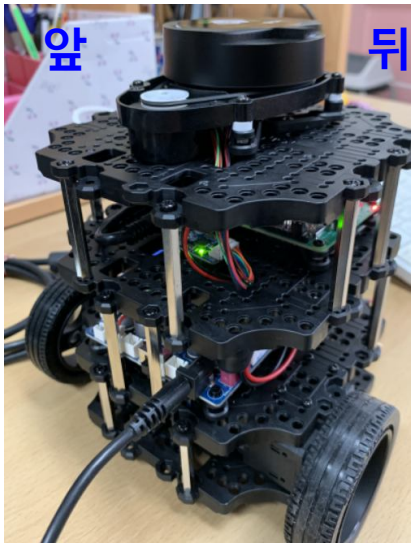


만일 실험을 마치려면, 모든 terminal은 CTL+C로 중단 후에 종료

Turtlebot은 sudo shutdown now로 종료

ROS Programming (SLAM)

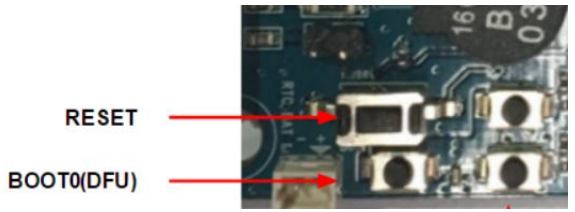
(file #5 / ?) ver1.0 Noetic

Battery 사전에 충전하기

(미충전 경우 Buzzer 소리 발생)

➔ 방치 시 완전 방전됨

Yongseok Chi



만일 회전 안하면, openCR의 RESET누르고, Ctl+c, 새 터미널

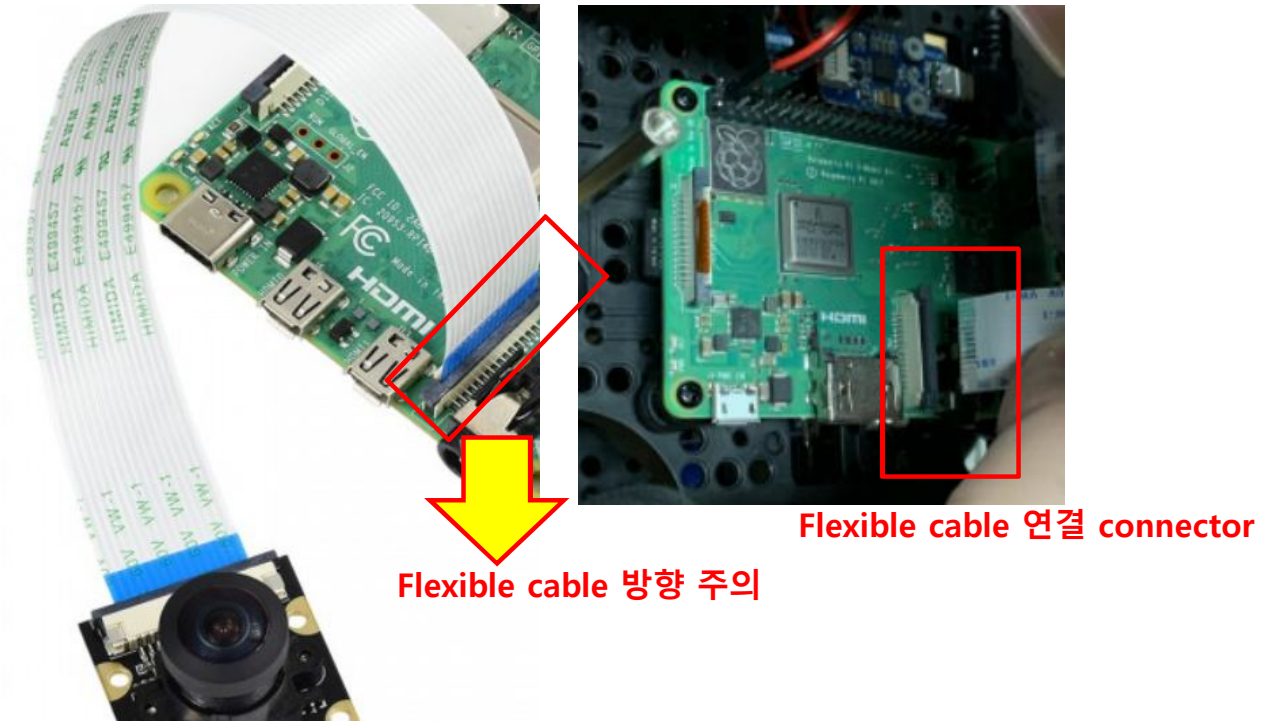
roslaunch turtlebot3_bringup turtlebot3_robot.launch

8. Autonomous Driving

8-1. RPi Camera(G) - Fisheye Lens 연결하기

(1) RPi Camera(G) - Fisheye Lens specification

- supports all revisions of the Pi
- Fisheye Lens, Angle of View : 160 degree (while other normal cameras are typically 72 degree)
- 5 megapixel OV5647 image sensor
- CCD size : 1/4inch
- Aperture (F) : 2.35
- Focal Length : 3.15mm
- Sensor best resolution : 1080p
- Provides 3.3V power output



8. Autonomous Driving

8-1. RPi Camera(G) - Fisheye Lens 연결하기

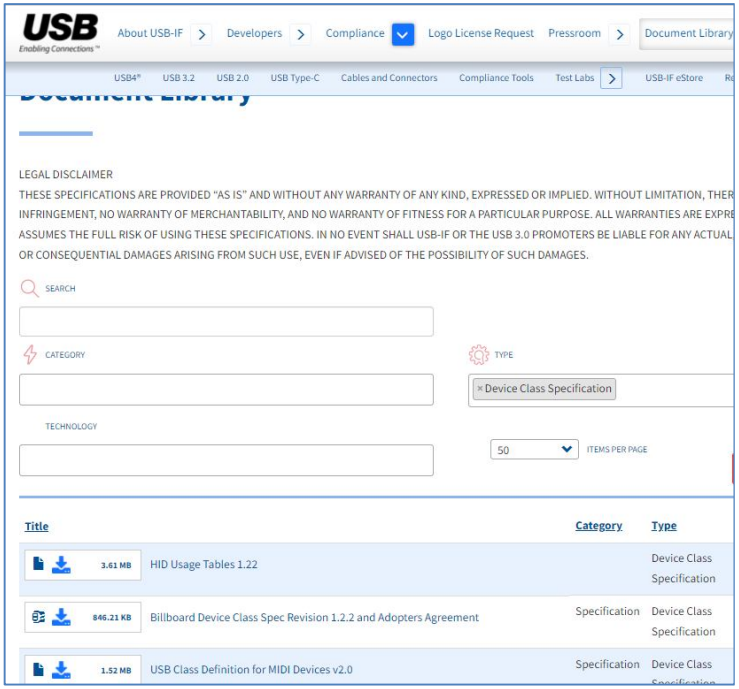
(2) RPi Camera(G) - Fisheye Lens를 Turtlebot3에 조립 및 Raspberry PI 3 B+ 연결

- Camera USB interface

  : USB video device class (UVC) 라고 정의 함

  ➔ USB video device class Ref. https://en.wikipedia.org/wiki/USB_video_device_class

  https://www.usb.org/documents?search=&type%5B0%5D=55&items_per_page=50



- USB camera package

  : libuvc-camera - UVC 표준 카메라 사용 위한 인터페이스 패키지

  : uvc-camera - 상세 카메라 설정 변경 기능

  - 카메라 두 대를 이용하여 스테레오 카메라 활용 가능 패키지

  : usb-cam - Bosch에서 사용 카메라 드라이버

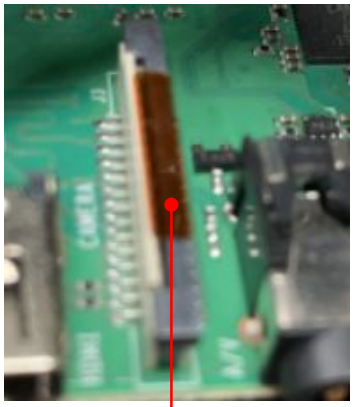
  : freenect-camera, openni-camera, openni2-camera

  - Kinect나 Xtion과 같은 심도 카메라

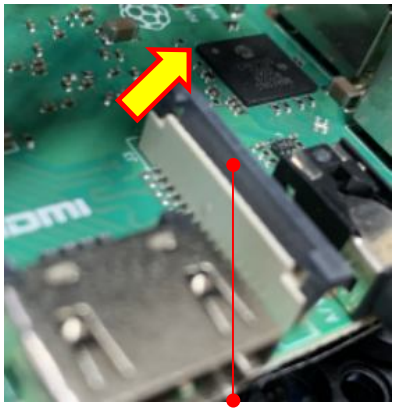
8. Autonomous Driving

8-1. RPi Camera(G) - Fisheye Lens 연결하기

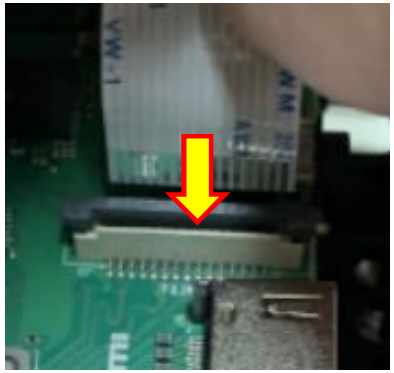
(2) RPi Camera(G) - Fisheye Lens를 Turtlebot3에 조립 및 Raspberry PI 4B 연결



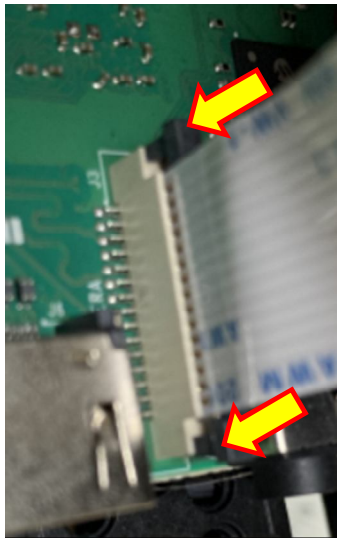
tape 제거



Connector 올리기



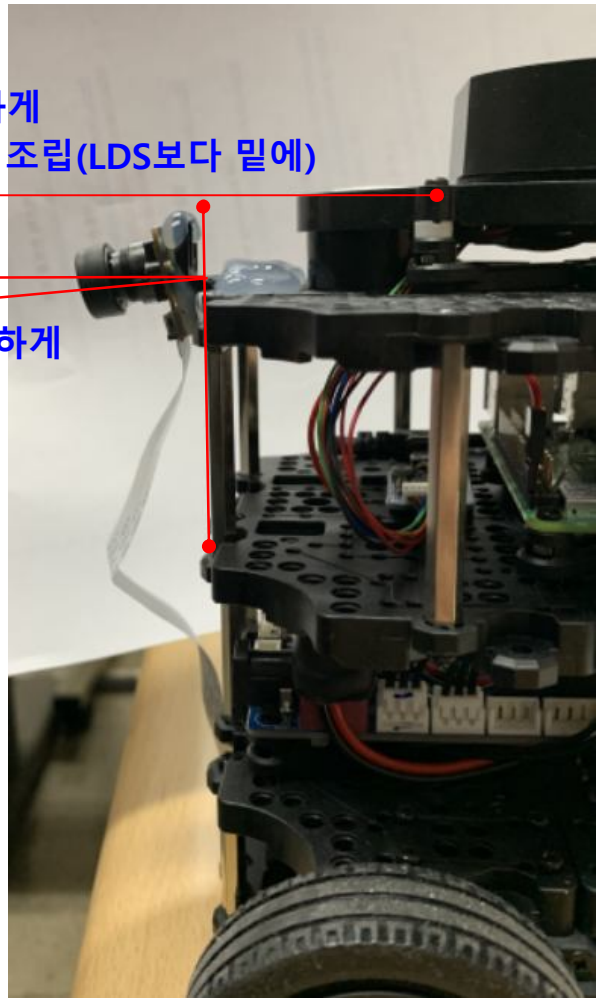
Flexible cable 연결

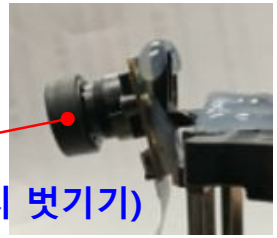


Connector 누르기

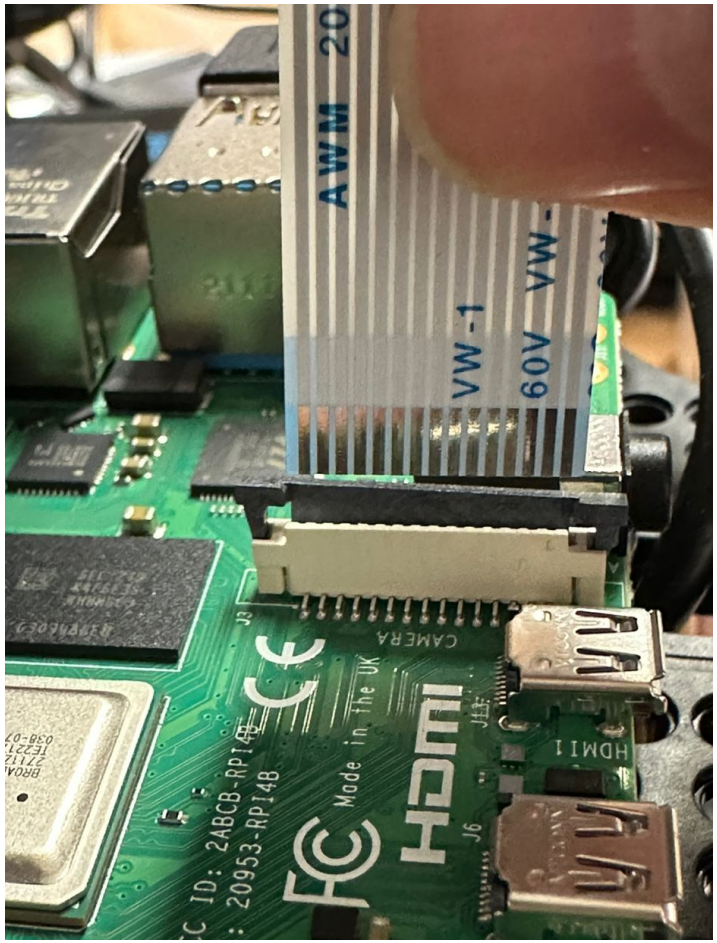
: 정면을 향하게
: 글루건으로 조립(LDS보다 밑에)

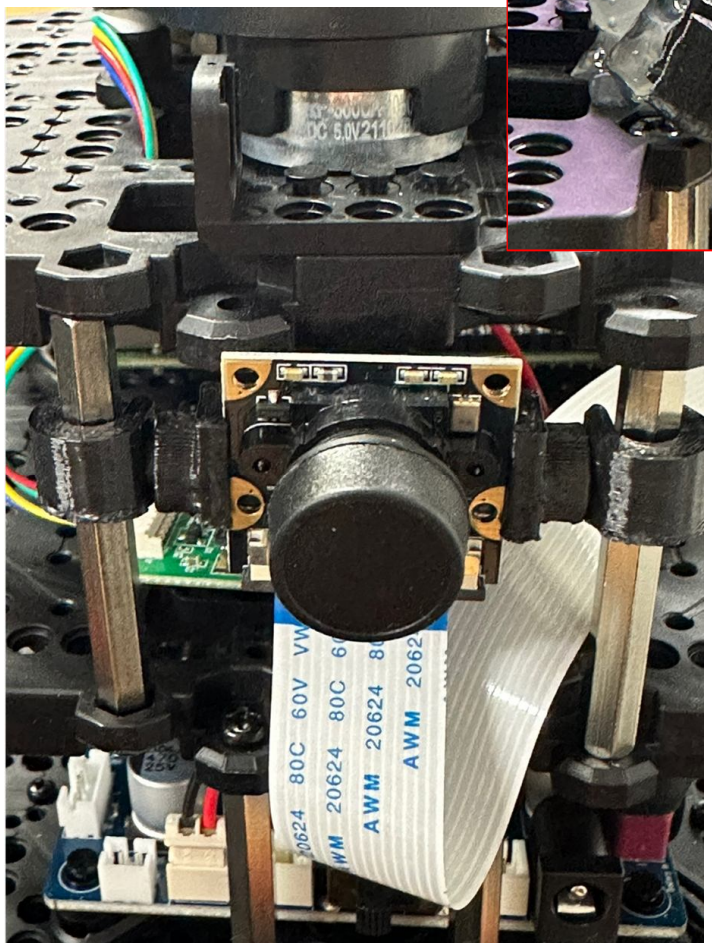
5 degree 밑을 향하게

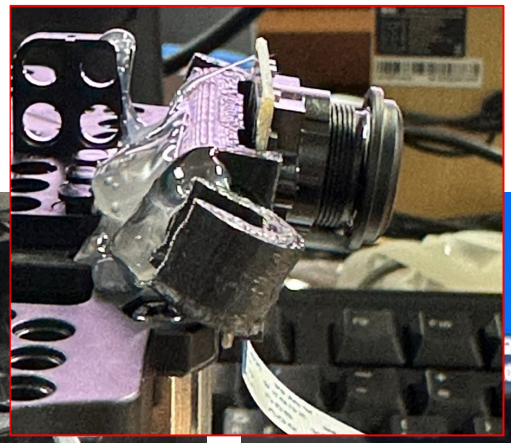


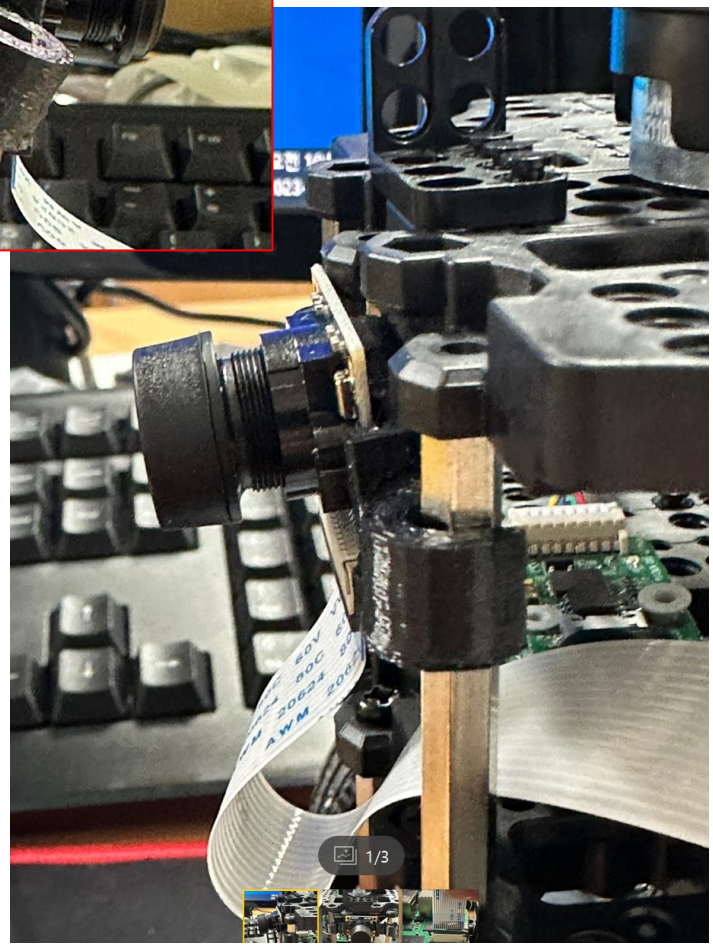


Lens cover (사용시 벗기기)


AWM 20
VW-1
60V VW-
CAMERA
Made in The UK
HDMI
HDMI1
ID: 2ABCB-RPI4
: 20953-RPI4B

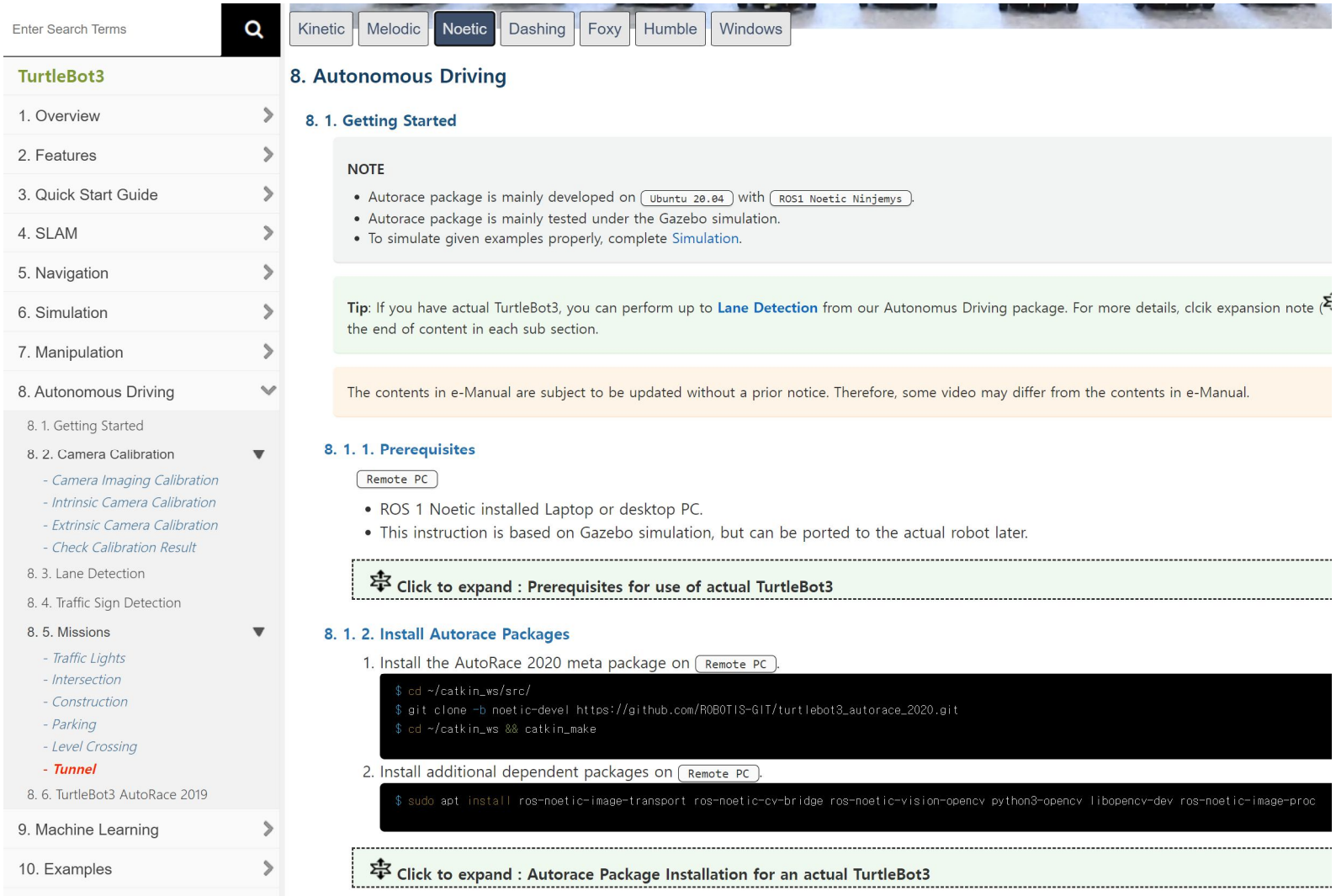

AWM 20624 80C 60V VW-1




1/3

: https://wiki.ros.org

: Robotis e-manual 8장 https://emanual.robotis.com/docs/en/platform/turtlebot3/autonomous_driving/#getting-started



8. Autonomous Driving

8.1.3 Install Autorace Packages

주의 : Turtlebot3 에 전원을 넣고 실행하기(Battery X ← 설치 시간 소요)

Remote PC에서 새터미널 열고

실행

8. 1. 3. Install Autorace Packages

in Gazebo simulation.

1. Install the AutoRace 2020 meta package on `Remote PC`.

```
$ cd ~/catkin_ws/src/
$ git clone -b noetic-devel https://github.com/ROBOTIS-GIT/turtlebot3_autorace_2020.git
$ cd ~/catkin_ws && catkin_make
```

2. Install additional dependent packages on `Remote PC`.

```
$ sudo apt install ros-noetic-image-transport ros-noetic-cv-bridge ros-noetic-vision-opencv python3-opencv libopencv-dev ros-noetic-image-proc
```

Click to expand : Autorace Package Installation for an actual TurtleBot3

The following instructions describes how to install packages and to calibrate camera.

1. Install AutoRace package on both `Remote PC` and `SBC`

```
$ cd ~/catkin_ws/src/
$ git clone -b feature-raspicam https://github.com/ROBOTIS-GIT/turtlebot3_autorace_2020.git
$ cd ~/catkin_ws && catkin_make
```

8.1.3 Install Autorace Packages

fallocate failed
swap 파일을 생성하려고 하지만 같은 이름의 파일이 이미 존재하고 아직 사용 중임을 의미. 비활성화

실행 | 새 터미널 열고 $ ssh ubuntu@192.168.0.21 (IP is Raspberry, password turtlebot)

⇩ Ubuntu(SBC)에서 아래 실행 (1h 30m소요)

실행

Click to expand : Autorace Package Installation for an actual TurtleBot3

The following instructions describes how to install packages and to calibrate camera.

1. Install AutoRace package on both Remote PC and SBC.

```
$ cd ~/catkin_ws/src/
$ git clone -b feature-raspicam https://github.com/ROBOTIS-GIT/turtlebot3_autorace_2020.git
$ cd ~/catkin_ws && catkin_make
```

2. Install additional dependent packages on SBC.
   - Create a swap file to prevent lack of memory in building OpenCV.

```
$ sudo fallocate -l 4G /swapfile
$ sudo chmod 600 /swapfile
$ sudo mkswap /swapfile
$ sudo swapon /swapfile
```

만일 fallocate faile 발생하면,
sudo swapoff -a
실행(비활성화)하고 다시 진행

```
ubuntu@ubuntu1:~/catkin_ws$ cd ..
ubuntu@ubuntu1:~$ sudo fallocate -l 4G /swapfile
fallocate: fallocate failed: Text file busy
ubuntu@ubuntu1:~$ sudo swapoff -a
ubuntu@ubuntu1:~$ sudo fallocate -l 4G /swapfile
ubuntu@ubuntu1:~$ sudo chmod 600 /swapfile
ubuntu@ubuntu1:~$ sudo mkswap /swapfile
mkswap: /swapfile: warning: wiping old swap signature.
Setting up swapspace version 1, size = 4 GiB (4294963200 bytes)
no label, UUID=3546a200-c603-45f1-9197-3b5afbcefb35
ubuntu@ubuntu1:~$ sudo swapon /swapfile
ubuntu@ubuntu1:~$
```

fallocate()
: Linux 전용 파일 공간 할당에 관련된 작업
블록 할당, 해제, 초기화 등 여러가지 작업

https://blog.stackframe.dev/31#:~:text=fallocate()%EB%8A%94%20EB%A6%AC%EB%88%85%EC%8A%A4%20EC%A0%84%EC%9A%A9,%EB%B3%B5%EC%9E%A1%ED%95%B4%EC%A7%80%EA%B8%B0%20EB%95%8C%EB%AC%B8%EC%97%90%20EC%A0%95%EB%A6%AC%ED%95%9C%EB%8B%A4.

⇩

8.1.3 Install Autorace Packages

실행

◦ Install required dependencies.

```
$ sudo apt-get update
$ sudo apt-get install build-essential cmake gcc g++ git unzip pkg-config
$ sudo apt-get install libjpeg-dev libpng-dev libtiff-dev libavcodec-dev libavformat-dev libswscale-dev libgtk2.0-dev libcanberra-gtk* libxvidcore-dev libx264-dev python3-dev pythc
```

주의
끝까지 drag한 후에 실행

◦ Build with opencv & opencv_contrib

```
$ cd ~
$ wget -O opencv.zip https://github.com/opencv/opencv/archive/4.5.0.zip
$ wget -O opencv_contrib.zip https://github.com/opencv/opencv_contrib/archive/4.5.0.zip

$ unzip opencv.zip
$ unzip opencv_contrib.zip

$ mv opencv-4.5.0 opencv
$ mv opencv_contrib-4.5.0 opencv_contrib
```

실행

주의
끝까지 drag한 후에 실행

○ Create cmake file.

```
$ cd opencv
$ mkdir build
$ cd build
$ cmake -D CMAKE_BUILD_TYPE=RELEASE ₩
        -D CMAKE_INSTALL_PREFIX=/usr/local ₩
        -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib/modules ₩
        -D ENABLE_NEON=ON ₩
        -D BUILD_TIFF=ON ₩
        -D WITH_FFMPEG=ON ₩
        -D WITH_GSTREAMER=ON ₩
        -D WITH_TBB=ON ₩
        -D BUILD_TBB=ON ₩
        -D BUILD_TESTS=OFF ₩
        -D WITH_EIGEN=OFF ₩
        -D WITH_V4L=ON ₩
        -D WITH_LIBV4L=ON ₩
        -D WITH_VTK=OFF ₩
        -D OPENCV_ENABLE_NONFREE=ON ₩
        -D INSTALL_C_EXAMPLES=OFF ₩
        -D INSTALL_PYTHON_EXAMPLES=OFF ₩
        -D BUILD_NEW_PYTHON_SUPPORT=ON ₩
        -D BUILD_opencv_python3=TRUE ₩
        -D OPENCV_GENERATE_PKGCONFIG=ON ₩
        -D BUILD_EXAMPLES=OFF ..
```

```
ubuntu@ubuntu1:~/opencv$ mkdir build
mkdir: cannot create directory ‘build’: File exists
ubuntu@ubuntu1:~/opencv$ ls
3rdparty  CMakeLists.txt   doc       opencv-4.5.0  SECURITY.md
apps      CONTRIBUTING.md  include   platforms
build     COPYRIGHT        LICENSE   README.md
cmake     data             modules   samples
ubuntu@ubuntu1:~/opencv$ cd build
ubuntu@ubuntu1:~/opencv/build$
```

실행 결과

```
--   Install to:                    /usr/local
-- ------------------------------------------------------------------
--
-- Configuring done
-- Generating done
-- Build files have been written to: /home/ubuntu/opencv/build
ubuntu@ubuntu1:~/opencv/build$
```

실행

◦ It will take an hour or two to build.

```
$ cd ~/opencv/build
$ make -j4
$ sudo make install
$ sudo ldconfig
$ make clean
$ sudo apt-get update
```

make 완료 화면

```
[100%] Building CXX object modules/python3/CMakeFiles/opencv_python3.dir/__/src2/cv2.cpp.o
[100%] Linking CXX executable ../../bin/opencv_perf_ximgproc
[100%] Built target opencv_perf_ximgproc
[100%] Linking CXX executable ../../bin/opencv_perf_superres
[100%] Built target opencv_perf_superres
[100%] Linking CXX shared module ../../lib/python3/cv2.cpython-38-aarch64-linux-gnu.so
[100%] Built target opencv_python3
ubuntu@ubuntu1:~/opencv/build$
```

완료 화면

```
ubuntu@ubuntu1:~/opencv/build$ sudo ldconfig
ubuntu@ubuntu1:~/opencv/build$ make clean
ubuntu@ubuntu1:~/opencv/build$ sudo apt-get update
Hit:1 http://packages.ros.org/ros/ubuntu focal InRelease
Hit:2 http://ports.ubuntu.com/ubuntu-ports focal InRelease
Get:3 http://ports.ubuntu.com/ubuntu-ports focal-updates InRelease [114
Hit:4 http://ports.ubuntu.com/ubuntu-ports focal-backports InRelease
Hit:5 http://ports.ubuntu.com/ubuntu-ports focal-security InRelease
Hit:6 http://ports.ubuntu.com/ubuntu-ports focal-proposed InRelease
Fetched 114 kB in 21s (5,529 B/s)
Reading package lists... Done
ubuntu@ubuntu1:~/opencv/build$
```

참고 : 우리는 실행 안함. Turtlebot3에 별도 모니터 연결하기 위해, 아래 실행하고 window에서 내용 수정하는 것임

- Turn off Raspberry Pi, take out the microSD card and edit the config.txt in system-boot section. add start_x=1 before the enable_uart=1 line.

여기부터 실행

```
$ sudo apt install ffmpeg
$ ffmpeg -f video4linux2 -s 640x480 -i /dev/video0 -ss 0:0:2 -frames 1 capture_test.jpg
```

- Install additional dependent packages

```
$ sudo apt install ros-noetic-cv-camera
```

완료 화면

```
ubuntu@ubuntu1:~/opencv/build$ sudo apt install ros-noetic-cv-camera
Reading package lists... Done
Building dependency tree
Reading state information... Done
ros-noetic-cv-camera is already the newest version (0.6.0-1focal.20230620.200022).
The following packages were automatically installed and are no longer required:
  libfwupdplugin1 libxmlb1 linux-headers-5.4.0-1043-raspi linux-image-5.4.0-1043-raspi
  linux-modules-5.4.0-1043-raspi linux-raspi-headers-5.4.0-1043
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 24 not upgraded.
ubuntu@ubuntu1:~/opencv/build$
```

8.1.3 Install Autorace Packages

SBC 실행이후 `ubuntu@ubuntu1:~`

Remote PC에서 새 터미널 열고 실행 `yongseok@yongseok:~$`

3. Install additional dependent packages on `Remote PC`.

```
$ sudo apt install ros-noetic-image-transport ros-noetic-image-transport-plugins ros-noetic-cv-bridge ros-noetic-vision-opencv python3-opencv libopencv-dev ros-noetic-image-proc ros-noe
```

주의
끝까지 drag한 후에 실행

완료 화면

```
yongseok@yongseok:~$ sudo apt install ros-noetic-image-transport ros-noetic-image-transport-plugins ros-noetic-cv-bridge ros-noeti
c-vision-opencv python3-opencv libopencv-dev ros-noetic-image-proc ros-noetic-cv-camera ros-noetic-camera-calibration
[sudo] password for yongseok:
Reading package lists... Done
Building dependency tree
Reading state information... Done
libopencv-dev is already the newest version (4.2.0+dfsg-5).
python3-opencv is already the newest version (4.2.0+dfsg-5).
ros-noetic-camera-calibration is already the newest version (1.17.0-1focal.20230620.192537).
ros-noetic-camera-calibration set to manually installed.
ros-noetic-cv-bridge is already the newest version (1.16.2-1focal.20230620.191830).
ros-noetic-image-proc is already the newest version (1.17.0-1focal.20230620.192452).
ros-noetic-image-transport is already the newest version (1.12.0-1focal.20230620.191833).
ros-noetic-image-transport-plugins is already the newest version (1.14.0-1focal.20230620.194116).
ros-noetic-image-transport-plugins set to manually installed.
ros-noetic-vision-opencv is already the newest version (1.16.2-1focal.20230620.193649).
The following NEW packages will be installed:
  ros-noetic-cv-camera
0 upgraded, 1 newly installed, 0 to remove and 5 not upgraded.
Need to get 61.7 kB of archives.
After this operation, 300 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://packages.ros.org/ros/ubuntu focal/main amd64 ros-noetic-cv-camera amd64 0.6.0-1focal.20230620.193132 [61.7 kB]
Fetched 61.7 kB in 2s (38.8 kB/s)
Selecting previously unselected package ros-noetic-cv-camera.
(Reading database ... 299968 files and directories currently installed.)
Preparing to unpack .../ros-noetic-cv-camera_0.6.0-1focal.20230620.193132_amd64.deb ...
Unpacking ros-noetic-cv-camera (0.6.0-1focal.20230620.193132) ...
Setting up ros-noetic-cv-camera (0.6.0-1focal.20230620.193132) ...
yongseok@yongseok:~$
```

8. Autonomous Driving

8-2. Camera calibration

(1) Install Autorace(2020) Packages

ⓐ Turtlebot3 Autorace 2020

- Mission에 따른 camera calibration

: Mission1. Traffic Light ➔ 신호등 green 확인 후 출발

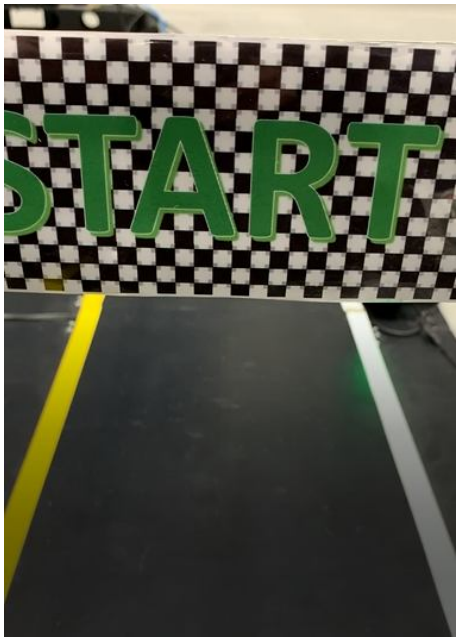
: Mission2. Intersection ➔ 좌우 교차로 표지판 인식하여 교차로 주행

: Mission3. Construction ➔ 공사 표지판 인식하여 3개의 장애물 피함

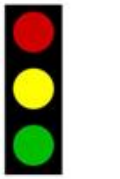
: Mission4. Parking ➔ 주차 표지판 인식 후 주차 빈 자리에 주차

: Mission5. Level Crossing ➔ 정지 표지판 인식 후 차단 바를 본 후 정지하고 출발

: Mission6. Tunnel ➔ 터널 표지판 인식 후 SLAM을 실행하여 터널 통과

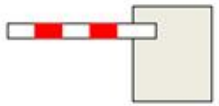


동영상 - Tracks image



신호등 미션

공사구간 미션



차단바 미션

터널 미션

8-2. Camera calibration

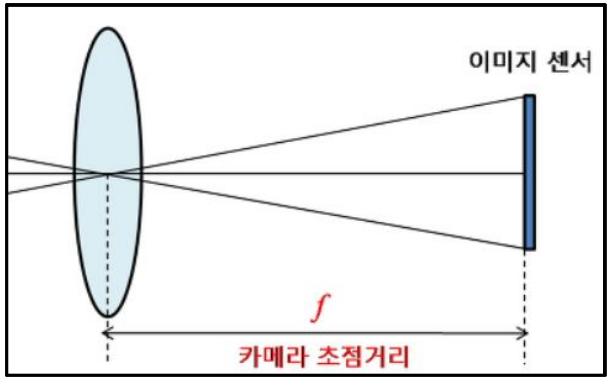
(2) Camera calibration

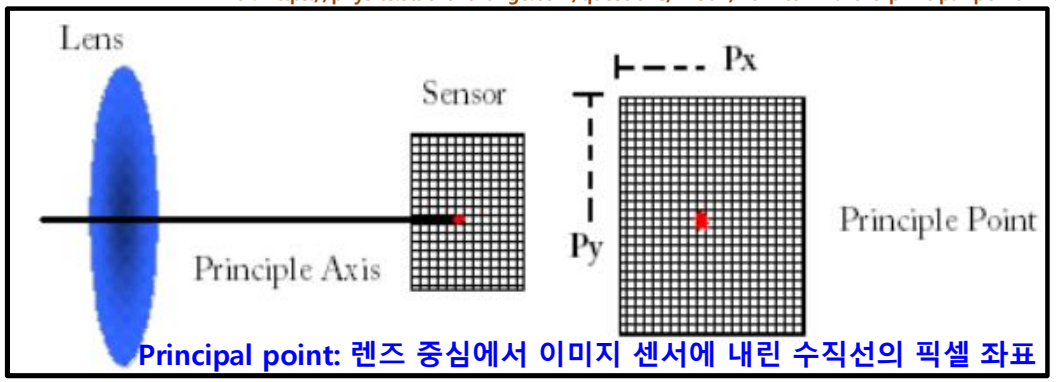
: 영상을 촬영할 때, 영상 품질은 카메라의 위치 및 방향 등 외부적(extrinsic) 요인에 의해 결정되나,

: 그러나 실제 이미지는 사용된 렌즈, 렌즈와 이미지 센서와의 거리, 렌즈와 이미지 센서가 이루는 각 등 카메라 내부(intrinsic) 기구 조립 요소에 의해 영향 받음.

➔ intrinsic calibration - 초점거리(focal length), 주점(principal point) 등 카메라 렌즈의 광학적 요소를 보정(calibration)을 의미

Ref. https://physics.stackexchange.com/questions/11594/how-to-find-the-principal-point-in-an-image





Principal point: 렌즈 중심에서 이미지 센서에 내린 수직선의 픽셀 좌표

➔ extrinsic calibration - 카메라의 설치 높이, 방향(tilt) 등 카메라와 외부 공간과의 기하학적 관계에 의한 카메라 보정

8-2. Camera calibration

(2) Camera calibration

: camera calibration parameter

➔ camera image calibration - imaging 보정

➔ intrinsic calibration - 초점거리(focal length), 주점(principal point) 등 카메라 렌즈의 광학적 요소를 보정(calibration)을 의미

➔ extrinsic calibration - 카메라의 설치 높이, 방향(tilt-기울기) 등 카메라와 외부 공간과의 기하학적 관계에 의한 카메라 보정

: intrinsic calibration parameter, extrinsic calibration parameter 를 camera matrix 라고 부름

: parameter 경로

➔ intrinsic calibration

catkin_ws / src / turtlebot3_autorace_2020 / turtlebot3_autorace_camera / calibration / intrinsic_calibration

➔ extrinsic calibration

catkin_ws / src / turtlebot3_autorace_2020 / turtlebot3_autorace_camera / calibration / extrinsic_calibration

8-2. Camera calibration

(3) Camera imaging calibration

: autonomous driving를 위해 camera calibration이 매우 중요

: Ref. 8.2.1장 https://emanual.robotis.com/docs/en/platform/turtlebot3/autonomous_driving/#autonomous-driving

ⓐ Remote PC에서 새 터미널 열기

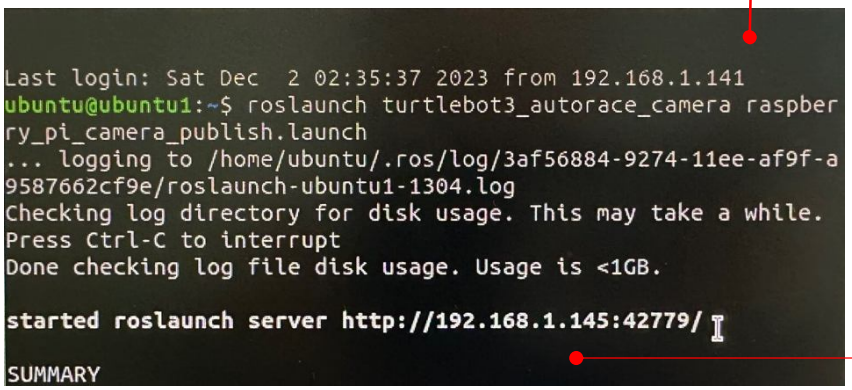
$ roscore

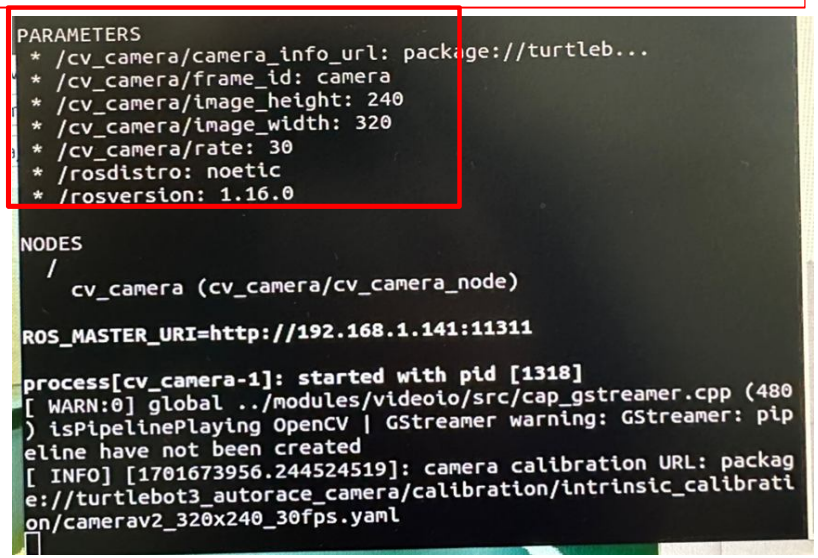
ⓑ Remote PC에서 새 터미널 열기

$ ssh ubuntu@192.168.0.25 ← Remote PC에서 TurtleBot3 Burger(Raspberry pi)을 원격접속

pw : turtlebot 입력

~$ roslaunch turtlebot3_autorace_camera raspberry_pi_camera_publish.launch

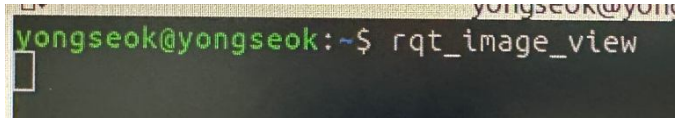




ⓒ Remote PC에서 새 터미널 열기

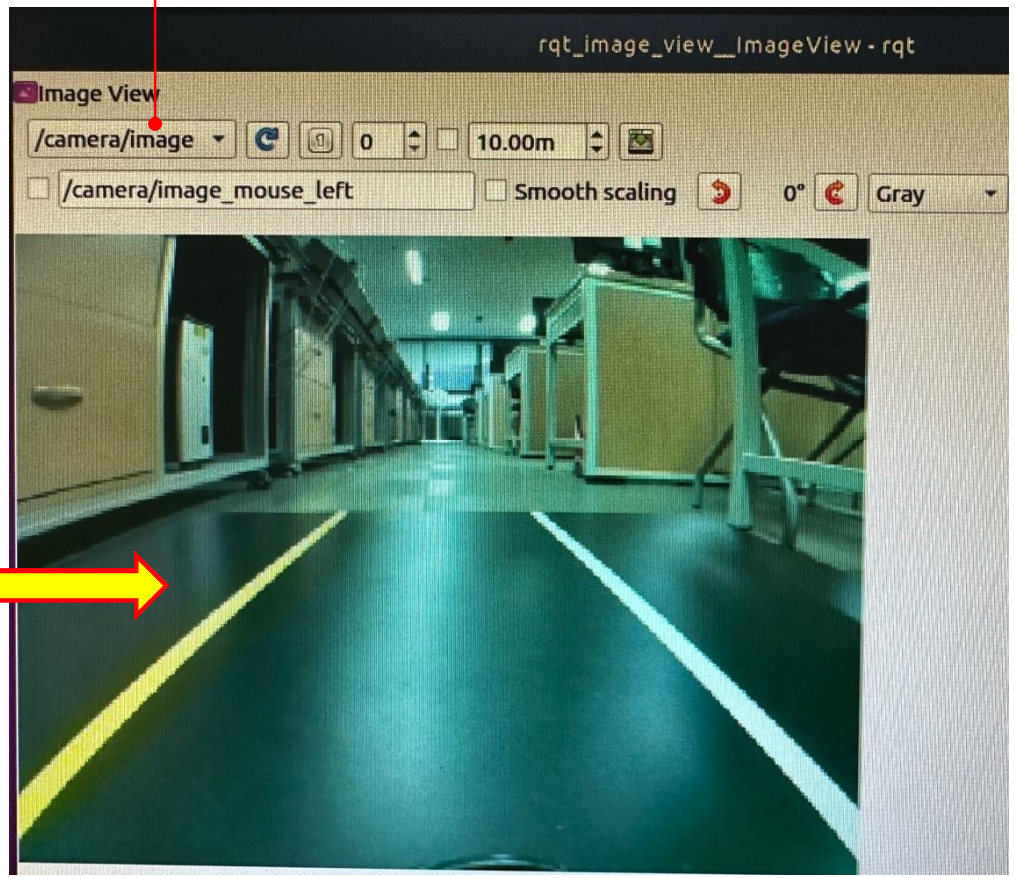
TurtleBot3 Burger를 경기장에 내려놓음

$ rqt_image_view



만일 실험을 마치려면, terminal은 CTL+C로 중단 후에 종료

/camera/image 선택 ➔ camera 화면 연결됨



실제 경기장

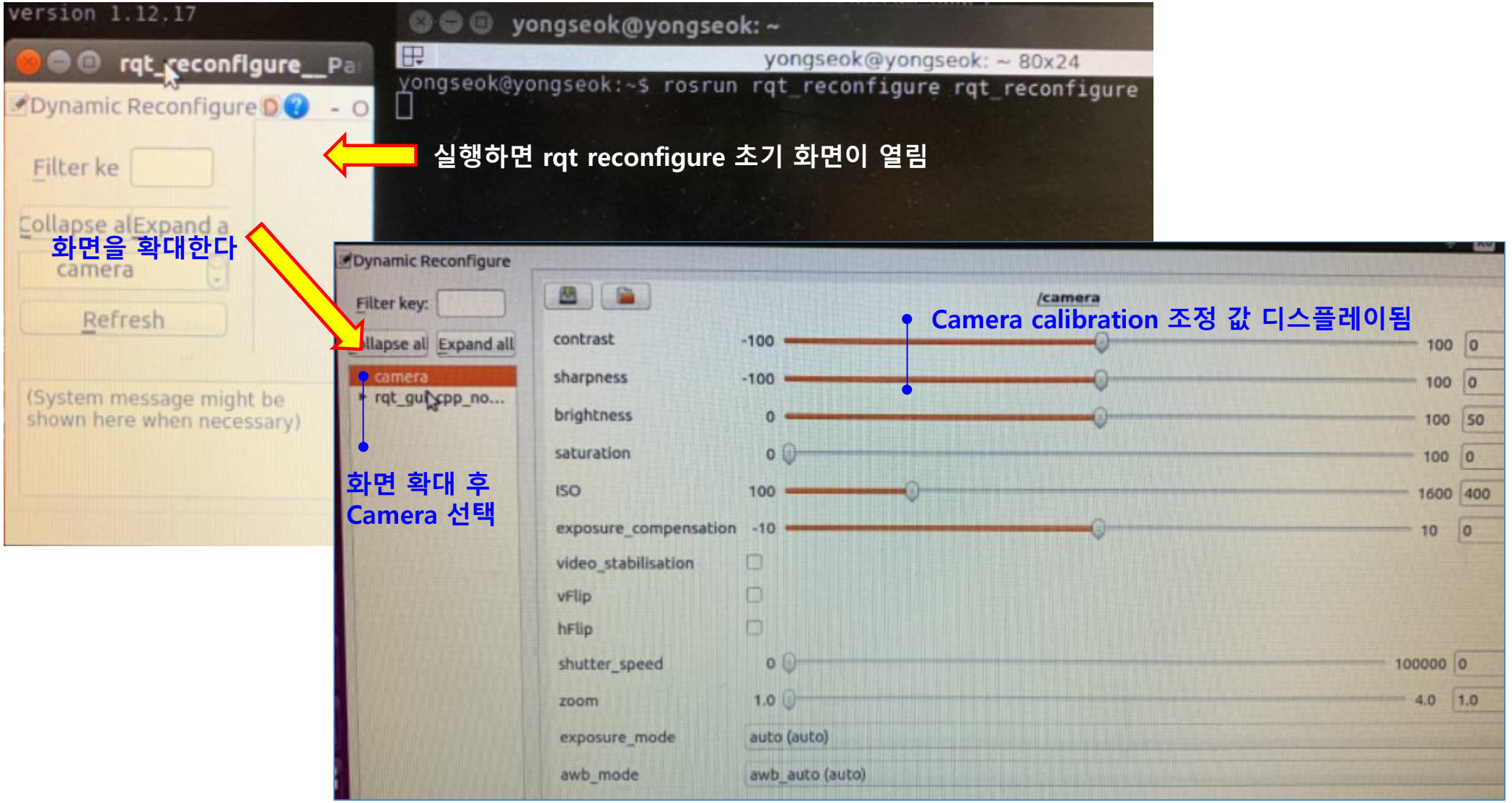
: Turtlebot의 왼쪽 Yellow, 오른쪽 white

: calibration에서는 관계없음

ⓓ Remote PC에서 새 터미널 열기

Skip for Kinectic

$ rosrun rqt_reconfigure rqt_reconfigure



ⓔ Remote PC에서

Skip for Kinectic

rqt_image_view 를 보면서, rqt_reconfigure의 camera image calibration 진행

← calibration에서 가장 중요한 것은 조도 값(외부 조명 조도)

: 야간, 주간 흰색과 노란색 구별 정도

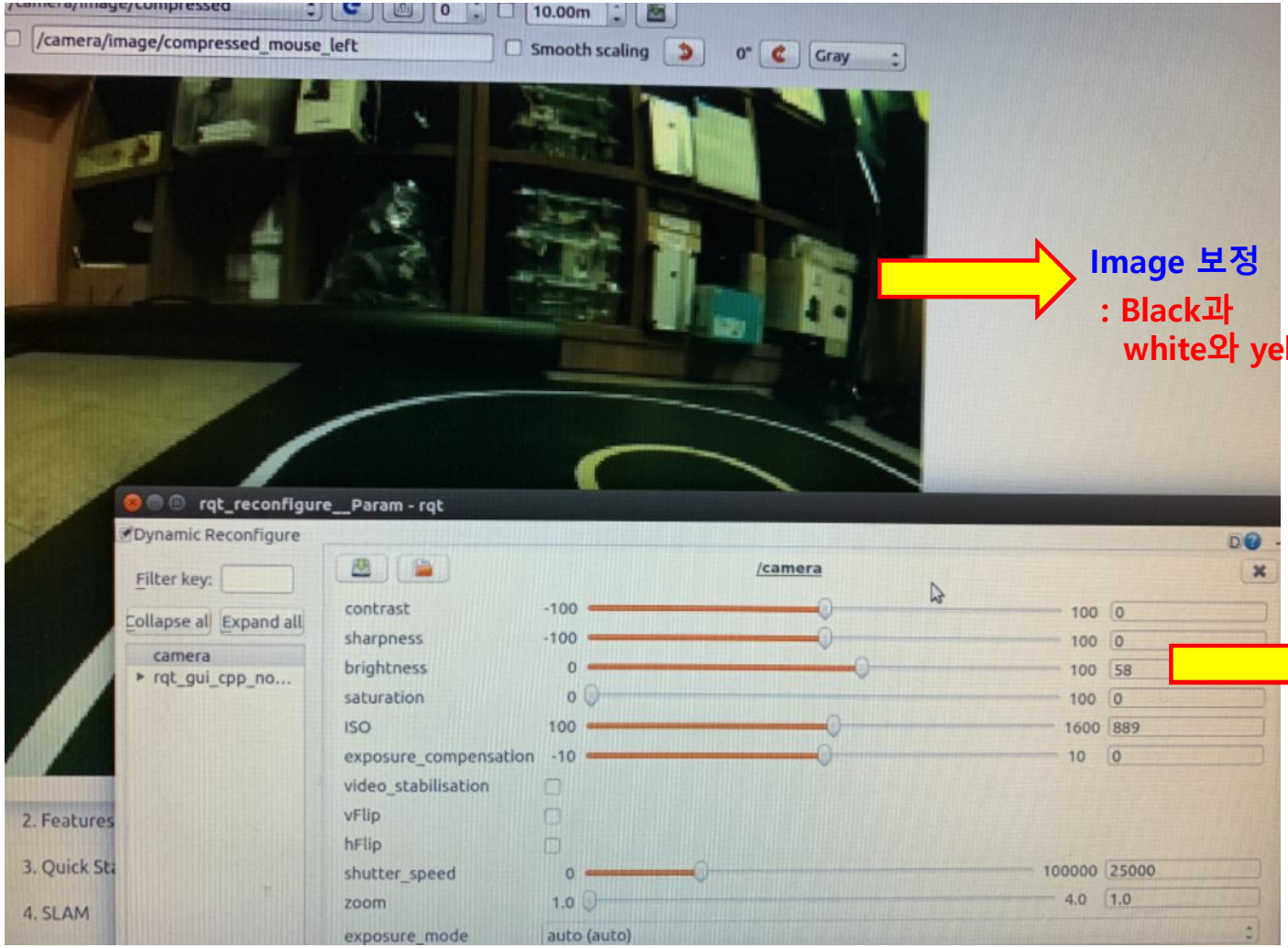


Image 보정

: Black과 white와 yellow가 서로 잘 구분되도록 영상 보정

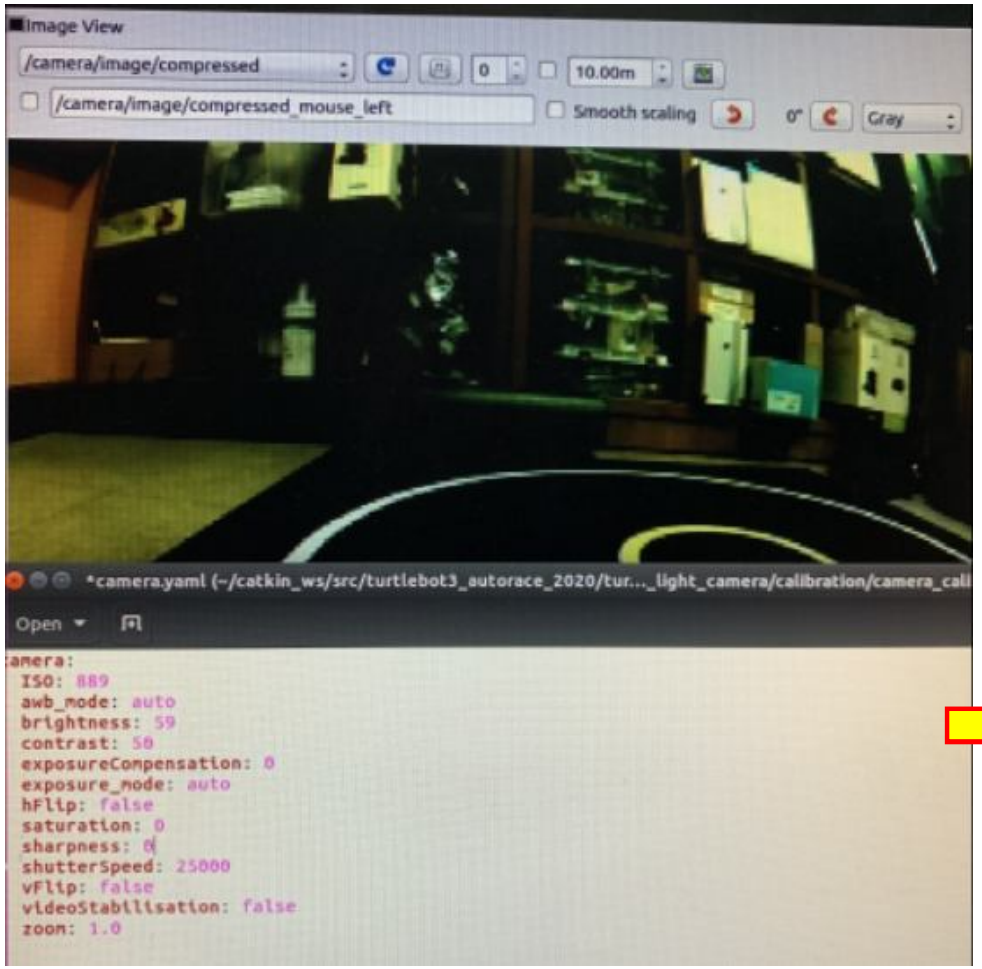
Camera calibration 조정

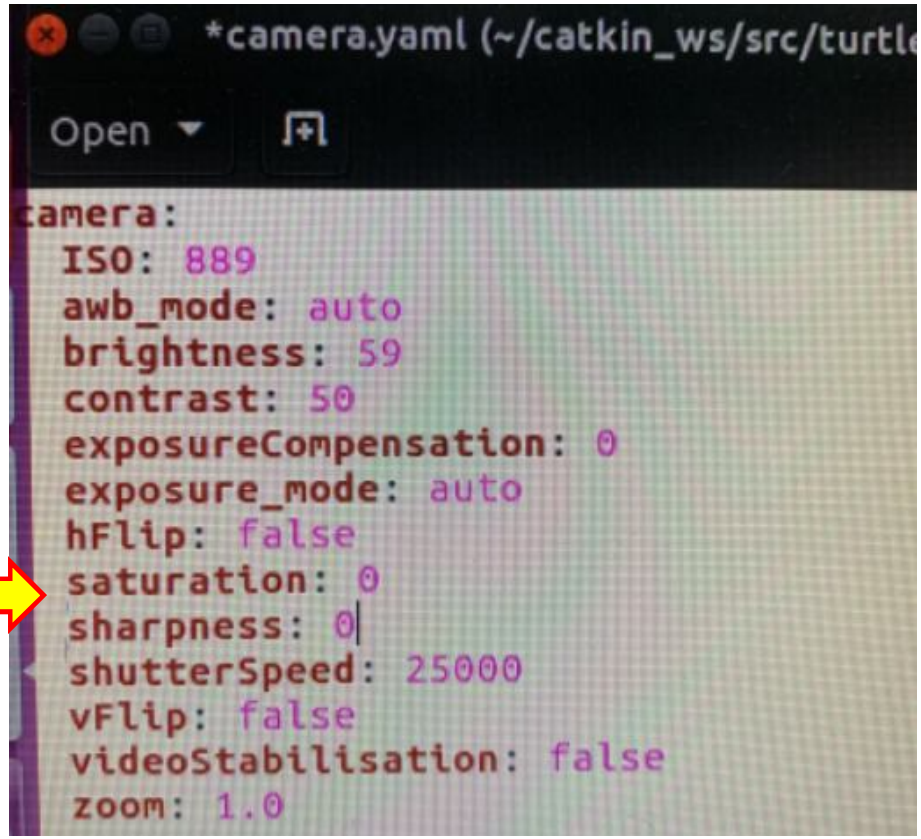
Skip for Kinectic

ⓕ Remote PC에서

rqt_reconfigure의 camera image calibration 값을 camera.yaml 에 저장

➔ 경로 catkin_ws / src / turtlebot3_autorace_2020 / turtlebot3_autorace_traffic_light /
turtlebot3_autorace_traffic_light_camera / calibration / camera_calibration





save

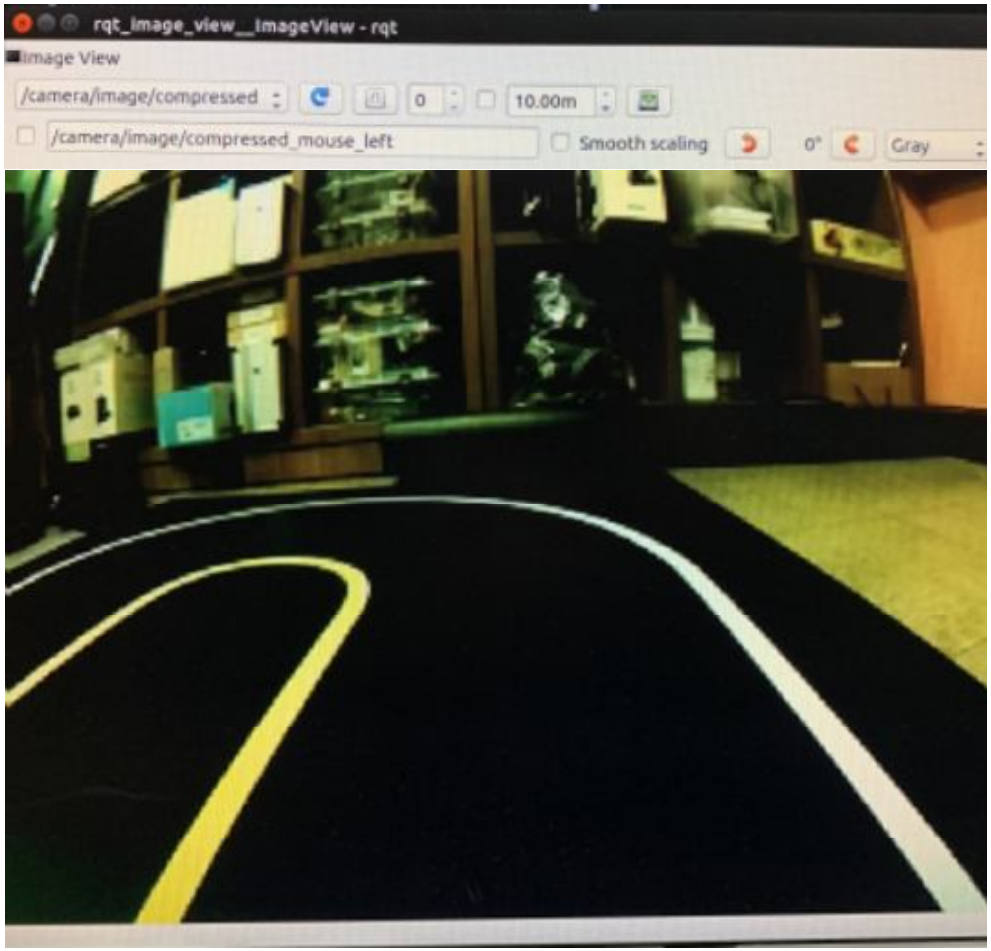
Skip for Kinectic

ⓖ Remote PC에서

: rqt_image_view 터미널 CTL +C 로 종료 후 닫기. rqt_image_view 화면 닫기. rqt_reconfigure 화면 닫기. camera.yaml 닫기

$ rqt_image_view

다시 열어서 camera.yaml에 의해 변경된 image 보정된 영상 확인하기



: yellow , white가 서로 잘 구분되도록 영상 보정 확인

8-2. RPi Camera(G) – Camera calibration

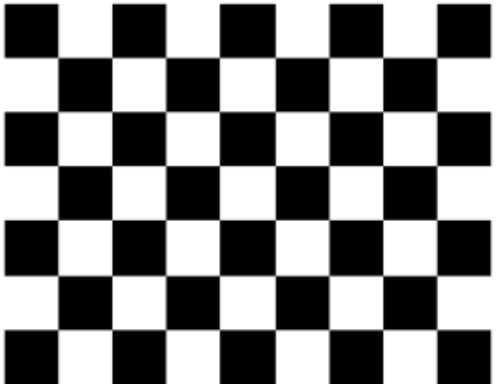
(4) Intrinsic Camera Calibration(터미널창 미리 6개 띄우기)

: Intrinsic Camera Calibration 를 위해 checkerboard 사용

: checkerboard

The checkerboard is stored at turtlebot3_autorace_camera/data/checkerboard_for_calibration.pdf

➔ checkerboard



: A4 출력하여, Intrinsic Camera Calibration 사용

➔ Calibration 진행 시, checkerboard는 흔들리지 않도록

(팔랑거리지 않도록 고정하는 것이 중요)

ⓐ Remote PC에서 새 터미널 열기

```
$ roscore
```

ⓑ Remote PC에서 새 터미널 열기

```
$ ssh ubuntu@192.168.0.25
```

← Remote PC에서 TurtleBot3 Burger(Raspberry pi)을 원격접속

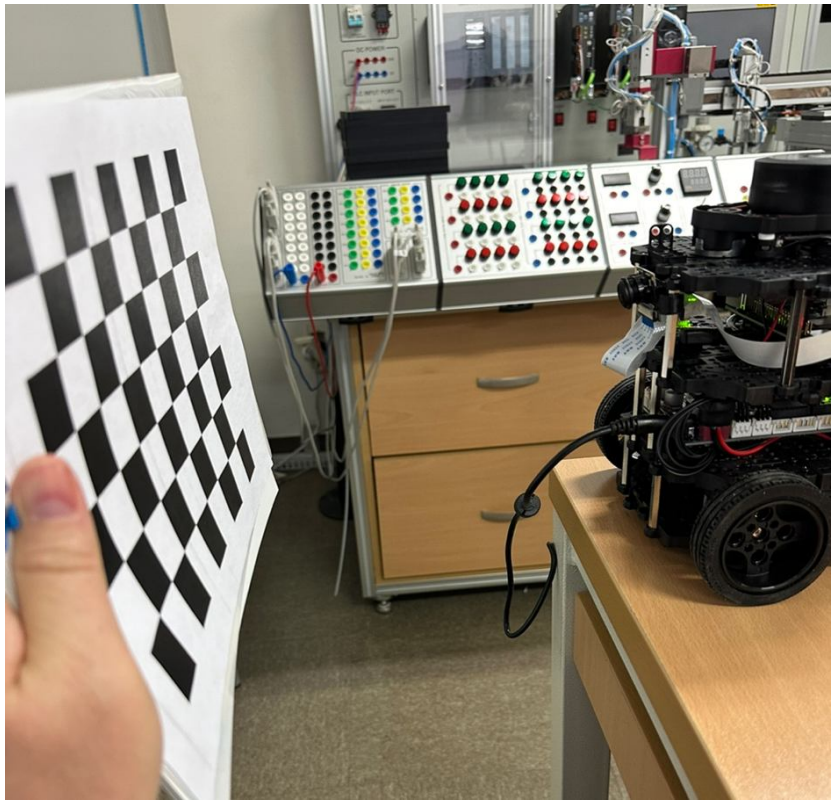
pw : turtlebot 입력

```
~$ roslaunch turtlebot3_autorace_camera raspberry_pi_camera_publish.launch
```

ⓒ **Remote PC**에서 새 터미널 열기

TurtleBot3 Burger(책상 끝에) 앞에 checkerboard(가로방향) 내려놓음

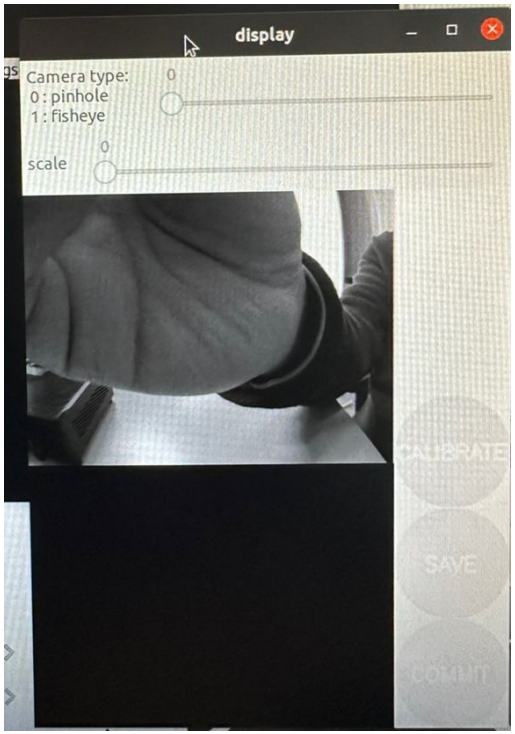
$ roslaunch turtlebot3_autorace_camera intrinsic_camera_calibration.launch mode:=calibration



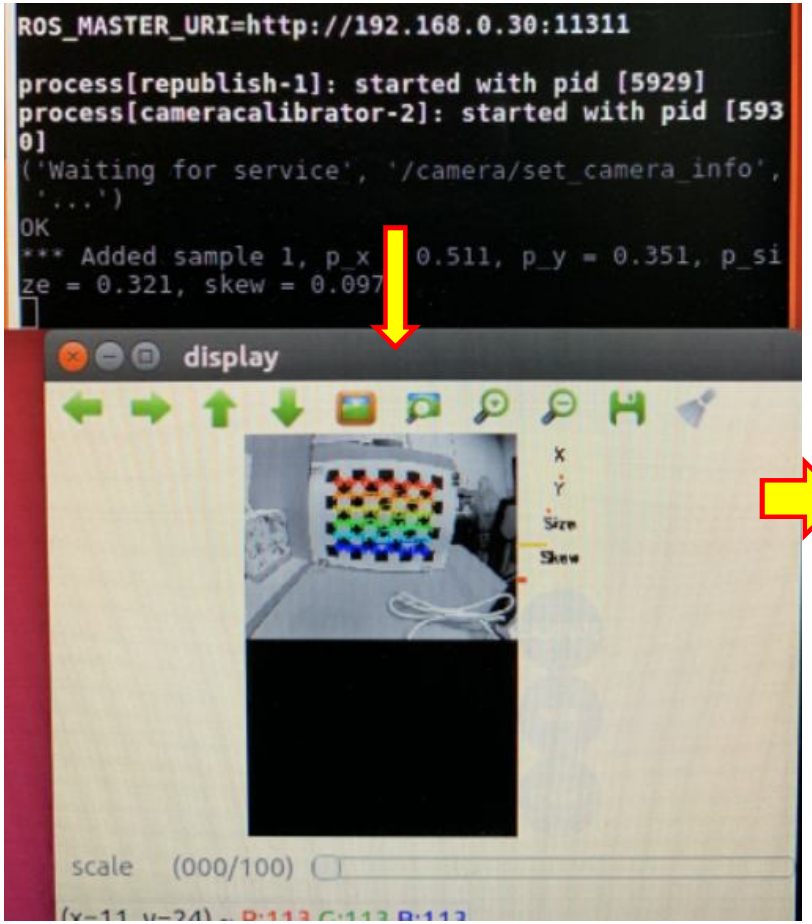
: A4 출력하여, Intrinsic Camera Calibration 사용

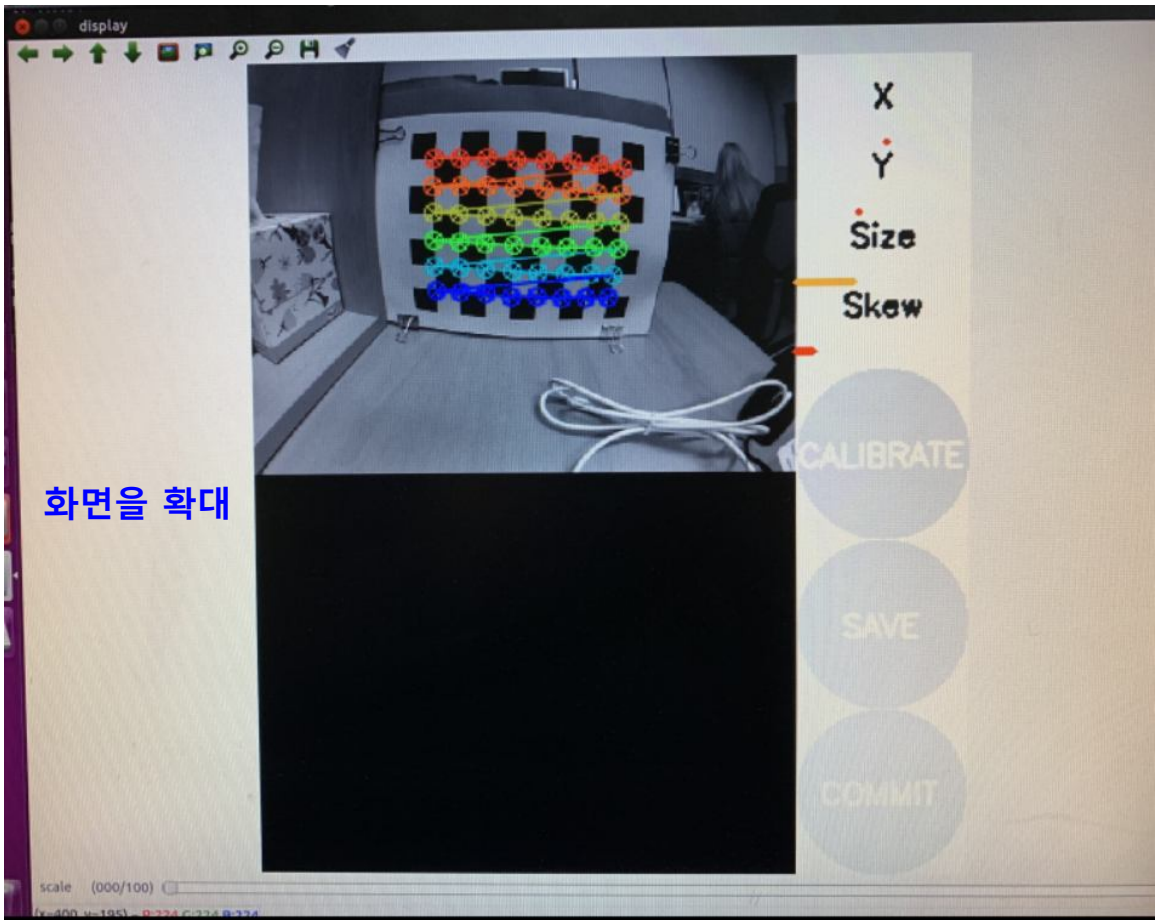
➔ Calibration 진행 시, checkerboard는 흔들리지 않도록

(팔랑거리지 않도록 고정하는 것이 중요)



실행 후 화면


ROS_MASTER_URI=http://192.168.0.30:11311
process[republish-1]: started with pid [5929]
process[cameracalibrator-2]: started with pid [5930]
('Waiting for service', '/camera/set_camera_info', '...')
OK
*** Added sample 1, p_x = 0.511, p_y = 0.351, p_size = 0.321, skew = 0.097
display
X
Y
Size
Skew
scale (000/100)


display
X
Y
Size
Skew
CALIBRATE
SAVE
COMMIT
scale (000/100)

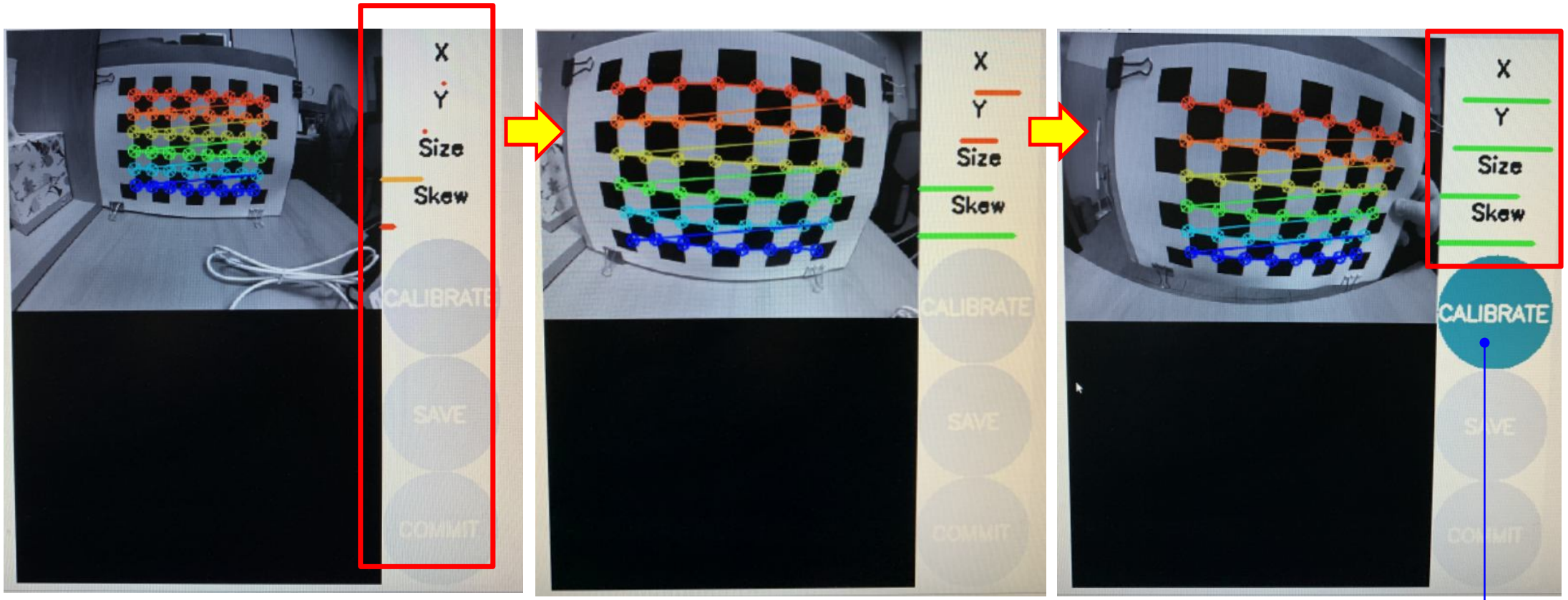
화면을 확대

ⓔ Intrinsic Camera Calibration 진행

: 카메라를 기준으로 왼쪽/오른쪽/위/아래로 checkerboard를 움직이기

: 카메라를 기준으로 앞/뒤로 움직이기, checkerboard를 상하, 좌우 비스듬히 기울이기

➔ 조건들이 채워질수록 X, Y, Size, Skew의 BAR 점점 채워지며 녹색으로 바뀜



초기화면

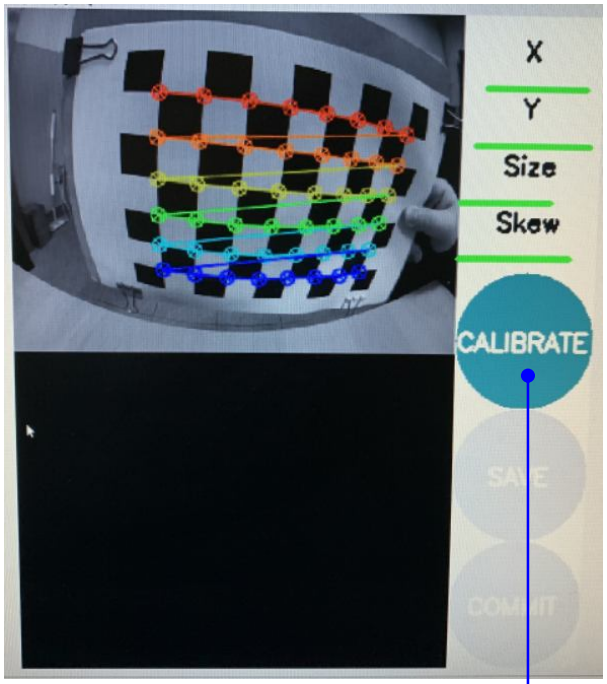
checkerboard를 움직임에 따라 초록색 BAR

초록색 BAR 최대 생성되면

Calibration 활성화 됨

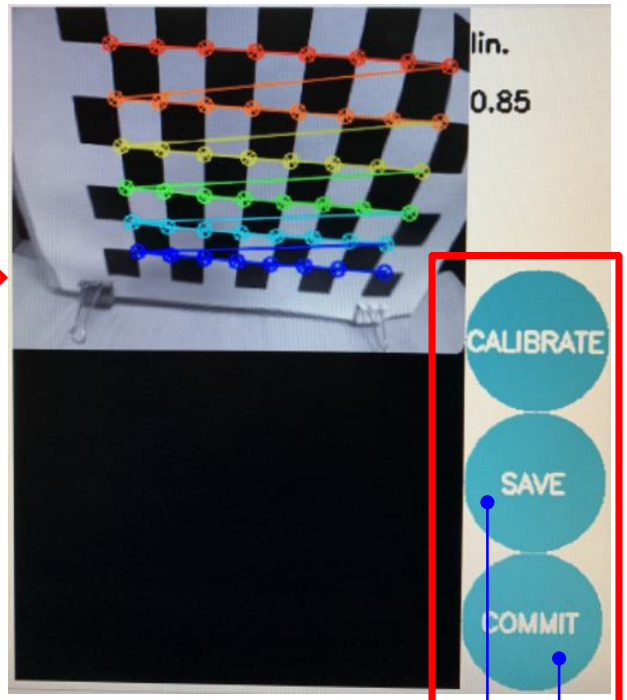
ⓕ click Calibration

: Calibration 누르고 1분 기다리기 ➔ SAVE 활성화 됨



click Calibration

1분 기다리기



활성화

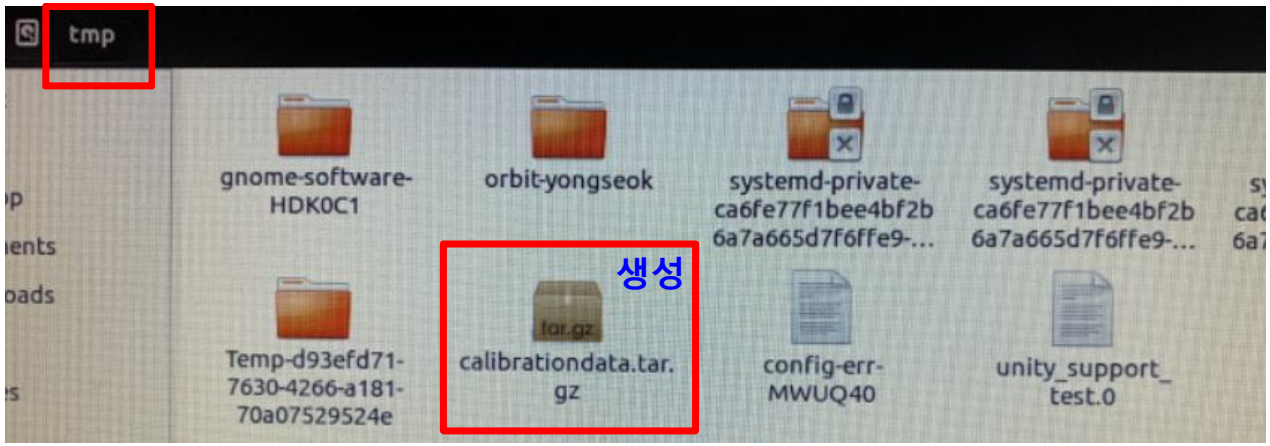
Button 활성화 됨

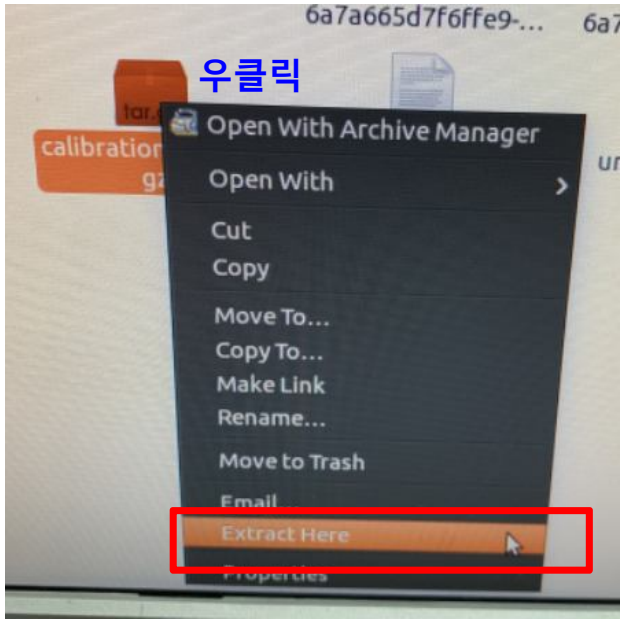
click SAVE

click (닫기)

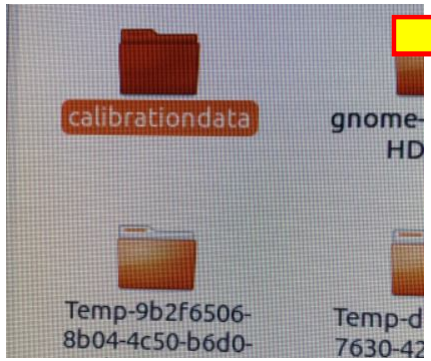
만일 실험을 마치려면, terminal은 CTL+C로 중단 후에 종료

ⓖ TMP 폴더에 calibrationdata.tar.gz 생성됨

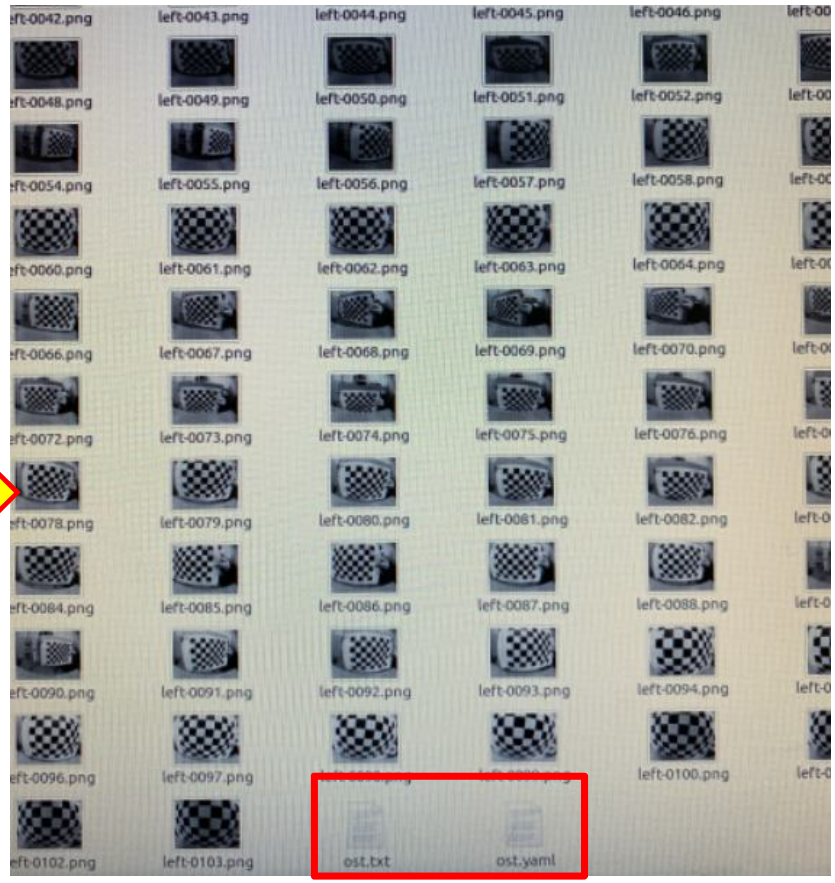




Extract Here – click 하여 압축 풀기



tmp 폴더에
Calibrationdata 생성



Calibrationdata 폴더 내용
그림 파일(png)와 ost.txt 그리고 ost.yaml

ⓗ ost.yaml 파일 열기(double click)

ost.yaml (/tmp/calibrationdata) - gedit

Open

```
image_width: 320
image_height: 240
camera_name: narrow_stereo
camera_matrix:
  rows: 3
  cols: 3
  data: [154.023918, 0.000000, 152.573255, 0.000000, 154.518473, 99.828186, 0.000000, 0.000000, 1.000000]
distortion_model: plumb_bob
distortion_coefficients:
  rows: 1
  cols: 5
  data: [-0.260440, 0.044973, 0.000049, 0.000158, 0.000000]
rectification_matrix:
  rows: 3
  cols: 3
  data: [1.000000, 0.000000, 0.000000, 0.000000, 1.000000, 0.000000, 0.000000, 0.000000, 1.000000]
projection_matrix:
  rows: 3
  cols: 4
  data: [104.956978, 0.000000, 153.489556, 0.000000, 0.000000, 116.299614, 86.441693, 0.000000, 0.000000, 0.000000, 1.000000,
0.000000]
```

카메라 이름을 narrow_stereo에서 camera 변경하기

카메라 내부 행렬

왜곡 계수

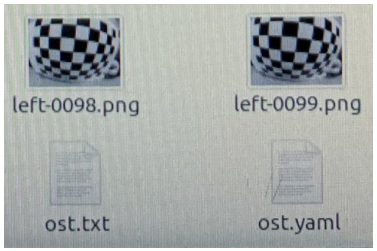
스테레오 카메라를 위한 보정 행렬

투영 행렬

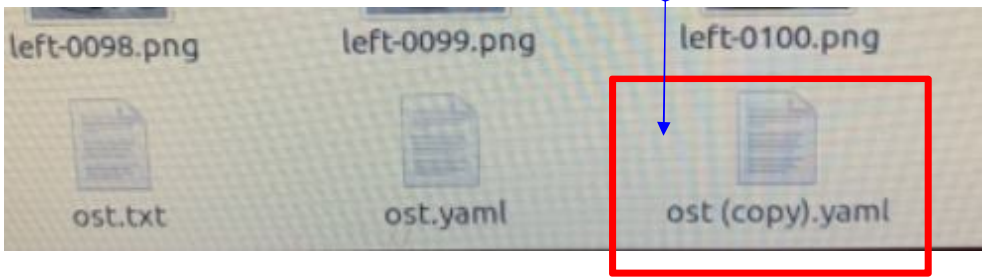
: 카메라 이름을 narrow_stereo에서 camera 변경하기 ➔ 저장하기

ⓘ tmp 폴더의

ost.yaml을 CTL + C(복사), CTL+V(붙여 놓기)



: 이름 변경하기 camerav2_320x240_30fps.yaml

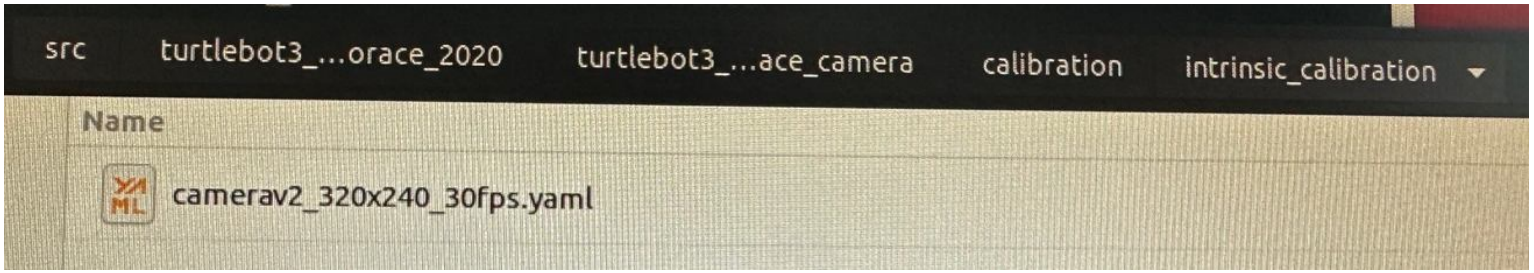


ⓙ tmp 폴더에서 생성한 camerav2_320x240_30fps.yaml 을 복사(CTL + C)하여

아래 경로에 붙여 넣기(CTL + V) – Replace

➔ 경로 catkin_ws / src / turtlebot3_autorace_2020 / turtlebot3_autorace_camera / calibration / intrinsic_calibration

➔ 위 경로에는 이미 camerav2_320x240_30fps.yaml 있으며, 이를 새로 업데이트 한다는 의미(덮어 쓰기)



➔ Intrinsic Calibration Data인 camerav2_320x240_30fps.yaml 를 덮어 씀(업데이트)

8-2. RPi Camera(G) – Camera calibration

(5) Extrinsic Camera Calibration

: Extrinsic Camera Calibration 를 위해 경기장(Lane) 사용

8. 2. 1. Extrinsic Camera Calibration

in Gazebo simulation.

1. Open a new terminal on `Remote PC` and launch Gazebo.

```
$ roslaunch turtlebot3_gazebo turtlebot3_autorace_2020.launch
```

2. Open a new terminal and launch the intrinsic camera calibration node.

```
$ roslaunch turtlebot3_autorace_camera intrinsic_camera_calibration.launch
```

3. Open a new terminal and launch the extrinsic camera calibration node.

```
$ roslaunch turtlebot3_autorace_camera extrinsic_camera_calibration.launch mode
```

4. Execute rqt on `Remote PC`.

```
$ rqt
```

5. Select **plugins** > **visualization** > **Image view**. Create two image view windo
6. Select `/camera/image extrinsic calib/compressed` topic on one window and

실행

turtlebot3_autorace_camera/calibration/extrinsic_calibration/`projection.yaml`

Click to expand : Extrinsic Camera Calibration with an actual TurtleBot3

1. Launch roscore on `Remote PC`.

```
$ roscore
```

2. Trigger the camera on `SBC`.

```
$ roslaunch turtlebot3_autorace_camera raspberry_pi_camera_publish.launch
```

3. Use the command on `Remote PC`.

```
$ roslaunch turtlebot3_autorace_camera intrinsic_camera_calibration.launch mode:=action
```

4. Run the extrinsic camera calibration launch file on `Remote PC`.

```
$ roslaunch turtlebot3_autorace_camera extrinsic_camera_calibration.launch mode:=calibration
```

5. Execute rqt on `Remote PC`.

```
$ rqt
```

6. Click **plugins** > **visualization** > **Image view**: Multiple windows will be present.

8-2. RPi Camera(G) – Camera calibration

(5) Extrinsic Camera Calibration(터미널창 미리 8개 띄우기)

: Extrinsic Camera Calibration 를 위해 경기장(Lane) 사용

ⓐ Remote PC에서 새 터미널 열기

$ roscore

ⓑ Remote PC에서 새 터미널 열기

$ ssh ubuntu@192.168.0.25 ← Remote PC에서 TurtleBot3 Burger(Raspberry pi)을 원격접속

pw : turtlebot 입력

~$ roslaunch turtlebot3_autorace_camera raspberry_pi_camera_publish.launch

ⓒ Remote PC에서 새 터미널 열기

$ roslaunch turtlebot3_autorace_camera intrinsic_camera_calibration.launch mode:=action

Calibration에서 action 바뀜

```
ROS_MASTER_URI=http://192.168.1.141:11311

process[republish-1]: started with pid [8897]
process[relay_camera_info-2]: started with pid [8898]
process[camera/image_proc-3]: started with pid [8899]
```

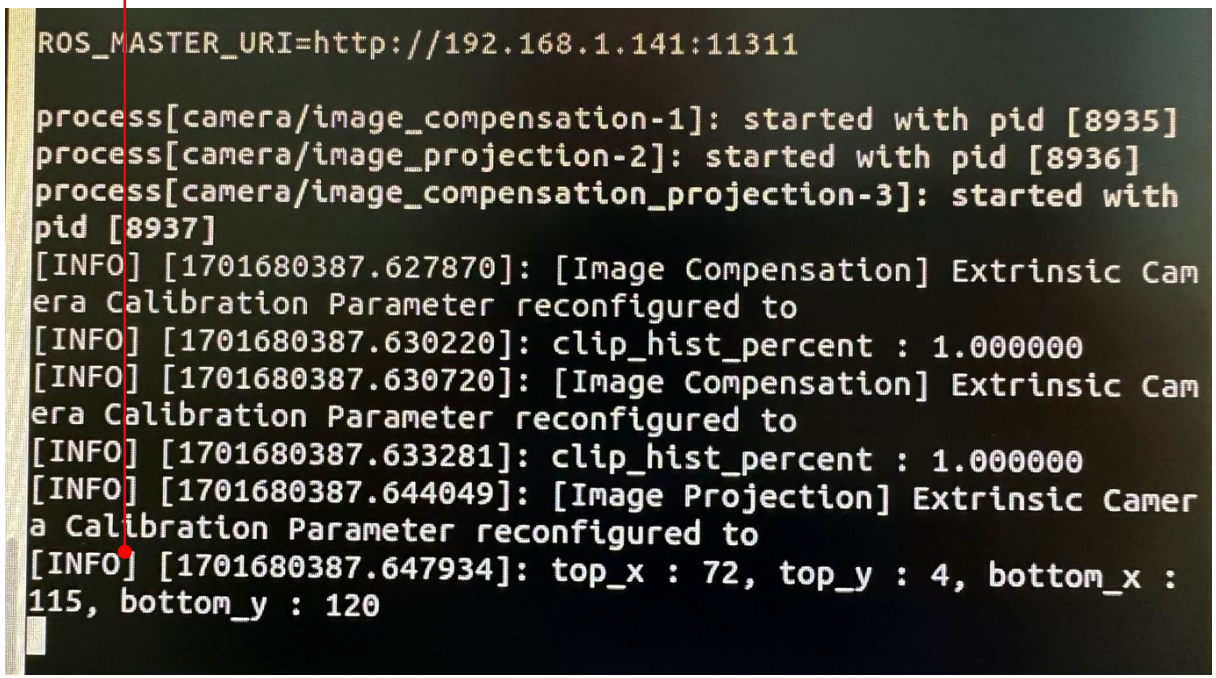
실제 경기장

: Turtlebot의 왼쪽 Yellow, 오른쪽 white

: calibration에서는 관계없음

ⓓ **Remote PC**에서 새 터미널 열기

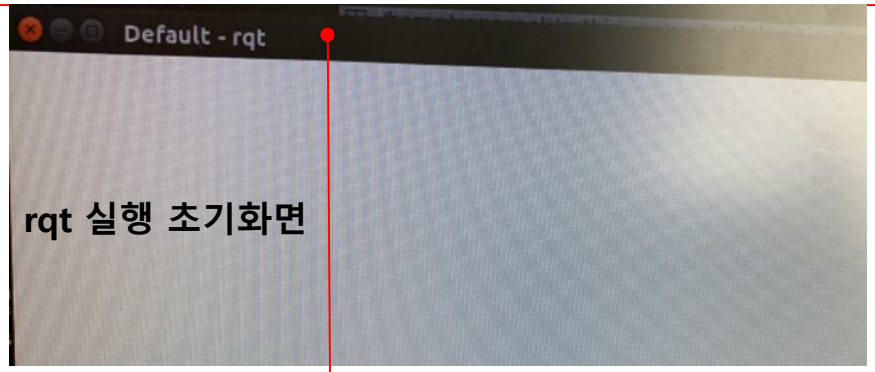
$ roslaunch turtlebot3_autorace_camera extrinsic_camera_calibration.launch mode:=calibration



ⓕ **Remote PC에서 새 터미널 열기**

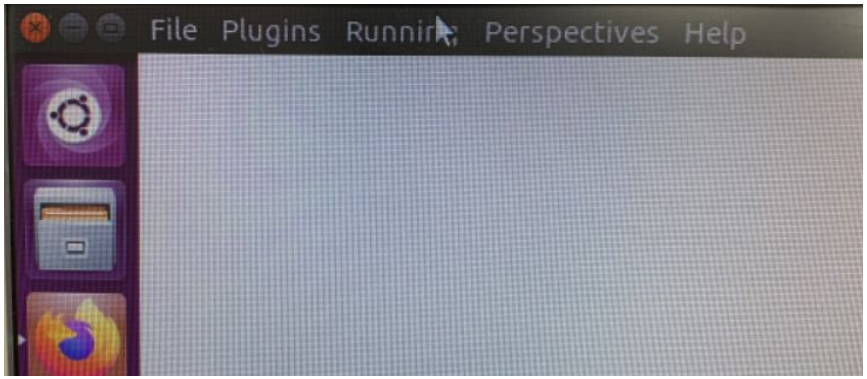
$ rqt

➔ 또는 $ rqt_image_view



rqt 실행 초기화면

double click 하여 전체 화면으로 만들기



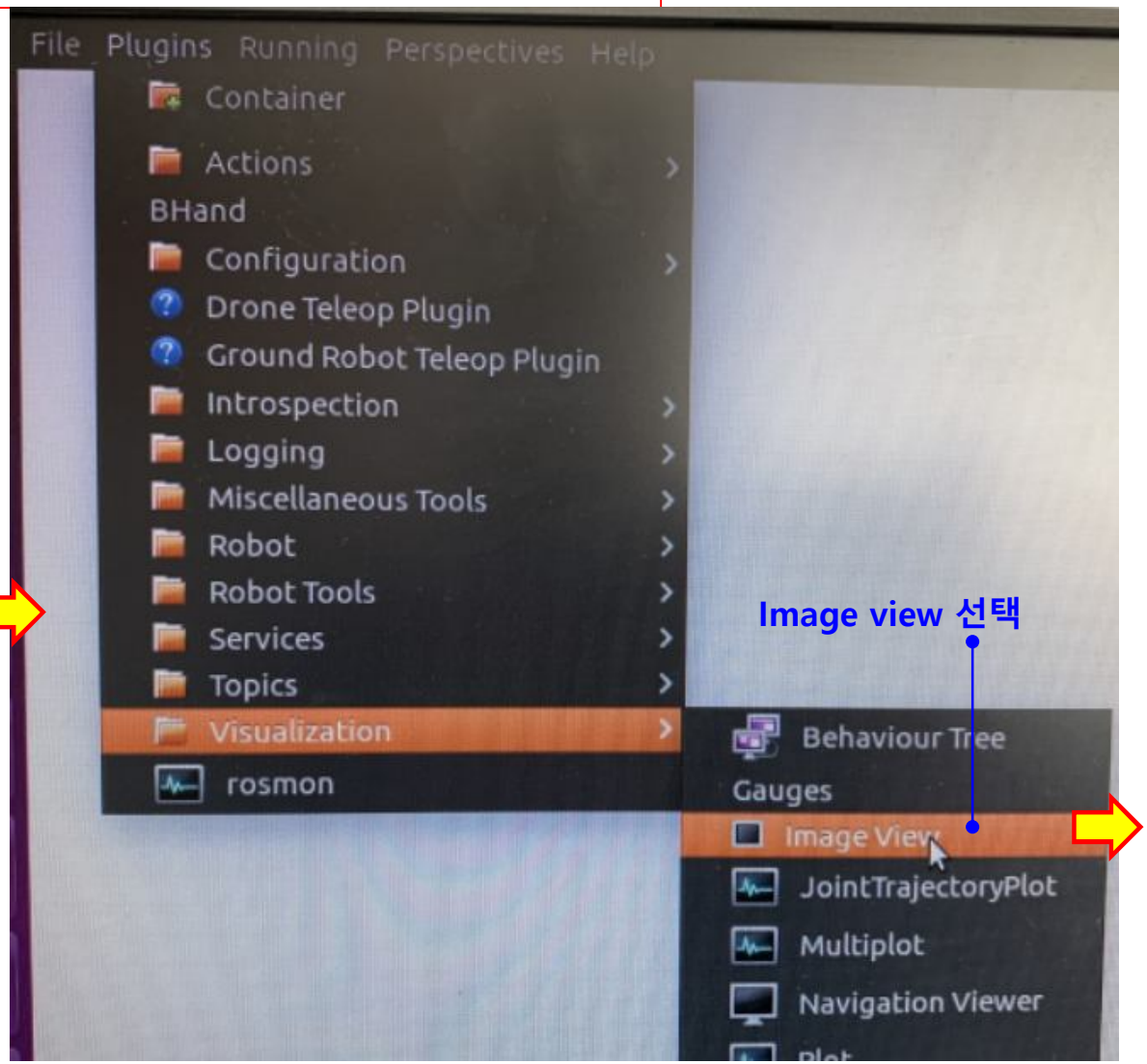
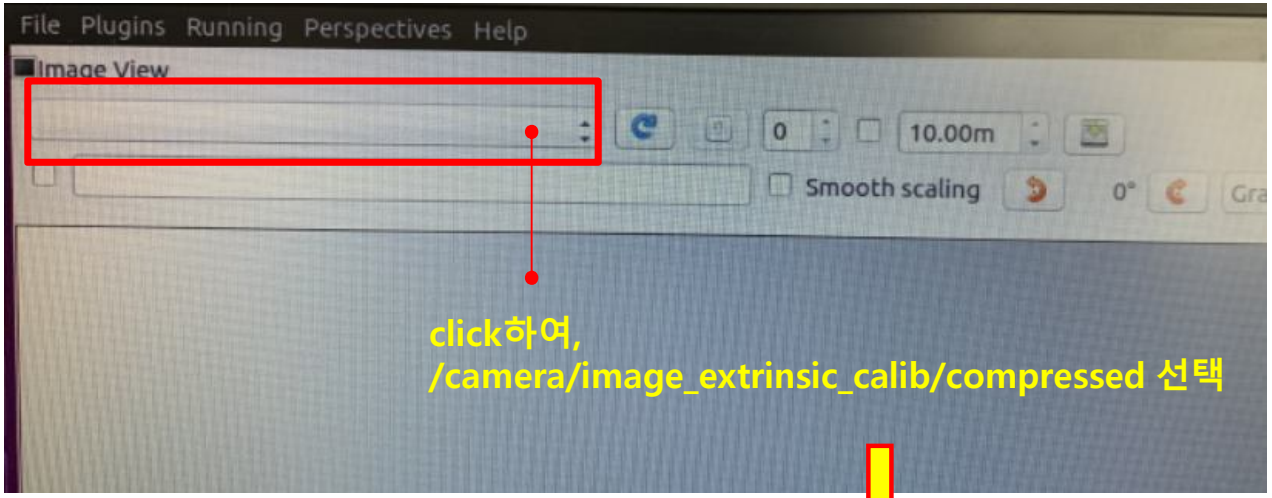
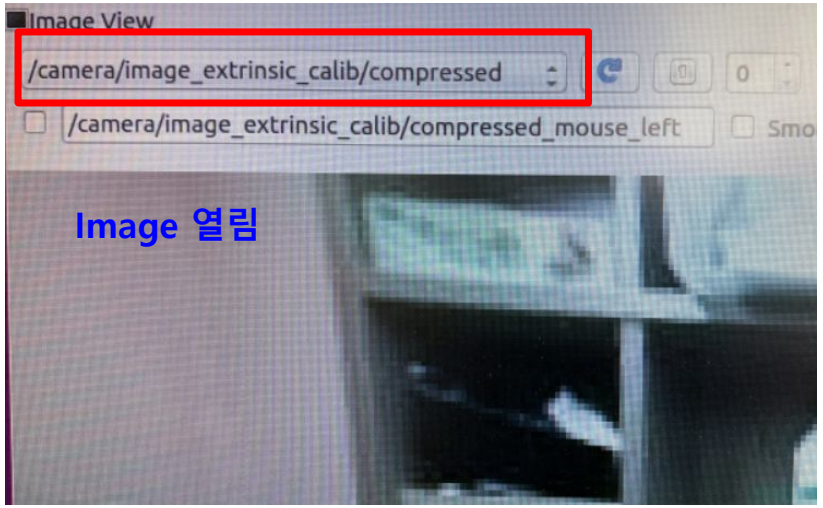


Image view 선택

$ rqt 실행 후, image viewer 선택 후 화면



rqt image viewer 초기 화면



주간(외부 조도가 강할 때 실험 영상)

야간(외부 조도가 약할 때 실험 영상)

Yellow 와

White color 구별되지 않음

실험 환경에 따라,

Camera imaging calibration

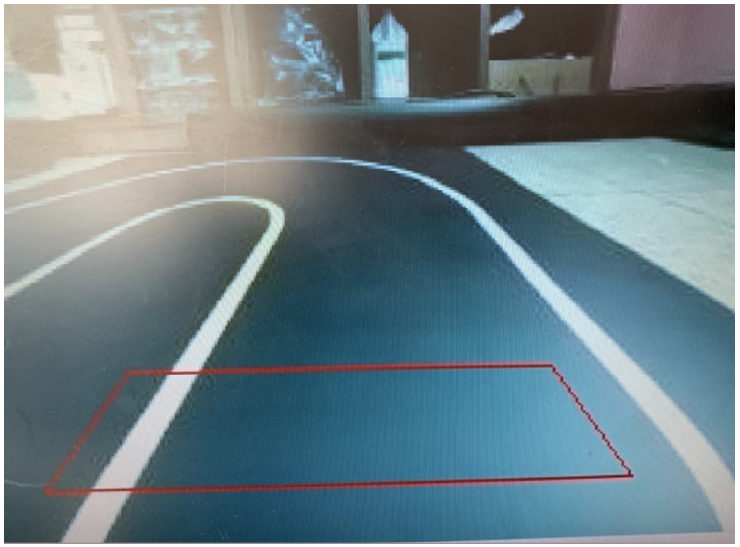
적용 결과가 다르게 나타남

[주의]

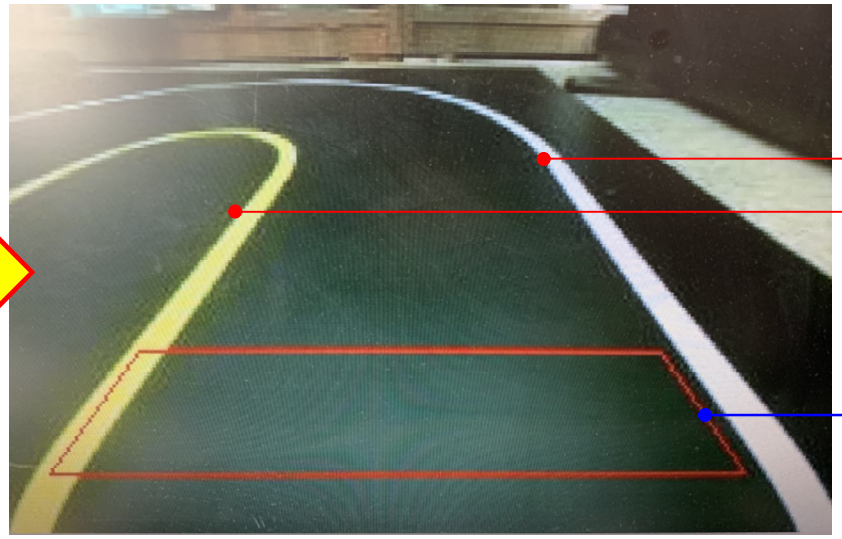
White, yellow lane 인식하는

자율주행 알고리즘

ⓖ 외부 조도를 차단하고, 카메라 부착 부위를 조금씩 조정하여 projected view (Bird's eye view)가 중앙에 위치하도록 수정



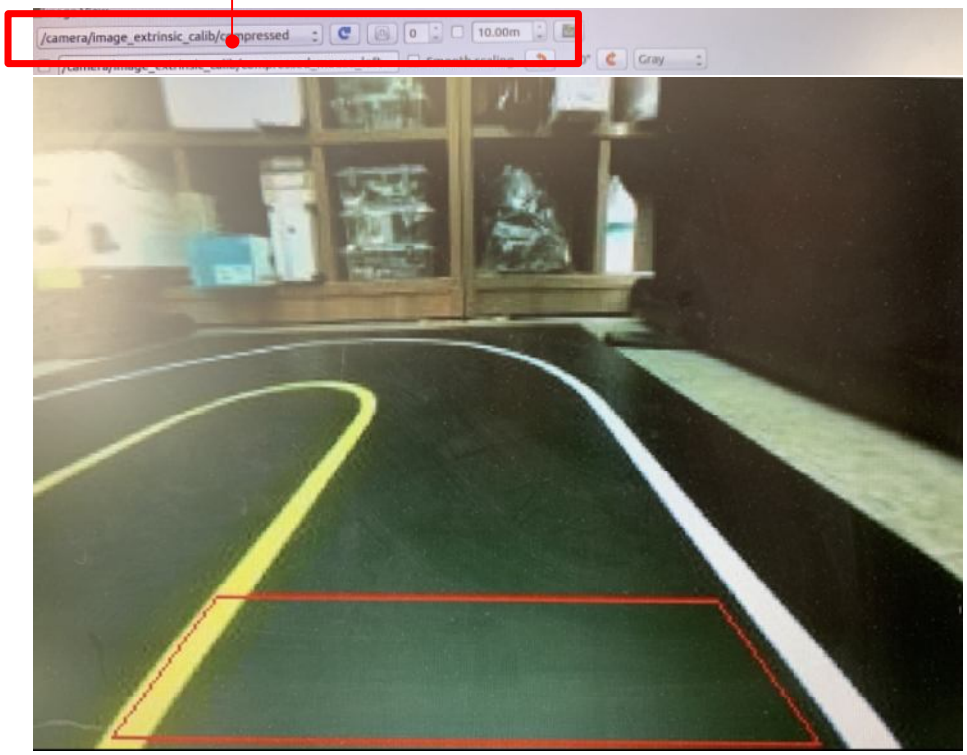
외부 조도를 차단 전, 카메라 부착 부위 조정 전

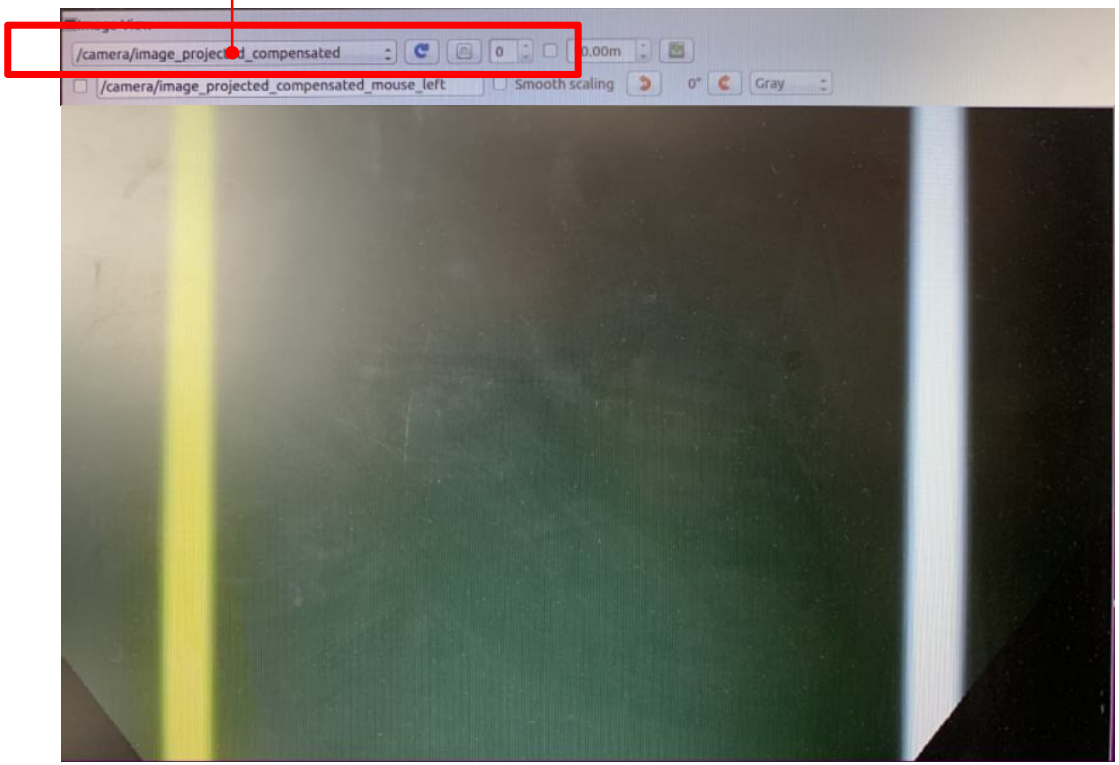


외부 조도를 차단, 카메라 부착 부위 조정

ⓗ rqt image viewer의 topic 변경

/camera/image_extrinsic_calib/compressed ⟺ /camera/image_projected_compensated

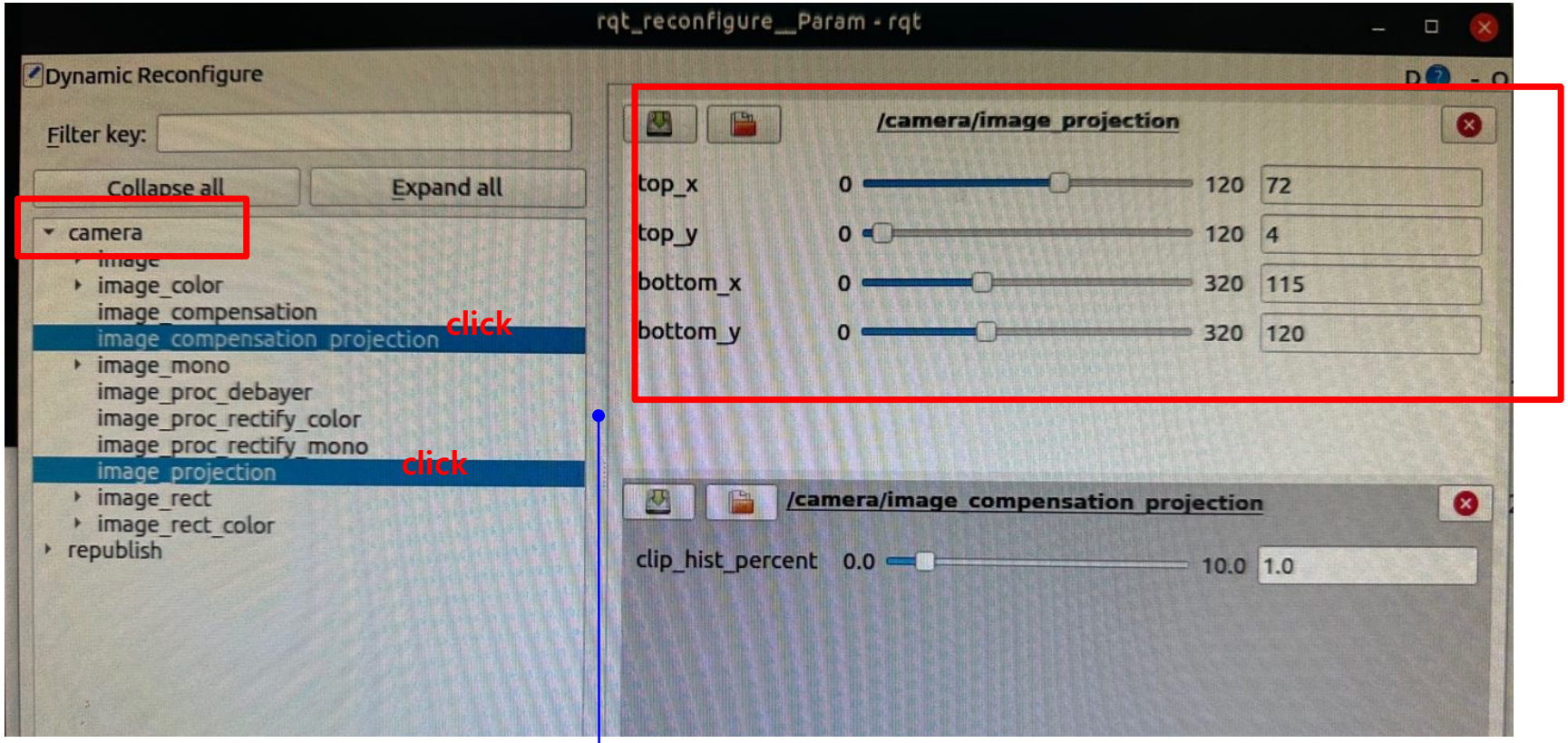




ⓘ **Remote PC에서 새 터미널 열기**

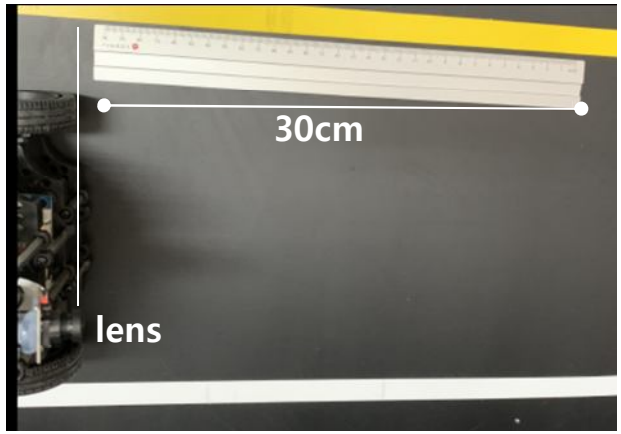
$ **rosrun rqt_reconfigure rqt_reconfigure**

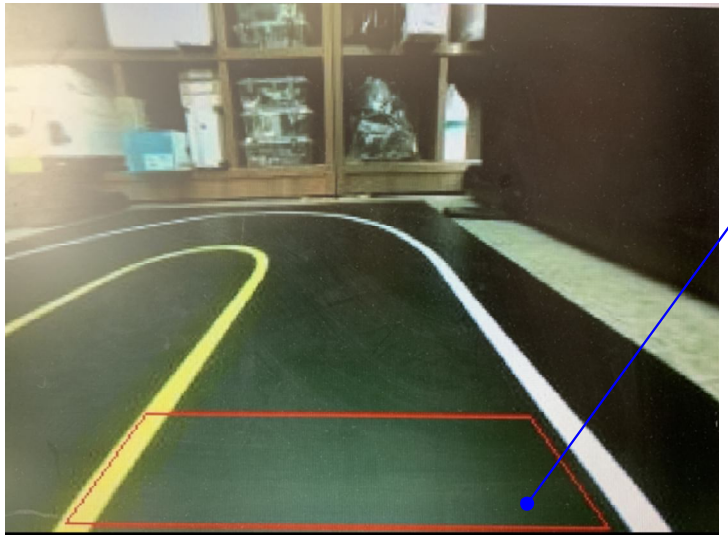
: rqt_reconfigure 초기 화면

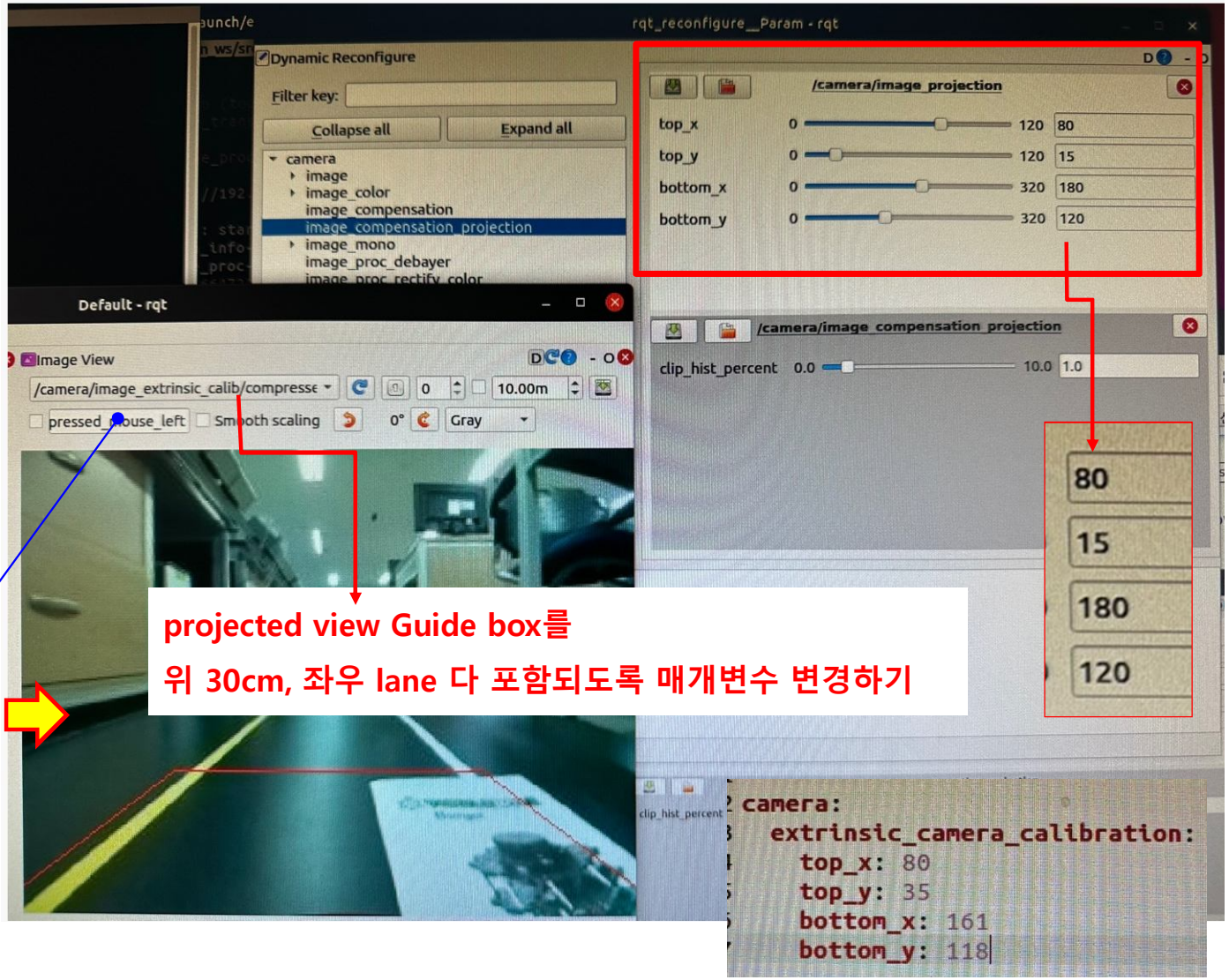


화면 넓히기

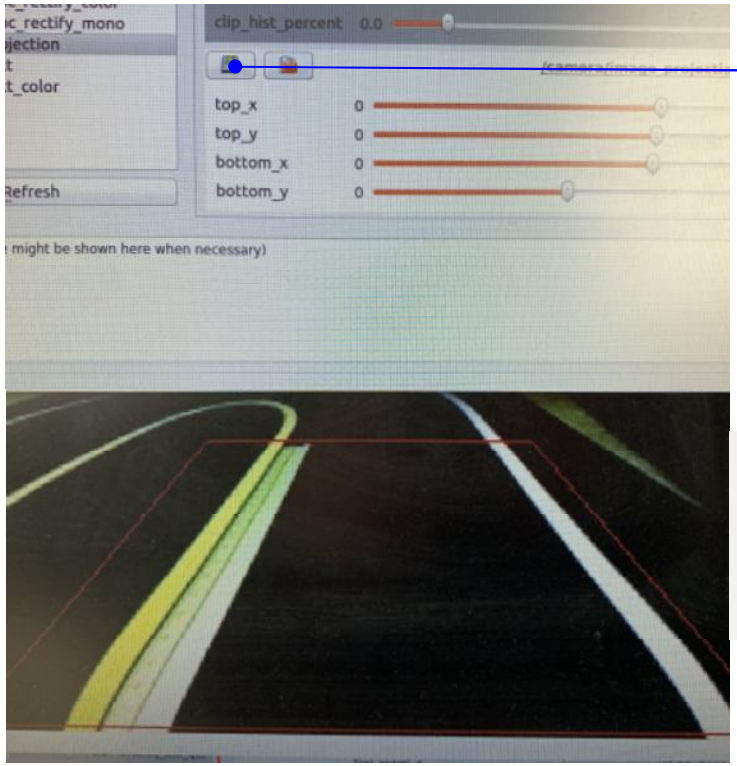
둘째 image_projection : extrinsic calibration(camera를 통한 lane 검출을 위한 area 설정)



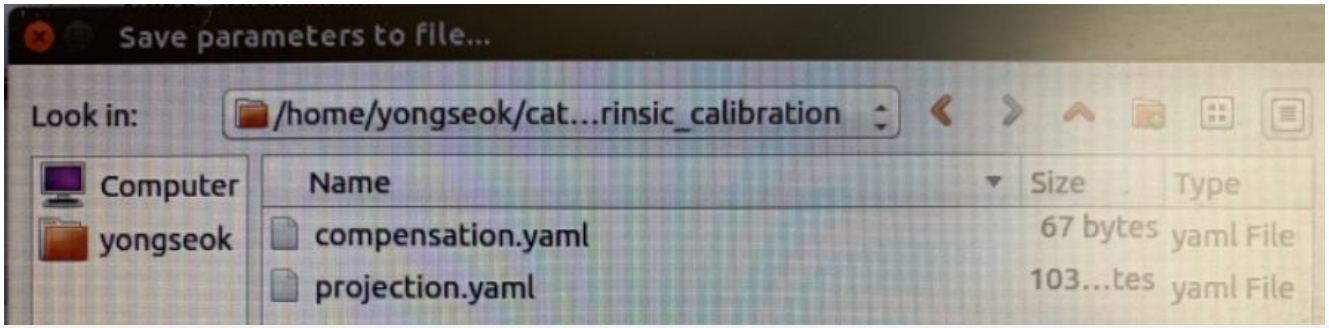




둘째 image_projection : extrinsic calibration(camera를 통한 lane 검출을 위한 area 설정)



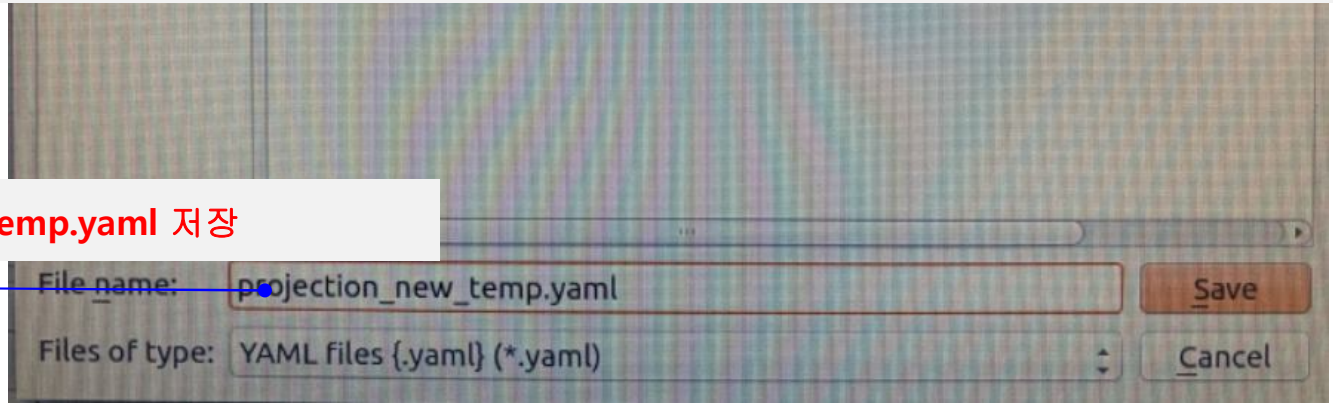
매개변수 수정 후 저장 : click



방금 수정한 값을 다른 이름으로 저장 후에, 보정한 값을 projection.yaml 반영해야

➔ 경로 catkin_ws / src / turtlebot3_autorace_2020 / turtlebot3_autorace_camera / calibration / extrinsic_calibration

Projection_new_temp.yaml 저장



➔ 매개변수 반영하기

경로 catkin_ws / src / turtlebot3_autorace_2020 / turtlebot3_autorace_camera / calibration / extrinsic_calibration 의 projection.yaml 에 반영하기

```
camera:
  extrinsic_camera_calibration:
    top_x: 80
    top_y: 35
    bottom_x: 161
    bottom_y: 118
```

Projection_new_temp.yaml

```
!!python/object/new:dynamic_reconfigure.encoding.Config
dictitems:
  bottom_x: 180
  bottom_y: 120
  groups: !!python/object/new:dynamic_reconfigure.encoding.Config
    dictitems:
      bottom_x: 180
      bottom_y: 120
      groups: !!python/object/new:dynamic_reconfigure.encoding.Config
        state: []
      id: 0
      name: Default
      parameters: !!python/object/new:dynamic_reconfigure.encoding.Config
        state: []
      parent: 0
      state: true
      top_x: 80
      top_y: 15
      type: ''
    state: []
  top_x: 80
  top_y: 15
state:
```

매개변수 반영하기

projection.yaml 수정 후 저장

```
---
camera:
  extrinsic_camera_calibration:
    top_x: 72
    top_y: 4
    bottom_x: 115
    bottom_y: 120
```

80
15
180
120

만일 실험을 마치려면, terminal은 CTL+C로 중단 후에 종료

8-2. RPi Camera(G) – Camera calibration test(터미널창 미리 8개 띄우기)

(6) test

: Intrinsic Camera Calibration, Extrinsic Camera Calibration 반영 값 확인하기

ⓐ Remote PC에서 새 터미널 열기

$ roscore

ⓑ Remote PC에서 새 터미널 열기

$ ssh ubuntu@192.168.0.25 ← Remote PC에서 TurtleBot3 Burger(Raspberry pi)을 원격접속

pw : turtlebot 입력

~$ roslaunch turtlebot3_autorace_camera raspberry_pi_camera_publish.launch

ⓒ Remote PC에서 새 터미널 열기

$ roslaunch turtlebot3_autorace_camera intrinsic_camera_calibration.launch mode:=action Calibration에서 action 바뀜

ⓓ Remote PC에서 새 터미널 열기

$ roslaunch turtlebot3_autorace_camera extrinsic_camera_calibration.launch mode:=action Calibration에서 action 바뀜

ⓕ Remote PC에서 새 터미널 열기

$ rqt

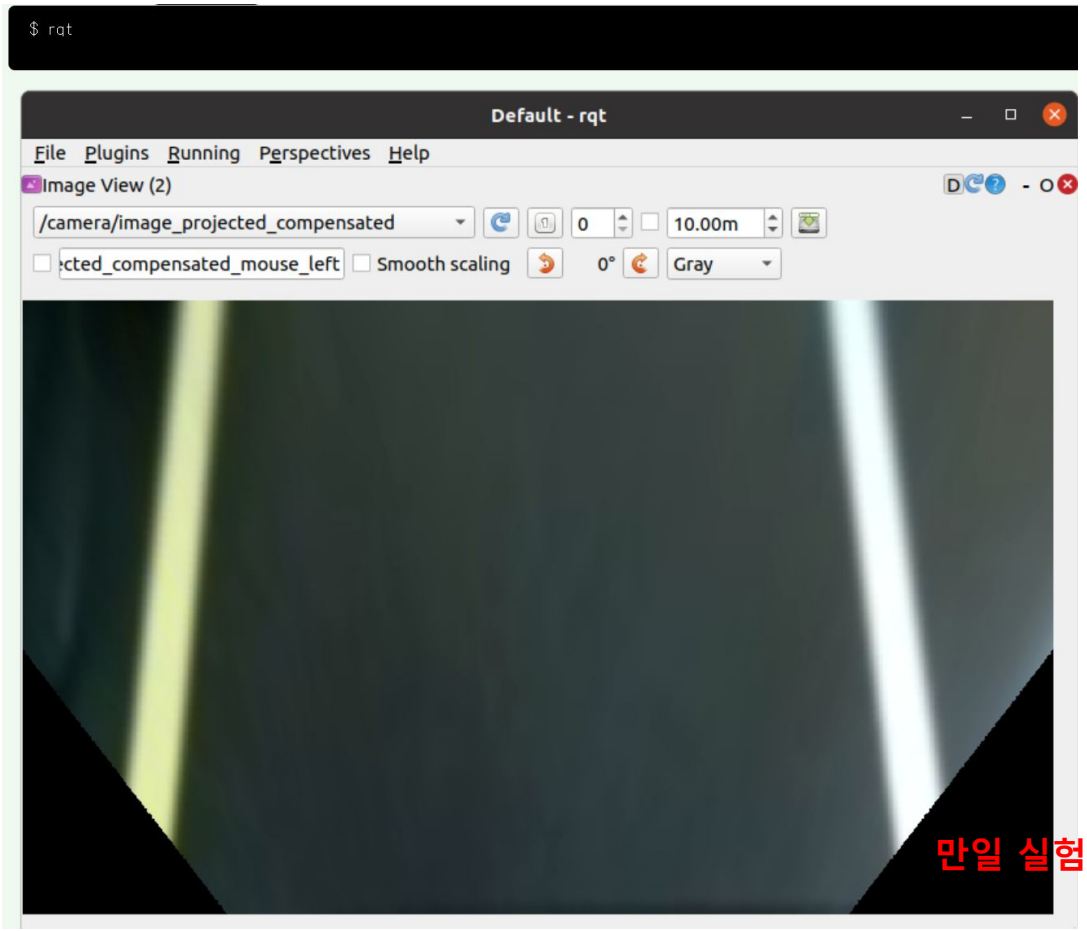
8-2. RPi Camera(G) – Camera calibration test

(6) test

: Intrinsic Camera Calibration, Extrinsic Camera Calibration 반영 값 확인하기

ⓕ Remote PC에서 새 터미널 열기

$ rqt



만일 실험을 마치려면, terminal은 CTL+C로 중단 후에 종료

8. Autonomous Driving

Battery 사전에 충전하기(미충전 경우 Buzzer 소리 발생)

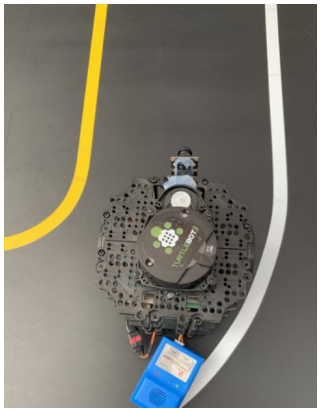
➔ 방치 시 완전 방전됨

Detect lane 실행 시 반드시 battery 충전되어 있어야 함

8-3. Lane detection

(1) Setting : Place TurtleBot3 between yellow and white lanes

➔ yellow(left) – Turtlebot3 burger – white(right)



8. 3. Lane Detection

Lane detection package that runs on the `Remote PC` receives camera images either from TurtleBot3 or Gazebo simu
driving lanes and to drive the Turtlebot3 along them.
The following instructions describe how to use and calibrate the lane detection feature via rqt.

1. Place the TurtleBot3 inbetween yellow and white lanes.

   NOTE: The lane detection filters yellow on the left side while filters white on the right side. Be sure that the yellow lane is c
   robot.

in Gazebo simulation.

2. Open a new terminal and launch Autorace Gazebo simulation. The `roscore` will be automatically launched with
   command.

```
$ roslaunch turtlebot3_gazebo turtlebot3_autorace_2020.launch
```

3. Open a new terminal and launch the intrinsic calibration node.

```
$ roslaunch turtlebot3_autorace_camera intrinsic_camera_calibratio
```

4. Open a new terminal and launch the extrinsic calibration node.

```
$ roslaunch turtlebot3_autorace_camera extrinsic_camera_calibratio
```

5. Open a new terminal and launch the lane detection calibration r

```
$ roslaunch turtlebot3_autorace_detect detect_lane.launch mode:=ca
```

6. Open a new terminal and launch the rqt.

```
$ rqt
```

실행

Click to expand : How to Perform Lane Detection with Actual TurtleBot3?

Lane detection package allows Turtlebot3 to drive between two lanes without external influence.

The following instructions describe how to use the lane detection feature and to calibrate camera via rqt.

1. Place TurtleBot3 between yellow and white lanes.

   NOTE: Be sure that yellow lane is placed left side of the robot and White lane is placed right side of the robot.

2. Launch roscore on `Remote PC`.

```
$ roscore
```

3. Trigger the camera on `SBC`.

```
$ roslaunch turtlebot3_autorace_camera raspberry_pi_camera_publish.launch
```

4. Run a intrinsic camera calibration launch file on `Remote PC`.

8-3. Lane detection(터미널창 미리 8개 띄우기)

ⓐ Remote PC에서 새 터미널 열기

$ roscore

ⓑ Remote PC에서 새 터미널 열기

$ ssh ubuntu@192.168.0.25 ← Remote PC에서 TurtleBot3 Burger(Raspberry pi)을 원격접속

pw : turtlebot 입력

~$ roslaunch turtlebot3_autorace_camera raspberry_pi_camera_publish.launch

ⓒ Remote PC에서 새 터미널 열기

$ roslaunch turtlebot3_autorace_camera intrinsic_camera_calibration.launch mode:=action

ⓓ Remote PC에서 새 터미널 열기

$ roslaunch turtlebot3_autorace_camera extrinsic_camera_calibration.launch mode:=action

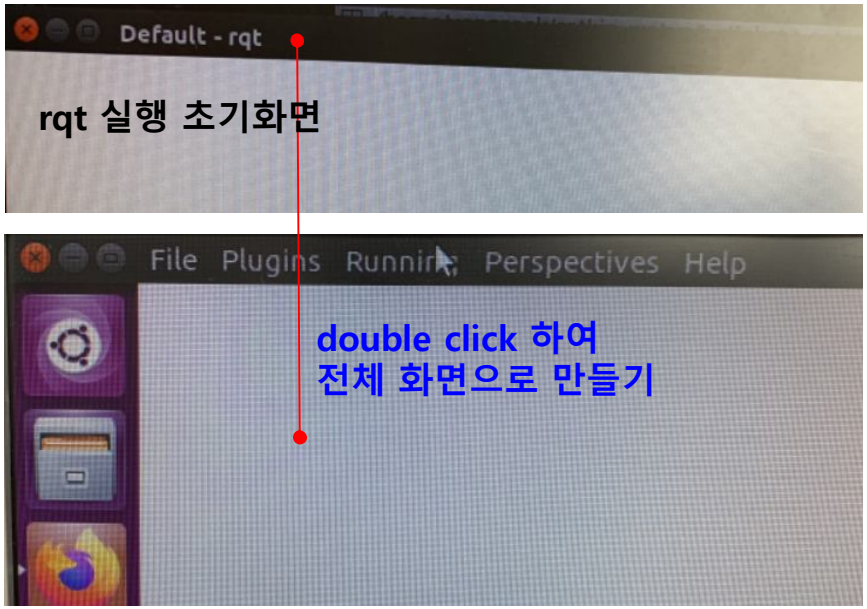
ⓕ Remote PC에서 새 터미널 열기

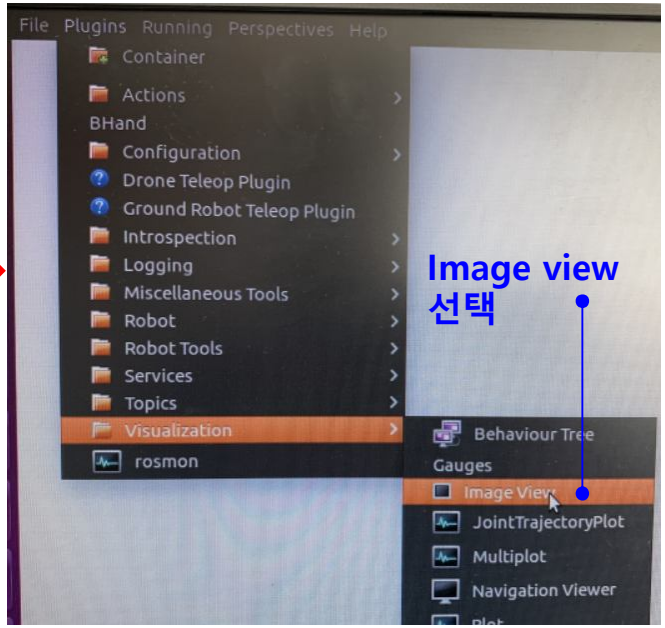
$ roslaunch turtlebot3_autorace_detect detect_lane.launch mode:=calibration

Remote PC에서 새 터미널 열기

$ rqt

$ rqt


Default - rqt
rqt 실행 초기화면
File Plugins Running Perspectives Help
double click 하여
전체 화면으로 만들기

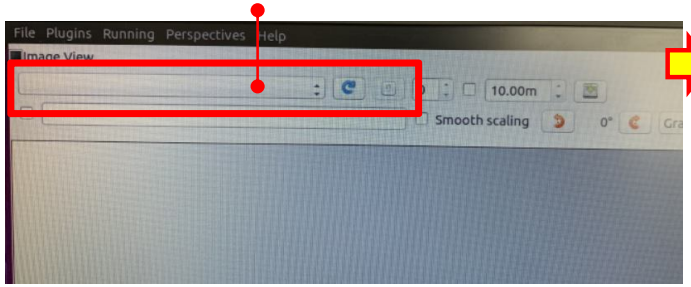

File Plugins Running Perspectives Help
Container
Actions
BHand
Configuration
Drone Teleop Plugin
Ground Robot Teleop Plugin
Introspection
Logging
Miscellaneous Tools
Robot
Robot Tools
Services
Topics
Visualization
rosmon
Image view
선택
Behaviour Tree
Gauges
Image View
JointTrajectoryPlot
Multiplot
Navigation Viewer

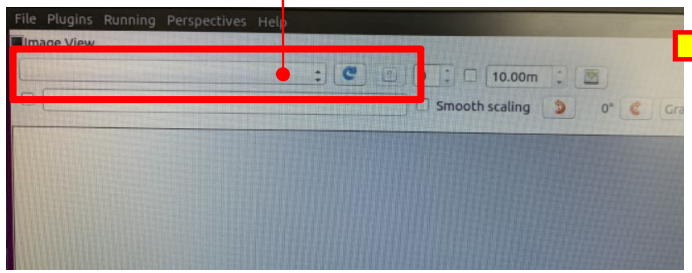
plugins > visualization > Image view 3번 반복 ★

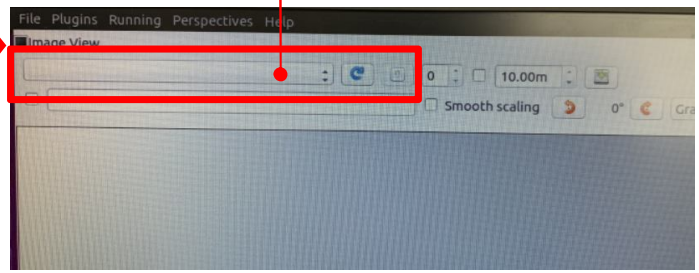
click하여, 아래 선택
/detect/image_yellow_lane_marker/compressed

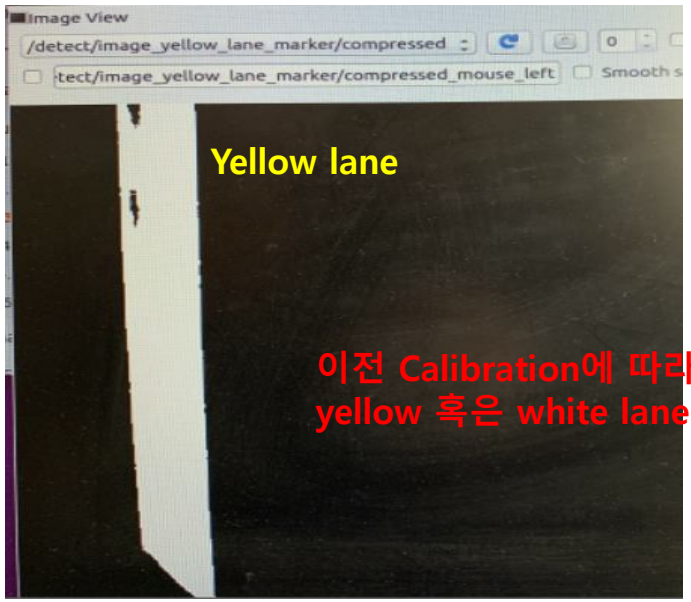
click하여, 아래 선택
/detect/image_white_lane_marker/compressed

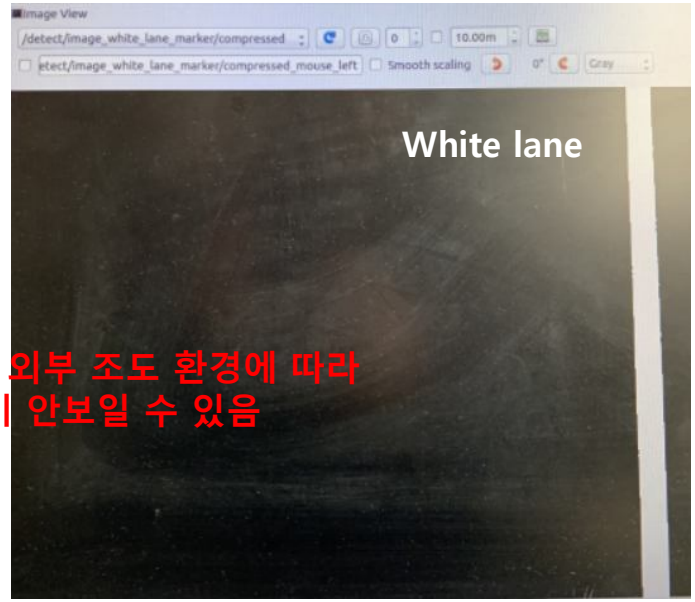
click하여, 아래 선택
/detect/image_lane/compressed

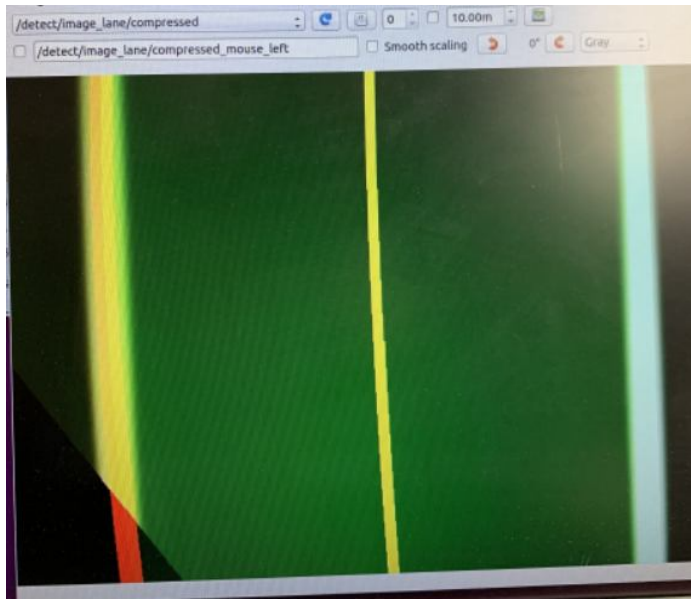












이전 Calibration에 따라, 외부 조도 환경에 따라
yellow 혹은 white lane이 안보일 수 있음

rqt는 multi window를 지원하기에 세개의 image를 실행하여 window에 적절히 배치하기

Red Yellow

Yellow

Cyan White

rqt 3개 image 실행 배치 화면

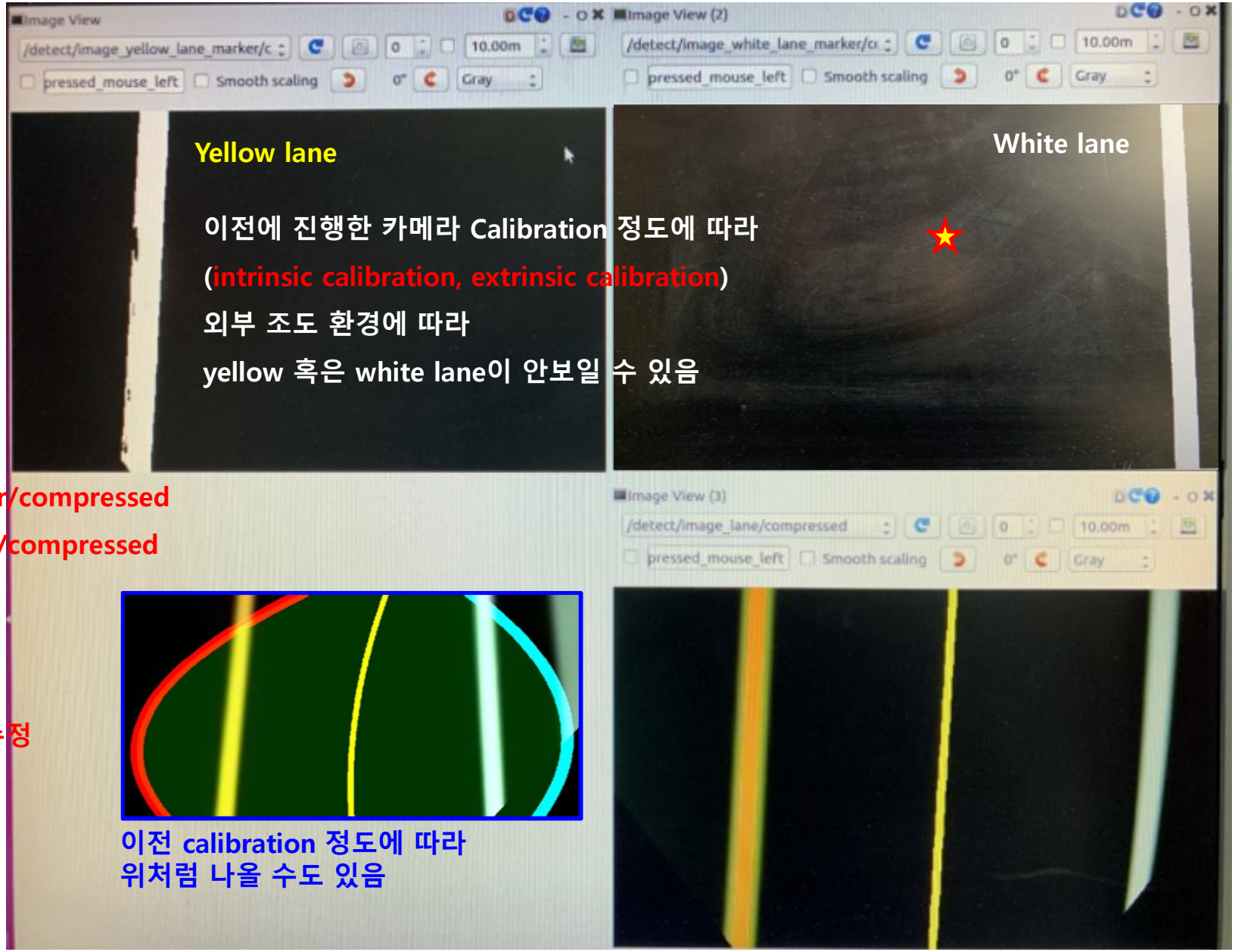
/detect/image_yellow_lane_marker/compressed

/detect/image_white_lane_marker/compressed

/detect/image_lane/compressed

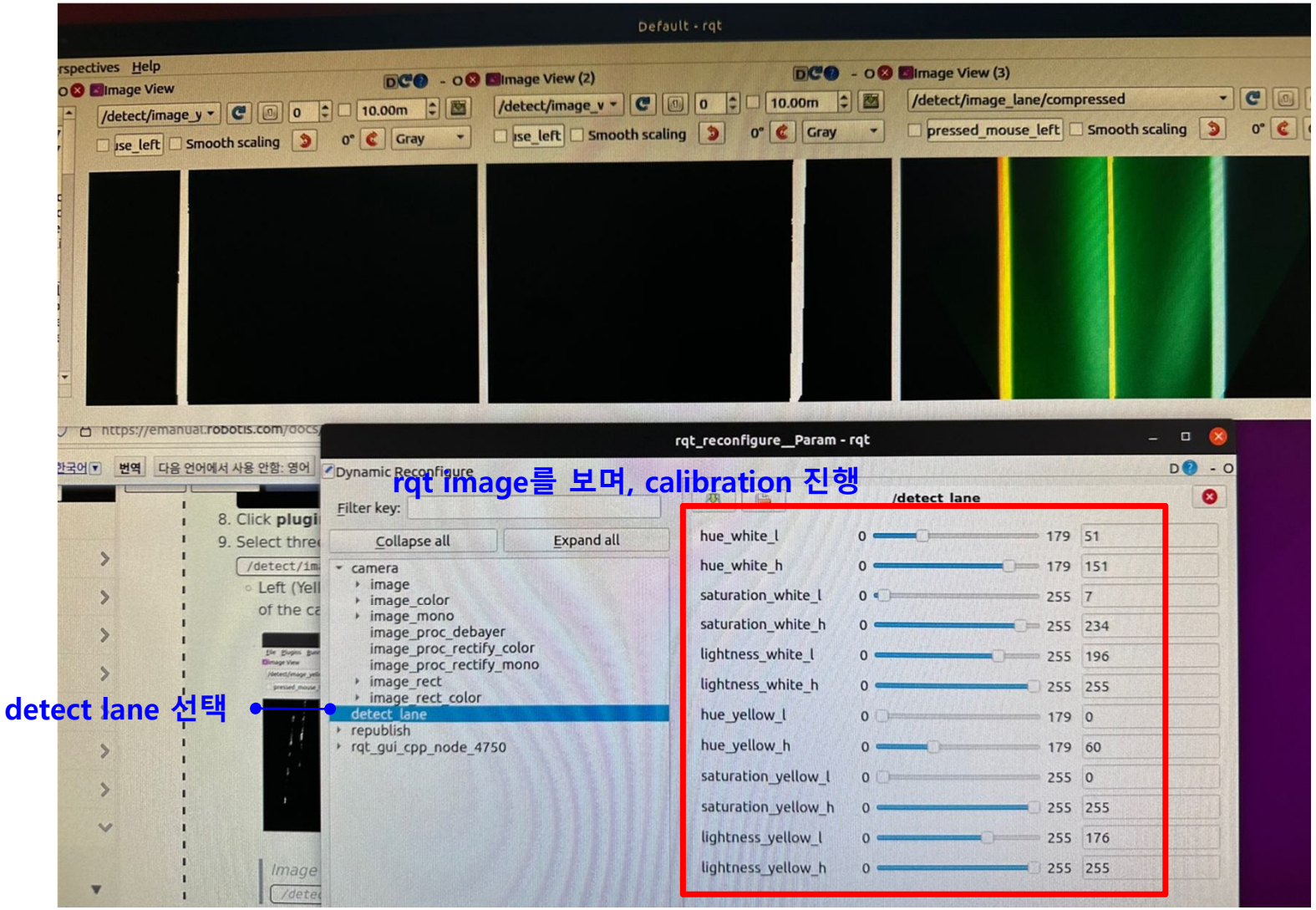
/detect/image_lane/compressed

화면이 안보이면, 다음 페이지 값 수정

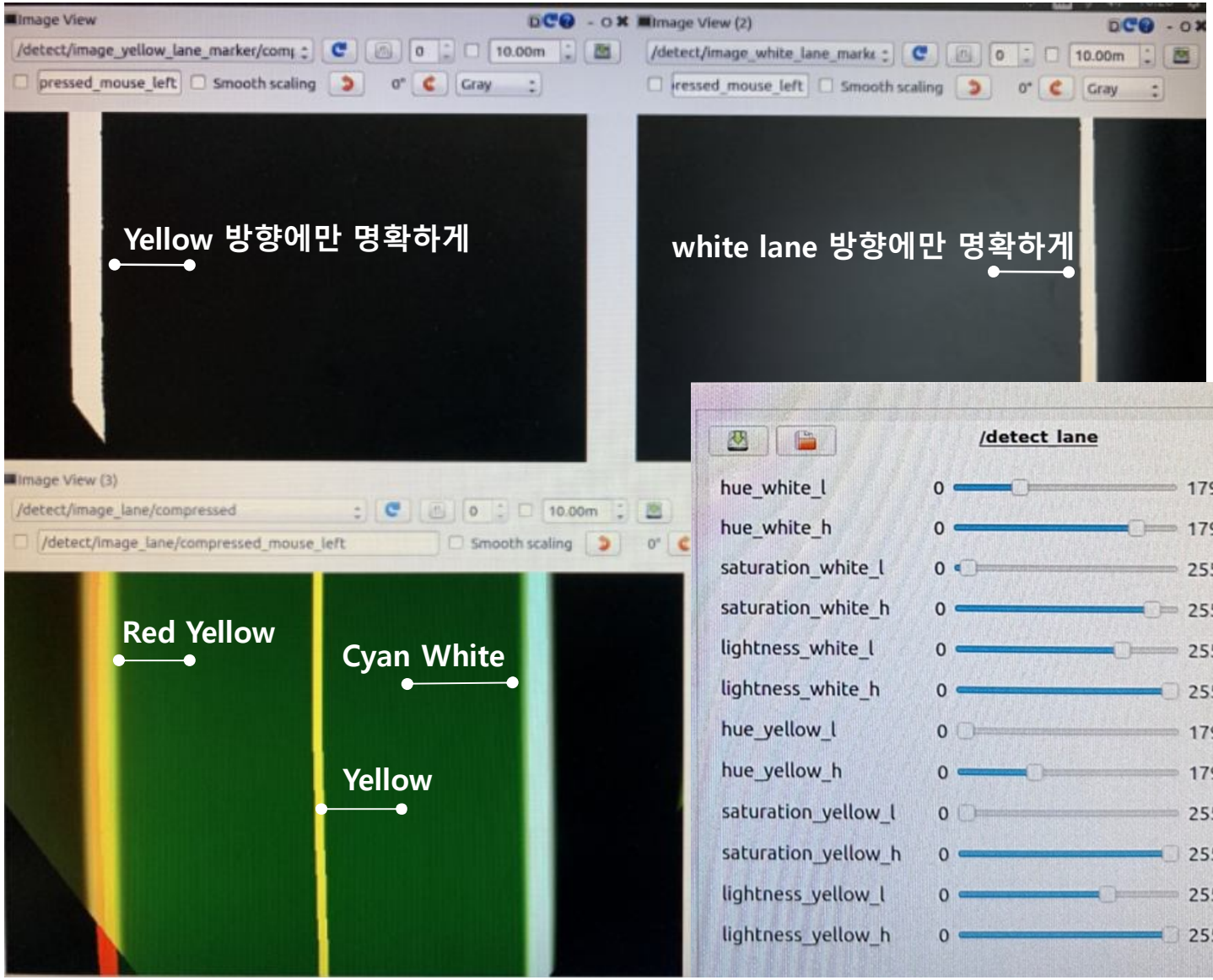


ⓖ **Remote PC에서 새 터미널 열기**

```
$ rosrun rqt_reconfigure rqt_reconfigure
```

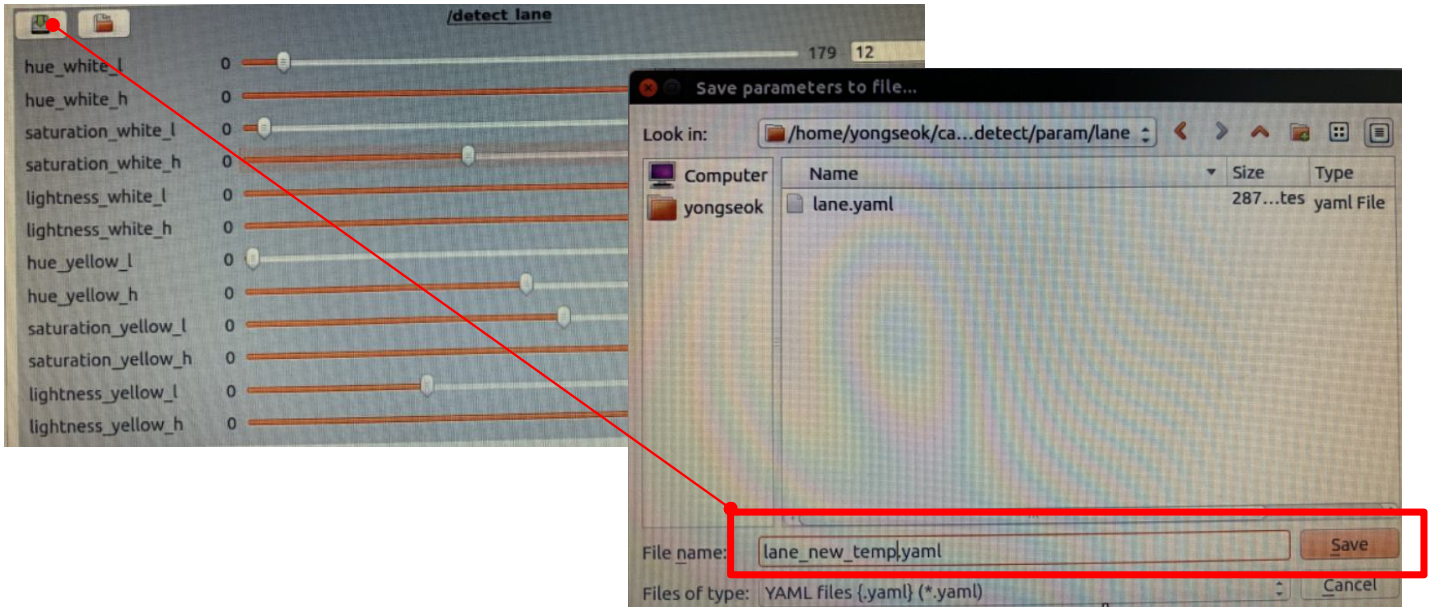


ⓗ rqt image를 보며, calibration 진행



rqt image를 보며, calibration 진행

ⓘ 매개변수 반영(저장)하기



다른 이름(lane_new_temp)으로 저장 후

변경 값을 lane.yaml 반영위해 tmep로 저장

➔ 경로

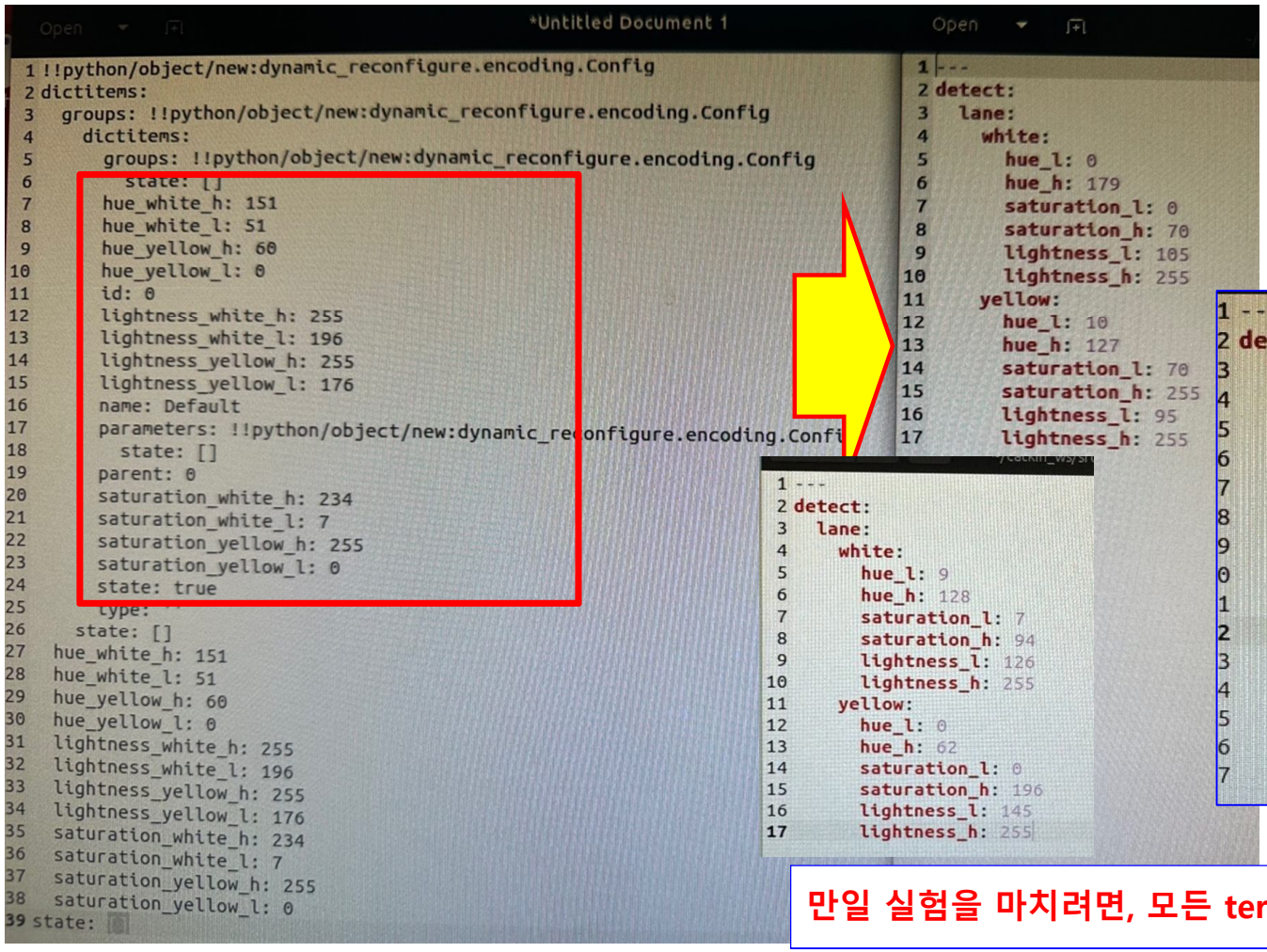
catkin_ws / src / turtlebot3_autorace_2020 / turtlebot3_autorace_detect / param / lane

➔ 매개변수 반영하기

➔ 경로

catkin_ws / src / turtlebot3_autorace_2020 / turtlebot3_autorace_detect / param / lane

temp.yaml



lane.yaml

lane.yaml 수정 후 저장

```
---
detect:
  lane:
    white:
      hue_l: 51
      hue_h: 151
      saturation_l: 7
      saturation_h: 234
      lightness_l: 196
      lightness_h: 255
    yellow:
      hue_l: 0
      hue_h: 60
      saturation_l: 0
      saturation_h: 255
      lightness_l: 176
      lightness_h: 255
```

만일 실험을 마치려면, 모든 terminal은 CTL+C로 중단 후에 종료

8-3. Lane detection 정리

ⓐ Remote PC에서 새 터미널 열기

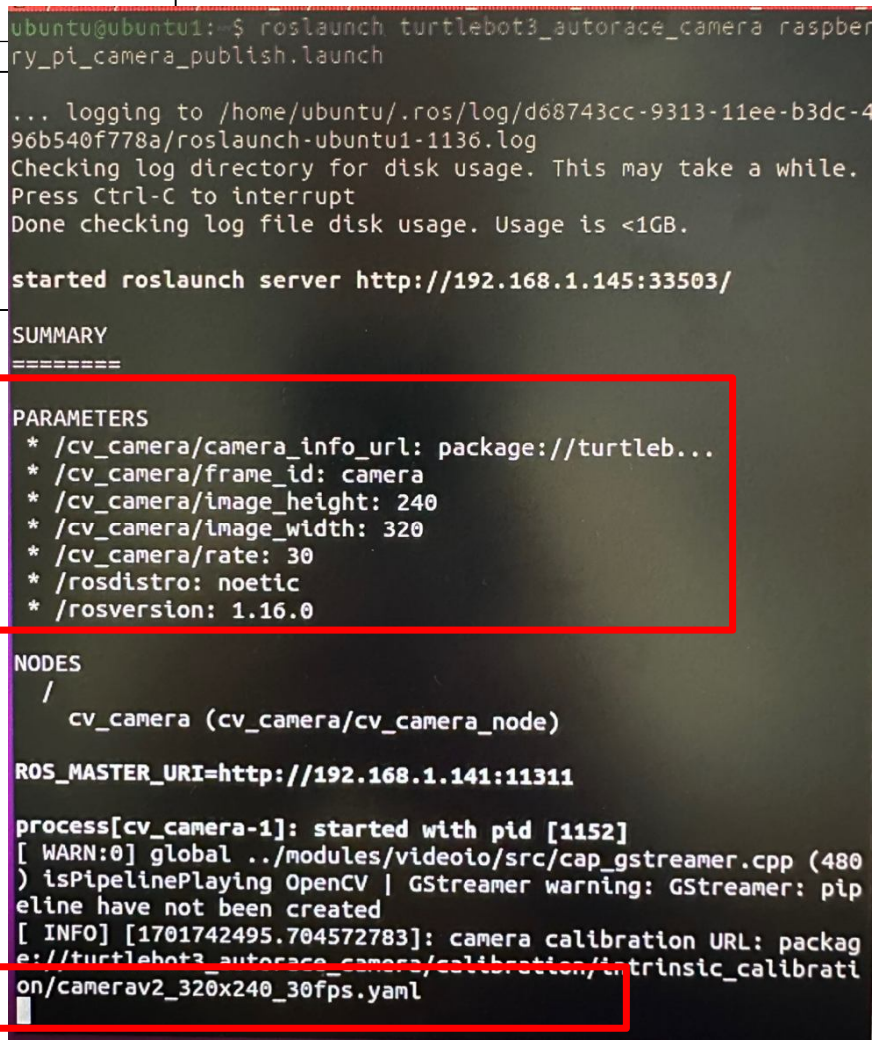
$ roscore

ⓑ Remote PC에서 새 터미널 열기

$ ssh ubuntu@192.168.0.25

pw : turtlebot 입력

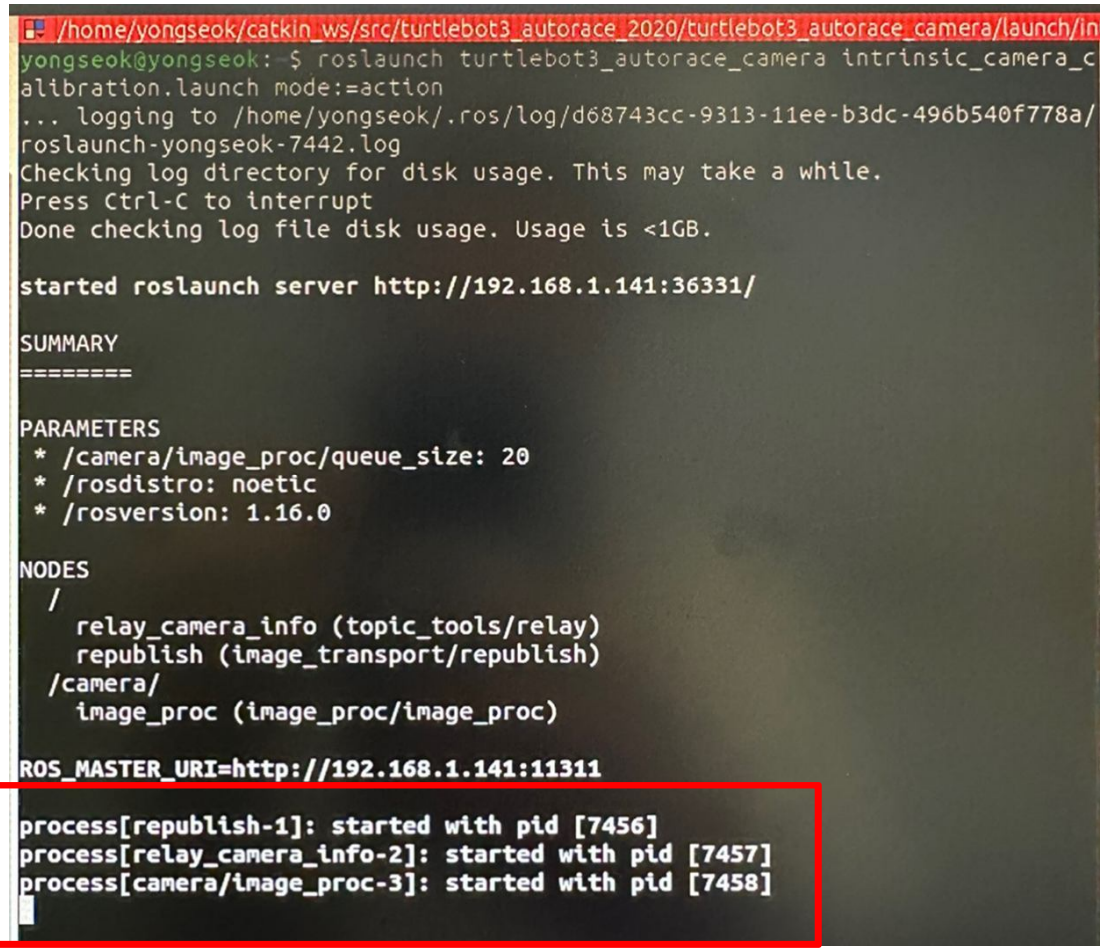
~$ roslaunch turtlebot3_autorace_camera raspberry_pi_camera_publish.launch



8-3. Lane detection 정리

ⓒ Remote PC에서 새 터미널 열기

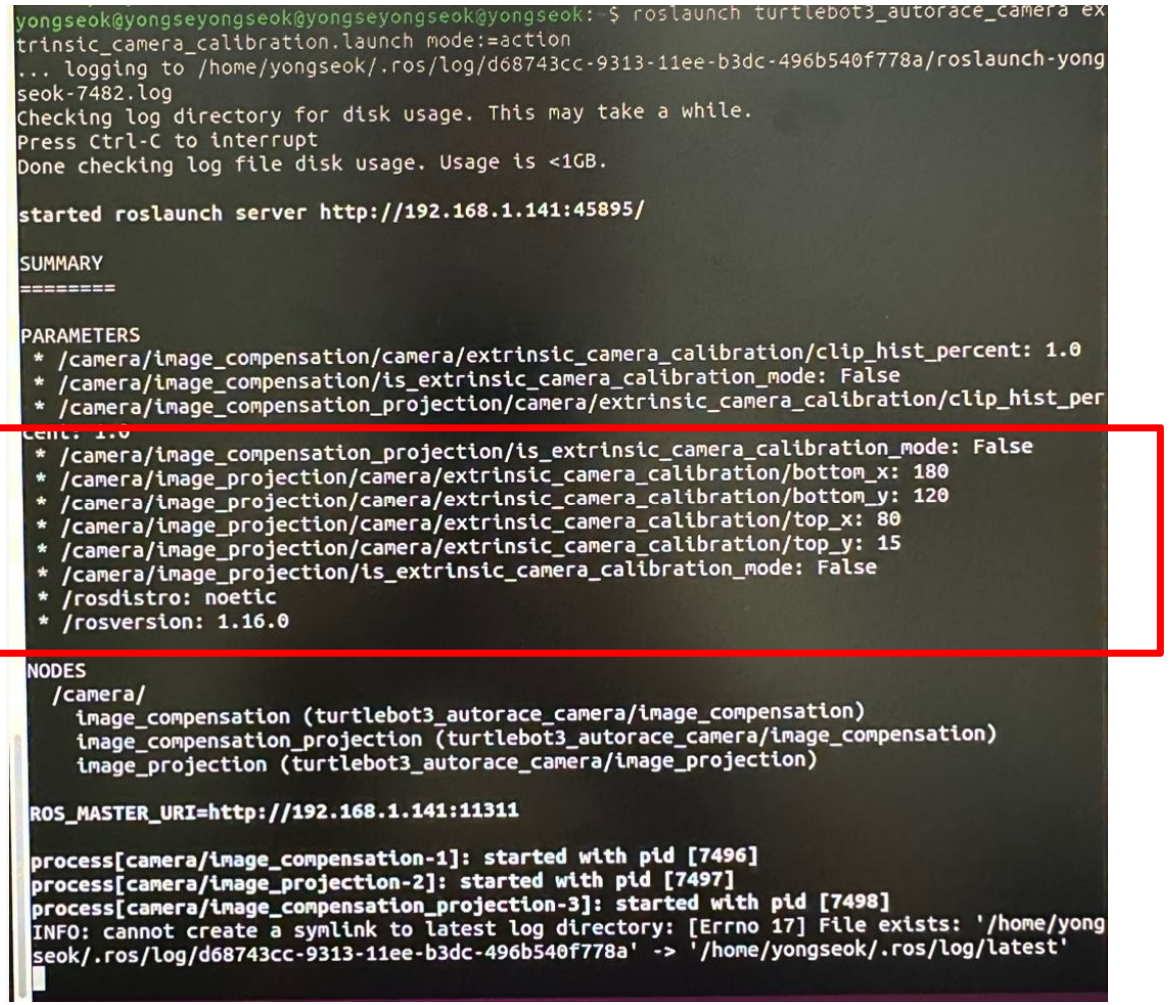
$ roslaunch turtlebot3_autorace_camera intrinsic_camera_calibration.launch mode:=action



8-3. Lane detection 정리

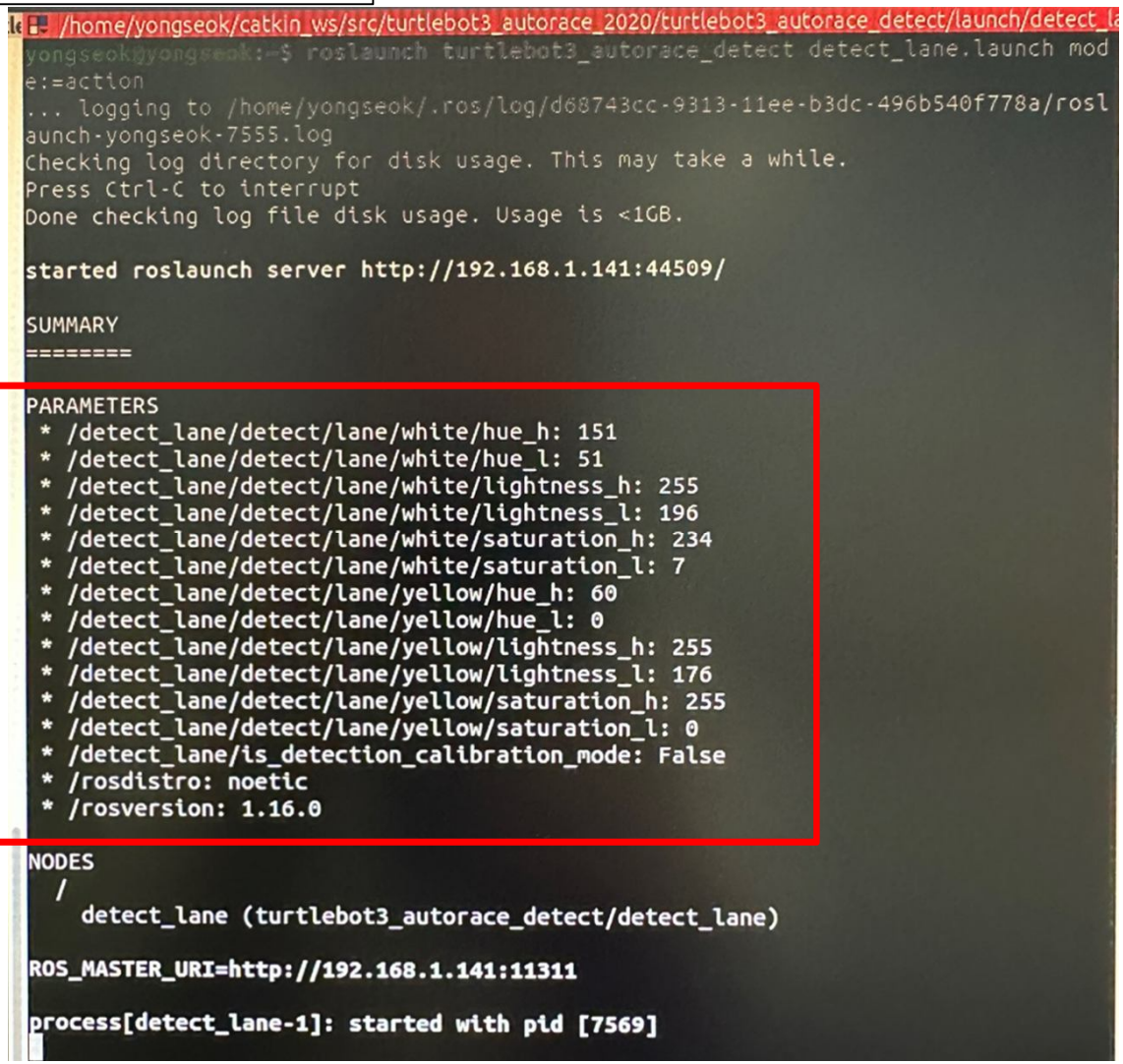
ⓓ Remote PC에서 새 터미널 열기

$ roslaunch turtlebot3_autorace_camera extrinsic_camera_calibration.launch mode:=action



ⓕ **Remote PC에서 새 터미널 열기**

$ roslaunch turtlebot3_autorace_detect detect_lane.launch mode:=calibration



8-3. Lane detection 최종 실습 (터미널창 미리 8개 띄우기) – 방법1 (turtlebot3 느리게 이동)

ⓐ Remote PC에서 새 터미널 열기

$ roscore

ⓑ 새 터미널 열기

$ ssh ubuntu@192.168.0.25 ← Remote PC에서 TurtleBot3 Burger(Raspberry pi)을 원격접속

pw : turtlebot 입력

~$ roslaunch turtlebot3_autorace_camera raspberry_pi_camera_publish.launch

ⓑ 새 터미널 열기

$ ssh ubuntu@192.168.0.25

~$ roslaunch turtlebot3_bringup turtlebot3_robot.launch ← TurtleBot3 Burger bringup 기다리기

ⓒ Remote PC에서 새 터미널 열기

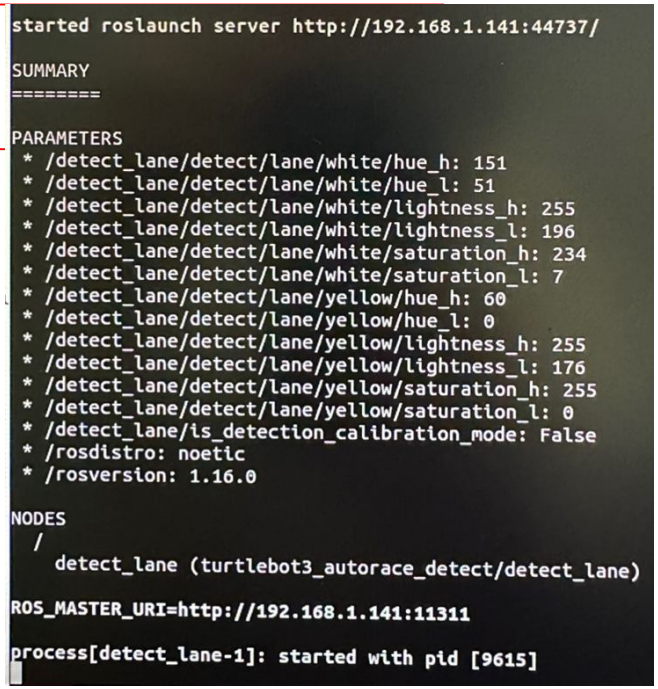
$ roslaunch turtlebot3_autorace_camera intrinsic_camera_calibration.launch mode:=action

ⓓ Remote PC에서 새 터미널 열기

$ roslaunch turtlebot3_autorace_camera extrinsic_camera_calibration.launch mode:=action

ⓕ **Remote PC에서 새 터미널 열기**

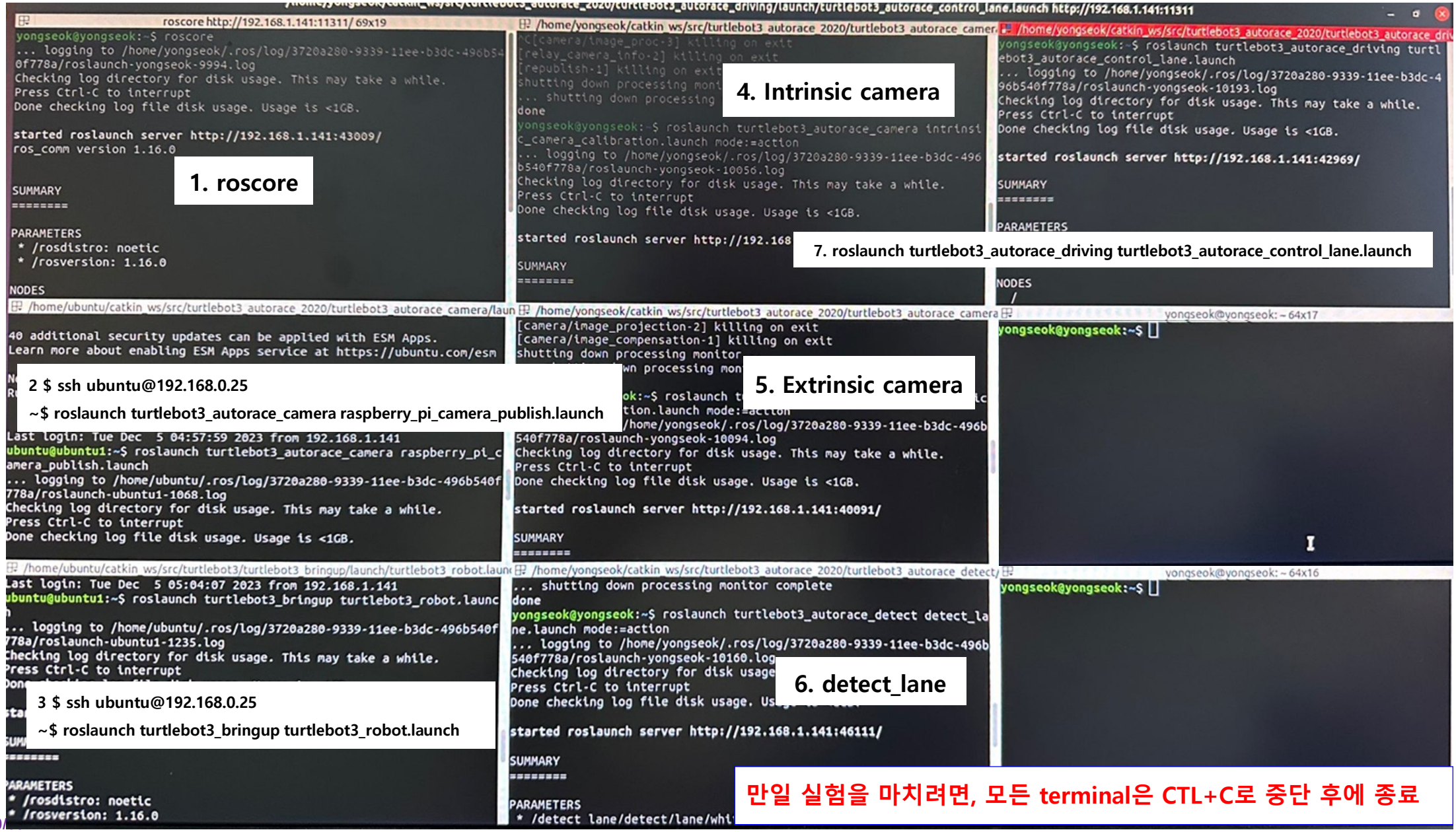
$ roslaunch turtlebot3_autorace_detect detect_lane.launch mode:=action



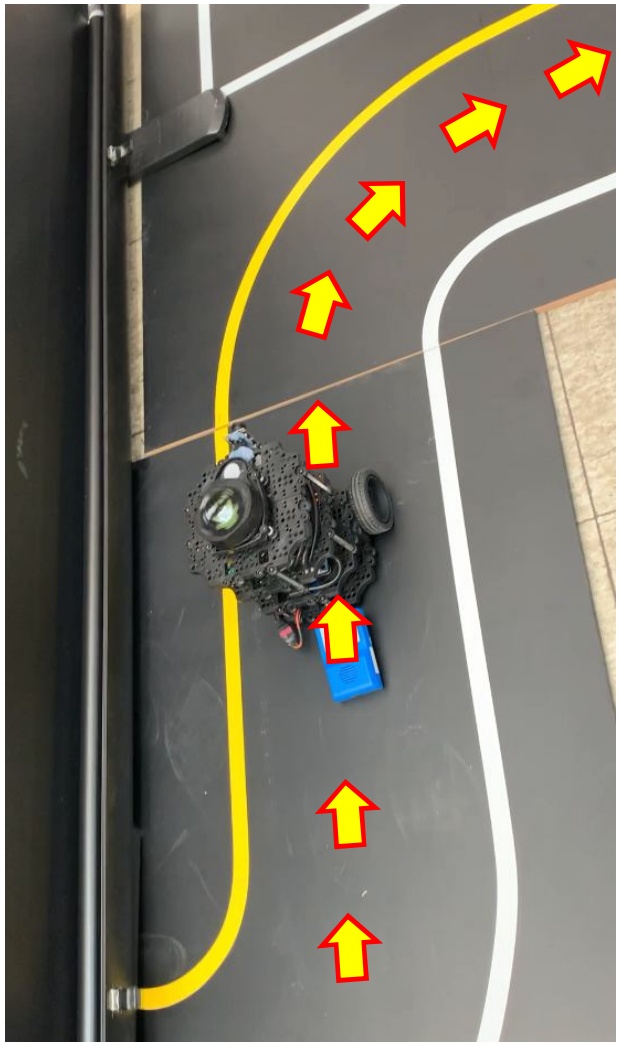
Ⓖ **Remote PC에서 새 터미널 열기**

$ roslaunch turtlebot3_autorace_driving turtlebot3_autorace_control_lane.launch

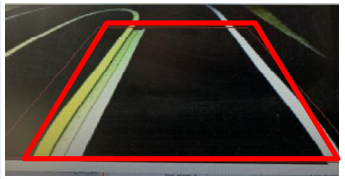
Turtlebot은 lane을 detecting하며 자동으로 이동함


1. roscore
2 $ ssh ubuntu@192.168.0.25
~$ roslaunch turtlebot3_autorace_camera raspberry_pi_camera_publish.launch
3 $ ssh ubuntu@192.168.0.25
~$ roslaunch turtlebot3_bringup turtlebot3_robot.launch
4. Intrinsic camera
5. Extrinsic camera
6. detect_lane
7. roslaunch turtlebot3_autorace_driving turtlebot3_autorace_control_lane.launch
만일 실험을 마치려면, 모든 terminal은 CTL+C로 중단 후에 종료

Detect lane 결과 : 자율주행 진행 : 명령을 사용하면 잠시 후 터틀봇이 실행. 터틀봇을 멈추고 싶을 경우 Open CR 리셋 버튼



장소가 바뀌면
=calibration 수행 후
=action 으로



만일 실행이 되지 않으면, 원인 - turtlebot 위치가 바뀜

: Extrinsic camera calibration 수행 후, =action 으로

만일 실험을 마치려면, 모든 terminal은 CTL+C로 중단 후에 종료

Turtlebot은 sudo shutdown now로 종료

8-3. Lane detection 최종 실습 (터미널창 미리 8개 띄우기) – 방법2 (turtlebot3 정상 이동)

ⓐ Remote PC에서 새 터미널 열기

$ roscore

ⓑ 새 터미널 열기

$ ssh ubuntu@192.168.0.25 ← Remote PC에서 TurtleBot3 Burger(Raspberry pi)을 원격접속

pw : turtlebot 입력

~$ roslaunch turtlebot3_autorace_camera raspberry_pi_camera_publish.launch

ⓒ Remote PC에서 새 터미널 열기

$ roslaunch turtlebot3_autorace_camera intrinsic_camera_calibration.launch mode:=action

ⓓ Remote PC에서 새 터미널 열기

$ roslaunch turtlebot3_autorace_camera extrinsic_camera_calibration.launch mode:=action

ⓕ Remote PC에서 새 터미널 열기

$ roslaunch turtlebot3_autorace_detect detect_lane.launch mode:=action

Ⓖ Remote PC에서 새 터미널 열기

$ roslaunch turtlebot3_autorace_driving turtlebot3_autorace_control_lane.launch

ⓑ 새 터미널 열기

$ ssh ubuntu@192.168.0.25

~$ roslaunch turtlebot3_bringup turtlebot3_robot.launch

감사합니다