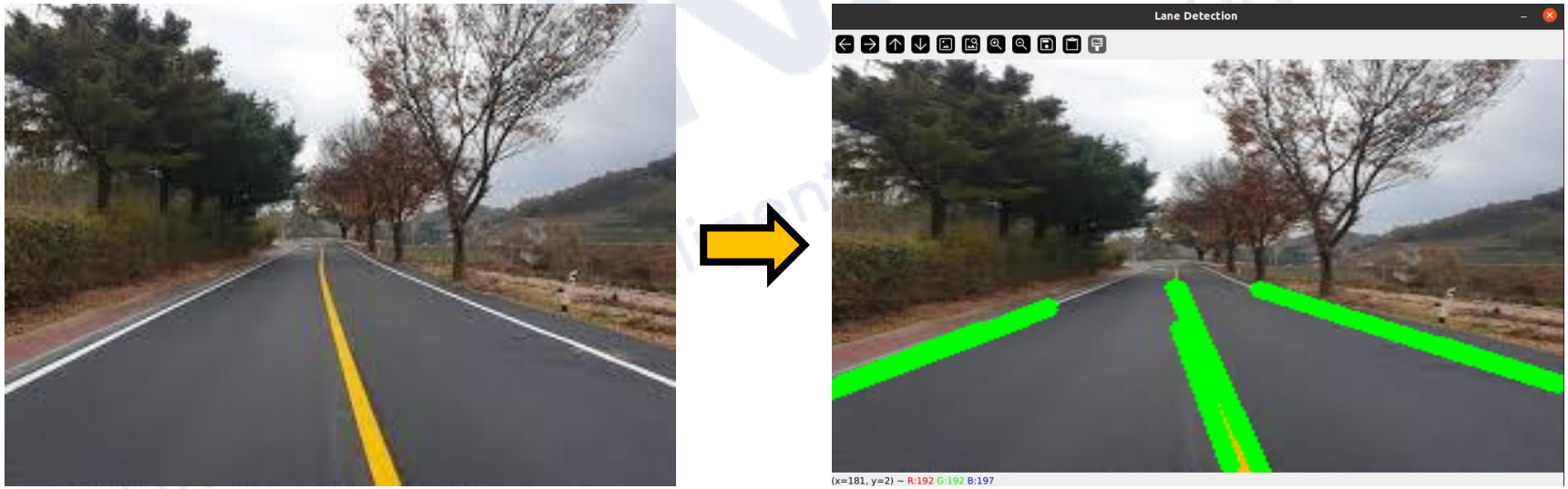
2. Open CV 실습

8) 차선 인식 (Hough / HSV / Hough + HSV)

- Python 환경에서 Open CV의 내장 함수를 이용하여 도로 이미지 속 차선 검출 수행하기 (실습)

  → 주의 사항 1 : Hough transform과 HSV 색상 공간을 모두 사용하여 **이미지 속 차선을 검출**할 것.

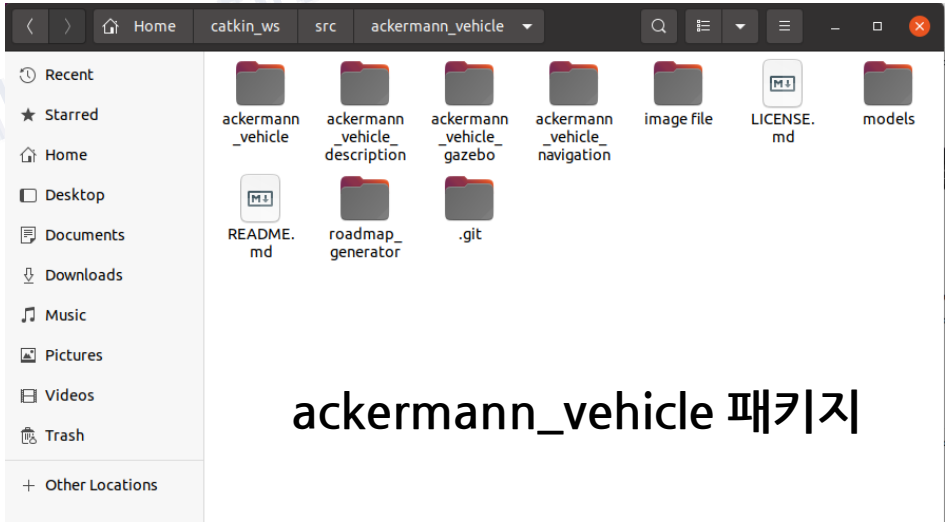
  → 주의 사항 2 : 도로 이미지에 따라 검출 결과의 정확도가 다를 수 있음.



2. Open CV 실습

9) ROS Gazebo 시뮬레이션 기반 차선 인식

- ROS Gazebo simulation environment setup
  - → $ cd ~/catkin_ws/src/
  - → $ git clone https://github.com/gihoonbackend/ackermann_vehicle.git
  - → $ sudo apt install ros-noetic-ackermann-msgs
  - → $ cd ..
  - → $ rosdep install --from-paths src --ignore-src -r -y
  - → $ catkin_make
  - → $ source devel/setup.bash



ackermann_vehicle 패키지

2. Open CV 실습

9) ROS Gazebo 시뮬레이션 기반 차선 인식

- ROS Gazebo simulation environment setup

  → $ gedit ~/.bashrc

```
119 alias eb='nano ~/.bashrc'
120 alias sb='source ~/.bashrc'
121 alias gs='git status'
122 alias gp='git pull'
123 alias cw='cd ~/catkin_ws'
124 alias cs='cd ~/catkin_ws/src'
125 alias cm='cd ~/catkin_ws && catkin_make'
126 source /opt/ros/noetic/setup.bash
127 source ~/catkin_ws/devel/setup.bash
128 export ROS_MASTER_URI=http://localhost:11311
129 export ROS_HOSTNAME=localhost
130 export TURTLEBOT3_MODEL=burger
131 export GAZEBO_MODEL_PATH=$GAZEBO_MODEL_PATH:/home/gihoon/catkin_ws/src/ackermann_vehicle/roadmap_generator
```

→ $ export GAZEBO_MODEL_PATH=$GAZEBO_MODEL_PATH

:/home/user/catkin_ws/src/ackermann_vehicle/roadmap_generator

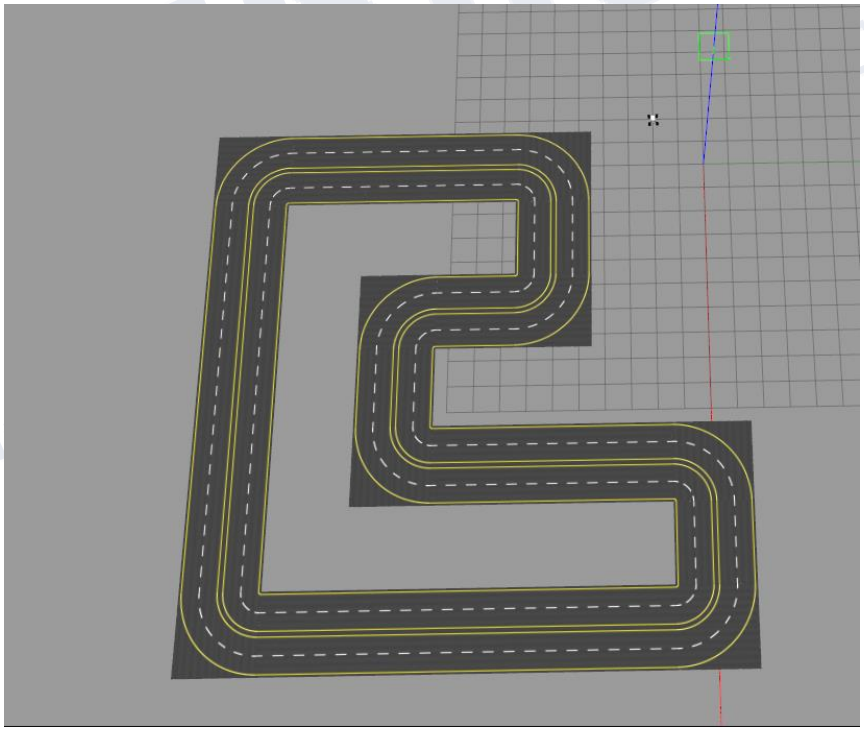
본인의 경로를 정확하게 작성 !

→ $ source ~/.bashrc

2. Open CV 실습

9) ROS Gazebo 시뮬레이션 기반 차선 인식

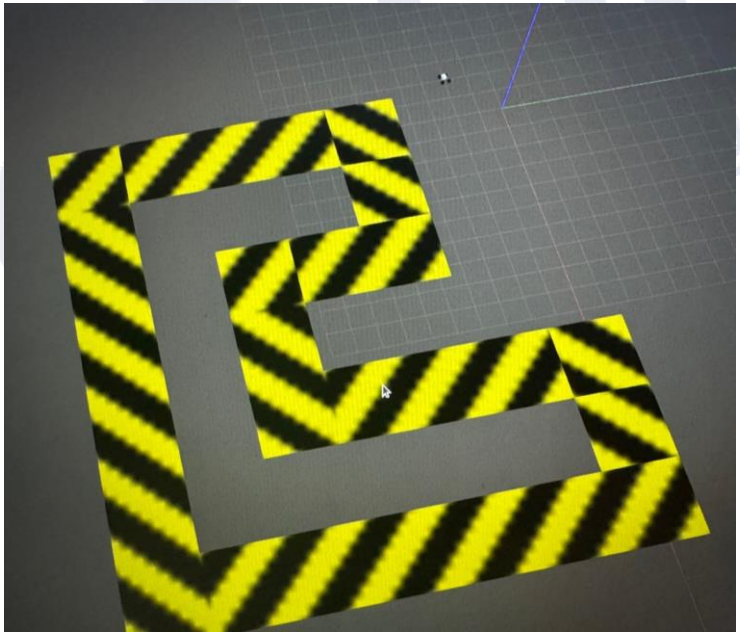
- ROS Gazebo simulation environment setup
  - → $ cd catkin_ws/src/ackermann_vehicle/ackermann_vehicle_gazebo/launch
  - → $ roslaunch ackermann_vehicle_noetic.launch
    - → 위 명령어를 입력하면, Gazebo simulation world와 Robot model이 생성됨

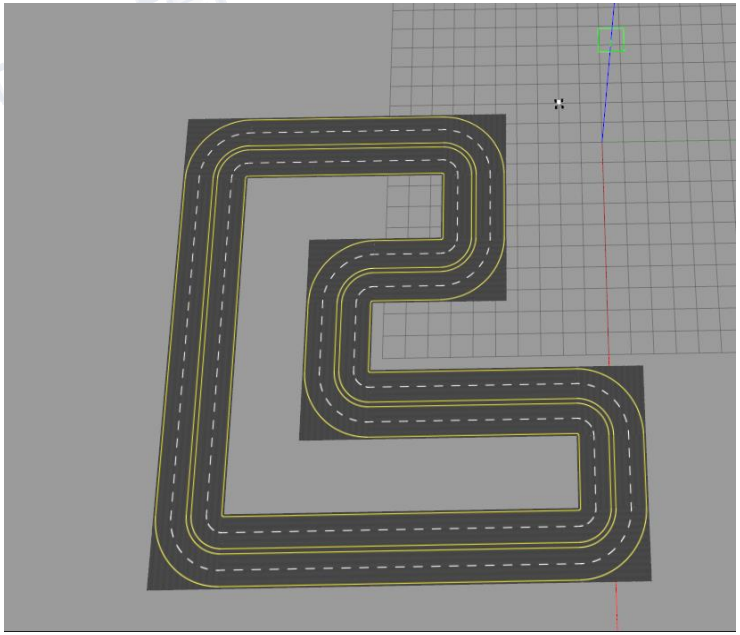


2. Open CV 실습

9) ROS Gazebo 시뮬레이션 기반 차선 인식

- 만약 아래의 사진과 같이 track이 생성되지 않는다면 다음의 경로로 진입하여 이미지 파일(png 파일)을 교체 후, 앞장의 launch 파일 재 실행
  - → $ cd ~/catkin_ws/src/ackermann_vehicle/roadmap_generator/road_straight/materials/textures
  - → "road_straight.png" 파일을 새로운 이미지 파일로 교체
  - → $ cd ~/catkin_ws/src/ackermann_vehicle/roadmap_generator/road_curve/materials/textures
  - → "road_curve.png" 파일을 새로운 이미지 파일로 교체
  - → $ roslaunch ackermann_vehicle_gazebo ackermann_vehicle_noetic.launch





2. Open CV 실습

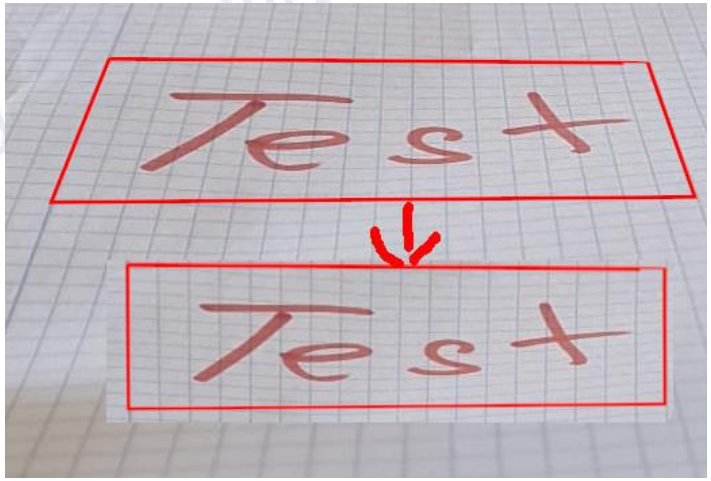
9) ROS Gazebo 시뮬레이션 기반 차선 인식

- Bird-Eye View

→ Bird-Eye View(BEV)란 로봇이나 차량이 마치 하늘에서 내려다보는 것처럼 도로와 주변 환경을 평평하게 볼 수 있도록 하는 변환 기법

→ BEV **변환**은 자율주행 및 로봇 공학에서 주로 사용

→ BEV는 실 세계의 좌표를 이미지 평면에 정확히 투영하므로, 실제 거리와 각도를 계산할 때 더 정확하게 제공 가능



2. Open CV 실습

9) ROS Gazebo 시뮬레이션 기반 차선 인식

- Bird-Eye View

→ Bird-Eye View(BEV)는 Open CV의 내장 함수를 통해 구현 가능

$$w \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

→ Open CV의 getPerspectiveTransform() 함수는 변환 전과 후를 짝짓는 4개의 매핑 좌표를 통해 원근 변환에 필요한 3x3 크기의 원근 변환 행렬을 계산하여 도출

→ Open CV의 warpPerspective() 함수는 getPerspectiveTransform() 함수를 통해 계산된 3x3 크기의 원근 변환 행렬을 기반으로 특정 이미지에 대한 원근 변환을 수행

2. Open CV 실습

9) ROS Gazebo 시뮬레이션 기반 차선 인식 실습

- **ROS Gazebo simulation 환경에서 Camera image 데이터를 Subscribe한 뒤,**

  Camera image 속 흰색 차선과 노란색 차선을 인식하기 (실습 1)

  → **주의 사항 1 :** 차량(로봇) 모델을 Gazebo simulation 환경 속 도로로 이동 후 명령어를 입력할 것.

  → **주의 사항 2 :** 앞서 설명한 Gazebo simulation 환경을 먼저 실행 후, 새로운 터미널에서 실행할 것.

  → Gazebo simulation 실행

  → 새로운 Terminal에서 다음의 명령어를 입력하여 해당 폴더로 이동

  $ cd catkin_ws/src/ackermann_vehicle/ackermann_vehicle_gazebo/scripts

  → Python 파일 생성 및 코드 작성

  $ gedit lane_detection.py

  → Python 파일 실행

  $ python3 lane_detection.py

2. Open CV 실습

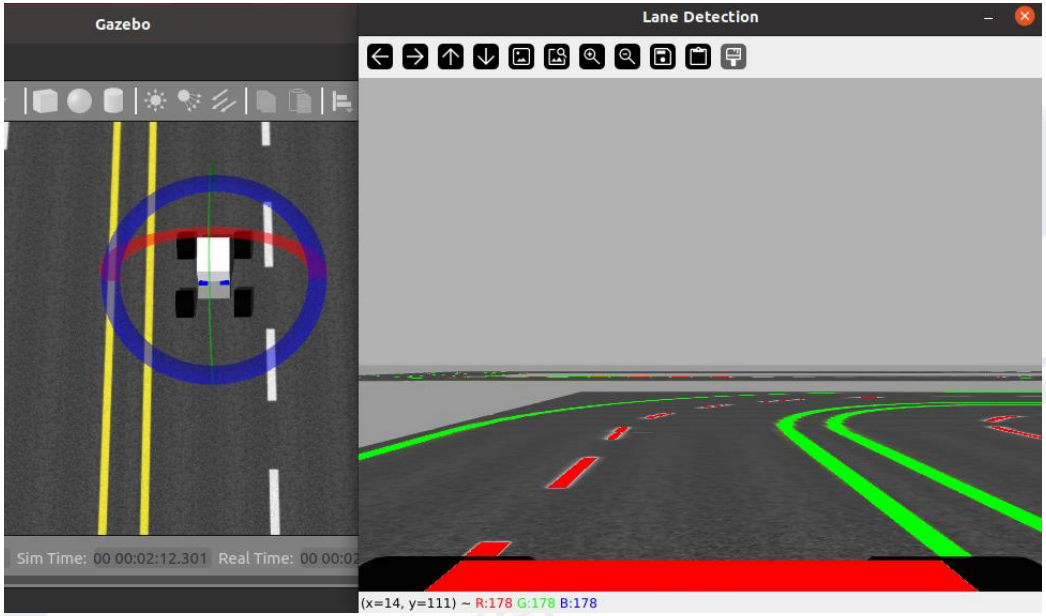
9) ROS Gazebo 시뮬레이션 기반 차선 인식 실습 : 주요 코드 분석

```python
def camera_callback(self, data):
    try:
        # Read camera data
        cv_image_raw = self.bridge.imgmsg_to_cv2(data, "bgr8")

        hsv = cv2.cvtColor(cv_image_raw, cv2.COLOR_BGR2HSV)

        # White mask HSV range
        lower_white = np.array([0, 0, 180])
        upper_white = np.array([255, 30, 255])

        # Yellow mask HSV range
        lower_yellow = np.array([20, 100, 100])
        upper_yellow = np.array([30, 255, 255])

        mask_white = cv2.inRange(hsv, lower_white, upper_white)
        mask_yellow = cv2.inRange(hsv, lower_yellow, upper_yellow)

        # Find white and yellow lane points
        white_lane_points = cv2.findNonZero(mask_white)
        yellow_lane_points = cv2.findNonZero(mask_yellow)

        # Draw white lane points as red dots
        if white_lane_points is not None:
            for point in white_lane_points:
                x, y = point[0]
                cv_image_raw[y, x] = [0, 0, 255]  # Red color

        # Draw yellow lane points as green dots
        if yellow_lane_points is not None:
            for point in yellow_lane_points:
                x, y = point[0]
                cv_image_raw[y, x] = [0, 255, 0]  # Green color

        cv2.imshow('Lane Detection', cv_image_raw)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            rospy.signal_shutdown("User exit with 'q' key")
            cv2.destroyAllWindows()

    except CvBridgeError as e:
        print(e)
```

- **Camera 센서로부터 이미지 데이터 읽기**
  → ROS 이미지 메시지를 Open CV 이미지로 변환
- **HSV 색상 공간 변환**
  → BGR 이미지를 HSV 색상 공간으로 변환
- **흰색과 노란색 mask 생성**
  → HSV 색상 범위를 사용하여 흰색과 노란색 mask 생성
- **차선 포인트 찾기 및 차선 그리기**
  → cv2.findNonZero() 함수를 통해 포인트를 찾고 표시
- **이미지 화면에 출력**
  → Open CV 윈도우에 변환한 이미지를 출력

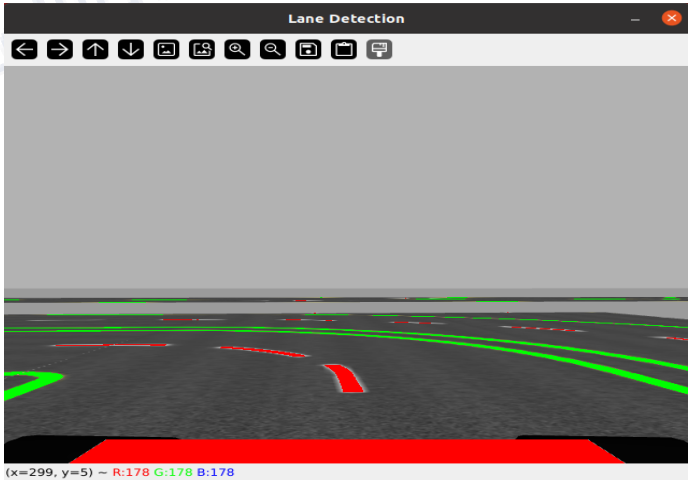
2. Open CV 실습

9) ROS Gazebo 시뮬레이션 기반 차선 인식 실습



다음의 그림과 같이

도로를 인식하면 성공 !!



2. Open CV 실습

9) ROS Gazebo 시뮬레이션 기반 차선 인식 실습

- ROS Gazebo simulation 환경에서 Camera image 데이터를 Subscribe한 뒤,

  Camera image를 “Bird-Eye View”로 변환 후 흰색 차선을 인식하기 (실습 2)

→ **주의 사항 1 :** 앞서 설명한 **Gazebo simulation 환경을 먼저 실행 후**, 새로운 터미널에서 실행할 것.

→ **주의 사항 2 : 차량(로봇) 모델을 Gazebo simulation 환경 속 도로로 이동 후** 명령어를 입력할 것.

→ **주의 사항 3 :** 인식된 흰색 차선에 **로봇이 추종해야 할 way point**가 표시될 수 있도록 할 것.

- **$ roslaunch ackermann_vehicle_gazebo ackermann_vehicle_noetic.launch**
  (Gazebo simulation world 실행)
- **$ cd catkin_ws/src/ackermann_vehicle/ackermann_vehicle_gazebo/scripts**
  (새로운 Terminal에서 다음의 명령어를 입력하여 해당 폴더로 이동)
- **$ gedit bev_lane_detection.py**
  (Python 파일 생성 및 코드 작성 후 저장)
- **$ python3 bev_lane_detection.py**
  (Python 파일 실행)

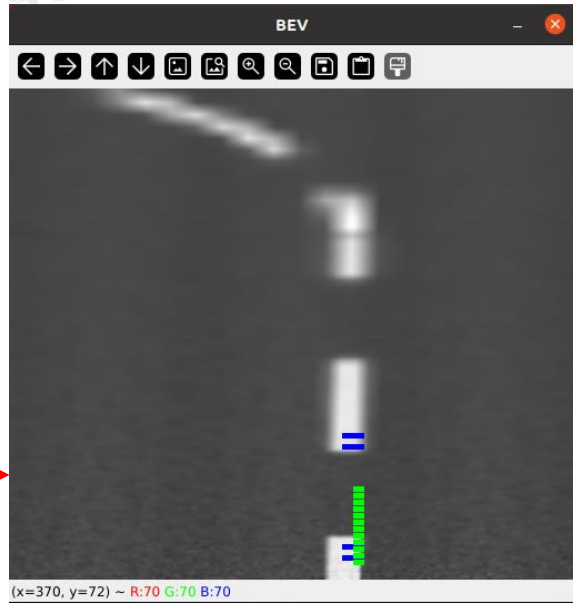
2. Open CV 실습

9) ROS Gazebo 시뮬레이션 기반 차선 인식 실습 : 주요 코드 분석

```python
def camera_setting(self, data):
    # read camera data, and ROI
    cv_image_raw = self.bridge.imgmsg_to_cv2(data, "bgr8")
    cv_image = cv_image_raw[0:640][300:480]

    # bev_tf
    A = [(251, 13), (400, 13), (580, 140), (70, 140)] # two line
    bev_image = self.bev.birdeyeview(cv_image, A)

    hsv = cv2.cvtColor(bev_image, cv2.COLOR_BGR2HSV)

    # white mask hsv range
    lower_white = np.array([0, 0, 200])
    upper_white = np.array([180, 25, 255])

    mask_white = cv2.inRange(hsv, lower_white, upper_white)

    image = bev_image.copy()

    return mask_white, image
```

- **Camera 센서로부터 이미지 데이터 읽기**
  → ROS 이미지 메시지를 Open CV 이미지로 변환
- **이미지 데이터의 관심 영역 설정**
  → ROI(Region of Interest)를 설정
- **ROI가 설정된 이미지의 시점 변환**
  → 변환에 사용될 4개의 점을 기반으로 Bird-Eye View(BEV) 변환 수행
- **이미지 데이터의 색상 공간 변환**
  → BGR 이미지를 HSV 색상 공간으로 변환

다음과 같은 이미지가 출력되면 성공!!



2. Open CV 실습

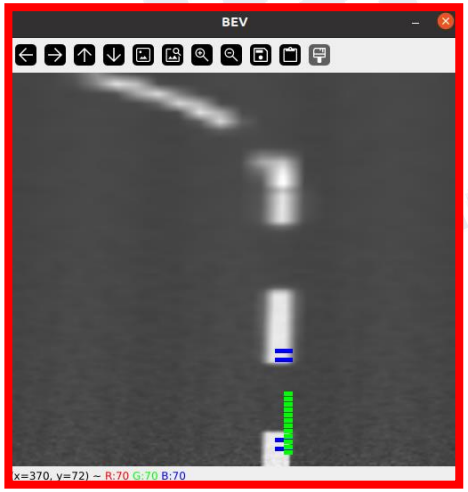
9) ROS Gazebo 시뮬레이션 기반 차선 인식 실습

- ROS Gazebo simulation 환경에서 Camera image 데이터를 Subscribe한 뒤,

  Camera image를 “Bird-Eye View”로 변환 후 흰색 차선을 인식 및 추종하기 (실습 3)

  → **주의 사항 1 :** 앞서 설명한 **Gazebo simulation 환경을 먼저 실행 후, 차량(로봇) 모델을 Gazebo simulation 환경 속 도로로 이동** 할 것.

  → **주의 사항 2 :** 흰색 차선에 계산된 **way point**가 표시되면, 이를 추종하며 주행할 수 있도록 할 것.



- **$ roslaunch ackermann_vehicle_gazebo ackermann_vehicle_noetic.launch**
  (Gazebo simulation world 실행)
- **$ cd catkin_ws/src/ackermann_vehicle/ackermann_vehicle_gazebo/scripts**
  (새로운 Terminal에서 다음의 명령어를 입력하여 해당 폴더로 이동)
- **$ gedit bev_lane_detection_and_tracking.py**
  (Python 파일 생성 및 코드 작성 후 저장)
- **$ python3 bev_lane_detection_and_tracking.py**
  (Python 파일 실행)

2. Open CV 실습

9) ROS Gazebo 시뮬레이션 기반 차선 인식 실습 : 주요 코드 분석

```
10  from geometry_msgs.msg import Twist
11  from purepursuit import estimate_line
```

- **Waypoint를 추종하기 위해 제어와 관련된 라이브러리**
  - → 속도 제어 입력을 Publish 하기 위해 Twist 메시지 import
  - → 차량의 횡방향 제어 입력을 계산하기 위해 Pure pursuit 라이브러리 import

```
96  if __name__ == "__main__":
97      try:
98          cmd_vel_pub = rospy.Publisher('/cmd_vel', Twist, queue_size=10)
99          move_cmd = Twist()
```

- **Control input Publisher 생성**
  - → 생성된 Waypoint를 로봇이 올바르게 추종할 수 있도록 하는 제어 입력을 Publish 하기 위한 Publisher 생성

```
73  # Calculate the steering angle based on the detected lane points
74  steering_angle = estimate_line(self.y_repeat, self.left_lane_pred, self.left_lane_pred, image)
```

- **Lateral control input 계산**
  - → import한 Pure pursuit 라이브러리를 통해 횡방향 제어 입력 계산

```
83          # Set fixed speed
84          move_cmd.linear.x = 0.3              # [m/s]
85          move_cmd.angular.z = steering_angle # [rad/s]
86          cmd_vel_pub.publish(move_cmd)
```

- **Control input Publish 수행**
  - → 특정 값(0.3m/s)으로 고정한 종방향 제어 입력과, Pure pursuit 라이브러리를 통해 계산된 횡방향 제어 입력을 Publish하여 로봇 제어

2. Open CV 실습

9) ROS Gazebo 시뮬레이션 기반 차선 인식 실습

- **ROS Gazebo simulation 환경에서 Camera image 데이터를 Subscribe한 뒤, image를 "Bird-Eye View"로 변환 후 흰색 차선과 노란색 차선을 인식 후 주행하기 (실습 4)**

  → 주의 사항 1 : 앞서 설명한 **Gazebo simulation 환경을 먼저 실행 후, 차량(로봇) 모델을 Gazebo simulation 환경 속 도로로 이동** 할 것.

  → 주의 사항 2 : 인식된 흰색 차선(왼쪽)과 노란색 차선(오른쪽)의 중간점이자 추종해야 하는 **way point**가 계산되면, 로봇이 이를 추종하며 주행할 수 있도록 알고리즘을 구현할 것.

  - Gazebo simulation 실행
  - 새로운 Terminal에서 다음의 명령어를 입력하여 해당 폴더로 이동

    $ cd catkin_ws/src/ackermann_vehicle/ackermann_vehicle_gazebo/scripts
  - Python 파일 생성 및 코드 작성 후 저장

    $ gedit bev_lane_detection_and_tracking2.py

    Python 파일 실행

    $ python3 bev_lane_detection_and_tracking2.py

2. Open CV 실습

9) ROS Gazebo 시뮬레이션 기반 차선 인식 실습 : 주요 코드 분석

```
20          self.y_repeat = 12
21          self.right_x = 0
22          self.left_x = 0
23          self.point_scale = 10
24
25          self.left_lane_pred = np.zeros((self.y_repeat,2))
26          self.right_lane_pred = np.zeros((self.y_repeat,2))
```

- **알고리즘에 필요한 초기 변수 설정**

→ 왼쪽 차선과 오른쪽 차선을 통해 중앙점인 Waypoint를 생성하기 위해 필요한 변수 설정.

```
29          # read cameara data, and ROI
30          cv_image_raw = self.bridge.imgmsg_to_cv2(data, "bgr8")
31          cv_image = cv_image_raw[0:640][300:480]
```

- **이미지 데이터의 크기 재설정**

→ 차선 인식에 전체 이미지 데이터가 필요하지는 않으므로, ROI를 설정하여 이미지의 크기를 resize 수행.

```
39          # white mask hsv lange
40          lower_white = np.array([0, 0, 200])
41          upper_white = np.array([180, 25, 255])
42          #lower_white = np.array([0, 0, 180])
43          #upper_white = np.array([255, 30, 255])
44
45          # yello mask hsv lange
46          lower_yellow = np.array([20, 100, 70])
47          upper_yellow = np.array([30, 255, 255])
48
49          mask_white = cv2.inRange(hsv, lower_white, upper_white)
50          mask_yellow = cv2.inRange(hsv, lower_yellow, upper_yellow)
```

- **차선 인식을 위한 HSV 범위 설정**

→ 하얀색 및 노란색 차선 인식을 위해 각 색에 맞는 HSV 범위를 설정 수행

2. Open CV 실습

9) ROS Gazebo 시뮬레이션 기반 차선 인식 실습 : 주요 코드 분석

```python
    def find_certain_color(self,mask_certain,threshold,j):
        certain_pixels = cv2.countNonZero(mask_certain[self.yy:self.yy+5, j*10:(j+1)*10])

        if certain_pixels > threshold:
            return [(j+1)*10,self.yy+2]
        else:
            return []
```

```python
def camera_callback(self, data):
    try:
        mask_white, mask_yellow, image = self.camera_setting(data)

        for i in range(self.y_repeat):
            self.yy = 430 - (i + 1) * self.point_scale # point_scale 간격으로 위로 이동
            self.left_x = 0
            self.right_x = 0

            for j in range(64):
                self.left_lane_points = self.find_certain_color(mask_white,15,j)
                self.right_lane_points = self.find_certain_color(mask_yellow,35,j)

                if len(self.left_lane_points) > 0:
                    self.left_x = self.left_lane_points[0]
                if len(self.right_lane_points) > 0:
                    self.right_x = self.right_lane_points[0]

            self.left_lane_pred[i][0] = self.left_x
            self.left_lane_pred[i][1] = self.yy+2
            image[self.yy:self.yy+5, self.left_x-20:self.left_x] = [255, 0, 0]

            self.right_lane_pred[i][0] = self.right_x
            self.right_lane_pred[i][1] = self.yy+2
            image[self.yy:self.yy+5, self.right_x-20:self.right_x] = [0, 0, 255]
```

- **특정 색상의 픽셀 탐색**
  → 주어진 mask에서 특정 영역의 픽셀 수를 센 뒤, 임계값(Threshold)을 초과하는 경우 좌표를 반환.
  → 임계값을 초과하지 않는 경우, 빈 리스트를 반환.

- **Waypoint 계산 및 추종을 위한 제어 입력 계산**
  → self.camera_setting() 함수로부터 흰색 및 노란색 차선 mask와 변환된 이미지를 반환.
  → 이중 반복문(for문)을 통해 차선 좌표 탐색 수행. (좌표 탐색 시, x좌표만을 탐색 수행)
  → 탐색한 좌표를 저장 후, 이미지 업데이트 수행.

2. Open CV 실습

9) ROS Gazebo 시뮬레이션 기반 차선 인식 실습 : 주요 코드 분석

```
 99                move_cmd.linear.x = 0.3              # [m/s]
100                move_cmd.angular.z = steering_angle # [rad/s]
101                cmd_vel_pub.publish(move_cmd)
```

- **Control input Publish 수행**

→ 특정 값(0.3m/s)으로 고정한 종방향 제어 입력과, Pure pursuit 라이브러리를 통해 계산된 횡방향 제어 입력을 Publish하여 로봇 제어

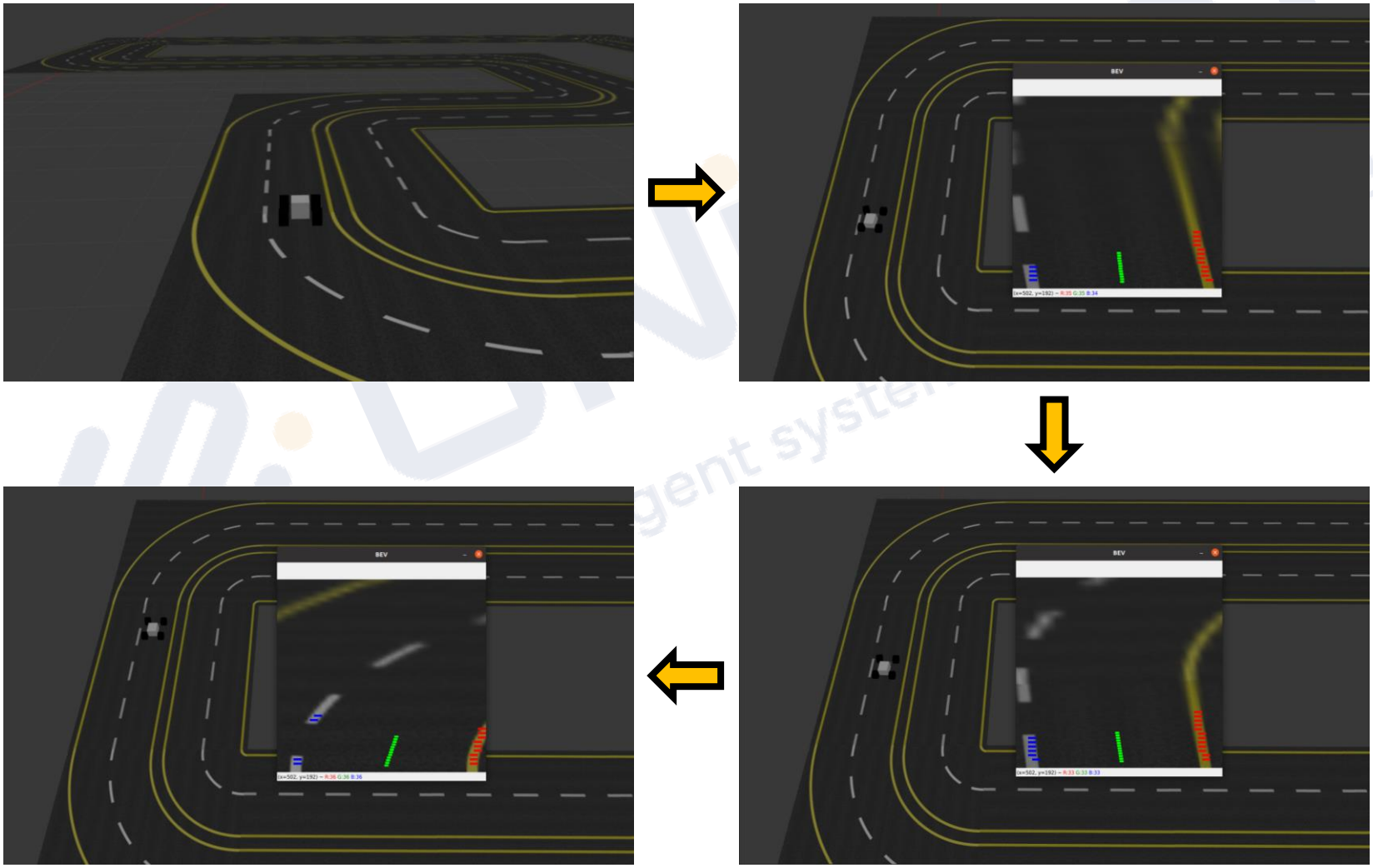
```
110  if __name__ == "__main__":
111      try:
112          cmd_vel_pub = rospy.Publisher('/cmd_vel', Twist, queue_size=10)
113          move_cmd = Twist()
114
115          rospy.init_node("lane_detect")
116          unicon_cv = Unicon_CV()
117          unicon_cv.main()
```

- **Control input Publisher 생성**

→ 생성된 Waypoint를 로봇이 올바르게 추종할 수 있도록 하는 제어 입력을 Publish 하기 위한 Publisher 생성

2. Open CV 실습

9) ROS Gazebo 시뮬레이션 기반 차선 인식 실습 결과



2. Open CV 실습

10) ROS 환경 내 Intel Realsense D455 Camera 구동

Intel Realsense D455 Model

- **Realsense D455 모델은 인텔(Intel)사가 개발한 스테레오 깊이 카메라(Stereo Depth Camera)로, 주로 깊이 감지(depth detection)와 3D 비전(3D Vision) 애플리케이션에 사용됨**

  → **깊이 감지 기술 :** D455 모델은 스테레오 비전 기술을 활용하여 깊이 정보를 캡쳐 가능

  → **고해상도 :** 고해상도 깊이 및 RGB 이미지를 제공

  → **넓은 시야각 :** D455 모델은 넓은 시야각을 제공함으로써 한 번에 넓은 영역을 캡쳐 가능

  → **ROS 호환성 :** D455 모델은 ROS 환경과 호환되어 로봇 공학 및 컴퓨터 비전 응용 프로그램에 쉽게 통합하여 사용 가능

2. Open CV 실습

10) ROS 환경 내 Intel Realsense D455 Camera 구동

- Realsense D455 모델 사용을 위한 SDK 설치

→ 서버의 공개 키를 등록

```
$  sudo mkdir -p /etc/apt/keyrings
$ curl -sSf https://librealsense.intel.com/Debian/librealsense.pgp |
  sudo tee /etc/apt/keyrings/librealsense.pgp > /dev/null
```

→ 저장소 목록에 서버를 추가

```
$ lsb_release -cs (your linux distribution check)
$  echo "deb [signed-by=/etc/apt/keyrings/librealsense.pgp]
   https://librealsense.intel.com/Debian/apt-repo (your linux distribution) main" |
   sudo tee /etc/apt/sources.list.d/librealsense.list
$ sudo apt-get update
```

* 참고 : https://github.com/IntelRealSense/librealsense/blob/master/doc/distribution_linux.md

2. Open CV 실습

10) ROS 환경 내 Intel Realsense D455 Camera 구동

- Realsense D455 모델 사용을 위한 ROS 패키지 설치

→ Realsense 관련 라이브러리 설치

```
$ sudo apt-get install librealsense2-dkms
$ sudo apt-get install librealsense2-utils
```

→ ROS 환경에서 Realsense Camera를 사용하기 위한 패키지를 설치

```
$ sudo apt-get install ros-noetic-realsense2-camera
$ sudo apt-get install ros-noetic-realsense2-description
```

2. Open CV 실습

10) ROS 환경 내 Intel Realsense D455 Camera 구동

- Realsense D455 Camera와 Local PC 연결 및 실행

→ Camera를 Local PC와 연결 후, USB 권한 설정

$ lsusb

$ sudo chmod 666 /dev/bus/usb/001/004 → 해당 숫자는 본인의 Terminal상에 출력되는 값으로 !!

```
gihoon@gihoon-B650M-HDV-M-2:~$ lsusb
Bus 008 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 007 Device 004: ID 25a7:fa61
Bus 007 Device 003: ID 248a:8373 Maxxter USB2.0 Hub
Bus 007 Device 002: ID 05e3:0610 Genesys Logic, Inc. 4-port hub
Bus 007 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 006 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 005 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 004 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 002: ID 0a12:0001 Cambridge Silicon Radio, Ltd Bluetooth Dongle (
HCI mode)
Bus 001 Device 004: ID 8086:0b5c Intel Corp. Intel(R) RealSense(TM) Depth Camera
 455
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

2. Open CV 실습

10) ROS 환경 내 Intel Realsense D455 Camera 구동

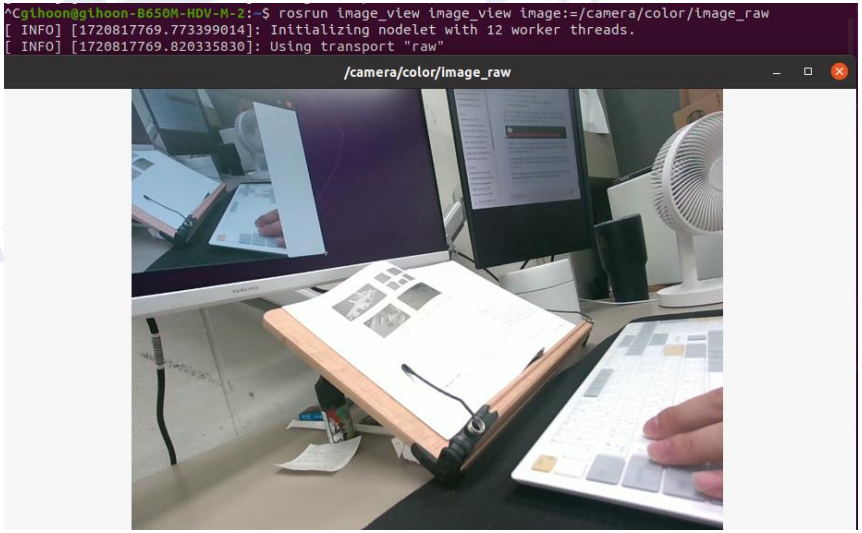
- Realsense D455 Camera 기반 RGB 이미지 출력

  → ROS 이미지 뷰 패키지 설치

  $ sudo apt-get install ros-noetic-image-view

  $ roslaunch realsense2_camera rs_camera.launch (첫 번째 터미널)

  $ rosrun image_view image_view image:=/camera/color/image_raw (두 번째 터미널)



2. Open CV 실습

10) ROS 환경 내 Intel Realsense D455 Camera 구동

- Realsense D455 Camera 기반 Depth(깊이) 이미지 출력

  $ roslaunch realsense2_camera rs_camera.launch (첫 번째 터미널)

  $ rosrun image_view image_view image:=/camera/depth/image_rect_raw (두 번째 터미널)

QnA

E-mail : leehwasu9696@inu.ac.kr
smr4207@inu.ac.kr

Mobile : 010-8864-5585
010-7142-7344

청주대학교
미래형자동차 인력양성사업단

자율주행 분야

단기집중 교육과정

6월 11일(화) 오후 4시~8시	리눅스 설치, ROS 설치 및 ROS 개발환경 구축	융합관 410호 *개인 노트북 필수 지참(대여불가)
온라인 사전교육	Python 교육(ROS 기반)	파이썬 기초 / 파이썬 중급 / ROS 기초강의
7월 2일~ 5일(화~금) 오후 1시~6시	라이다 센서 교육	새천년종합정보관 107A호 경진대회 참여학생
7월 8일~ 10일(월~수) 오후 1시~6시	ROS 활용 교육(Python 기반)	융합관 410호 *개인 노트북 필수 지참(대여불가)
7월 15일~ 17일(월~수) 오후 1시~6시	Python OpenCV 설치 및 응용	새천년종합정보관 409호 *개인 노트북 필수 지참(대여불가)

청주대학교
CHEONGJU UNIVERSITY

Thank you.