Ros에 대한 기본 개념은 필수적으로 공부가 된 상태여야 합니다. 그 다음에 뒷장부터 보세요.

- Package, node, topic, msg, 등 ros에서 쓰는 용어와 그것들이 어떤 역할을 하는지
- https://wiki.ros.org/ko/ROS/Tutorials
- 위 url 들어가서 아래 사진의 8번까지는 꼭 한번씩 해보시기 바랍니다 (얼마안걸림)
- 하다보면 많은 추가 개념이 있지만
- 나머지는 하면서 배워도 될 것 같아요.

1. 초보자 수준

ROS Tutorial Video Demos at ANU

1. ROS 환경 설치와 설정
   이 자습서는 ROS의 설치와 설정 방법에 대해 보여줍니다.
2. ROS 파일시스템의 탐색
   이 자습서는 ROS의 파일시스템 개념과, 명령행 도구인 roscd, rosls, rospack을 다루고 있습니다.
3. ROS 패키지의 작성
   이 자습서는 roscreate-pkg 또는 catkin을 이용해 새로운 패키지를 작성하는 방법에 대해 설명합니다. 또한 rosp
   이용해 패키지의 의존성을 확인하는 법도 알아봅니다.
4. catkin 환경에서 작업공간 만들기
   이 자습서는 catkin 작업공간을 어떻게 설정하는지에 대하여 다루고 있습니다.
5. ROS Package 빌드하기
   이 자습서는 패키지를 빌드하는 툴체인들을 다루고 있습니다.
6. ROS Node 이해하기
   이 자습서는 ROS graph의 개념을 소개하고 roscore, rosnode, rosrun을 다루고 있습니다.
7. ROS Topics 이해하기
   This tutorial introduces ROS topics as well as using the rostopic and rqt_plot commandline tools.
8. ROS Service와 Parameter 이해하기
   이 예제는 ROS service와 parameter를 소개할뿐만 아니라 rosservice와 rosparam 명령어 도구를 소개합니다.

carla_ctl

- carla_ctl 노드에 대한 정보
- /carla_ctl/back_view_topic – ?
- /carla_ctl/bird_view_topic – ?
- Imu 는 관성센서
- Scan은 Lidar 센서
- Usb_cam/image_raw 는 카메라
- 구독하고 있는 xycar_motor에 토픽을 쏴서 제어하는 것입니다.

```
doldolmeng2@DESKTOP-LGJVU1S:~/deb$ rosnode info carla_ctl
--------------------------------------------------------------------------------
Node [/carla_ctl]
Publications:
 * /carla_ctl/back_view_topic [sensor_msgs/Image]
 * /carla_ctl/bird_view_topic [sensor_msgs/Image]
 * /imu [sensor_msgs/Imu]
 * /rosout [rosgraph_msgs/Log]
 * /scan [sensor_msgs/LaserScan]
 * /usb_cam/image_raw [sensor_msgs/Image]

Subscriptions:
 * /xycar_motor [unknown type]

Services:
 * /carla_ctl/get_loggers
 * /carla_ctl/set_logger_level
```

carla_ctl Topic

Imu 관성 센서 정보

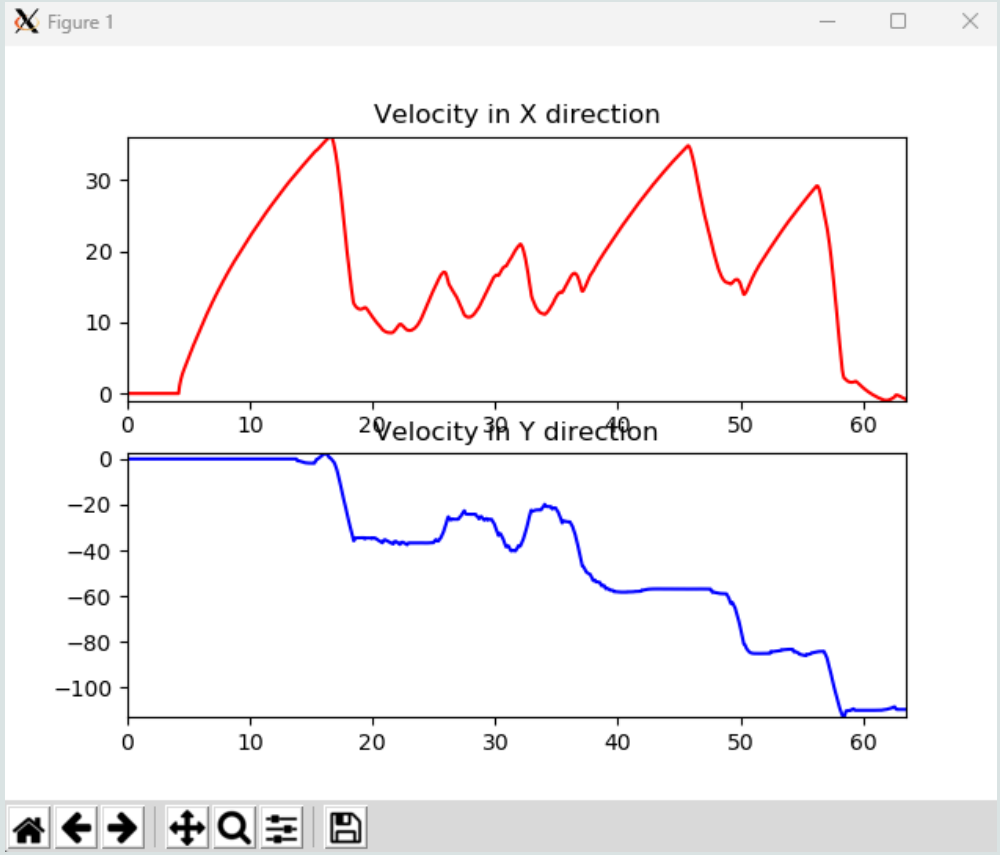
- Seq : 시퀀스 번호
- Stamp 시간 (초, 나노초)
- Orientation : 현재 방향
- angular_velocity : 각 속도
- linear_acceleration : 선 가속도 (속도 변화)

- 각 측정값의 covariance 는 3x3 공분산 행렬로,
  대각선 원소들은 각각 x, y, z 축의 불확실성을,
  나머지 원소들은 축 사이의 상관관계를 나타낸다.

```
---
header:
  seq: 3709
  stamp:
    secs: 1716916548
    nsecs: 206301020
  frame_id: "imu_link"
orientation:
  x: 8.905007874551927e-07
  y: -1.6507007588917203e-05
  z: 0.004057984747855352
  w: 0.9999917662093583
orientation_covariance: [0.0001, 0.0, 0.0, 0.0, 0.0001, 0.0, 0.0, 0.0, 0.0001]
angular_velocity:
  x: 0.0
  y: 0.0
  z: 0.0
angular_velocity_covariance: [1e-06, 0.0, 0.0, 0.0, 1e-06, 0.0, 0.0, 0.0, 1e-06]
linear_acceleration:
  x: 0.039679754525423 05
  y: 0.0002841840323526412
  z: 9.811251640319824
linear_acceleration_covariance: [1e-05, 0.0, 0.0, 0.0, 1e-05, 0.0, 0.0, 0.0, 1e-05]
---
header:
  seq: 3710
  stamp:
    secs: 1716916548
    nsecs: 218127177
  frame_id: "imu_link"
orientation:
  x: 8.905007874551927e-07
  y: -1.6507007588917203e-05
  z: 0.004057984747855352
  w: 0.9999917662093583
orientation_covariance: [0.0001, 0.0, 0.0, 0.0, 0.0001, 0.0, 0.0, 0.0, 0.0001]
angular_velocity:
  x: 0.0
  y: 0.0
  z: 0.0
angular_velocity_covariance: [1e-06, 0.0, 0.0, 0.0, 1e-06, 0.0, 0.0, 0.0, 1e-06]
linear_acceleration:
  x: -0.010328679345548153
  y: -0.0005599766154773533
  z: 9.809375762939453
linear_acceleration_covariance: [1e-05, 0.0, 0.0, 0.0, 1e-05, 0.0, 0.0, 0.0, 1e-05]
---
```

carla_ctl Topic

관성센서로 속도 구하기

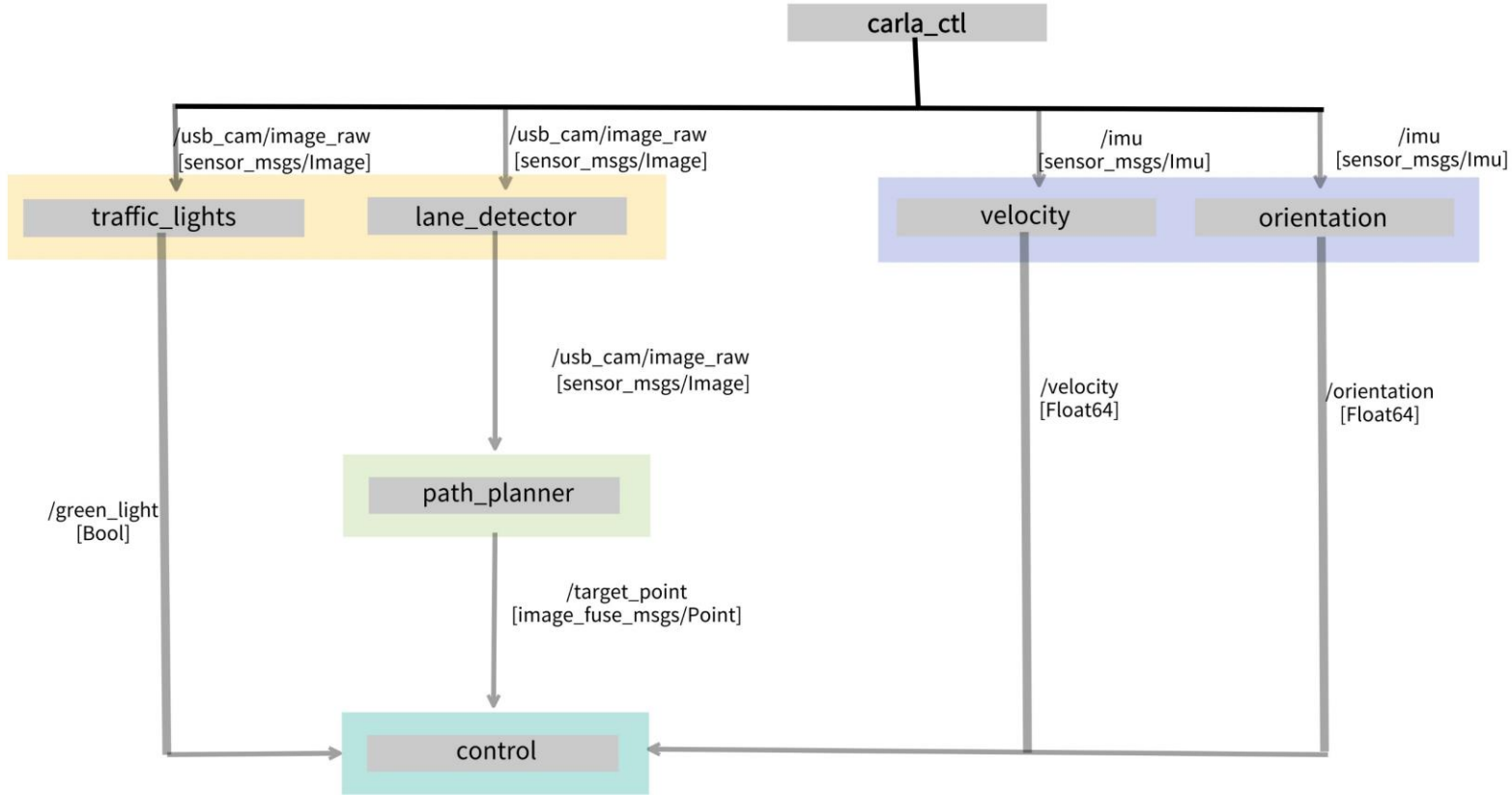


https://www.youtube.com/watch?v=nVPzLxtJPBQ

Usb_cam/image_raw

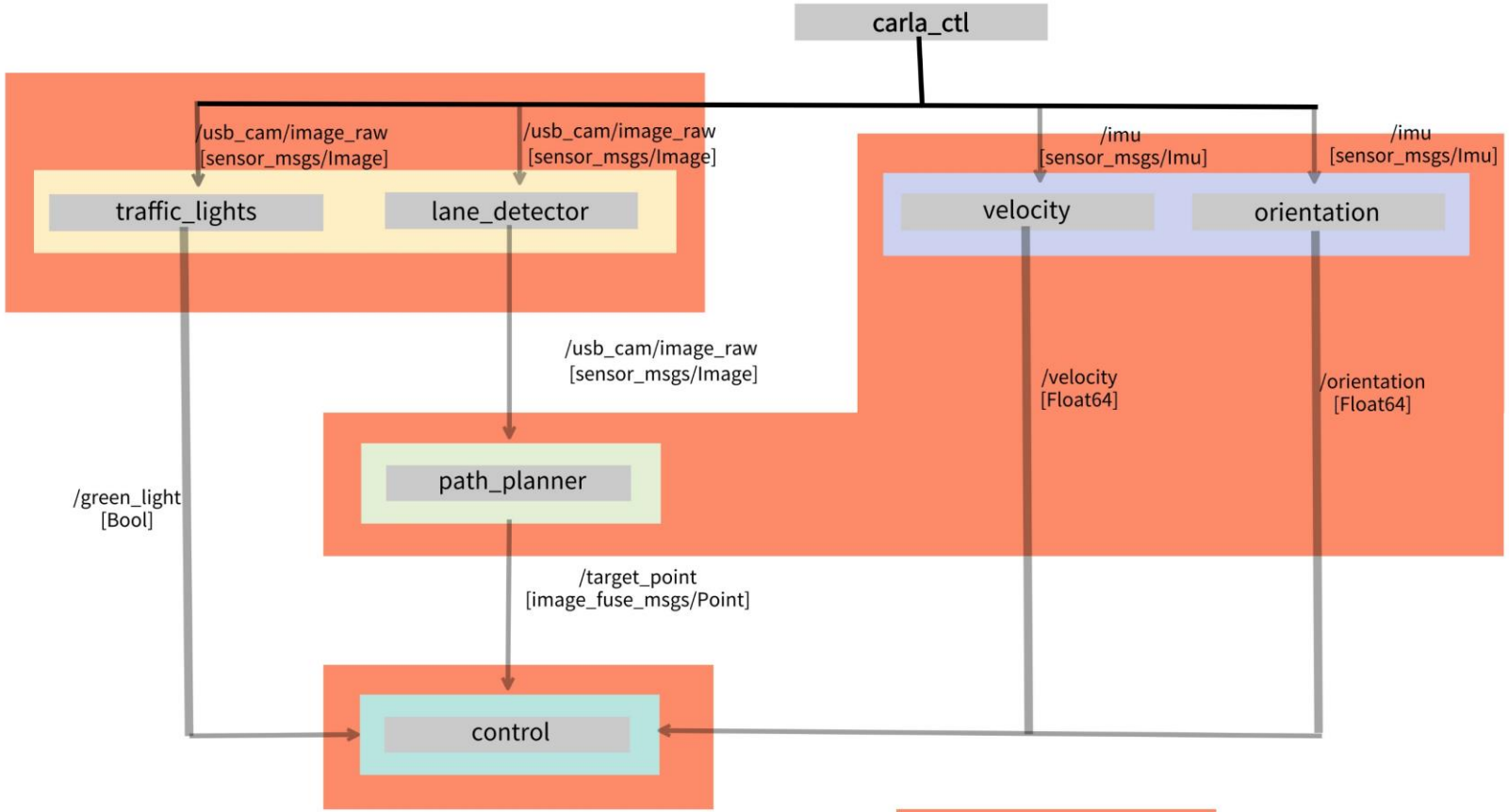
- 카메라의 데이터.

```
Header header        # Header timestamp should be acquisition time of image
                     # Header frame_id should be optical frame of camera
                     # origin of frame should be optical center of camera
                     # +x should point to the right in the image
                     # +y should point down in the image
                     # +z should point into to plane of the image
                     # If the frame_id here and the frame_id of the CameraInfo
                     # message associated with the image conflict
                     # the behavior is undefined

uint32 height A       # image height, that is, number of rows
uint32 width         # image width, that is, number of columns

# The legal values for encoding are in file src/image_encodings.cpp
# If you want to standardize a new string format, join
# ros-users@lists.sourceforge.net and send an email proposing a new encoding.

string encoding      # Encoding of pixels -- channel meaning, ordering, size
                     # taken from the list of strings in include/sensor_msgs/image_encodings.h

uint8 is_bigendian   # is this data bigendian?
uint32 step          # Full row length in bytes
uint8[] data         # actual matrix data, size is (step * rows)
```



/usb_cam/image_raw → 토픽 이름

[sensor_msgs/Image] → 데이터 form



작업 영역

package

image_fuse

path_plan

imu_fuse

control

/usb_cam/Image_raw → 토픽 이름

[sensor_msgs/Image] → 데이터 form

노드 설명

traffic_lights

- BRG -> HSV 변환
- 초록불이 되면
- /green_light topic에
- True를 20초간 publishing
  (그 이후 노드 종료)

HSV/YCbCr 색상 모델과 형태적 특징 기반 교통신호등 인식

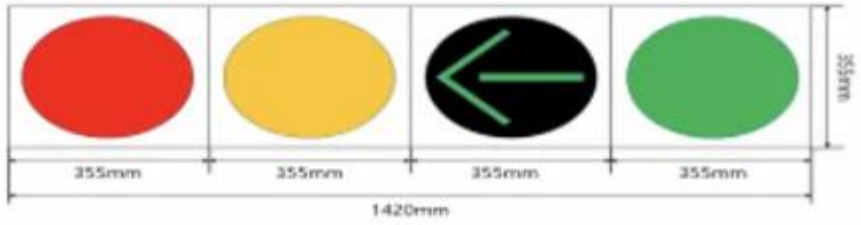
Table 1 Threshold values of each color light in the HSV color model

	H	S	V
적색	0~10 or 150~180	80~255	80~255
황색	11~30	100~255	80~255
녹색	60~100	75~255	80~255

Table 2 Threshold values of each color light in the YCbCr color model

	Y	Cb	Cr
적색	0~190	100~150	155~240
황색	100~210	60~100	140~180
녹색	150~240	115~145	60~110

H: 색상, S: 채도, V: 명도



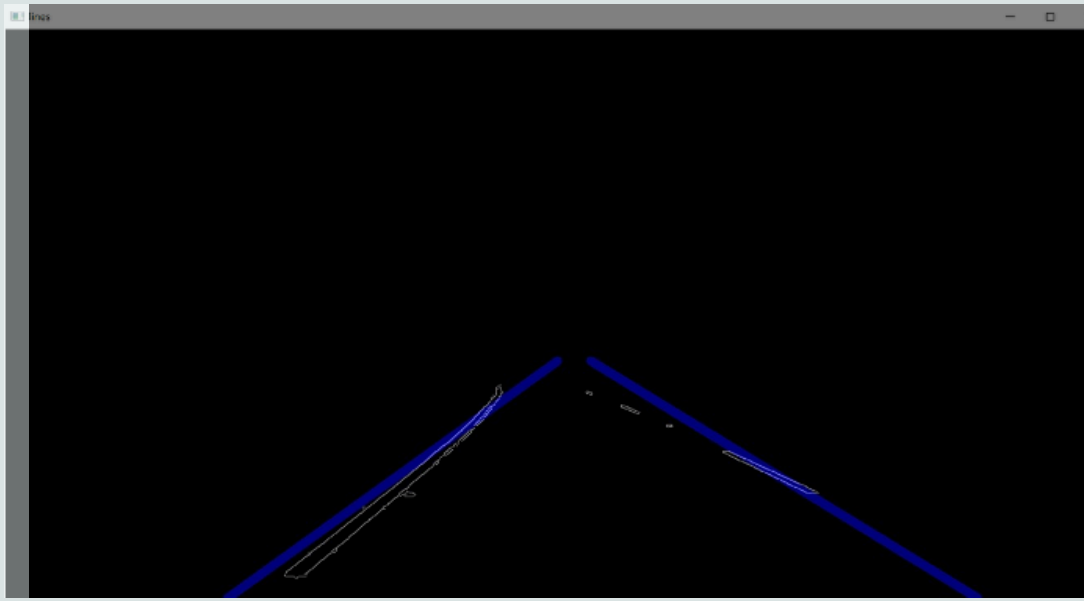
노드 설명

lane_detector

- https://codingwell.tistory.com/60
- https://webnautes.tistory.com/1244#google_vignette
- 위 링크 참고해서
- 우리는 중앙에 있는 점선까지 인식
- Subscribe -> usb_cam
- Publish ->오른쪽 그림과 같은 차선 '영상 이미지'

노드 설명

path_planner



- lane_detector 에서 받은 이미지로 직선을 검출하여 접점 구하기(lane_detector url 참고, 직선 추출까지 나와있음)
- 접점이 곧 목표 지점이 됨, 중앙에 점선 잘 이용
- 왼쪽 위에 좌표가 (0,0)이고 아래로, 오른쪽으로 갈수록 +
- Subscribe -> usb_cam 영상
- Publish -> image_fuse_msg (heigh, width, x, y)

velocity, orientation 각각 2개의 노드

- Imu 관성 센서 값 -> 직선 가속도, 각 가속도, 방향 정보를 가지고
- 현재 속도, 각 속도, 방향(각도)를 처리함
- Subscribe -> imu [sensor_msgs/Imu]
- Publish -> velocity [float64]
- Publish -> orientation [float64]

control

- 신호등 신호시 출발
- PID제어
- 목표 지점(좌표 지점), 속도, 방향등을 계산해 주행
- Subcribe -> 신호등, 목표지점, 관성센서 정보
- Publish -> 제어값 speed, angle