LiDAR End-to-End Information Map

김해린
조 일
서야결
유정근

목차

Part 1

프로젝트 개요

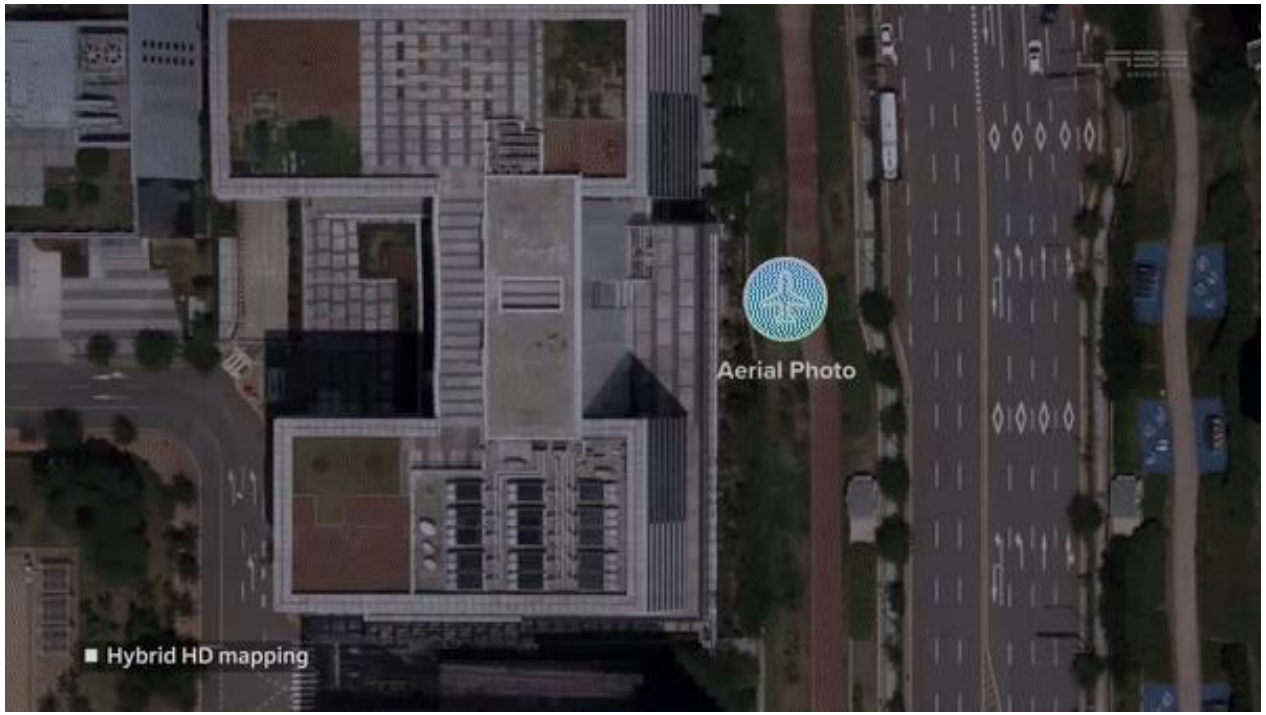
KITTI dataset

프로젝트 요약

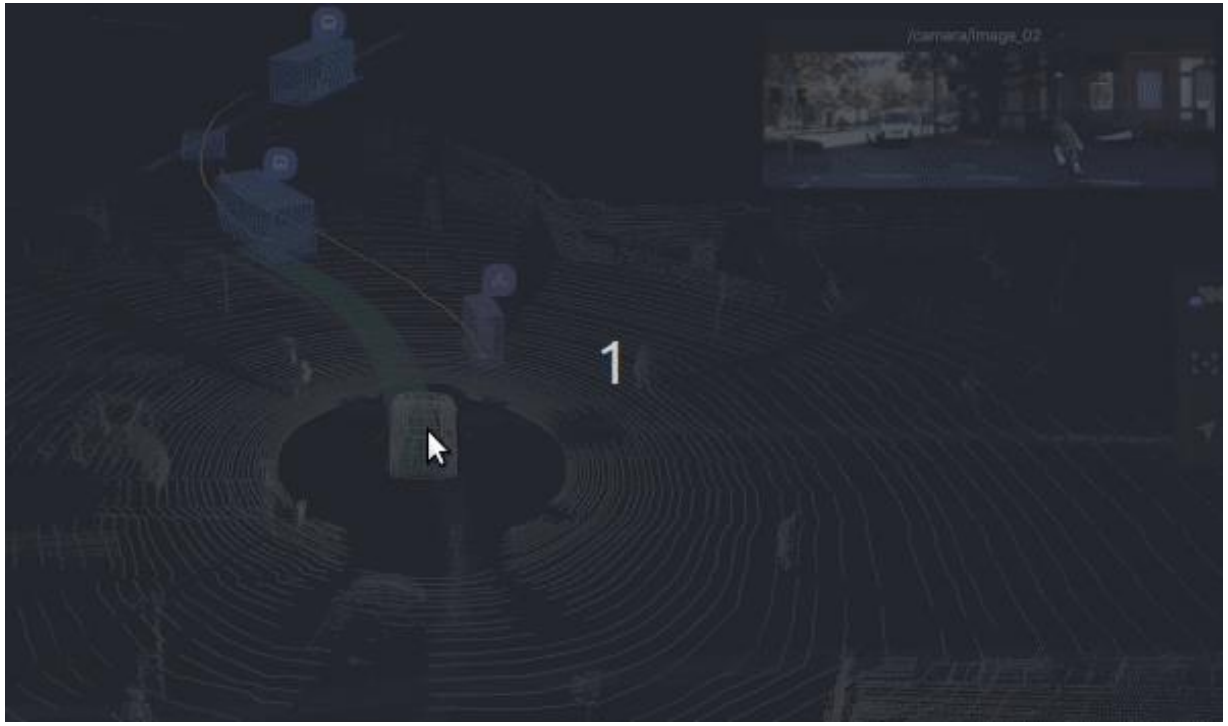
팀원 소개

프로젝트 개요

Self-driving car Visualization



Hybrid HD Mapping, A Map for Self-Driving Vehicles
FacebookTwitter2018.10.12 NAVER LABS



Uber and GM Cruise are making their respective AV 'visualization' tools open source
By Andrew J. Hawkins@andyjayhawk Feb 19, 2019, 9:00am EST

KITTI DATASET

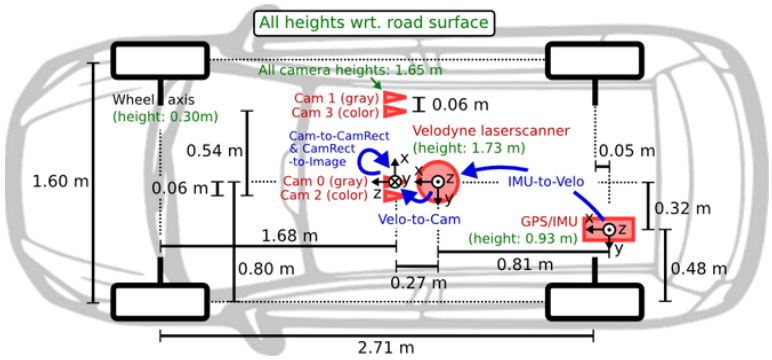
출처

Vision meets Robotics: The KITTI Dataset

Andreas Geiger, Philip Lenz, Christoph Stiller and Raquel Urtasun

↳ 3350회 인용

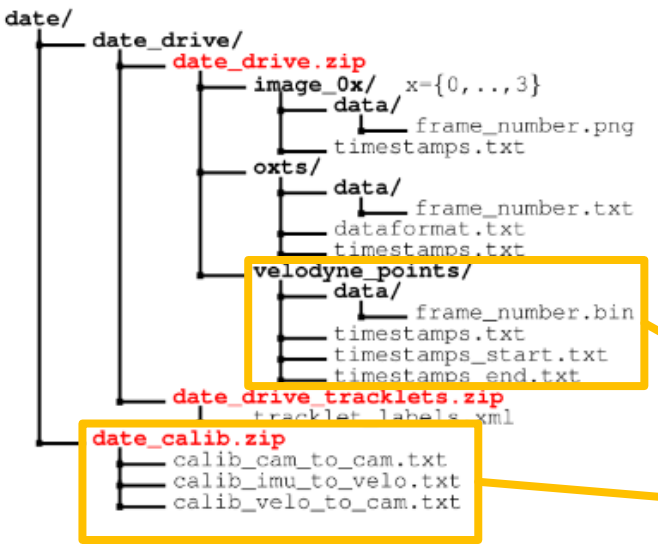
left-hand coordinate system



What information in the kitti-dataset	Sensor Setup
Raw (unsynced+unrectified) and processed (synced+rectified) grayscale stereo sequences	Grayscale cameras, 1.4 Megapixels: Point Grey Flea 2 (FL2-14S3M-C)
Raw (unsynced+unrectified) and processed (synced+rectified) color stereo sequences	Color cameras, 1.4 Megapixels: Point Grey Flea 2 (FL2-14S3C-C)
Calibration	
3D GPS/IMU data	Inertial Navigation System (GPS/IMU): OXTS RT 3003
3D Velodyne point clouds	Laserscanner: Velodyne HDL-64E
3D object tracklet labels	
Training labels of object data set	

HOW WE USE THE KITTI DATASET

- Change Kitti to Rosbag

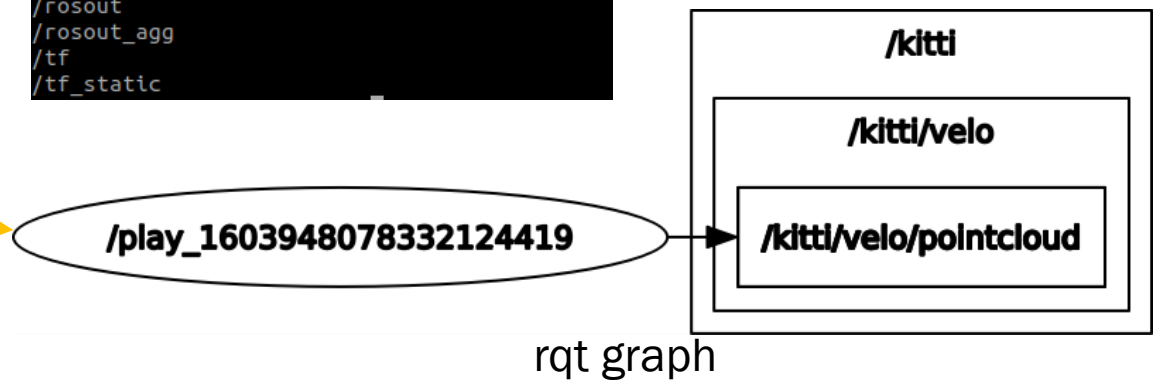


Vision meets Robotics: The KITTI Dataset

Andreas Geiger, Philip Lenz, Christoph Stiller and Raquel Urtasun

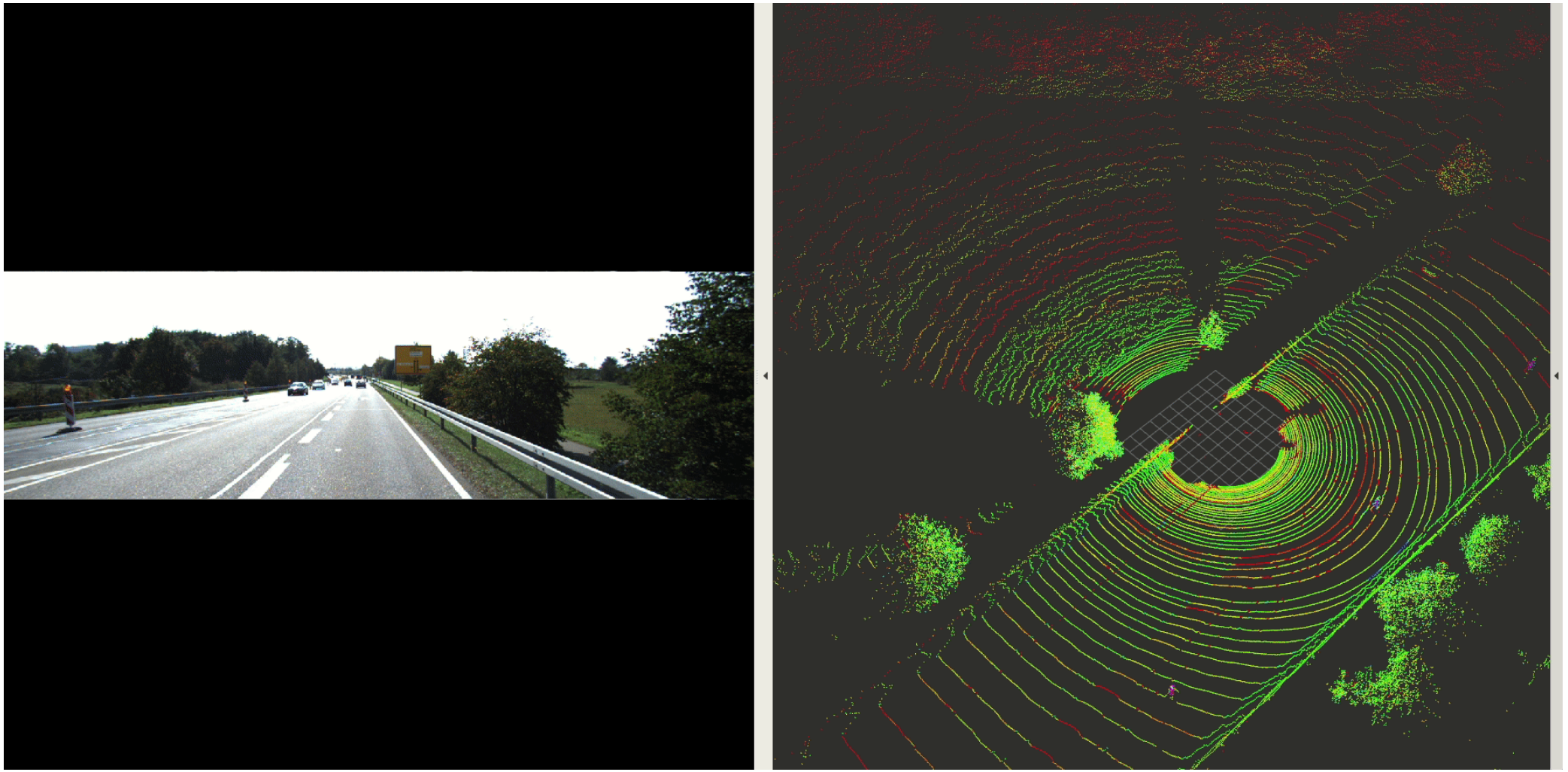
Ros Topic list

```
/clock
/kitti/camera_color_left/camera_info
/kitti/camera_color_left/image_raw
/kitti/camera_color_right/camera_info
/kitti/camera_color_right/image_raw
/kitti/camera_gray_left/camera_info
/kitti/camera_gray_left/image_raw
/kitti/camera_gray_right/camera_info
/kitti/camera_gray_right/image_raw
/kitti/oxts/gps/fix
/kitti/oxts/gps/vel
/kitti/oxts/imu
/kitti/velo/pointcloud
/rosout
/rosout_agg
/tf
/tf_static
```



rqt graph

HOW WE USE THE KITTI DATASET

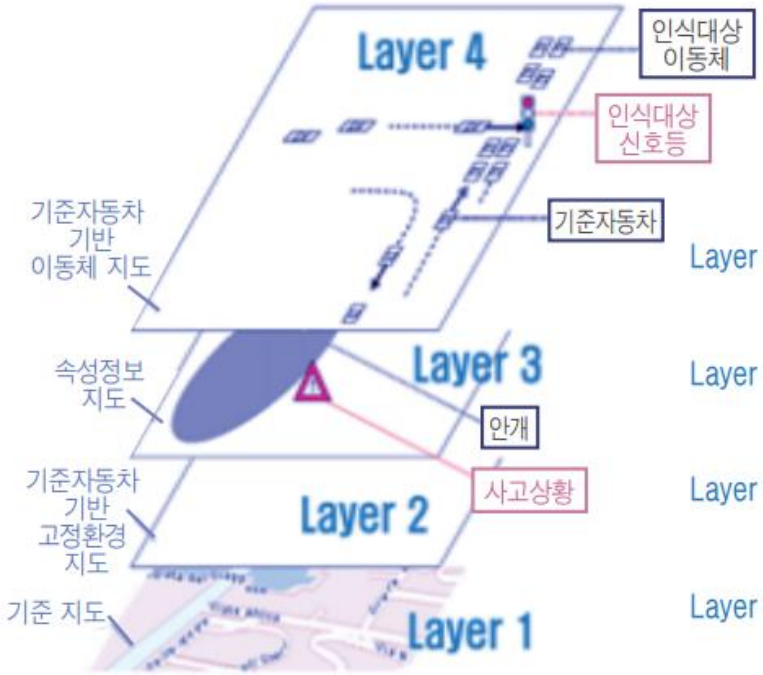


Camera image and 64 channel Velodyne laser scanner

프로젝트 요약

HD Map

•고정밀 지도(HD 지도)란 자율주행을 위해 센티미터(cm) 수준의 정밀도를 갖춘 3D 입체 지도
•도로 중심선, 경계선 등 차선 단위의 정보는 물론 신호등, 표지판, 연석, 노면마크, 각종 구조물 등의 정보가 3차원 디지털로 담긴다.



*referensce: LDM [Local Dynamic Map]

Layer 1 기준지도[고정밀 지상 정지측량]
– 건물, 도로, 연석, 시설물, 나무, 차선, 각종 교통표시 등

Layer 2 기준자동차 기반 고정환경 지도 [이동측량]
– 건물, 도로, 연석, 시설물, 나무, 차선, 각종 교통표시 등

Layer 3 속성정보 지도[속성정보, 운영자 입력]
– 날씨, 안개, 교통상황[사고상황], 신호등 상태 등

Layer 4 기준자동차 기반 인식대상 이동체 지도 [이동측량]
– 차량[대형트럭, 버스, SUV, 세단, 이륜차 등], 사람

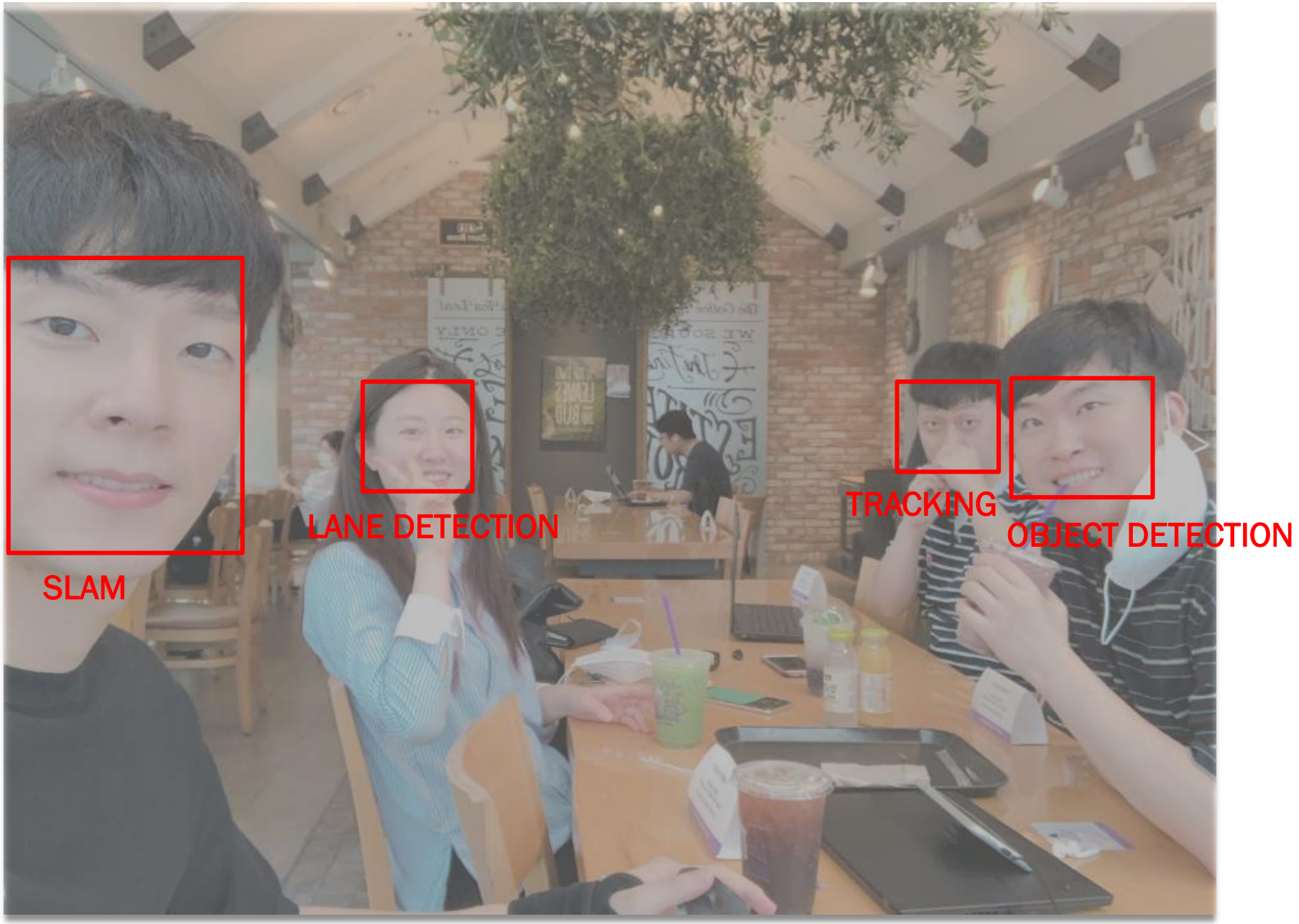
Lidar End-to-End Information Map

SLAM

Lane detection

Object Detection & Tracking

팀원 소개



Part 2

전체 시스템 구성도 & S/W 구성도

SLAM

Lane Detection

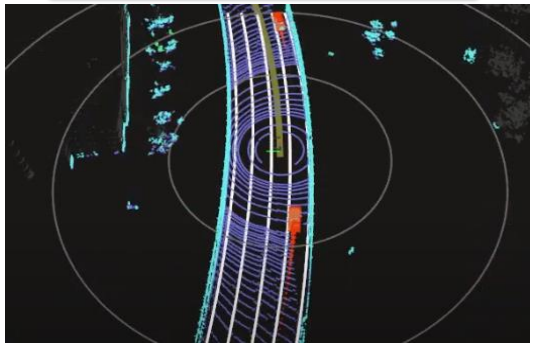
Object Detection

Tracking

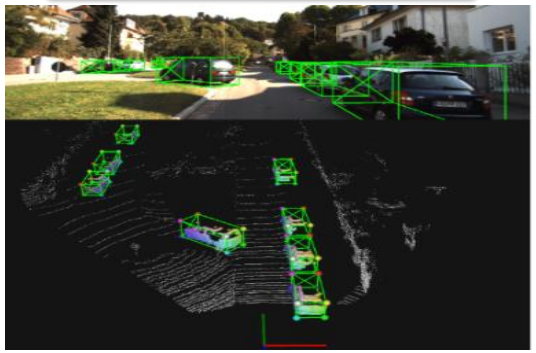
전체 시스템 구성도

KITTI DataSet

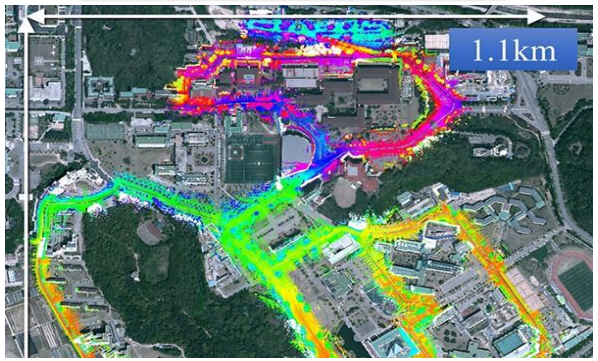
Lane Detection



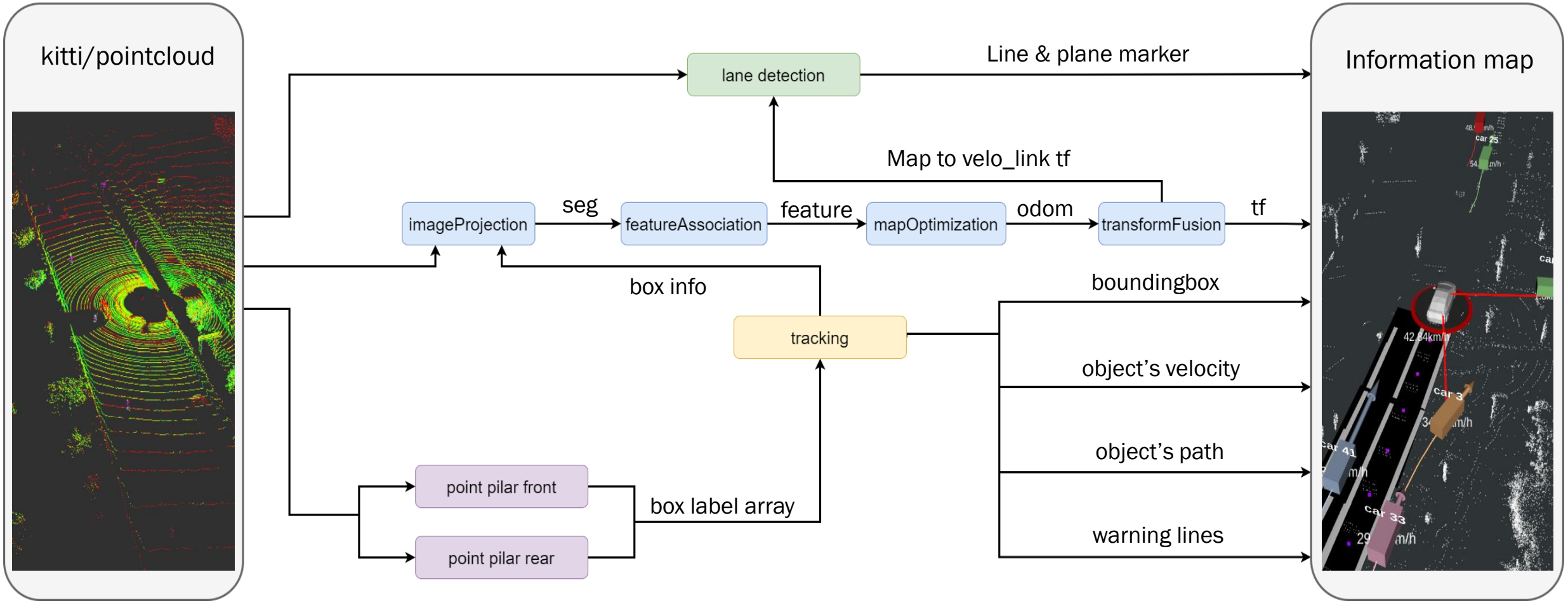
Object Detection, Tracking



SLAM



S/W 구성도


kitti/pointcloud
lane detection
Line & plane marker
Information map
Map to velo_link tf
imageProjection
seg
featureAssociation
feature
mapOptimization
odom
transformFusion
tf
box info
tracking
boundingbox
object's velocity
object's path
warning lines
point pilar front
point pilar rear
box label array

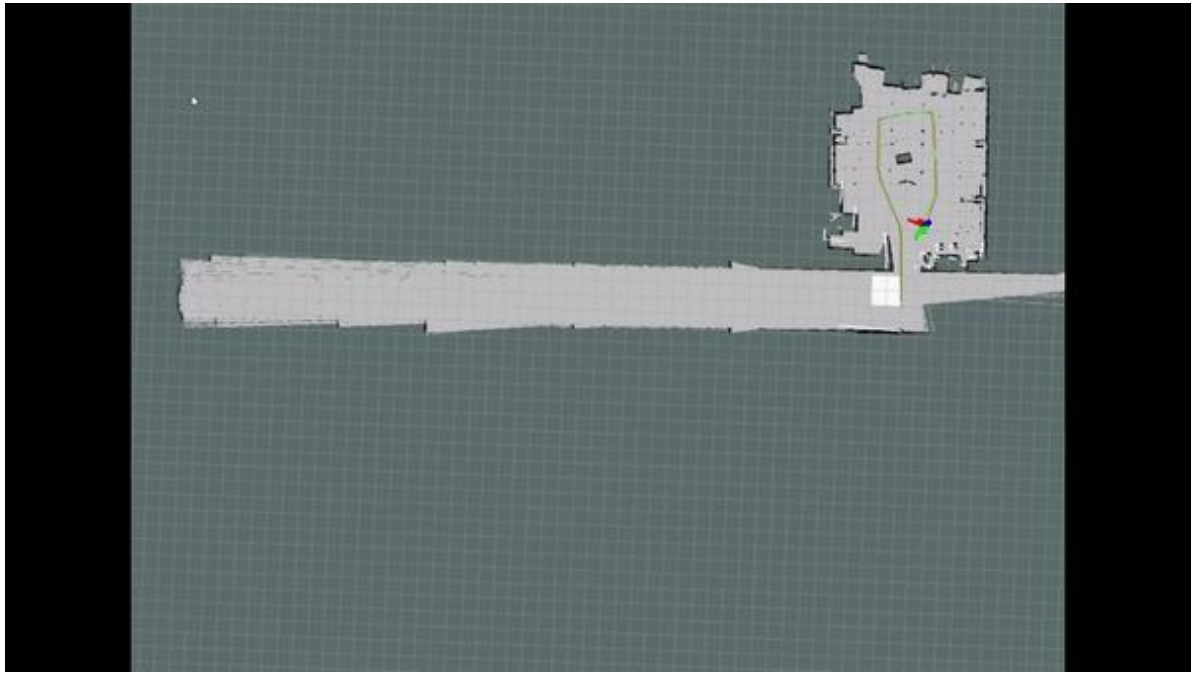
SLAM

- **SLAM(Simultaneous Localization and Mapping)**

  - 주변 환경을 센서로 감지해가며 Map을 만들고 만든 Map에서 자신의 위치를 추정

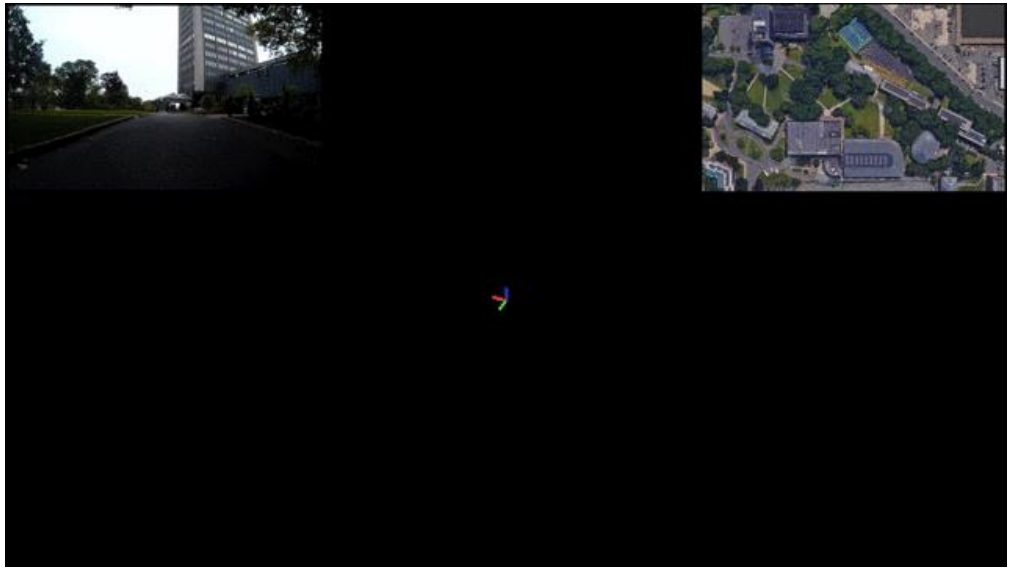
  - 자율 주행 차가 주변 환경을 인식하고 안전하게 자율 주행을 하는데 활용

  - 그 외 증강 및 가상현실 등 엔터테인먼트 분야에서도 사용



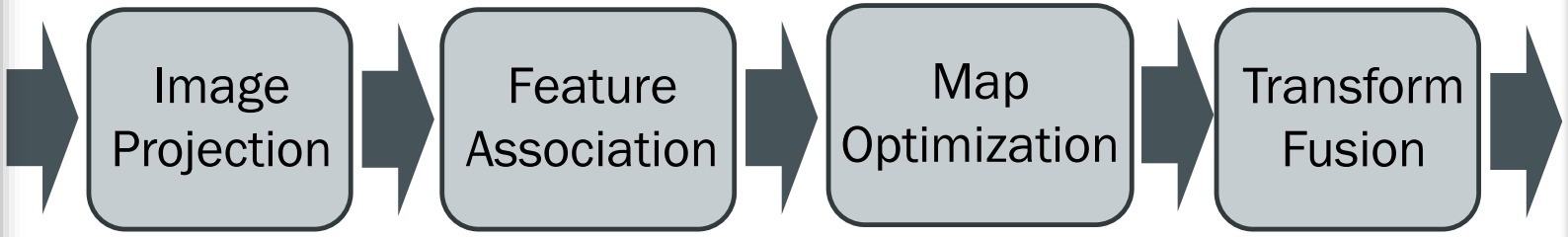
SLAM

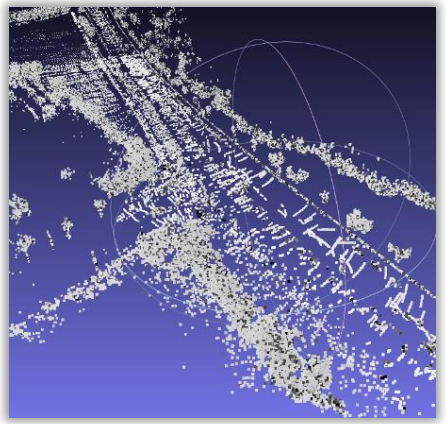
- **LeGO-LOAM**
  - lightweight and ground optimized lidar odometry and mapping라는 뜻의 경량 SLAM 알고리즘
  - Top down 방식 SLAM 알고리즘 스터디
  - Input : LiDAR
  - Output : 6 degree-of-freedom pose estimation (translation, rotation 값)



- **LeGO-LOAM Overview**

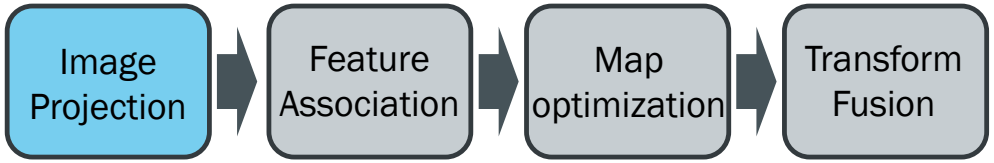
3D Point Cloud





SLAM

SLAM

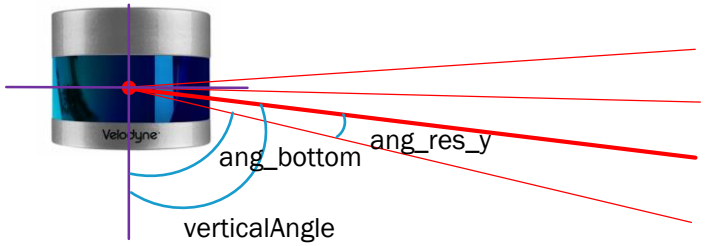


- **imageProjection**

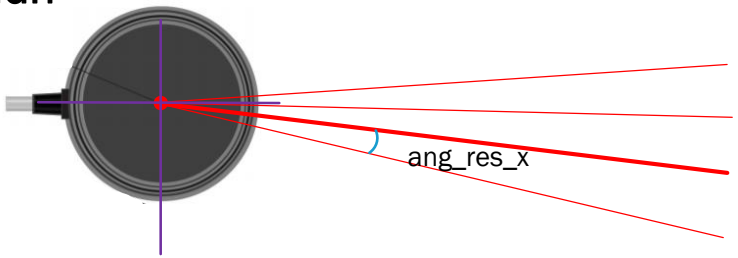
  - 3D -> 2D 좌표 변환, 각 point 거리 값 저장

```
rowIdn = (verticalAngle + ang_bottom) / ang_res_y;
        verticalAngle = atan2(thisPoint.z, sqrt(thisPoint.x * thisPoint.x + thisPoint.y * thisPoint.y)) * 180 / M_PI;
        ang_bottom = 15.0+0.1;
        ang_res_y = 2.0;
columnIdn = -round((horizonAngle-90.0)/ang_res_x) + Horizon_SCAN/2;
        ang_res_x = 0.2;
        Horizon_SCAN = 1800;
range = sqrt(thisPoint.x * thisPoint.x + thisPoint.y * thisPoint.y + thisPoint.z * thisPoint.z);
```

rowIdn



columnIdn

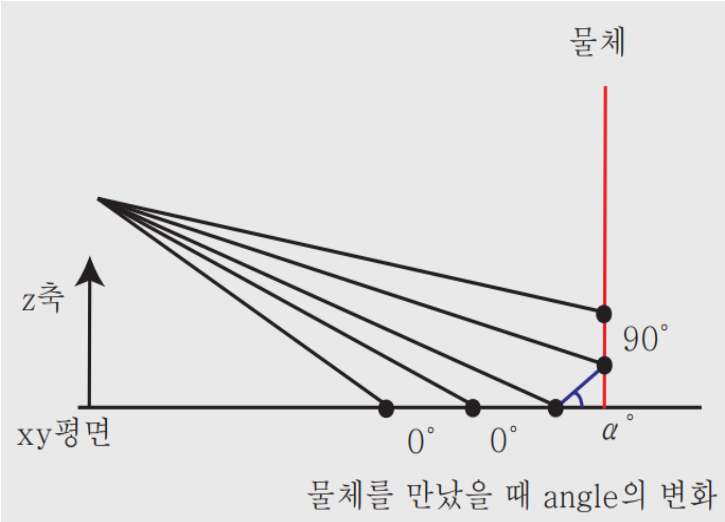


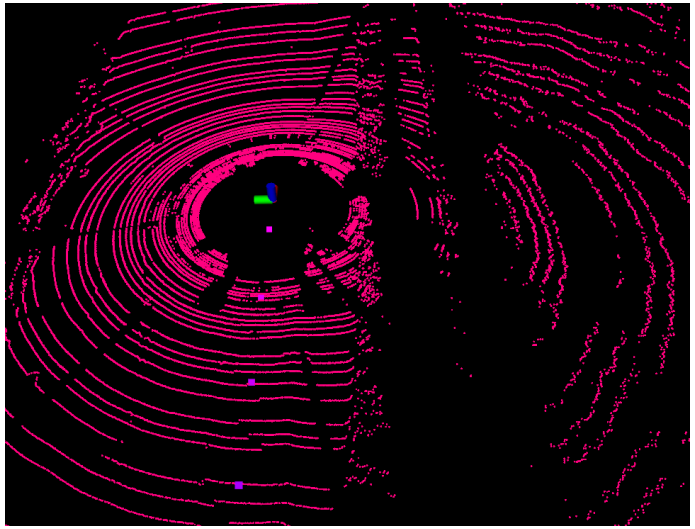
SLAM

Image Projection → Feature Association → Map optimization → Transform Fusion

- **imageProjection**

  - extract ground point

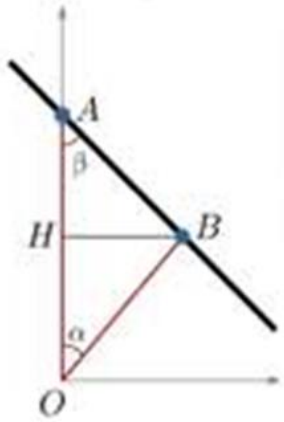


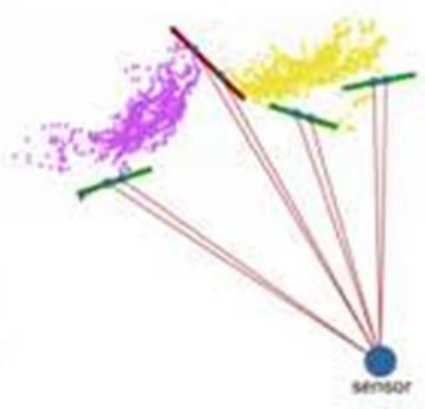


  - segment & labeling

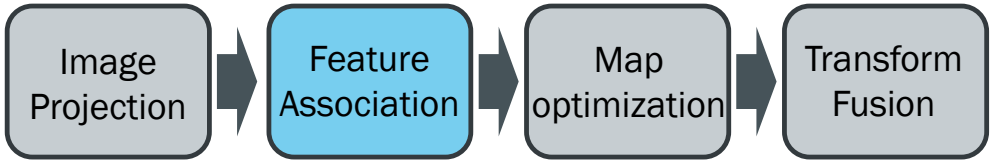
    이웃한 두 점의 range값에 의해 계산된 각도로 segment

    30개 이상의 point들이 segment 되었을 때 labeling





SLAM



- **featureAssociation**

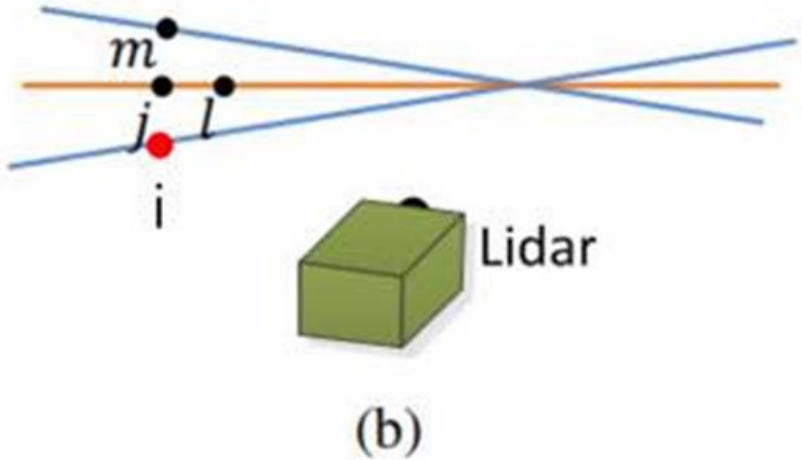
  - extract edge and surf feature

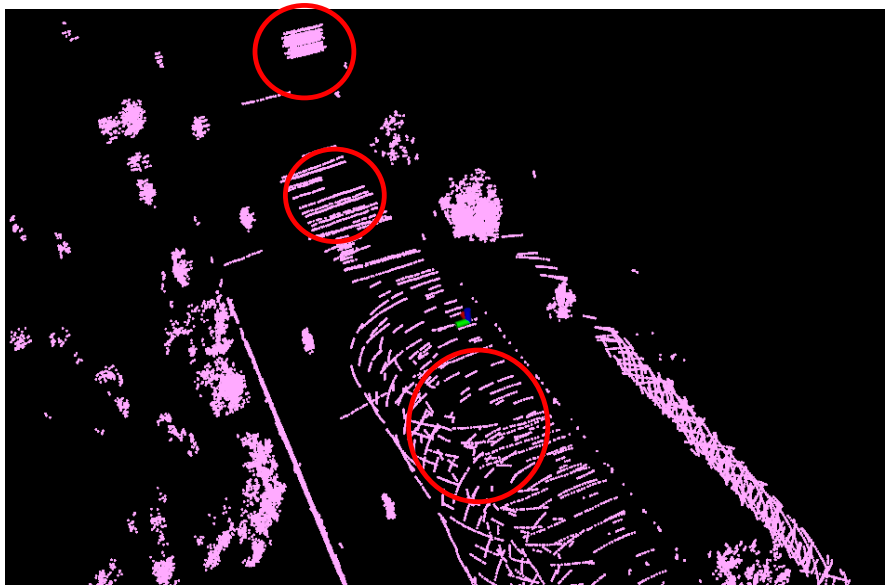
궁금한 점 **i**

i의 위 channel에서 i와 가장 이웃한 점 **j**
j와 같은 channel에서 j와 가장 이웃한 점 **l**

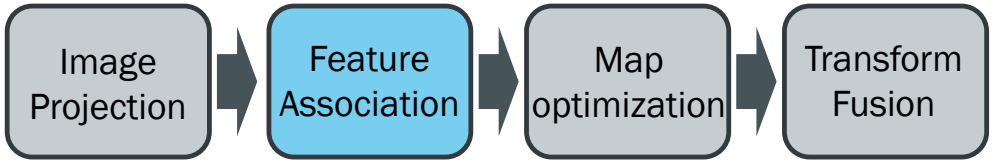
i의 아래 channel에서 i와 가장 이웃한 점 **m**

총 4점으로 평면 방정식 만들어서 surf 추출





SLAM



- **featureAssociation**

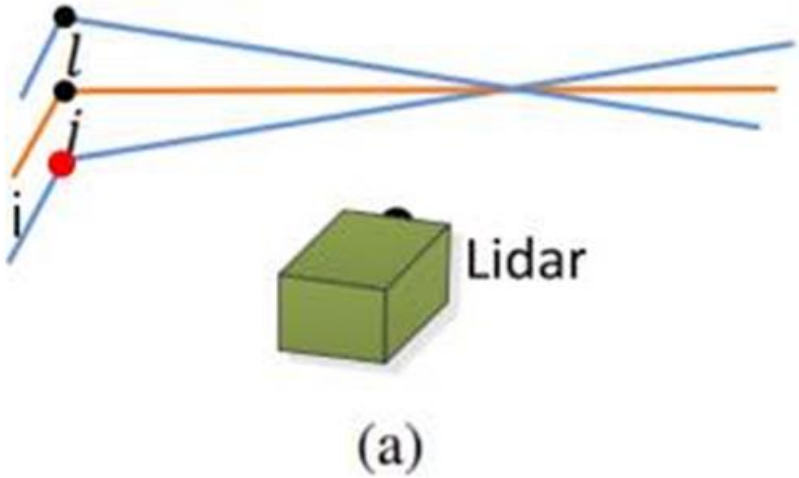
  - extract edge and surf feature

궁금한 점 **i**

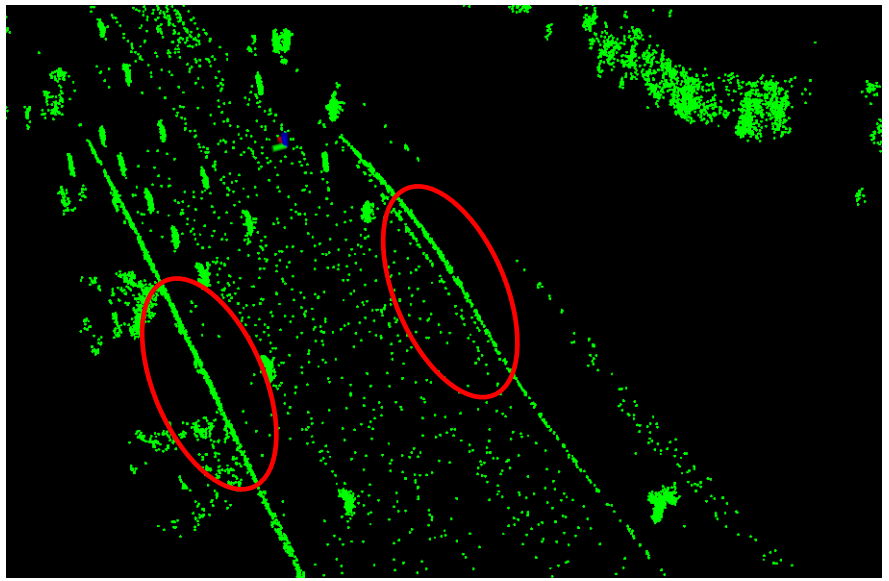
i의 위 channel에서 i와 가장 이웃한 점 **j**

i의 아래 channel에서 i와 가장 이웃한 점 **l**

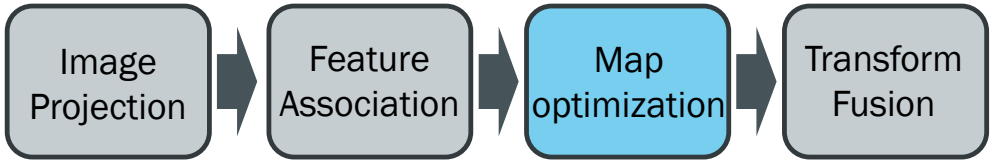
총 3점으로 직선 방정식 만들어서 edge 추출



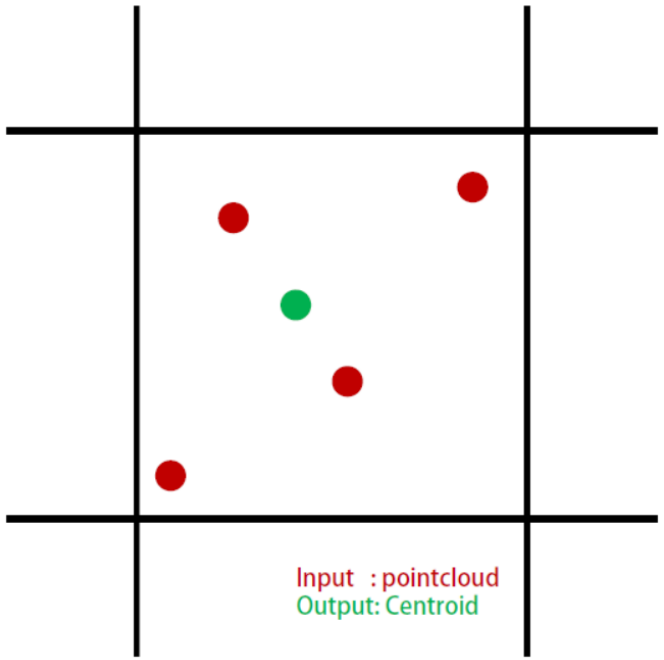
(a)

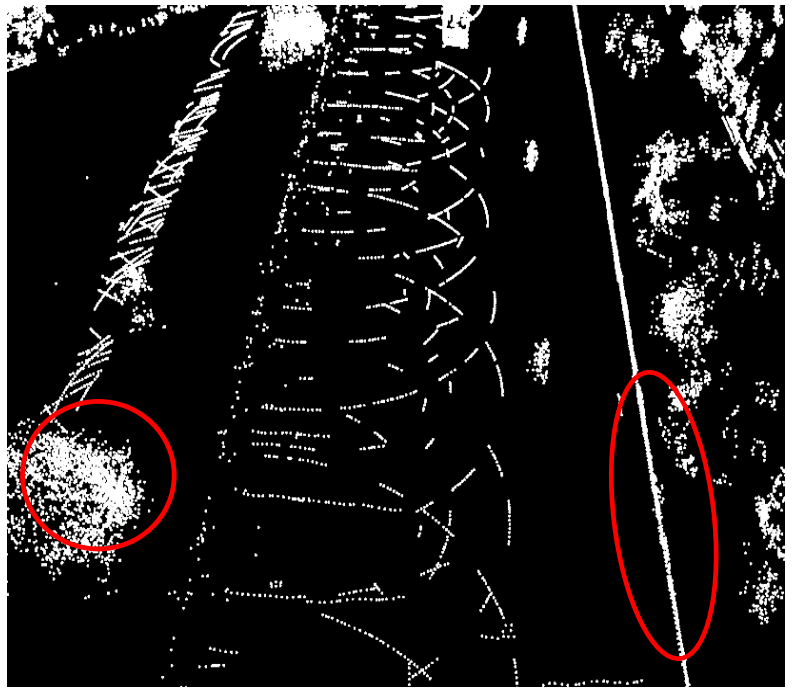


SLAM



- **mapOptimization**
  - downsample edge and corner feature

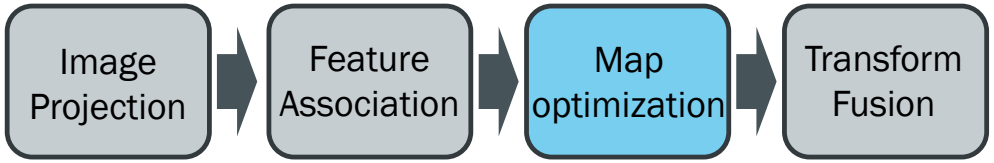




before

after

SLAM



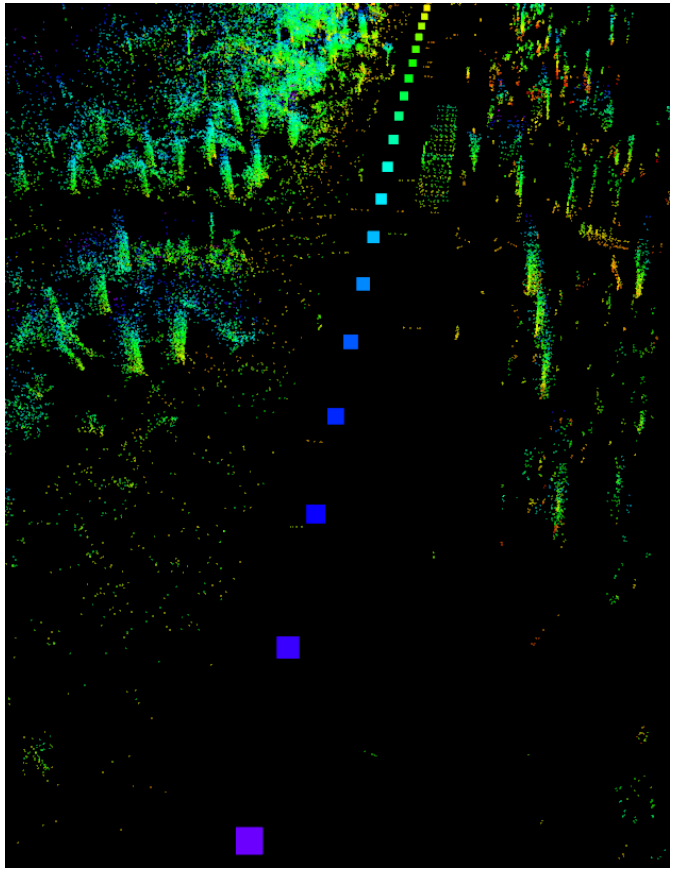
- **mapOptimization**

  - get odometry information

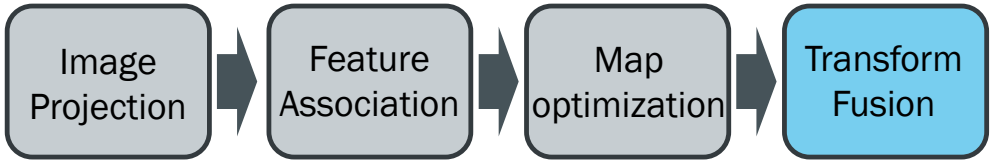
    계산된 Odometry data 추출

```
---
header:
  seq: 63
  stamp:
    secs: 1317011377
    nsecs: 588315010
  frame_id: "/camera_init"
child_frame_id: "/aft_mapped"
pose:
  pose:
    position:
      x: 28.3159122467
      y: -8.27009963989
      z: 367.929382324
    orientation:
      x: 0.0147947205593
      y: 0.0611305199303
      z: -0.00131205391567
      w: 0.998019265491
```

```
---
header:
  seq: 64
  stamp:
    secs: 1317011378
    nsecs:   2885103
  frame_id: "/camera_init"
child_frame_id: "/aft_mapped"
pose:
  pose:
    position:
      x: 28.9347877502
      y: -8.39725780487
      z: 372.495544434
    orientation:
      x: 0.0140013310925
      y: 0.0667706188217
      z: -0.00412920924241
      w: 0.99766156427
```



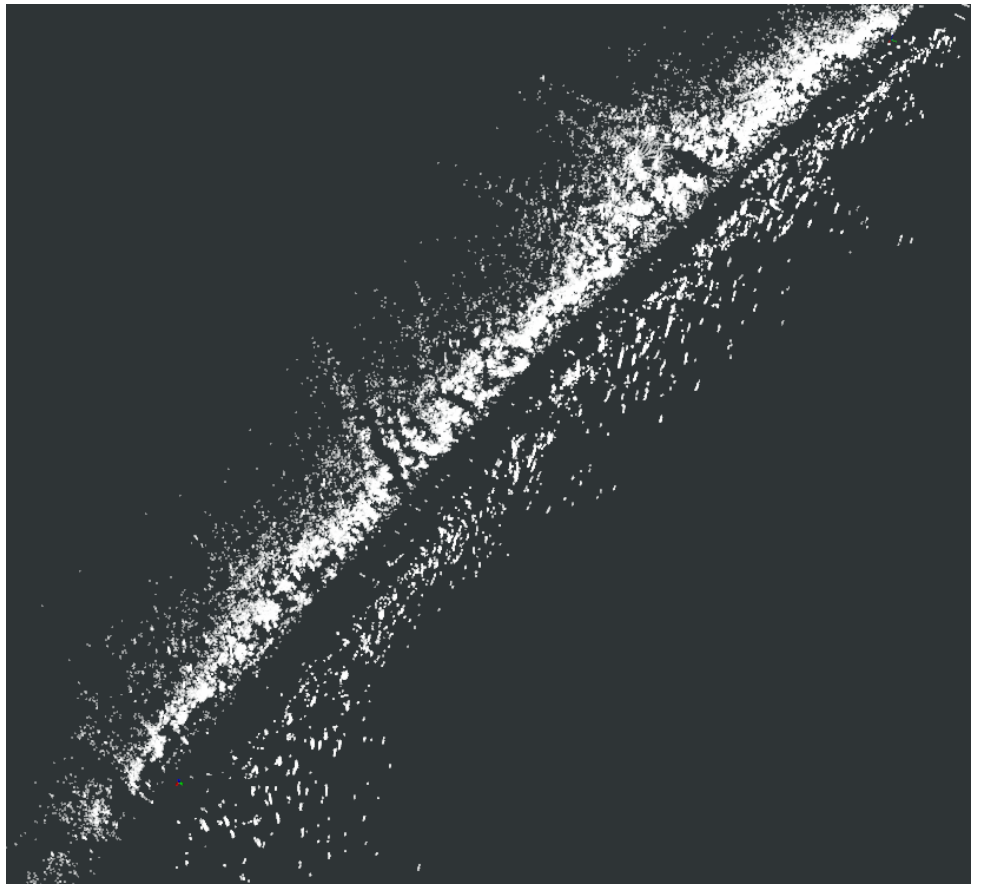
SLAM



- **transformFusion**

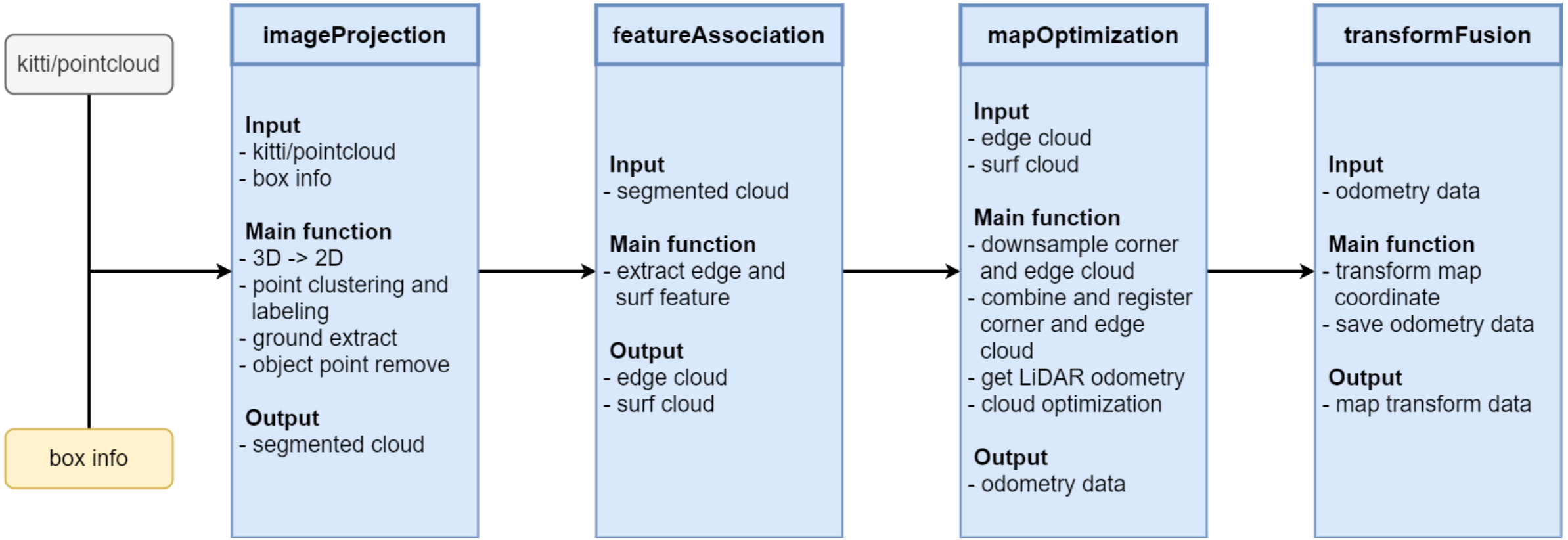
  - transform MAP coordinate

Odometry data를 이용하여
LiDAR sensor coordinate 포인트들을
map coordinate transform 후 map에 포인트 등록



SLAM

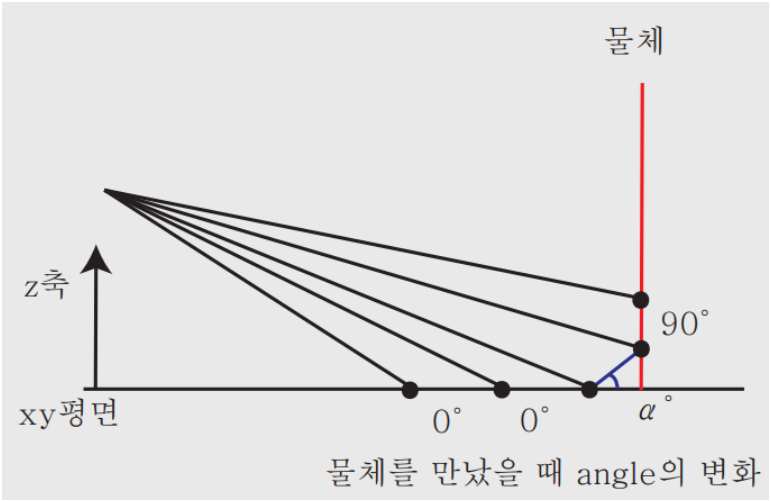
- **Overview**



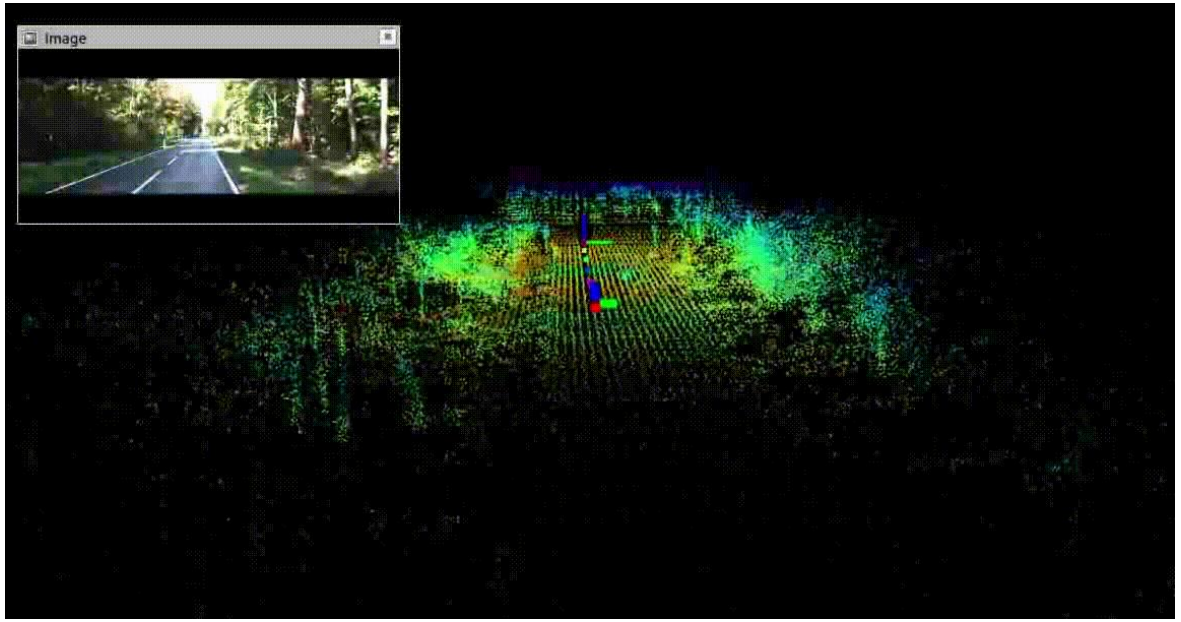
SLAM

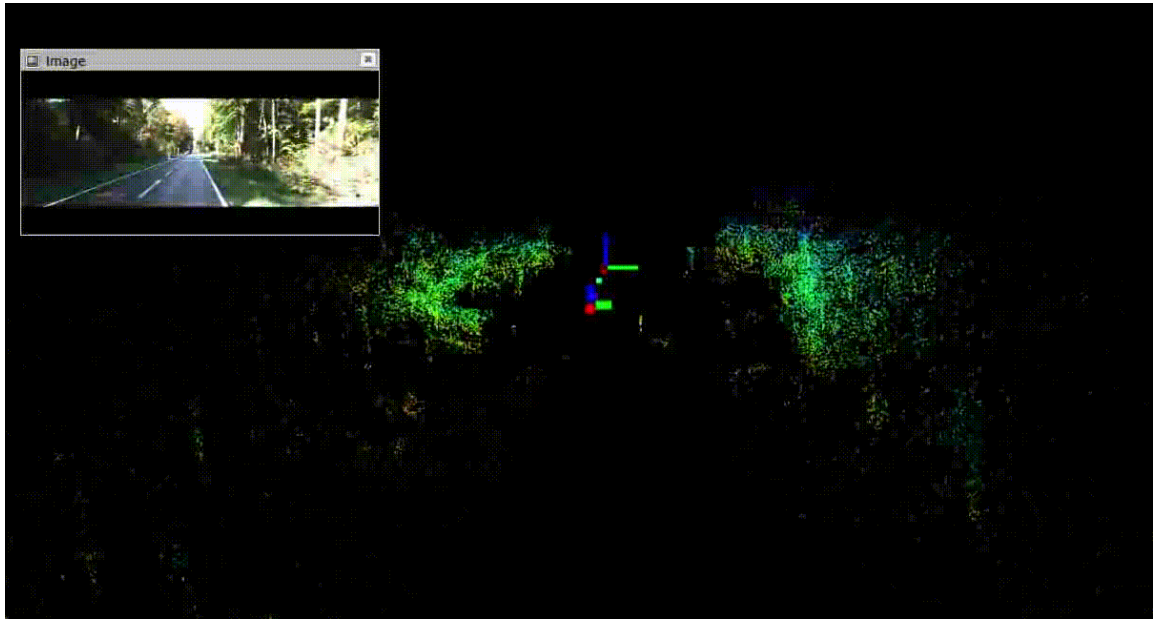
Issue1. Ground point

- 도로 vector 맵 생성의 방해요소
- 위 channel의 point와 각도 계산으로 제거
- raw data를 초기에 지움으로써 연산량 감소 효과



물체를 만났을 때 angle의 변화



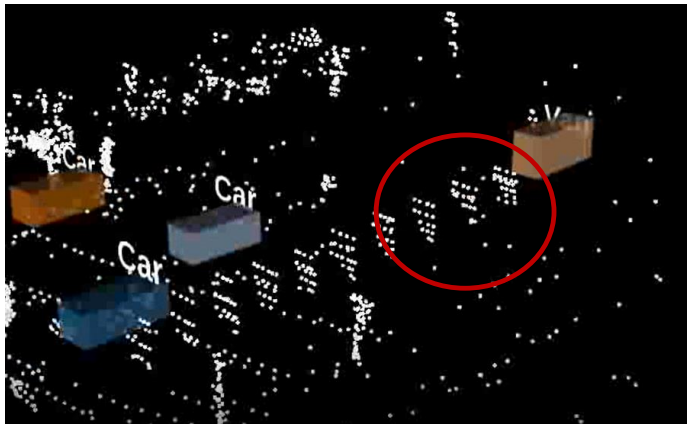


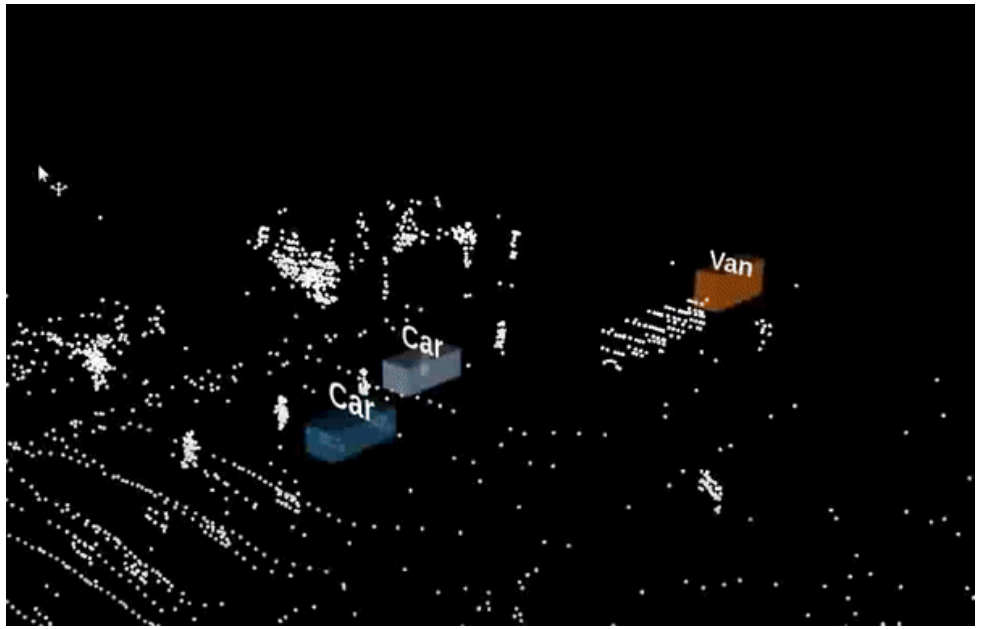
SLAM

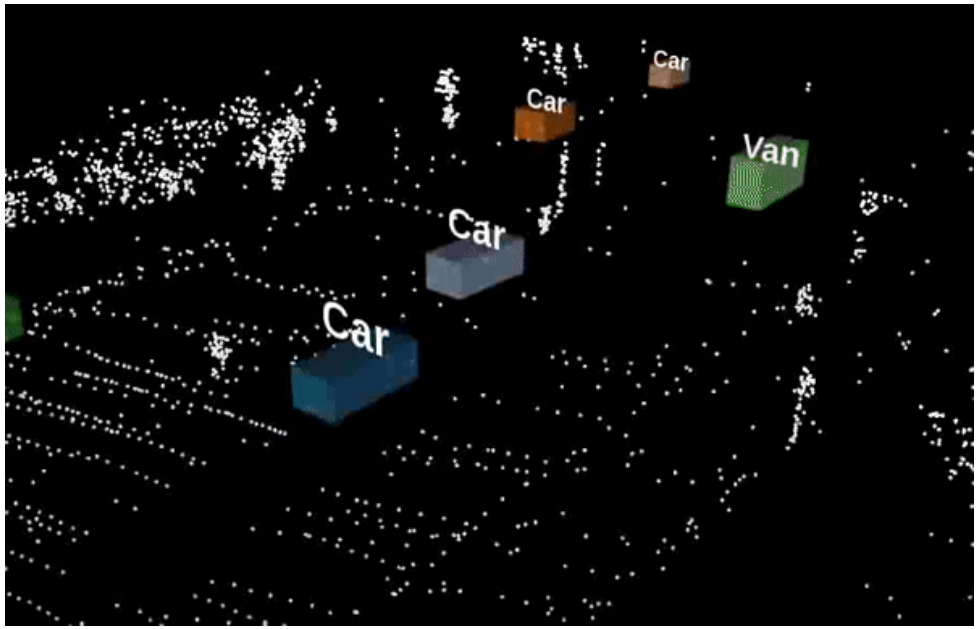
Issue2. Dynamic Objects point

- 완성된 Map에서 불필요한 정보

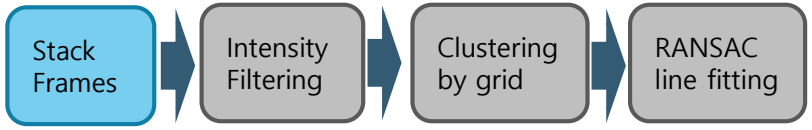
- Map 기반 자율주행 시 Localization 방해요소
(장애물 인식 가능성)





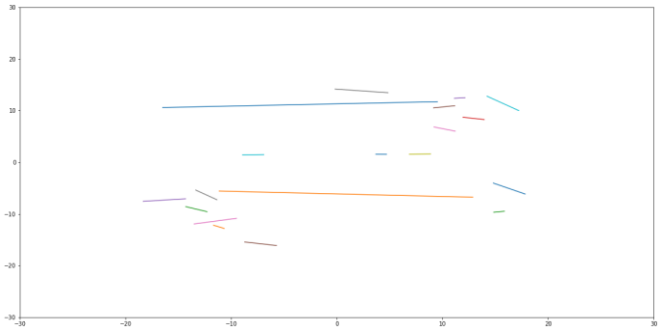


LANE DETECTION

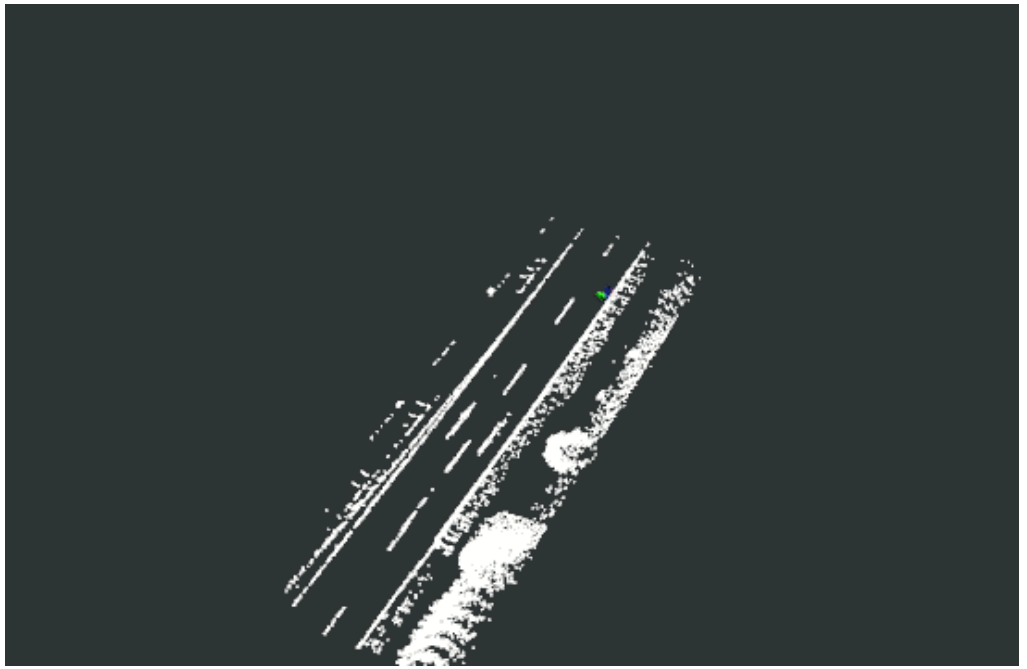


- **Stack Frame**

  - 단일 프레임 사용 시 정보 부족으로 노이즈가 극대화
  - SLAM에서의 tf data 변환을 이용하여 road data 축적
  - 10 frame마다 축적

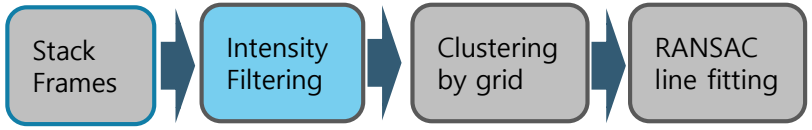


Lack of point cloud information(1 frame)

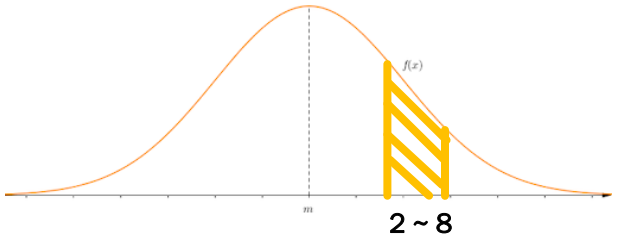


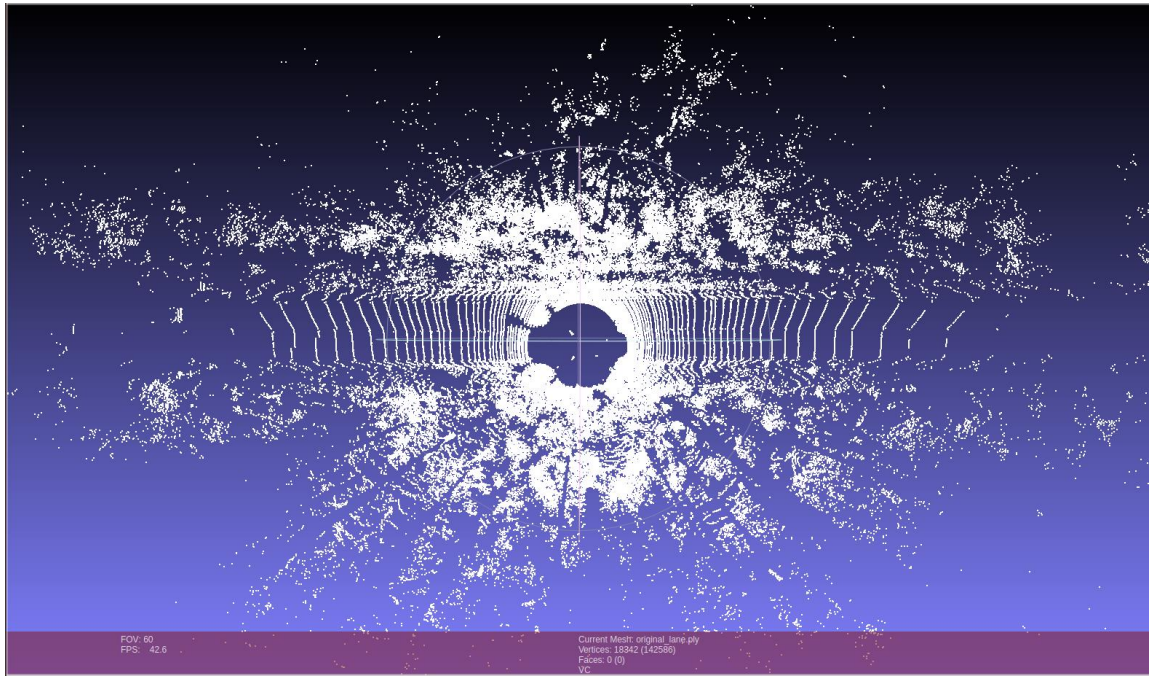
frames accumulated(30 frames)

LANE DETECTION

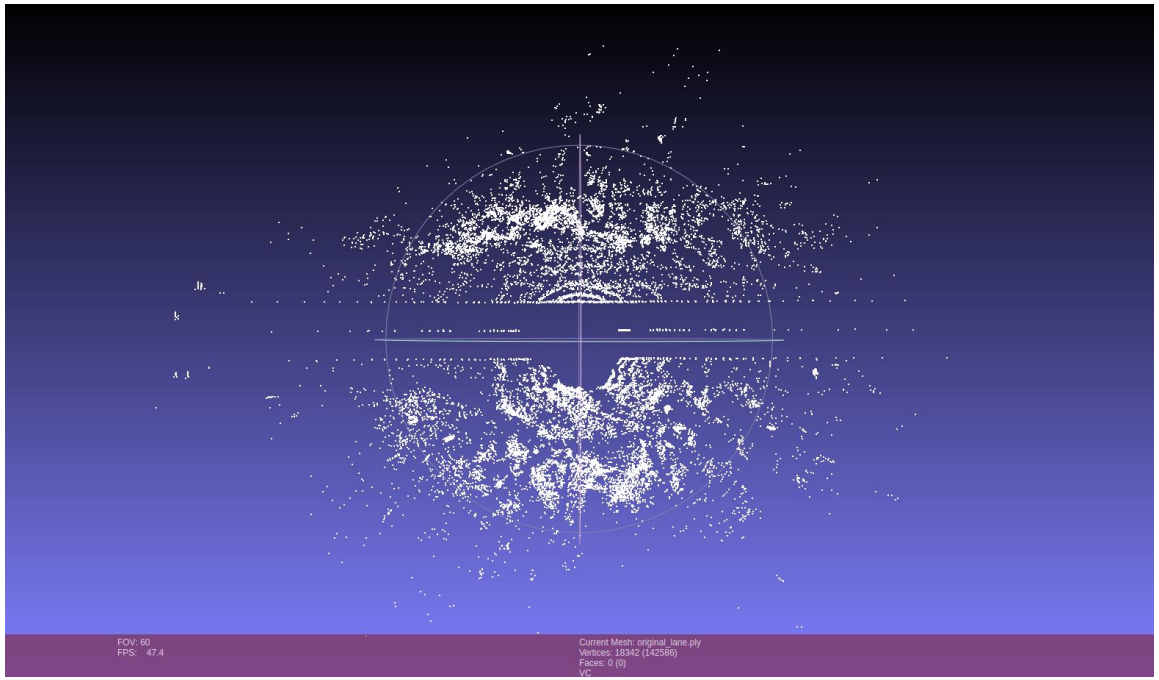


- **Intensity Filtering**
  - 특정 범위의 반사율(intensity)을 가진 차선 검출



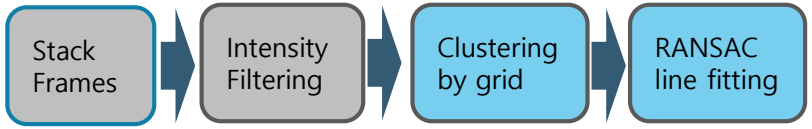


Velodyne Point Clouds



After Intensity Filtering

LANE DETECTION



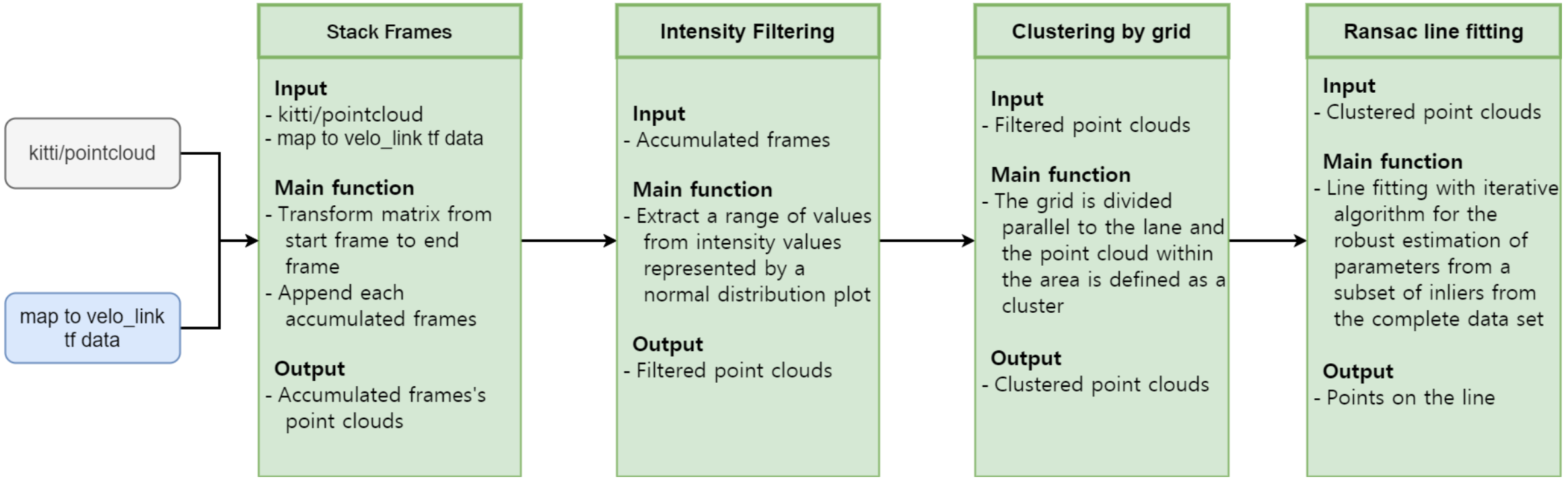
- **Clustering by grid**
  - 차선과 평행하도록 grid를 나누어 point가 설정값보다 많다면 Clustering 진행

- **RANSAC line fitting**
  - 무작위로 데이터를 뽑아 만족하는 모델 파라미터를 구하고 구한 모델과 가까이에 있는 데이터들의 개수가 크다면 기억해두는 과정반복
  - 반복 후 가장 데이터가 많았던 모델을 최종 line으로 도출

Lane vector graphics

LANE DETECTION

- **Overview**

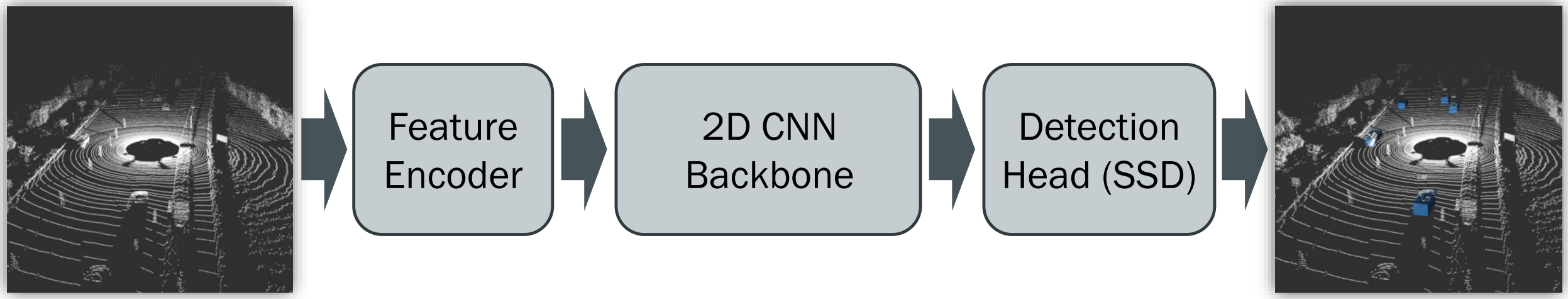


OBJECT DETECTION

- **PointPillars**

  - LiDAR data 기반의 3D 객체 검출 알고리즘
  - KITTI Challenge에서 3D box와 BEV box Benchmarks에 대해 SOTA 성능
  - 최대 60Hz의 실시간 검출속도와 높은 정확도 성능으로 Autoware에서 채택됨
  - 3D Point Cloud를 Pillar 형태의 2D data로 Encoding함으로서 2D Convolutional Network 사용

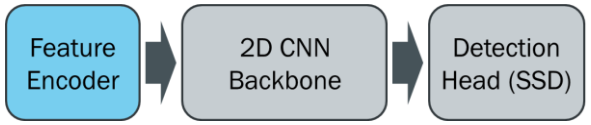
- **PointPillars Network Overview**



3D Point Cloud

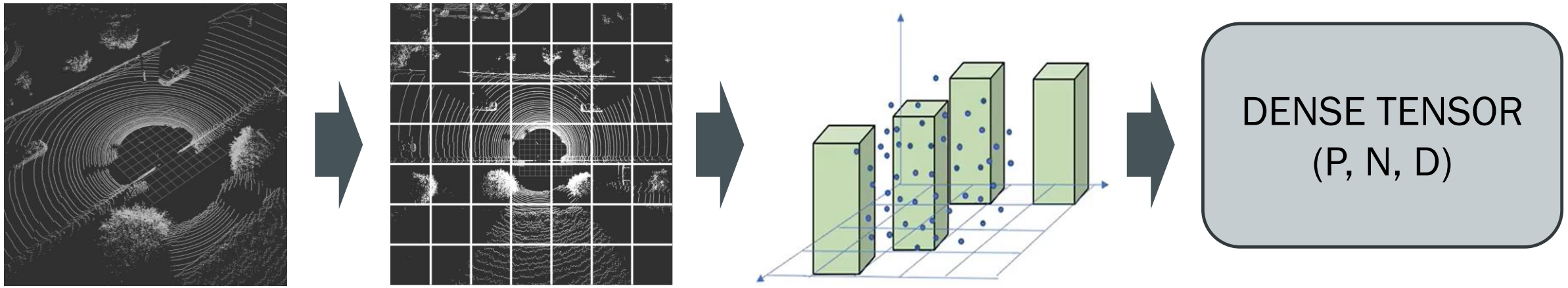
Detection

OBJECT DETECTION



- **Feature Encoder**

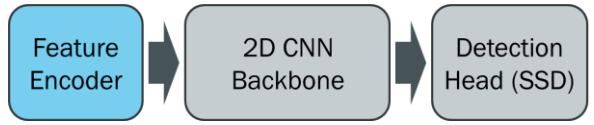
  - Point cloud to sparse pseudo image



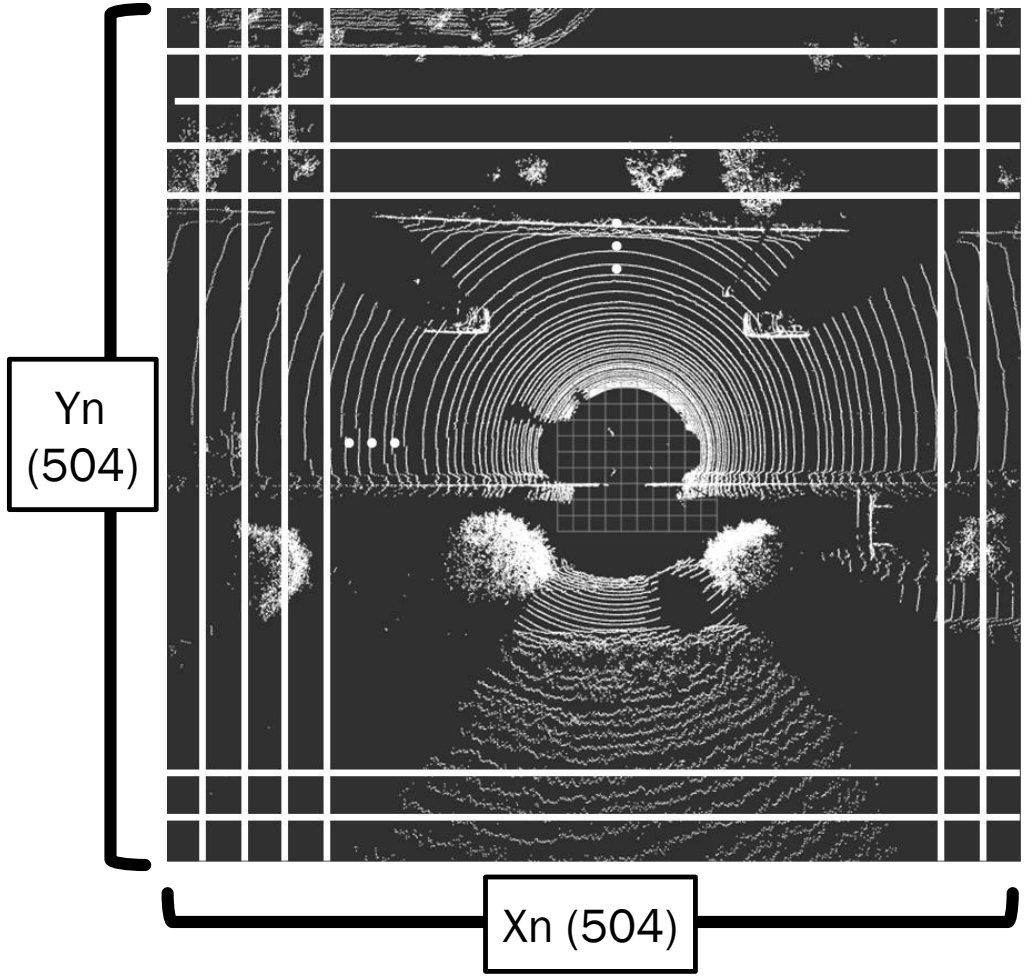
3D Point Cloud | X-Y grid | Pillars on X-Y grid

- P : Maximum Number of Pillars (None empty pillar)
- N : Maximum number of Points per Pillar
- D = [x, y, z, intensity, Xc, Yc, Zc, Xp, Yp]
  - Xc, Yc, Zc : Distance from the arithmetic mean of the pillar
  - Xp, Yp : Distance of the point from the center of the pillar

OBJECT DETECTION

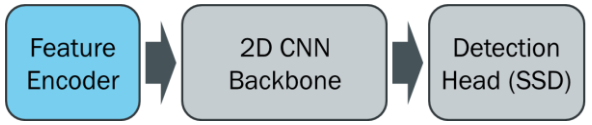


- **Pseudo Image : (Xn, Yn) Pixel**

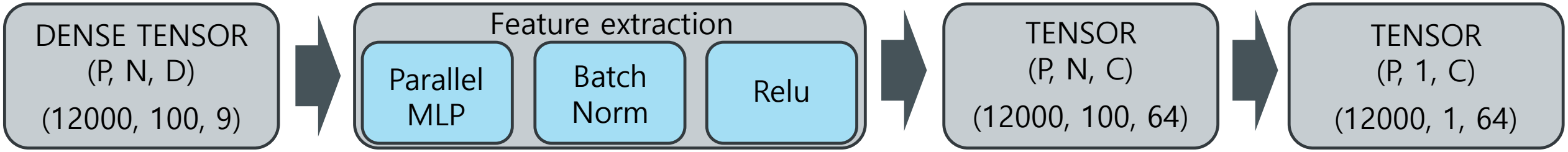



```python
class GridParameters:
    x_min = 0.0
    x_max = 80.64
    x_step = 0.16

    y_min = -40.32
    y_max = 40.32
    y_step = 0.16

    z_min = -1.0
    z_max = 3.0

    max_points_per_pillar = 100
    max_pillars = 12000
    # derived parameters
    Xn_f = float(x_max - x_min) / x_step
    Yn_f = float(y_max - y_min) / y_step
    Xn = int(Xn_f)
    Yn = int(Yn_f)
```

OBJECT DETECTION



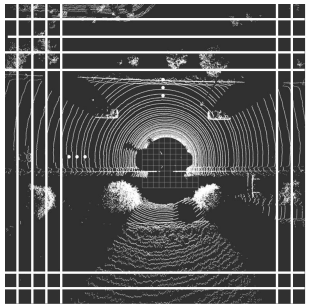
- **Feature Encoder**



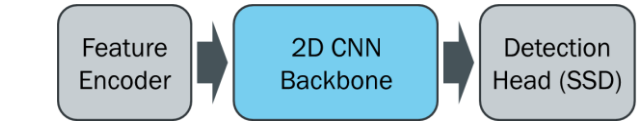
Layer (type)	Output Shape	Param #	Connected to
pillars/input (InputLayer)	[(4, 12000, 100, 9)]	0	
pillars/conv2d (Conv2D)	(4, 12000, 100, 64)	448	pillars/input[0][0]
pillars/batchnorm (BatchNormali	(4, 12000, 100, 64)	256	pillars/conv2d[0][0]
pillars/relu (Activation)	(4, 12000, 100, 64)	0	pillars/batchnorm[0][0]
pillars/indices (InputLayer)	[(4, 12000, 3)]	0	
pillars/maxpooling2d (MaxPoolin	(4, 12000, 1, 64)	0	pillars/relu[0][0]
lambda (Lambda)	(4, 12000, 3)	0	pillars/indices[0][0]
pillars/reshape (Reshape)	(4, 12000, 64)	0	pillars/maxpooling2d[0][0]
pillars/scatter_nd (Lambda)	(4, 504, 504, 64)	0	lambda[0][0] pillars/reshape[0][0]

Reshape P
& Scatter_nd

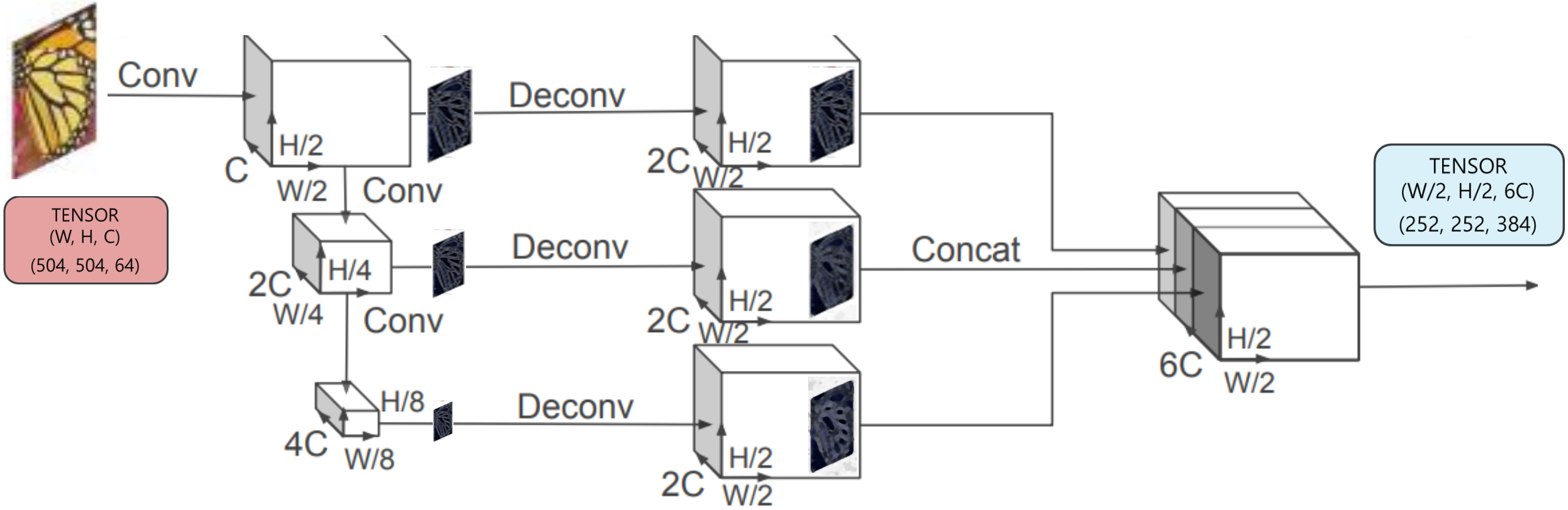
TENSOR
(W, H, C)
(504, 504, 64)



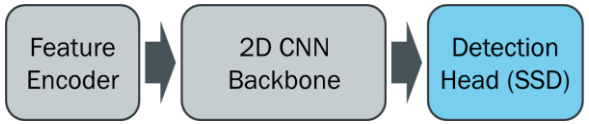
OBJECT DETECTION



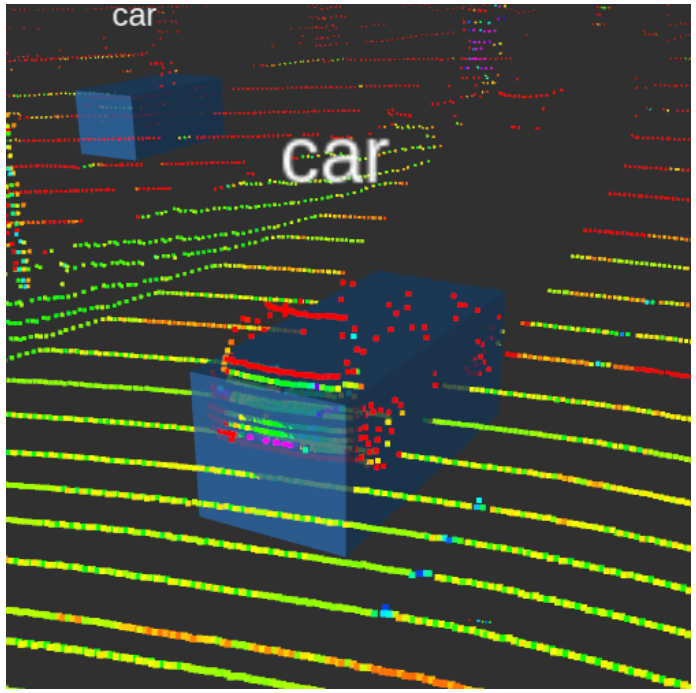
- **2D CNN Backbone**
  - Pseudo Image to High Level Representation



OBJECT DETECTION



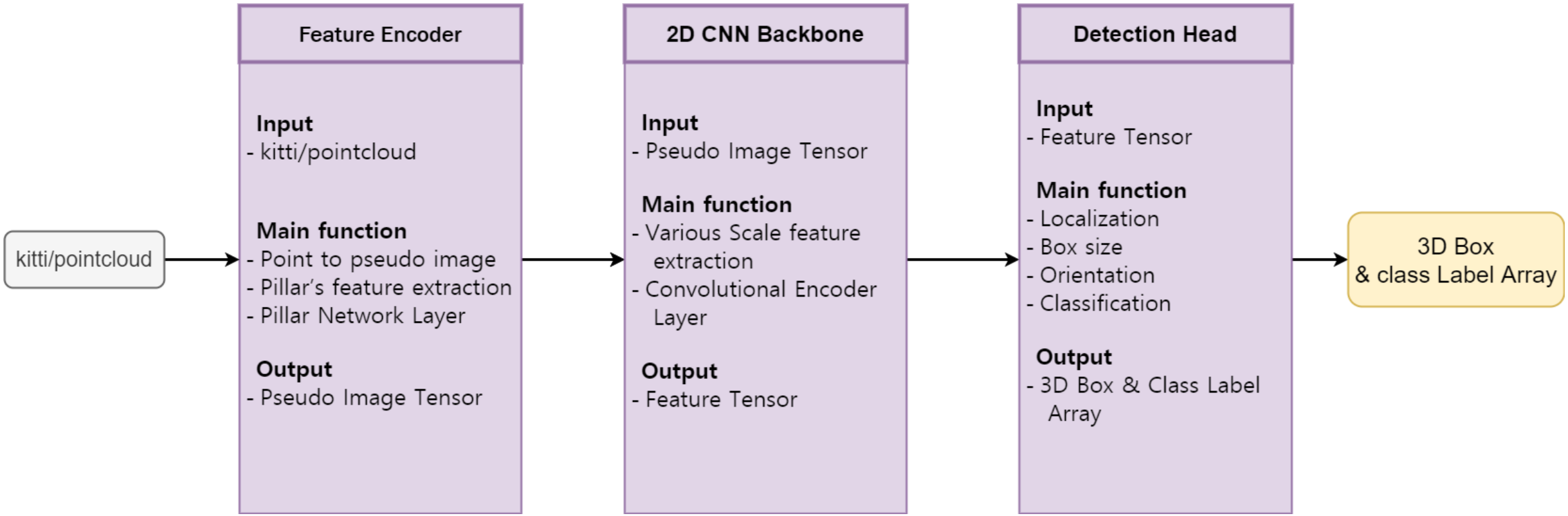
- **Detection Head**
  - Detects and Regresses 3D boxes
  - Estimating the Position and orientation and size of 3D bounding box & Class label



loc/conv2d (Conv2D)	(4, 252, 252, 12)	4620	cnn/concatenate[0][0]
size/conv2d (Conv2D)	(4, 252, 252, 12)	4620	cnn/concatenate[0][0]
clf/conv2d (Conv2D)	(4, 252, 252, 16)	6160	cnn/concatenate[0][0]
occupancy/conv2d (Conv2D)	(4, 252, 252, 4)	1540	cnn/concatenate[0][0]
loc/reshape (Reshape)	(4, 252, 252, 4, 3)	0	loc/conv2d[0][0]
size/reshape (Reshape)	(4, 252, 252, 4, 3)	0	size/conv2d[0][0]
angle/conv2d (Conv2D)	(4, 252, 252, 4)	1540	cnn/concatenate[0][0]
heading/conv2d (Conv2D)	(4, 252, 252, 4)	1540	cnn/concatenate[0][0]
clf/reshape (Reshape)	(4, 252, 252, 4, 4)	0	clf/conv2d[0][0]

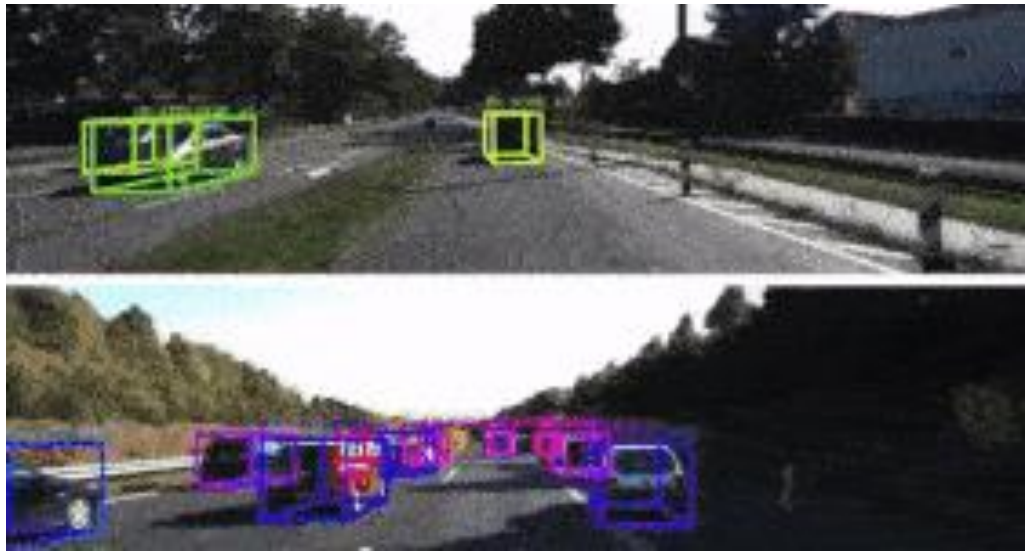
OBJECT DETECTION

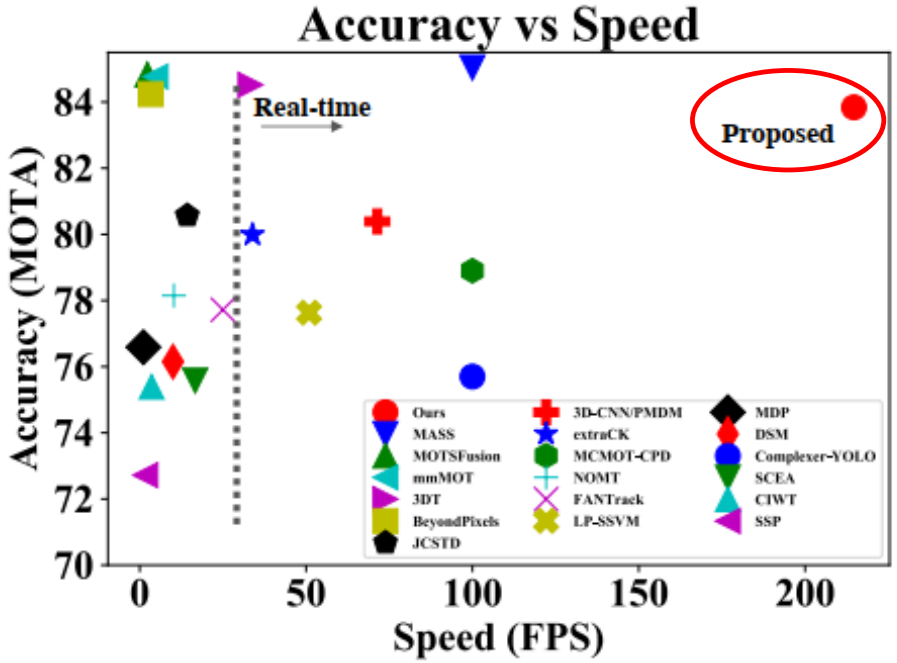
- **Overview**



TRACKING

- **AB3DMOT (A Baseline 3D Multi-Object Tracking)**
  - Kalman Filter & Hungarian Algorithm 기반 3D Multi-Object Tracking 알고리즘
  - Neural net Tracking 모델과 다르게 Trainning없이 3D MOT benchmarks에서 SOTA성능





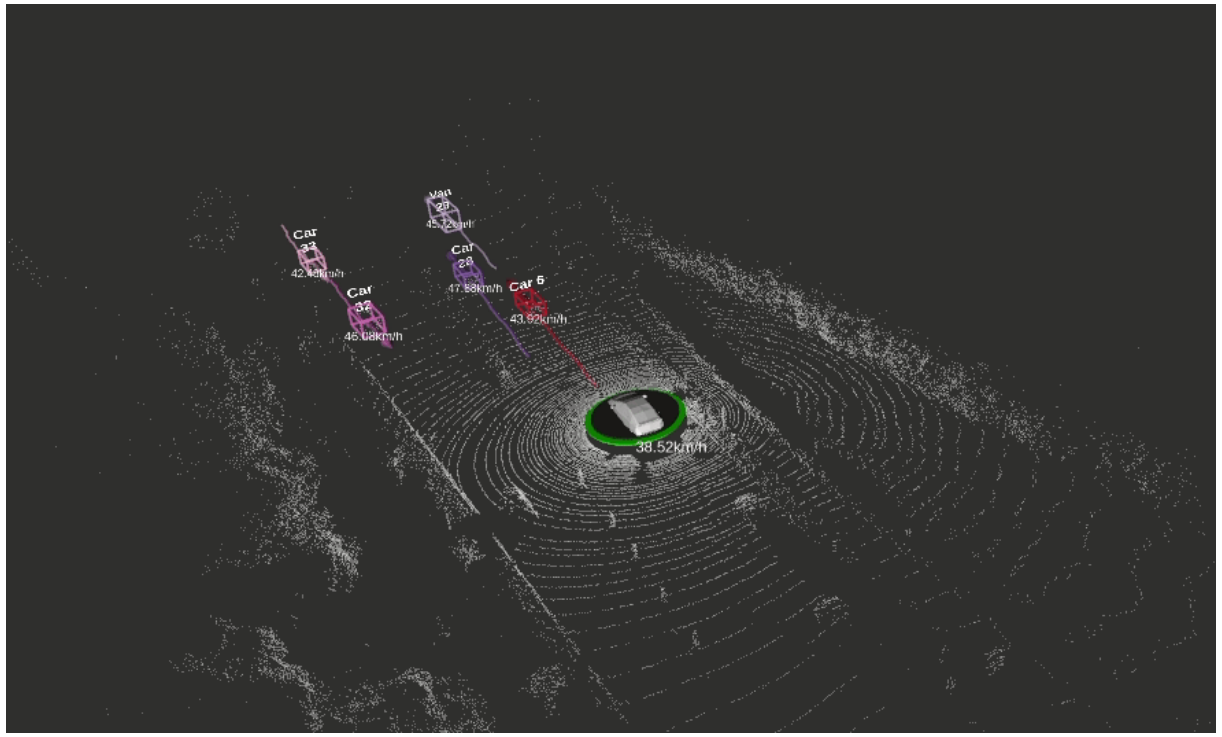
TRACKING

- **목표**

① Object Detection을 통해 얻은
Object들의 파라미터들을 이용하여
객체에 고정ID를 부여

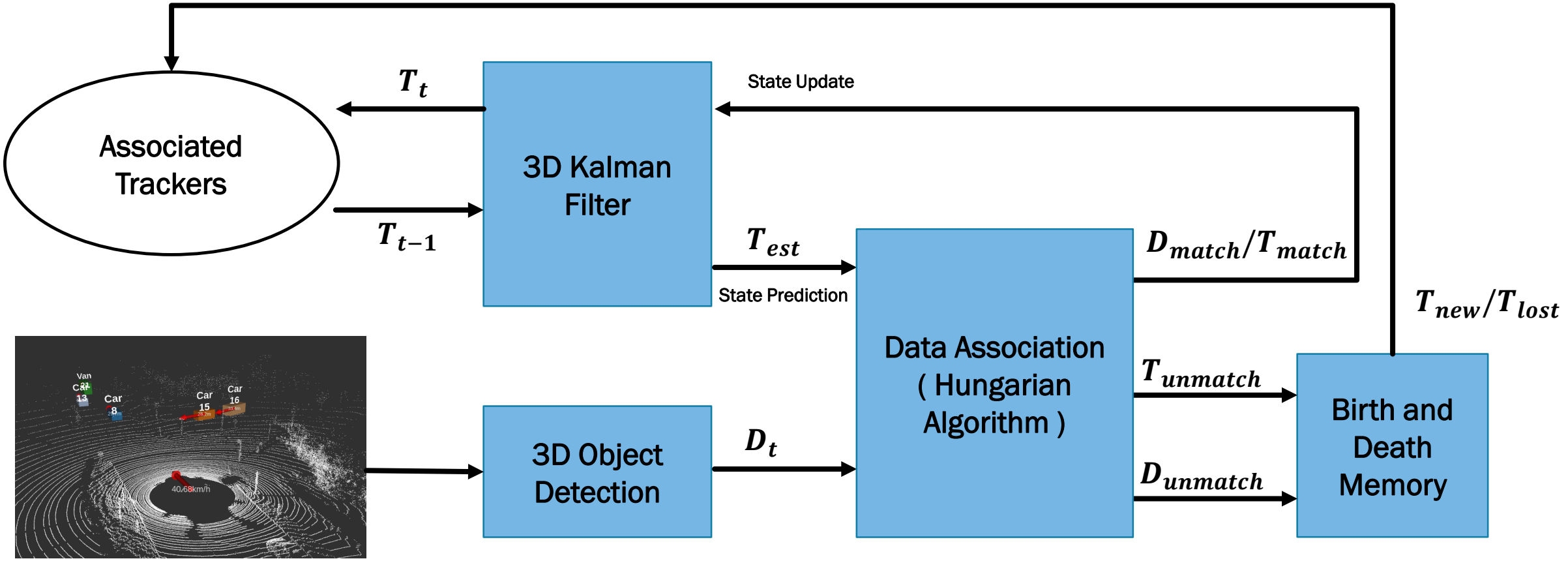
② ID가 부여된 Object들의 부가정보 추출

- Object들의 절대속도
- Object들의 누적경로
- Object와 ego car와의 위험거리



TRACKING

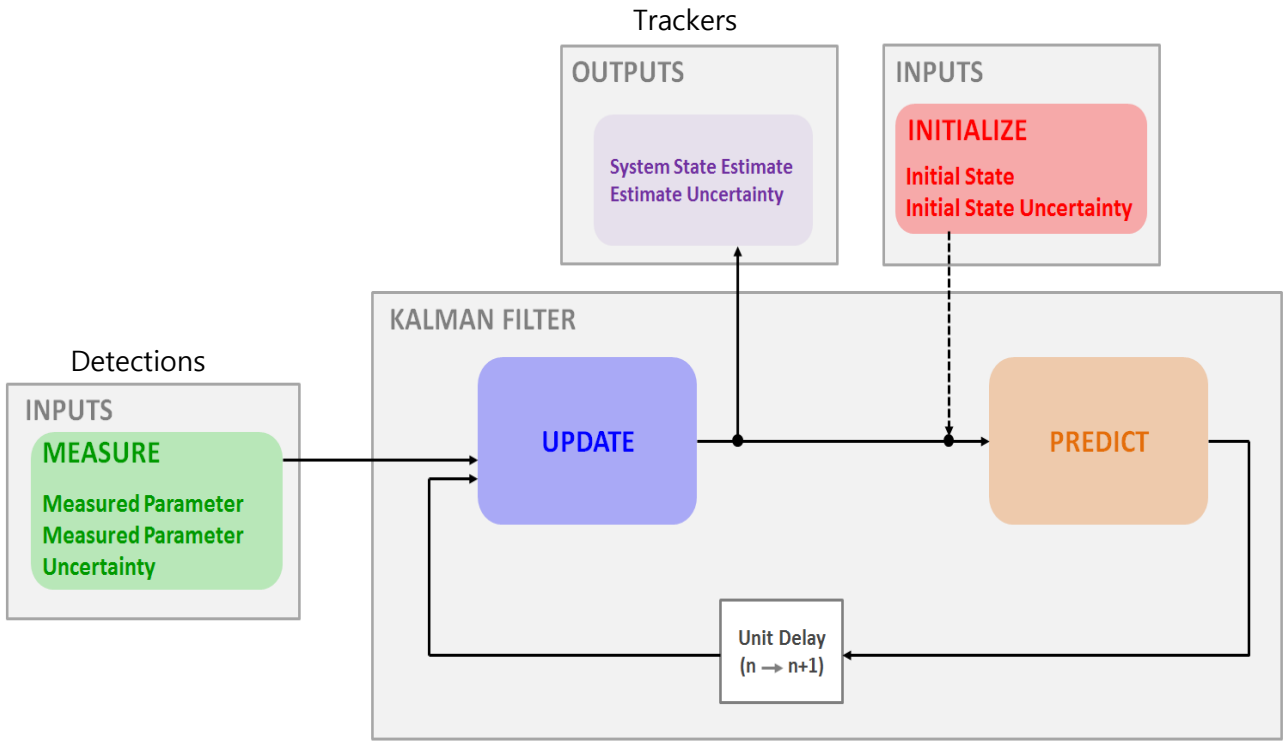
- 실행구조



LIDAR Point Cloud

TRACKING

- **Kalman Filter**



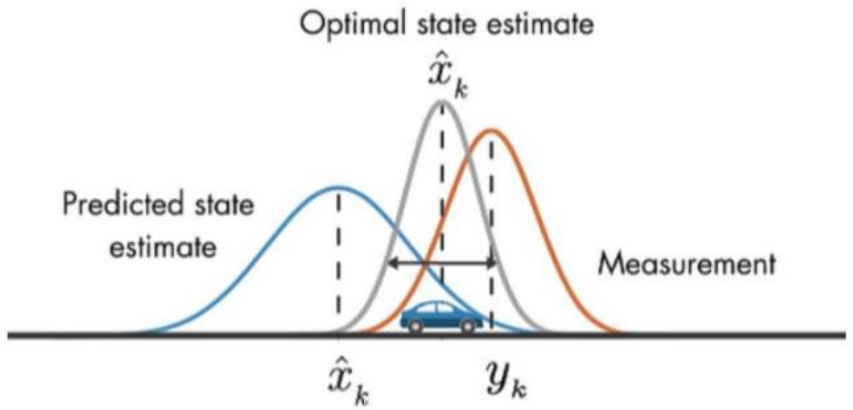
❖ **정의**

- 이전시점의 결과와 현재시점 입력 값을 가지고 다음 출력 값을 추정하는 알고리즘

❖ **프로젝트 적용**

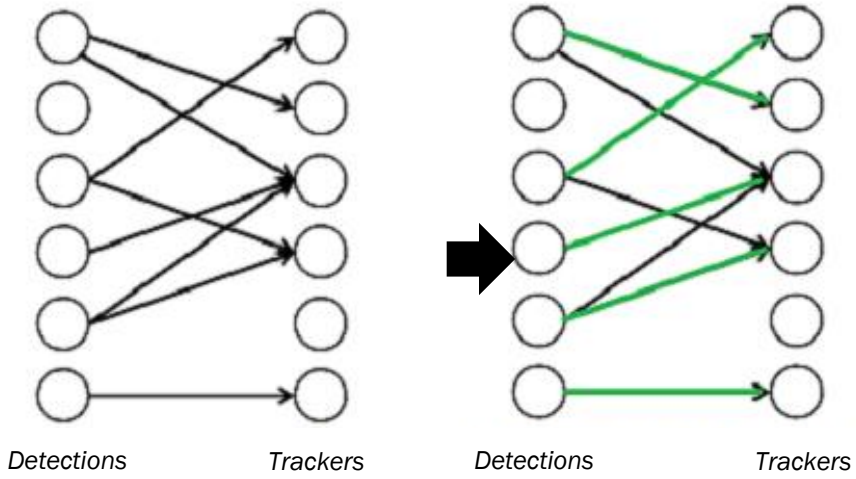
① 이전 시점 Object의 Tracker와 현 시점 Detection을 융합하여 Prediction

① Beyes Rule을 따라 Tracker를 Update



TRACKING

- **Hungarian Algorithm**



	T_1	T_2	T_3	T_4
D_1	3	6	9	10
D_2	1	8	0	6
D_3	15	1	10	7
D_4	0	7	10	9

- IOU Matrix 예시 -
(실제로는 0~1 사이의 백분위)

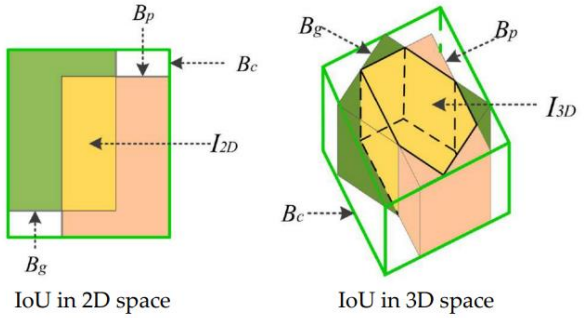
❖ **정의**

- 이분매칭 문제를 해결하는 알고리즘

❖ **프로젝트 적용**

① match된 Detection과 이전시점의 Tracker 들의 IOU Matrix 구성

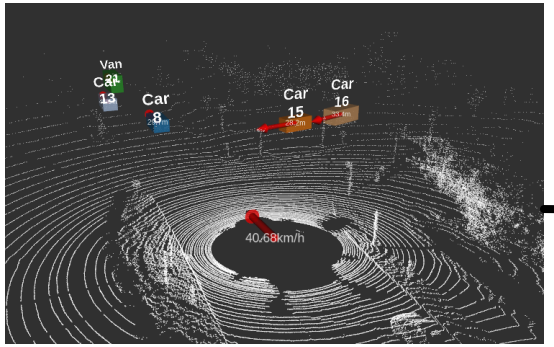
② Hungarian Algorithm을 적용하여 IOU를 최대로 하게하는 최적의 matching진행



B_p: the predicted box
B_g: the ground truth box
B_c: the smallest enclosing box
I_{2D}, I_{3D}: the intersection

TRACKING

- **실행구조**



LIDAR Point Cloud

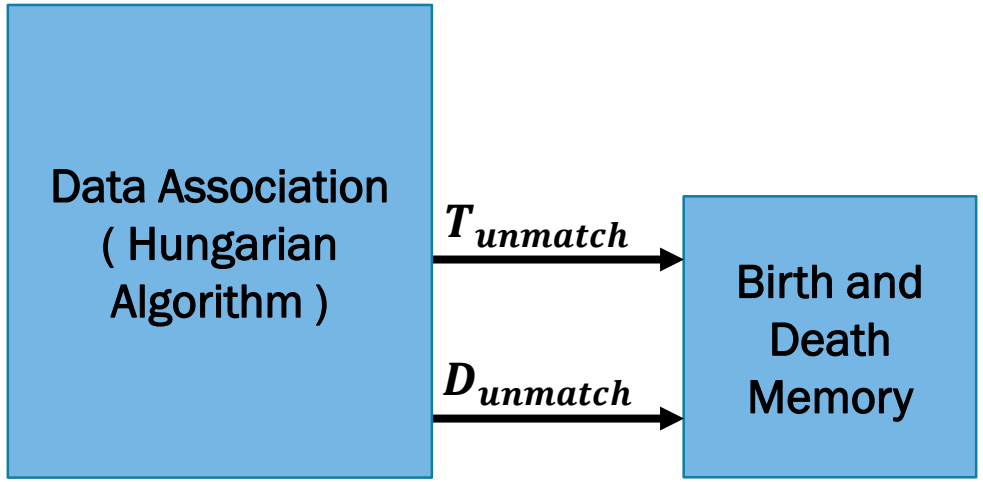
3D Object Detection

D_t

Data Association (Hungarian Algorithm)

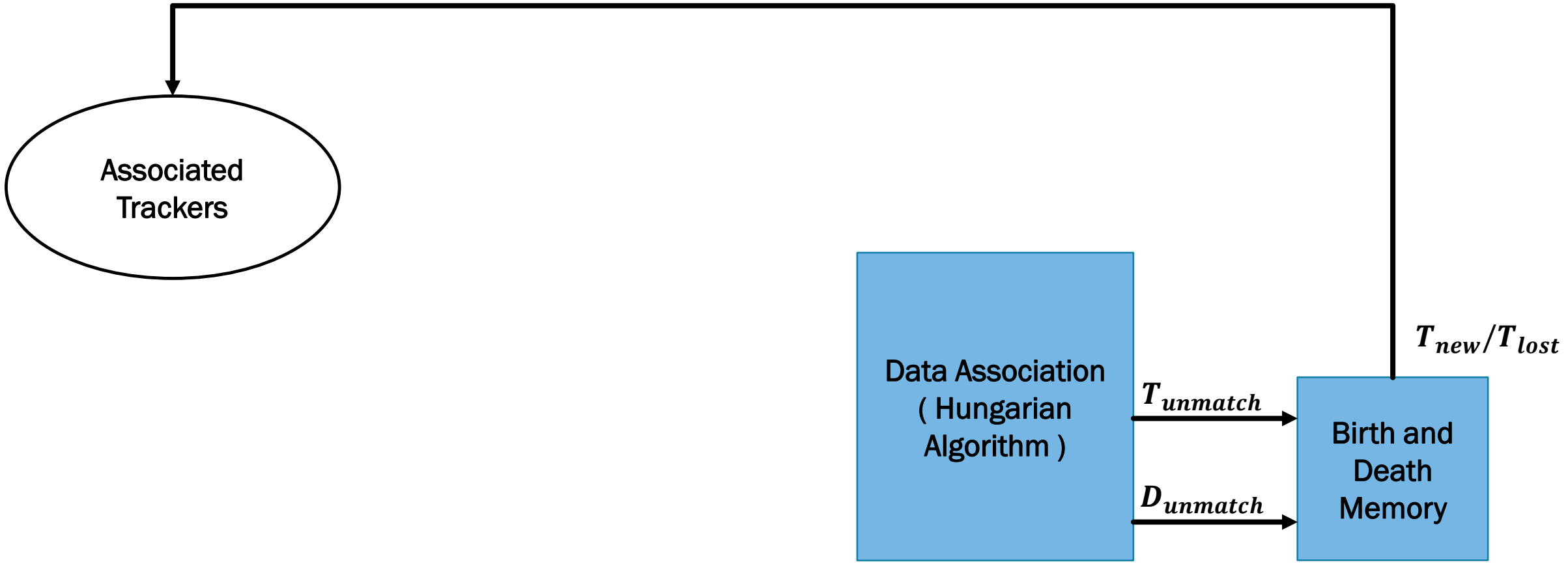
TRACKING

- **실행구조**



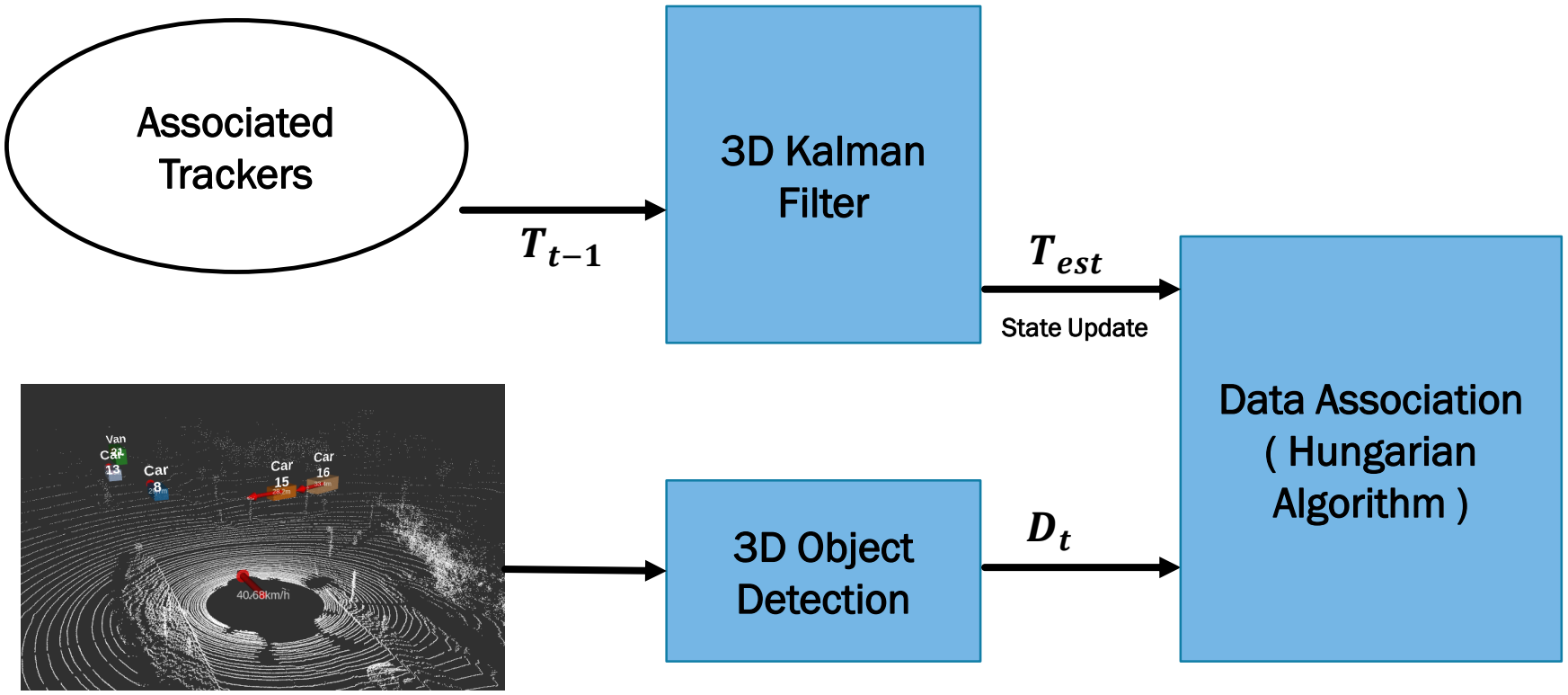
TRACKING

- 실행구조



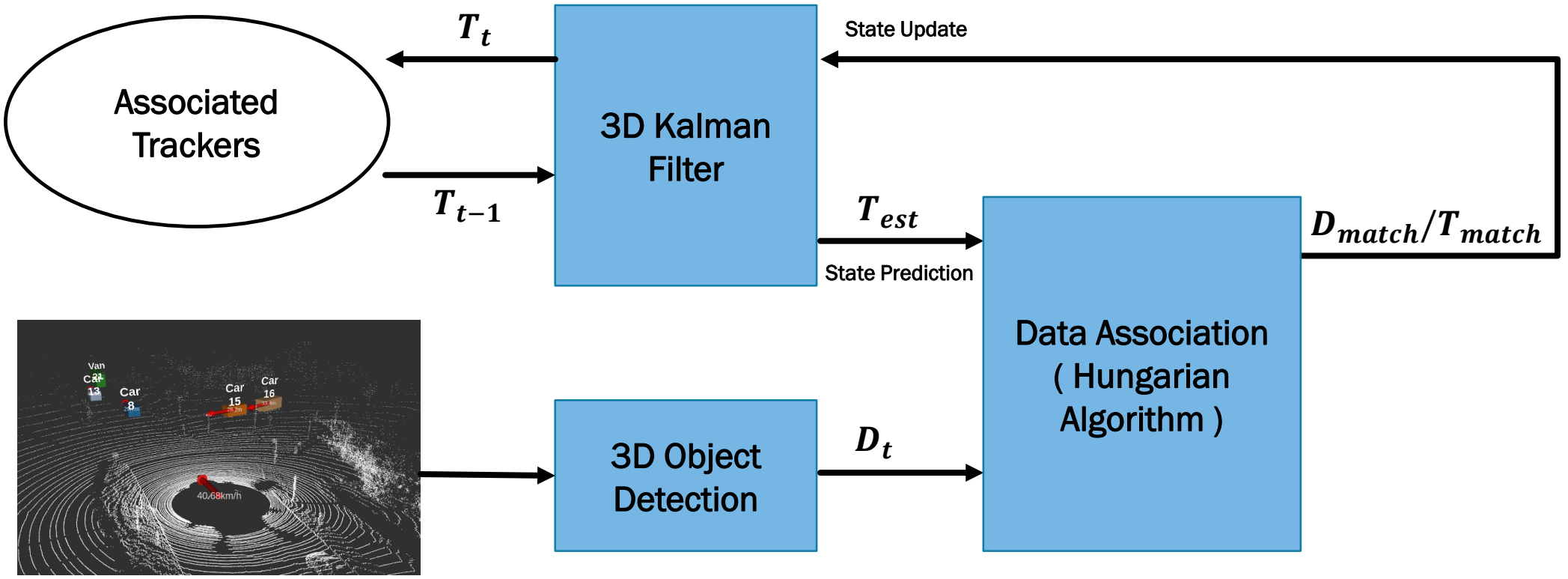
TRACKING

- **실행구조**



TRACKING

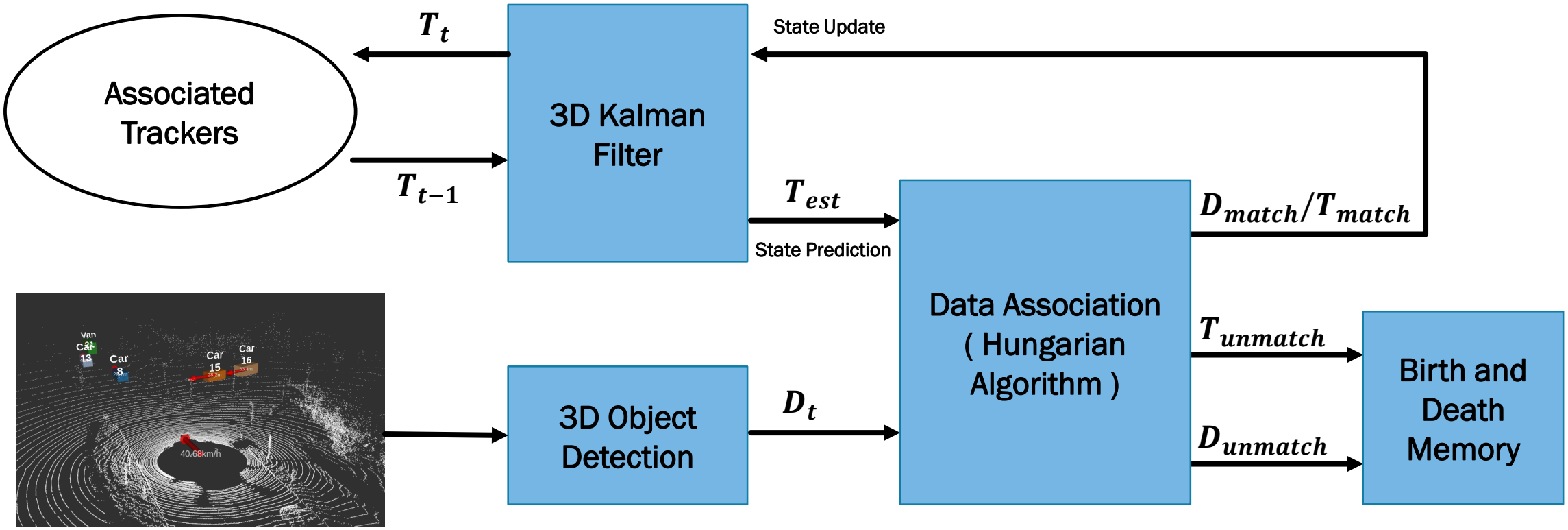
- **실행구조**



LIDAR Point Cloud

TRACKING

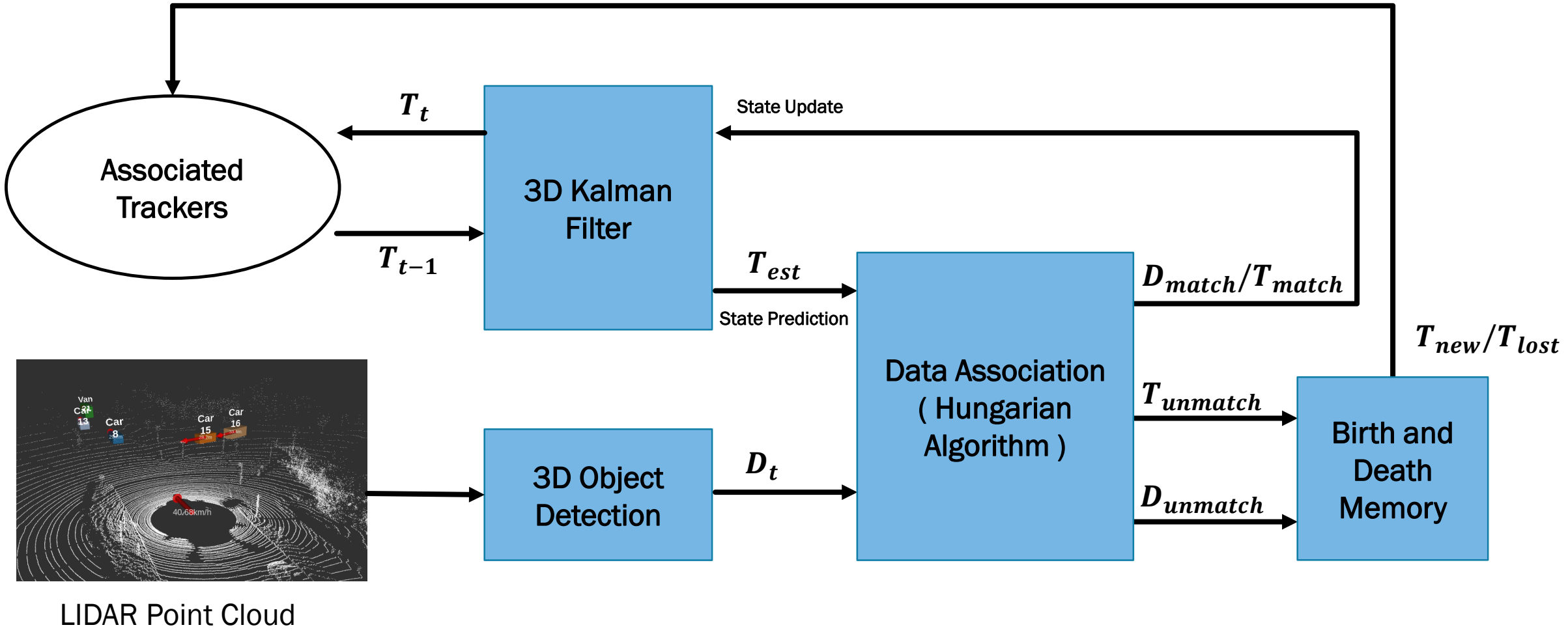
- 실행구조



LIDAR Point Cloud

TRACKING

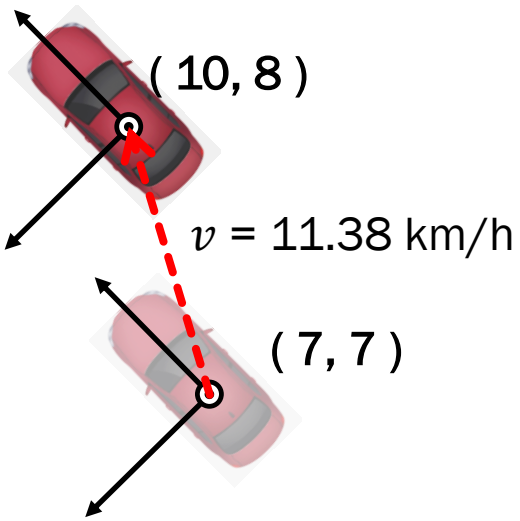
- 실행구조

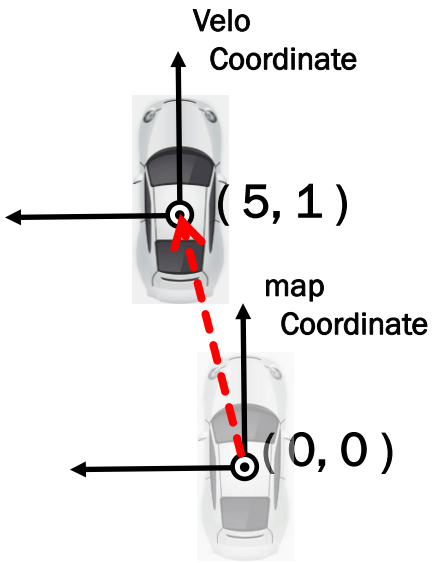


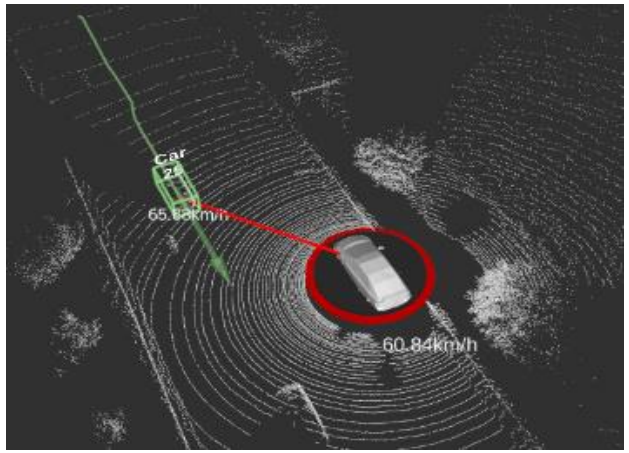
LIDAR Point Cloud

TRACKING

- **부가기능 구현 : Object들의 절대속도**







❖ **Object들의 절대 속도**

Map 좌표계 기준, Object의 이전프레임, 현재프레임에서 얻은 중심좌표를 이용하여 Euclidean Distance를 구한 후, frame당 걸리는 시간 0.1s로 나누어 준 후 km/h단위로 변경. 속도에 따라 Arrow marker의 크기를 조절

$$d = \sqrt{(x_t - x_{t-1})^2 + (y_t - y_{t-1})^2}\ ,\ obj_v = \frac{d\,(m)}{0.1\,(s)}$$

❖ **Object들의 누적경로**

Map 좌표계 기준, Detecting 되는 Object들의 중점을 15프레임을 저장하여, 누적경로를 선으로 표시한다.

❖ **Object와 Ego Car 사이의 주의신호 출력거리**

Velodyne 좌표계 기준, Object의 Boundary를 원으로 가정하고 Ego car와의 거리를 d, Object의 Boundary 원의 반지름을 r로, 원점가지의 거리를 $d - r$로 정의하고, 이 $d - r$이 일정 거리 이하로 들어온 경우 Ego car의 Warning circle을 빨간색으로 바꾸어 주의 표시

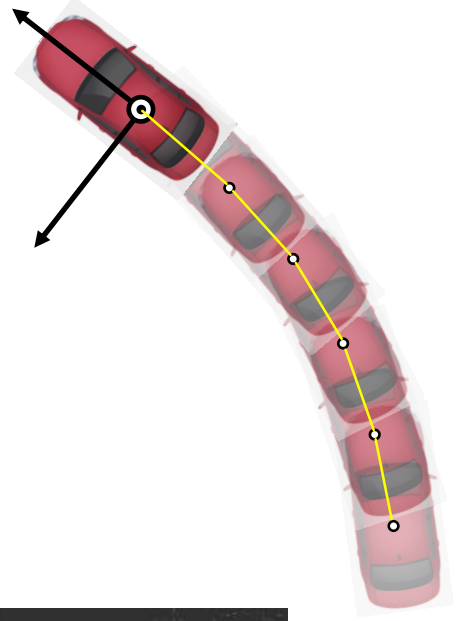
※ ***r* 구하기**

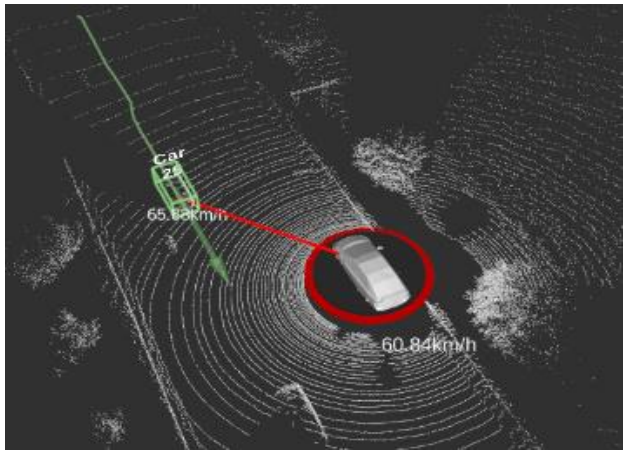
Tracking된 Boundingbox Size정보 h, w, l 정보들을 이용하여 BoundingBox와 접하는 원을 가정하고, 원의 반지름은 원의 중점과 접점까지의 거리이기 때문에 $(\boldsymbol{h/2,\ w/2,\ l/2})$의 Euclidean Distance 이기 때문에 아래 수식과 같다.

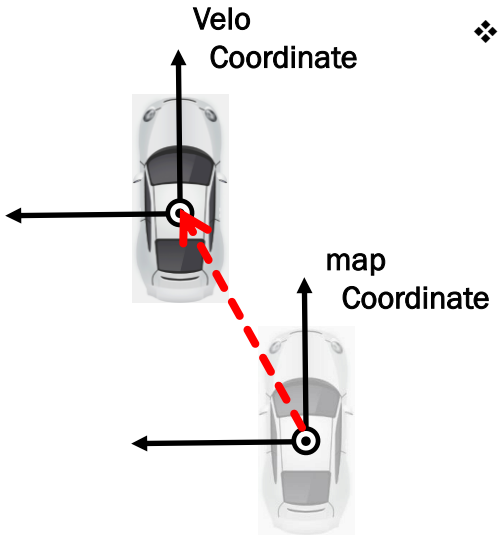
$$r = \sqrt{(h/2)^2 + (w/2)^2 + (l/2)^2}$$

TRACKING

- **부가기능 구현 : Object들의 누적경로**







❖ **Object들의 절대 속도**

Map 좌표계 기준, Object의 이전프레임, 현재프레임에서 얻은 중심좌표를 이용하여 Euclidean Distance를 구한 후, frame당 걸리는 시간 0.1s로 나누어 준 후 km/h단위로 변경. 속도에 따라 Arrow marker의 크기를 조절

$$d = \sqrt{(x_t - x_{t-1})^2 + (y_t - y_{t-1})^2}\ ,\ obj_v = \frac{d\,(m)}{0.1\,(s)}$$

❖ **Object들의 누적경로**

Map 좌표계 기준, Detecting 되는 Object들의 중점을 15프레임을 저장하여, 누적경로를 선으로 표시한다.

❖ **Object와 Ego Car 사이의 주의신호 출력거리**

Velodyne 좌표계 기준, Object의 Boundary를 원으로 가정하고 Ego car와의 거리를 d, Object의 Boundary 원의 반지름을 r로, 원점가지의 거리를 $d-r$로 정의하고, 이 $d-r$이 일정 거리 이하로 들어온 경우 Ego car의 Warning circle을 빨간색으로 바꾸어 주의 표시

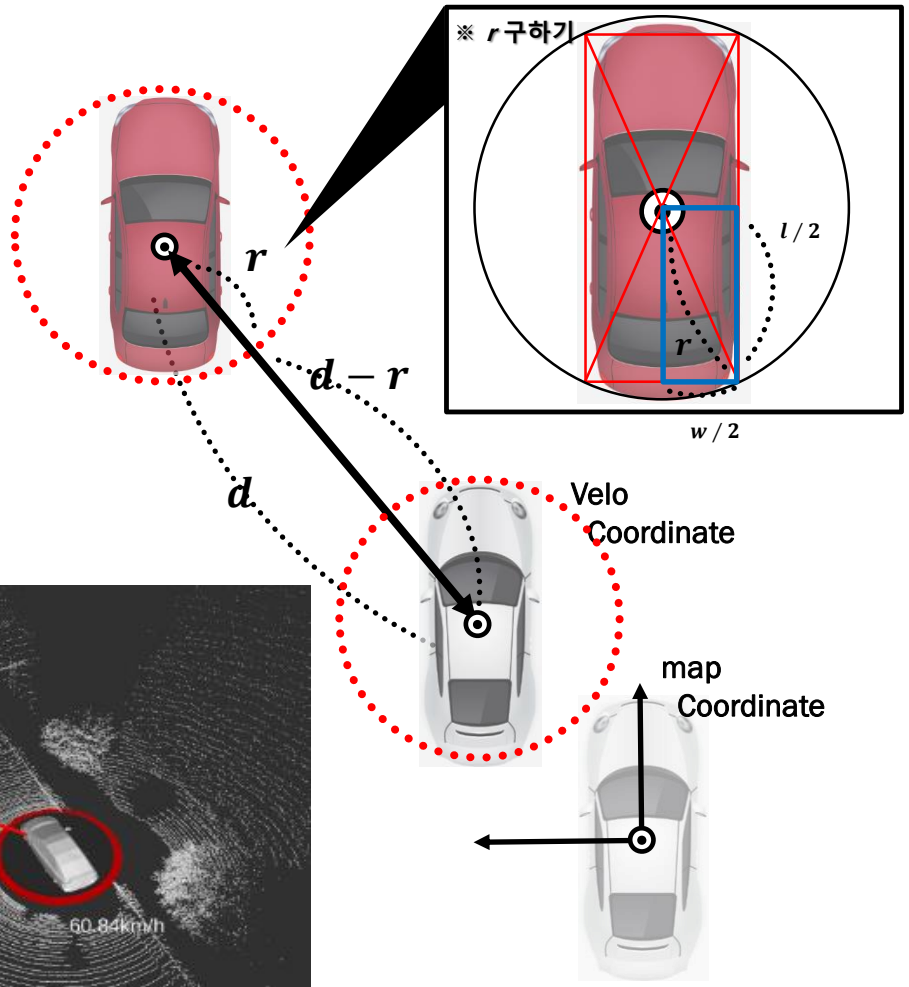
※ r **구하기**

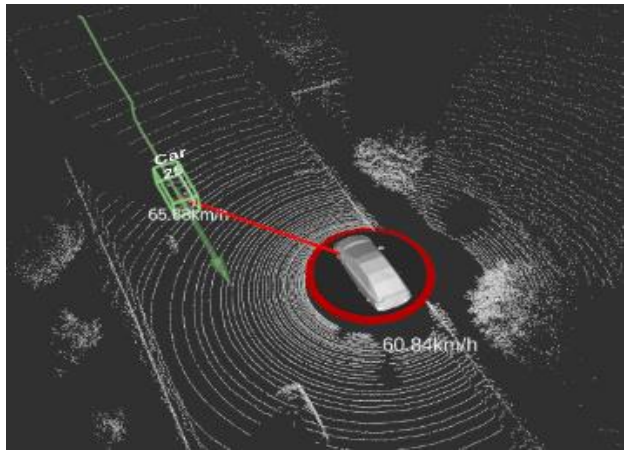
Tracking된 Boundingbox Size정보 h, w, l 정보들을 이용하여 BoundingBox와 접하는 원을 가정하고, 원의 반지름은 원의 중점과 접점까지의 거리이기 때문에 $(\boldsymbol{h/2,\ w/2,\ l/2})$의 Euclidean Distance 이기 때문에 아래 수식과 같다.

$$r = \sqrt{(h/2)^2 + (w/2)^2 + (l/2)^2}$$

TRACKING

- **부가기능 구현 : 위험 거리 감지**





❖ **Object들의 절대 속도**

Map 좌표계 기준, Object의 이전프레임, 현재프레임에서 얻은 중심좌표를 이용하여 Euclidean Distance를 구한 후, frame당 걸리는 시간 0.1s로 나누어 준 후 km/h단위로 변경. 속도에 따라 Arrow marker의 크기를 조절

$$d = \sqrt{(x_t - x_{t-1})^2 + (y_t - y_{t-1})^2}\ ,\ obj_v = \frac{d\,(m)}{0.1\,(s)}$$

❖ **Object들의 누적경로**

Map 좌표계 기준, Detecting 되는 Object들의 중점을 15프레임을 저장하여, 누적경로를 선으로 표시한다.

❖ **Object와 Ego Car 사이의 주의신호 출력거리**

Velodyne 좌표계 기준, Object의 Boundary를 원으로 가정하고 Ego car와의 거리를 d, Object의 Boundary 원의 반지름을 r로, 원점가지의 거리를 $d-r$로 정의하고, 이 $d-r$이 일정 거리 이하로 들어온 경우 Ego car의 Warning circle을 빨간색으로 바꾸어 주의 표시

※ **r 구하기**

Tracking된 Boundingbox Size정보 h, w, l 정보들을 이용하여 BoundingBox와 접하는 원을 가정하고, 원의 반지름은 원의 중점과 접점까지의 거리이기 때문에 $(h/2,\ w/2,\ l/2)$의 Euclidean Distance 이기 때문에 아래 수식과 같다.

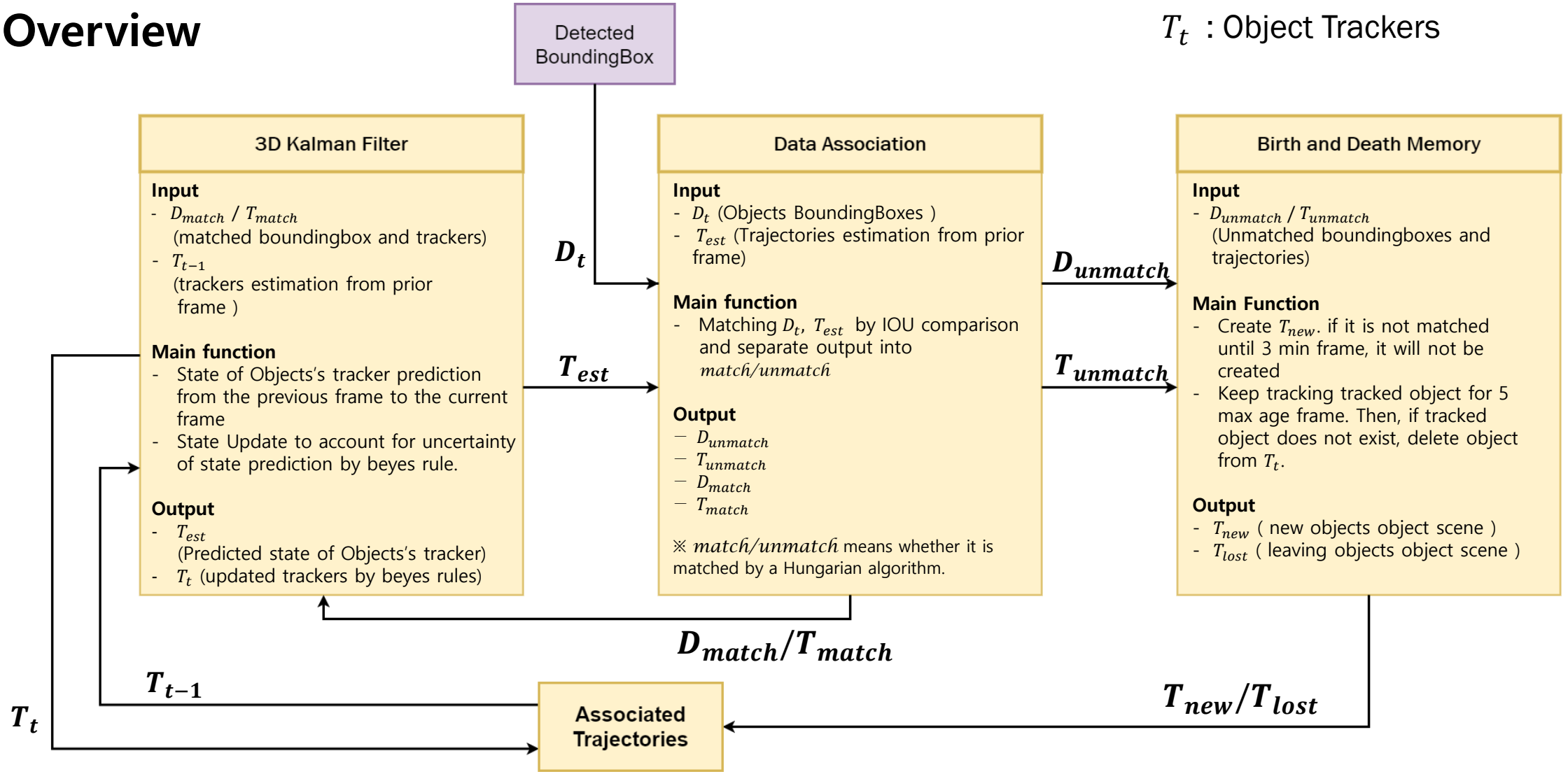
$$r = \sqrt{(h/2)^2 + (w/2)^2 + (l/2)^2}$$

TRACKING

- **Overview**

D_t : Detection BoundingBox Info

T_t : Object Trackers



Part 3

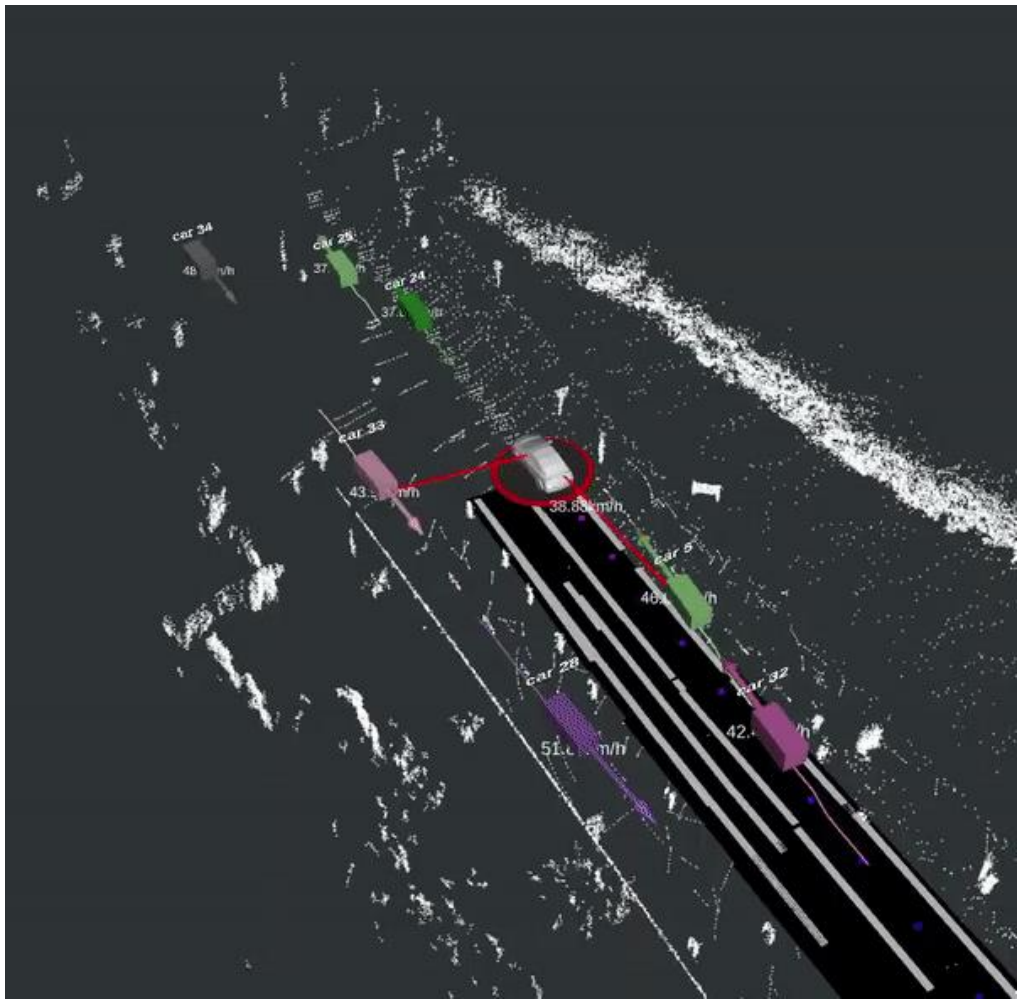
최종 결과물 & rqt graph

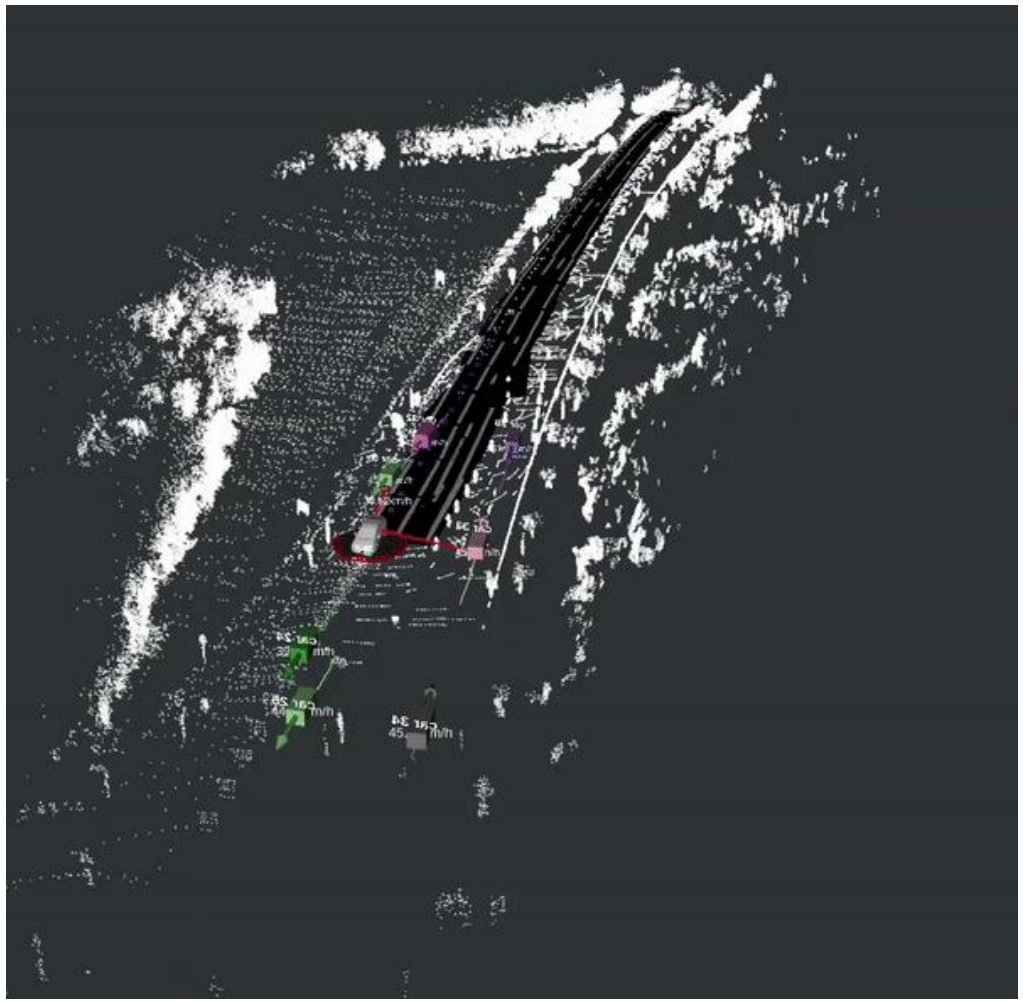
추가 연구 방향

활용 방안

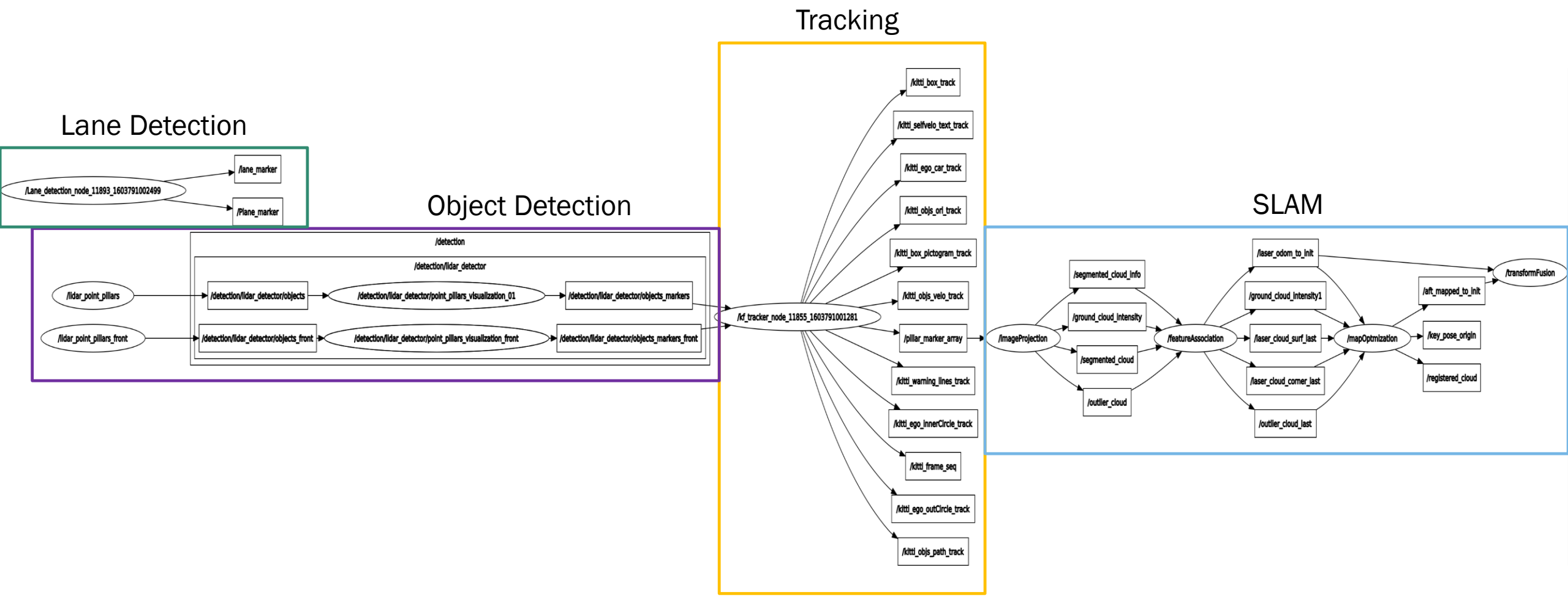
참고 논문

최종 결과물



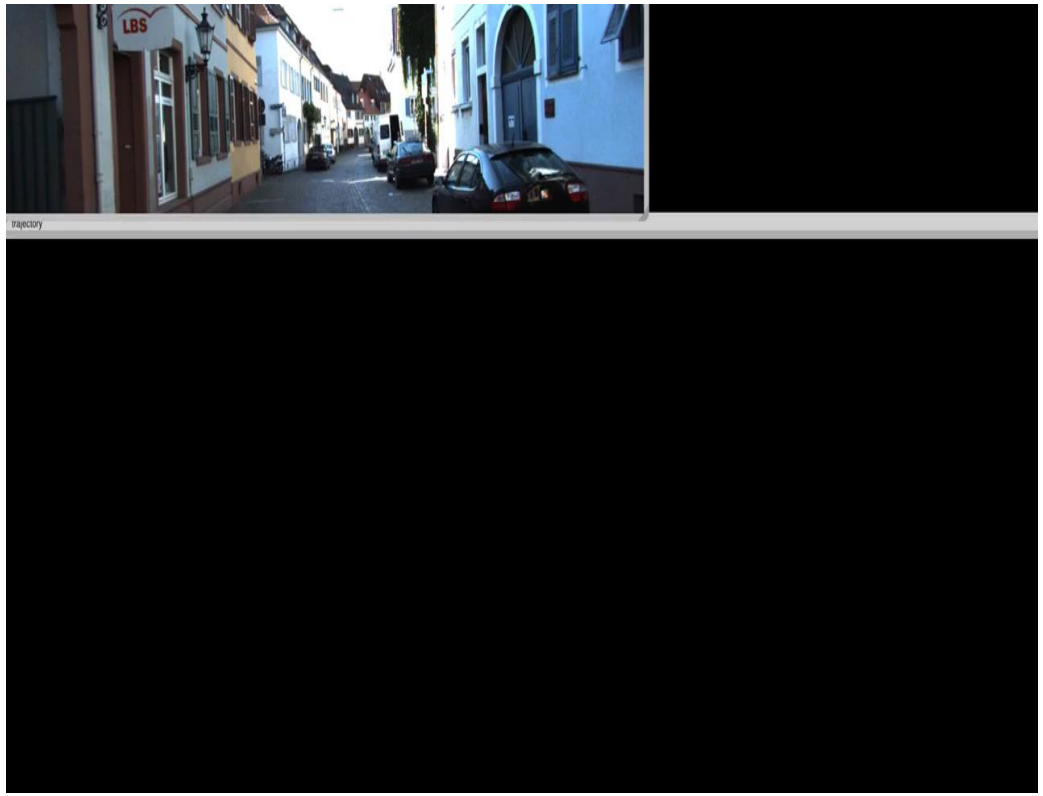


RQT_GRAPH


Lane Detection
/Lane_detection_node_11893_1603791002499
/lane_marker
/Plane_marker
Object Detection
/detection
/detection/lidar_detector
/lidar_point_pillars
/lidar_point_pillars_front
/detection/lidar_detector/objects
/detection/lidar_detector/objects_front
/detection/lidar_detector/point_pillars_visualization_01
/detection/lidar_detector/point_pillars_visualization_front
/detection/lidar_detector/objects_markers
/detection/lidar_detector/objects_markers_front
Tracking
/kf_tracker_node_11855_1603791001281
/kitti_box_track
/kitti_selfvelo_text_track
/kitti_ego_car_track
/kitti_objs_ori_track
/kitti_box_pictogram_track
/kitti_objs_velo_track
/pillar_marker_array
/kitti_warning_lines_track
/kitti_ego_innerCircle_track
/kitti_frame_seq
/kitti_ego_outCircle_track
/kitti_objs_path_track
SLAM
/imageProjection
/segmented_cloud_info
/ground_cloud_intensity
/segmented_cloud
/outlier_cloud
/featureAssociation
/laser_odom_to_init
/ground_cloud_intensity1
/laser_cloud_surf_last
/laser_cloud_corner_last
/outlier_cloud_last
/mapOptmization
/aft_mapped_to_init
/key_pose_origin
/registered_cloud
/transformFusion

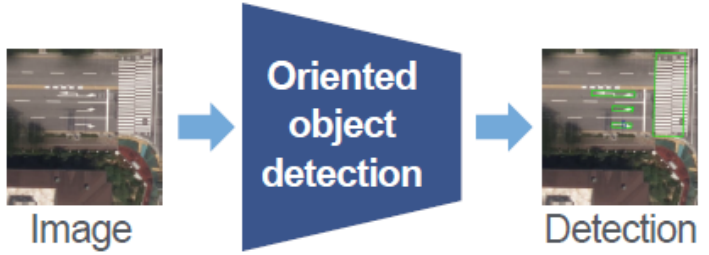
추가 연구 방향

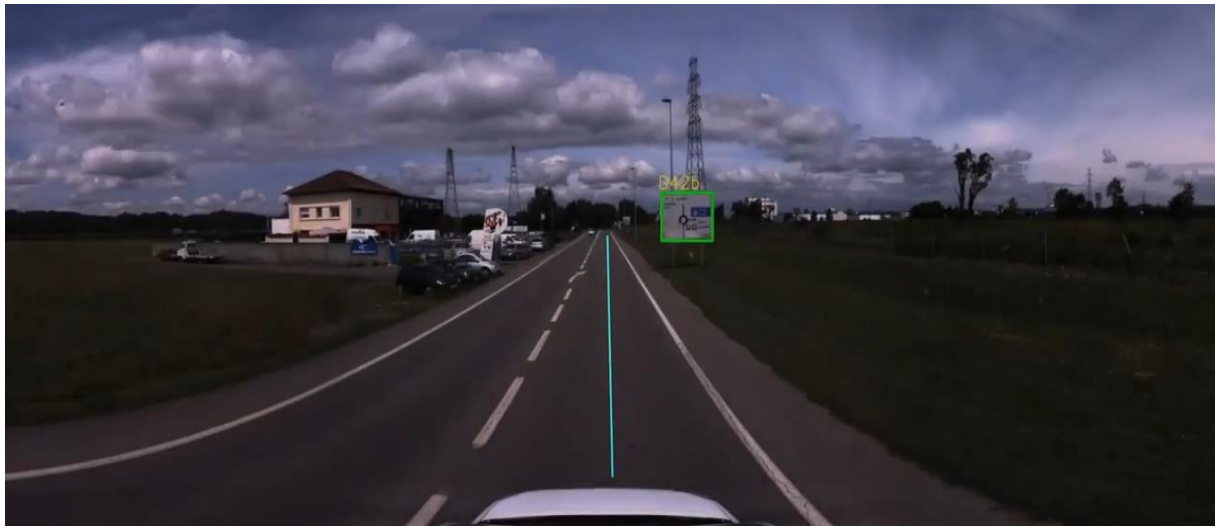
- **Sensor Fusion**
  - Image Pixel to Point data calibration
  - Visual & LiDAR Fusion SLAM



https://www.youtube.com/watch?v=jE-RNOiCjWM&ab_channel=KudanSLAM

- **HD Map Improvement**
  - Road Layout detection
  - Road Facilities and traffic sign Marking





https://www.youtube.com/watch?v=RIYDR8UE0hk&ab_channel=Geosat3D

활용 방안

Naver Labs M1, HD Mapping Robot

https://www.youtube.com/watch?v=BYWbf1iZ7co&ab_channel=NAVERLABS

Leica BLK2GO, Handheld Imaging Laser Scanner

https://www.youtube.com/watch?v=jJjAf79nmw0&ab_channel=Kuker-RankenInc.

참고 논문 출처

SLAM

"LOAM: Lidar Odometry and Mapping in Real-time", *Ji Zhang and Sanjiv Singh*, (2014)
"LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain", *shan Tixiao and Englot and Brendan*, (2018)
"Remove, then Revert: Static Point cloud Map Construction using Multiresolution Range Images", *Giseop Kim and Ayoung Kim, (*2020)

https://github.com/laboshinl/loam_velodyne
https://github.com/RobustFieldAutonomyLab/LeGO-LOAM

Object detection

"Pointpillars: Fast encoders for object detection from point clouds." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Lang, Alex H., et al.* (2019)

https://github.com/nutonomy/second.pytorch
https://github.com/seo-dev/PointPillars

Tracking

"3D Multi-Object Tracking: A Baseline and New Evaluation Metrics," *Xinshuo Weng, Jianren Wang, David Held and Kris Kitani*, (IROS 2020, ECCVW 2020)

https://github.com/xinshuoweng/AB3DMOT

Lane detection

어려운 고속도로 환경에서 Lidar를 이용한 안정적이고 정확한 다중 차선 인식 알고리즘, *이한슬, 서승우* (2015)

https://github.com/Lukas-Justen/Lane-Marking-Detection

감사합니다