

230324_김문갑

1. Replica City 시설물 갱신

a. 기존 프로그램 업데이트

1. 결과
2. 개요
3. 주요 업데이트 내용
4. 프로세스

b. 프로그램 입력방법 변경 (.pcd files → .pcap file)

2. ITSK 과제

a. 요약

3. To do

1. Replica City 시설물 갱신

a. 기존 프로그램 업데이트

1. 결과

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/a90e4fa2-1a61-4d85-be7d-985c94f87aec/3th_camera_advantage-2023-03-24_10.41.32.mp4

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/2cab2242-2255-4e3a-ad18-2c7cef7e4945/3th_camera_advantage-2023-03-24_10.55.58.mp4

2. 개요

a. 프로그램 입력 데이터

데이터(타입)

설명

.pcd files	라이다 점군 정보
.mic files	이미지 정보
detection files	이미지에서 얻은 바운딩박스 및 클래스네임 정보
calib.txt config.txt slam_coordinate.txt	lidar - camera calib lidar - imu imu - world(gps)
facility_info.txt	기존 시설물 객체 리스트

b. 프로그램 출력 데이터 : facility_info_updated.txt

c. 프로세스 요약

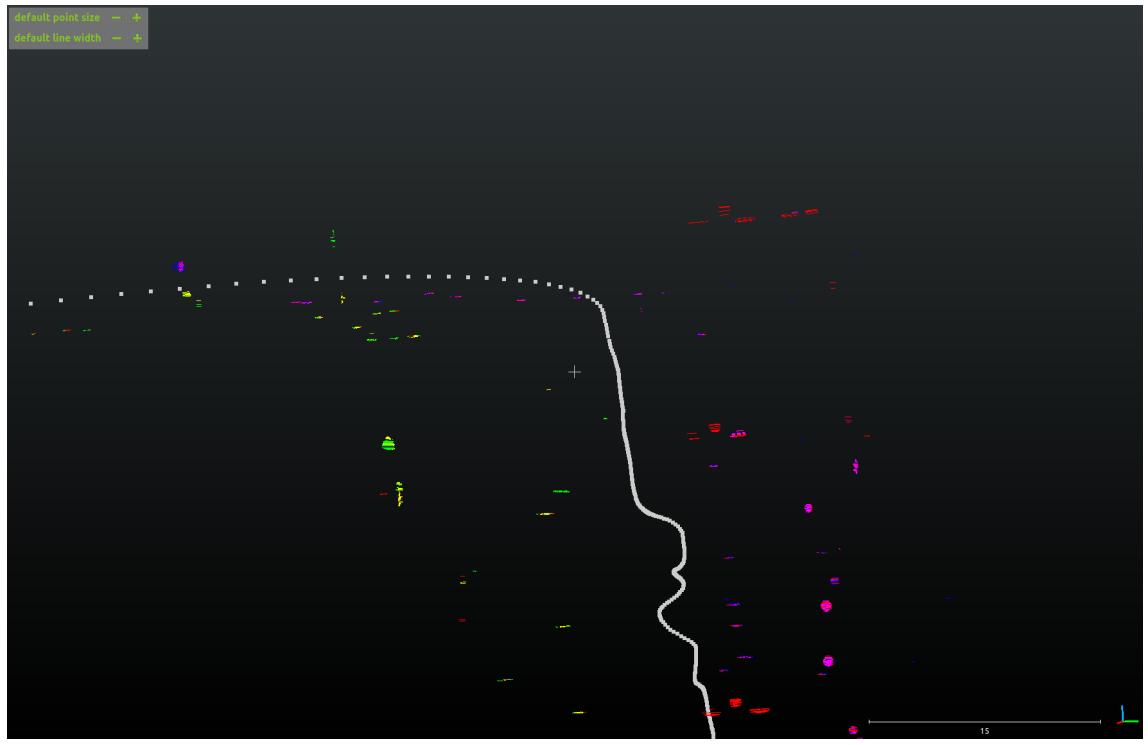
- i. 한 프레임(주행차 경로에서의 한 지점)에서 얻은 **라이다 포인트들을 카메라 이미지에 사영하여, 이 중 디텍션 결과 생성된 바운딩박스 안에 들어오는 포인트(즉, 디텍션정보를 담고 있는 포인트)들만 모은다.** 이때 포인트들의 위치를 라이다 좌표계에서 gps 좌표계로 변환 할 수 있다. 모든 프레임에 대하여 이 작업을 반복하여 디텍션정보를 담고있는 포인트들을 모은다.

```

class PointWithDetection
{
    Eigen::Vector3d m_pt_xyz;
    int m_data_frame_id;
    int m_cam_num;
    const DetectionInfo* m_detection_info;

    // ...
}

```



- ii. 모은 포인트들을 잘 분류하여 (공간상에서의 밀집정도, 클래스네임 등 포인트들의 속성을 활용) 시설물 객체로 지정한다

```

class ClusterInfo
{
    vector<PointWithDetection*> m_pts_with_det;
    string m_freq_classname;
    double m_freq_classname_score;
    double m_freq_classname_ratio;
    pcl::PointCloud<pcl::PointXYZI>::Ptr m_cloud;
    PointWithDetection* m_rep_pt_with_det;
    int m_level;
    ClusterInfo* m_parent;
    vector<ClusterInfo> m_children;

    // ...
}

```

iii. 위에서 새로 얻은 정보를 바탕으로 기존 시설물 리스트를 업데이트한다.

Key	Worldx	Worldy	Worldz	Classname	micFileName	Eventnumber	Posixtime	ImgX1	ImgY1	ImgX2	ImgY2	Time	Lat	Lon	Alt
2 1	205854.	363635	545654.	411441	16.819890	Hole	2302221414_M900C25L51G061_3267/01_camera/04/230222144813.mlc	18941	2023-02-22 05:48:13.643158(utc)	610	1807	1214	2109		
3 2	205481.	569303	545765.	421672	19.888024	TrafficLight_pedestrian	2302221414_M900C25L51G061_3267/01_camera/05/230222143825.mlc	16756	2023-02-22 05:49:56.964790(utc)	708	1824	1214	2109		
4 3	205763.	479713	545133.	306892	39.004107	TrafficLight_road	2302221414_M900C25L51G061_3267/01_camera/01/230222153758.mlc	34578	2023-02-22 06:40:18.671237(utc)	2956	1807	1214	2109		
5 4	205763.	179733	545134.	280619	38.896234	TrafficLight_pedestrian	2302221414_M900C25L51G061_3267/01_camera/04/230222157578.mlc	34590	2023-02-22 06:40:21.070902(utc)	519	979	1214	2109		
6 5	205733.	091964	545559.	162913	17.543355	Hole	2302221414_M900C25L51G061_3267/01_camera/02/230222158407.mlc	24153	2023-02-22 06:05:35.183085(utc)	1229	1913	1649	2089		
7 6	205987.	345016	544642.	518115	19.778762	TrafficLight_pedestrian	2302221414_M900C25L51G061_3267/01_camera/05/230222152911.mlc	31284	2023-02-22 06:29:20.382812(utc)	1614	1818	1214	2109		
8 7	205987.	387066	544642.	552781	21.218070	TrafficLight_road	2302221414_M900C25L51G061_3267/01_camera/01/230222152301.mlc	29095	2023-02-22 06:24:02.822826(utc)	536	3674	1214	2109		
9 8	205718.	916698	545561.	354267	19.629317	INST_cross_walk	2302221414_M900C25L51G061_3267/01_camera/05/230222144813.mlc	20015	2023-02-22 05:51:48.240802(utc)	701	1081	1123	2109		
10 9	205867.	720204	545579.	824679	21.945421	TrafficLight_pedestrian	2302221414_M900C25L51G061_3267/01_camera/04/230222153507.mlc	33549	2023-02-22 06:36:53.029753(utc)	1256	1540	944	2109		
11 10	205794.	979105	545861.	305931	38.570458	TrafficLight_pedestrian	2302221414_M900C25L51G061_3267/01_camera/04/230222152301.mlc	30156	2023-02-22 06:25:34.947799(utc)	913	697	1255	788		
12 11	205795.	265143	545060.	367012	38.500806	TrafficLight_road	2302221414_M900C25L51G061_3267/01_camera/01/230222152301.mlc	30150	2023-02-22 06:25:33.740763(utc)	517	3175	967	2109		
13 12	205703.	365853	545276.	572610	28.936599	Hole	2302221414_M900C25L51G061_3267/01_camera/04/230222153111.mlc	3254	2023-02-22 06:33:31.761077(utc)	701	1801	1115	2109		
14 13	205821.	998070	546187.	079546	22.662582	INST_straight_right	2302221414_M900C25L51G061_3267/01_camera/05/230222150207.mlc	23131	2023-02-22 06:02:10.945890(utc)	591	1801	1115	2109		
15 14	205822.	610811	546187.	517793	23.003103	TrafficLight_pedestrian	2302221414_M900C25L51G061_3267/01_camera/05/230222150207.mlc	23131	2023-02-22 06:02:10.945890(utc)	695	244	963	870		
16 15	205872.	368250	545636.	219466	21.553761	TrafficLight_pedestrian	2302221414_M900C25L51G061_3267/01_camera/04/230222144813.mlc	19138	2023-02-22 05:48:52.990329(utc)	713	704	958	2109		
17 16	205973.	314682	545637.	011092	21.642856	INST_bike_pedestrian	2302221414_M900C25L51G061_3267/01_camera/02/230222144813.mlc	19108	2023-02-22 05:48:46.999122(utc)	931	2915	1829	2109		
18 17	205365.	525994	546074.	840058	19.759697	TrafficLight_pedestrian	2302221414_M900C25L51G061_3267/01_camera/05/230222145543.mlc	22881	2023-02-22 06:01:20.960014(utc)	2037	754	2133	994		
19 18	205367.	772581	546079.	580867	19.892510	TrafficLight_pedestrian	2302221414_M900C25L51G061_3267/01_camera/04/230222145543.mlc	22896	2023-02-22 06:01:23.958060(utc)	1807	1807	1214	2109		

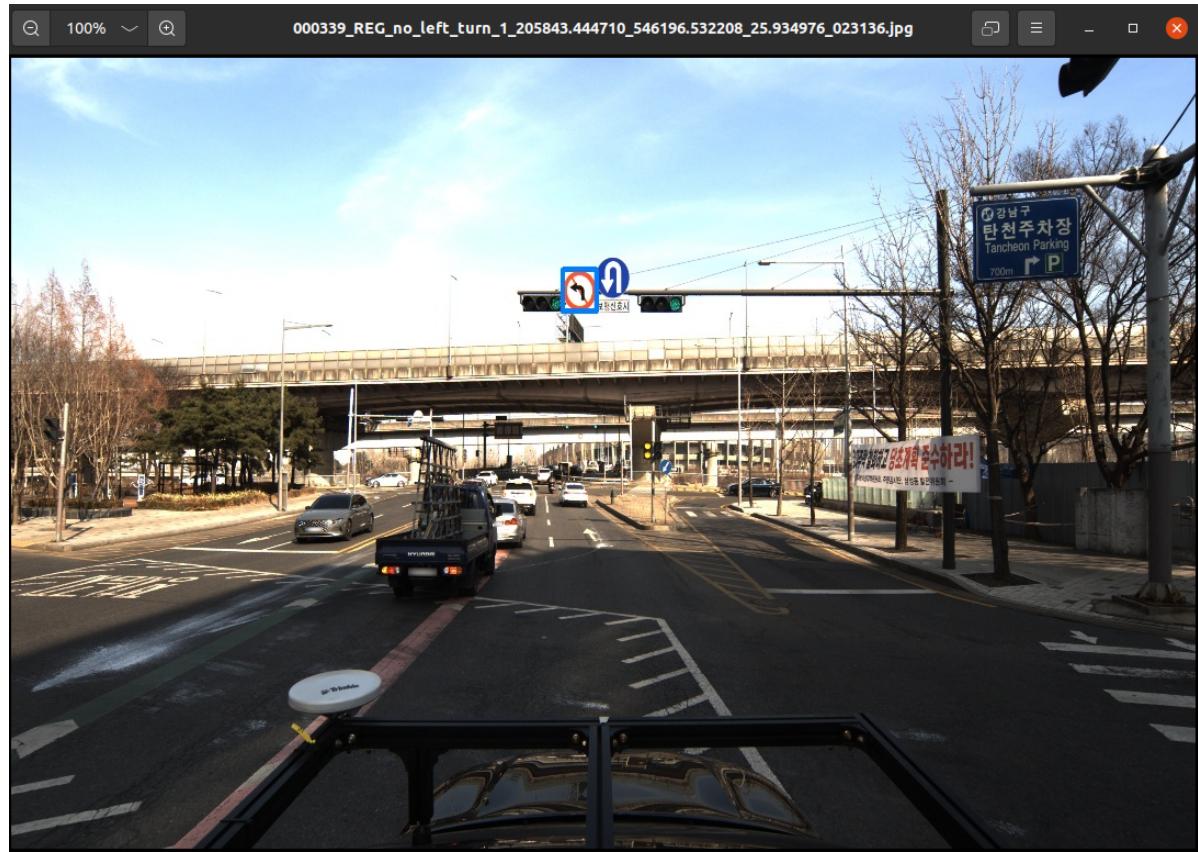


3. 주요 업데이트 내용

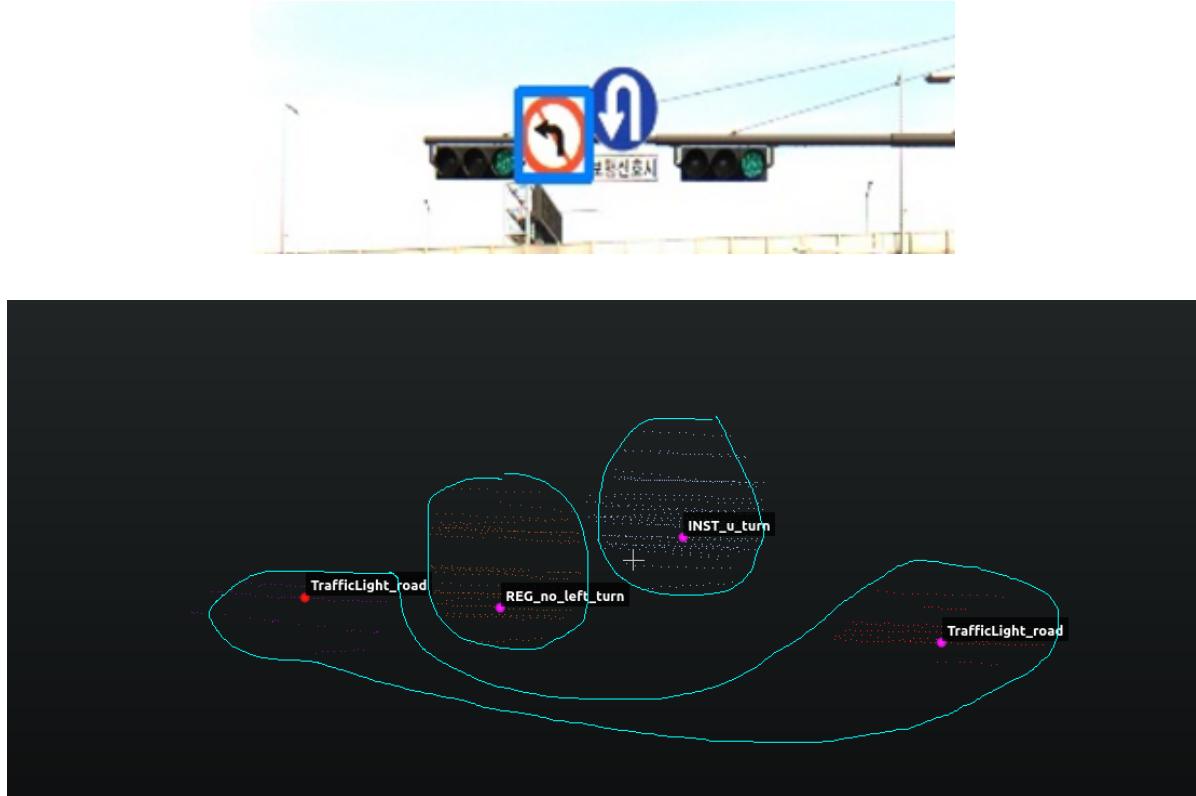
- 여러개의 카메라에서의 이미지들을 한번에 입력값으로 받는 기능을 추가. 원하는 번호의 카메라를 옵션으로 선택하면, 해당번호의 카메라 이미지와 디텍션 결과만 입력값으로 받게 된다.
- 위치정보와 디텍션정보를 담고있는 포인트들을 적절히 분류하여 시설물객체들로 클러스터링 하는 알고리듬 개선
- 이미지 디텍션 결과가 정확하지 않은 경우 발생할 수 있는 문제를 막기 위한 디텍션 결과 검증 방법 추가
- 기존 시설물 객체 리스트(DB)와 취득데이터를 통해 얻은 새로운 객체 리스트를 비교하는 방법 개선

4. 프로세스

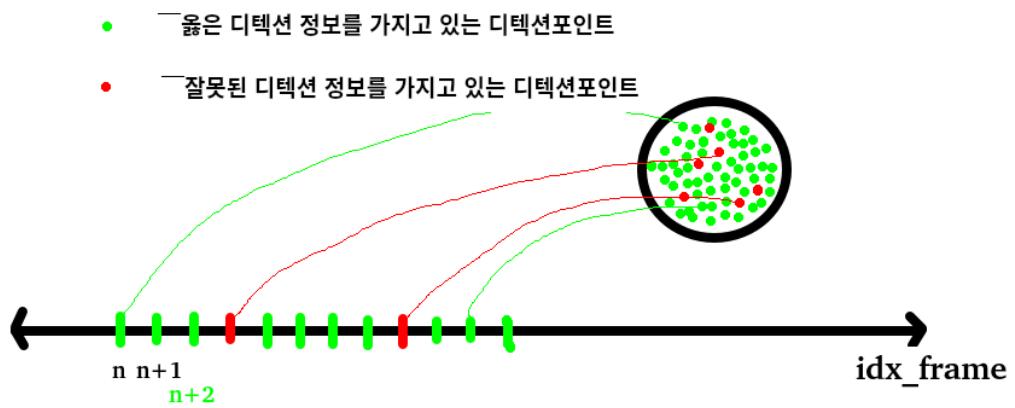
- 디텍션정보를 갖고있는 모든 포인트들을 대상으로 Euclidean Clustering 적용



- b. 포인트 사이의 거리에 따라 분류된 각각의 클러스터들에 대하여 클래스네임을 기준으로 재분류한다. 이미지 디텍션 결과가 정확하지 않은 경우 발생할 수 있는 문제를 막기 위한 디텍션 결과 검증방법 추가



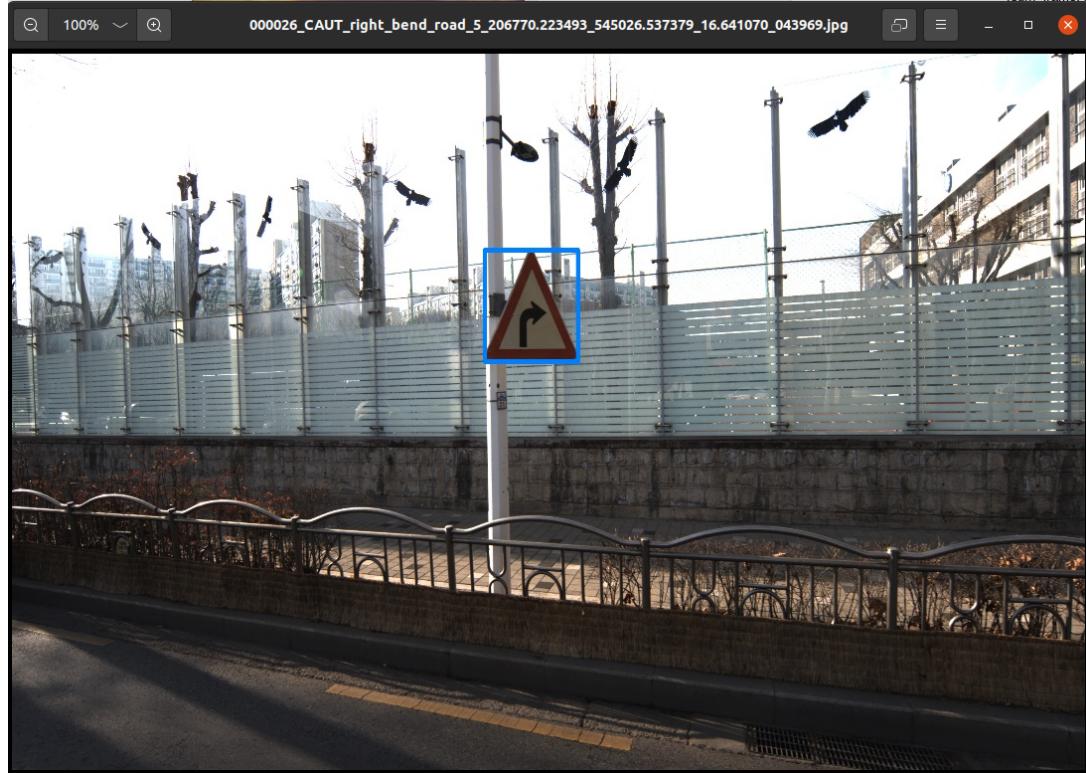
▼ 디텍션이 잘못되면?



1. 클래스네임별로 score 값을 구하여 전체 대비 비율이 기준치 이하이거나 해당 클러스터와 연관된 프레임 개수가 기준치 이하이면 잘못된 디텍션으로 간주한다.

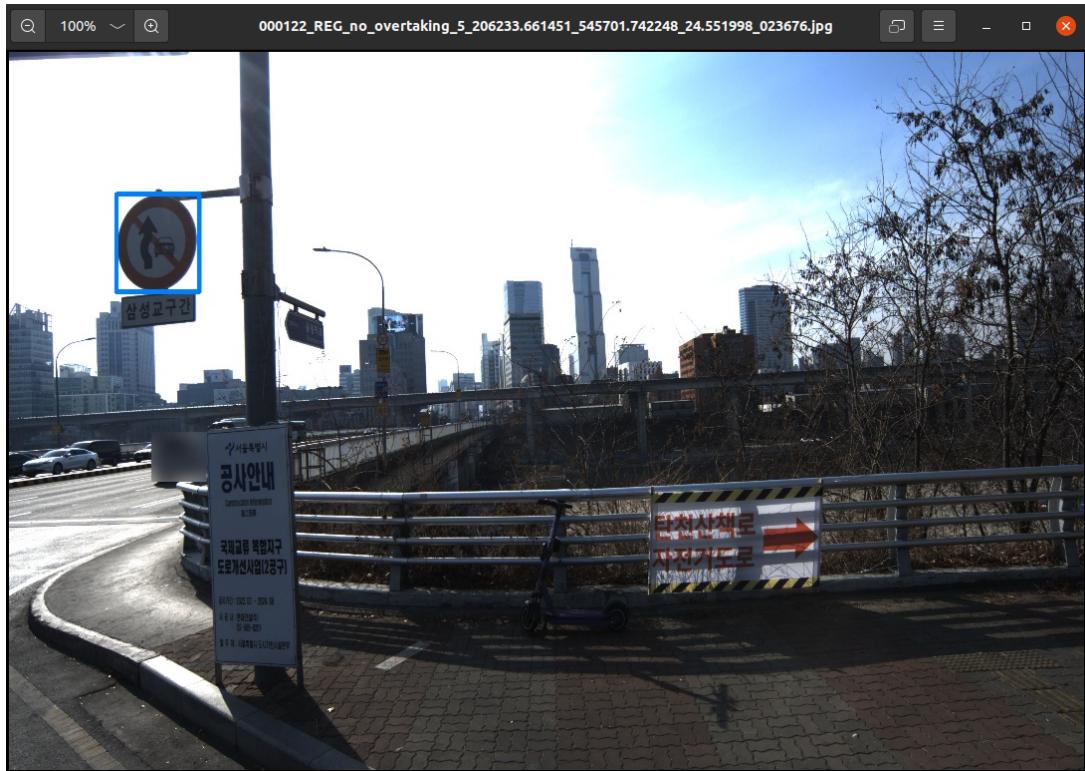
▼ ex 1. ratio < ratio_threshold = 0.04

```
20th cluster info :  
(classname, score, count) = (CAUT_right_bend_road, 0.9956101312, 6358)  
(classname, score, count) = (CAUT_trafficlight_ahead, 0.0043898688, 34)  
ratio is less than 0.0400000000  
(sum_frame, n_max_frame) : ( 56, 17)  
list_detection_status : ( true)
```



▼ ex 2. n_frame < frame_threshold

```
99th cluster info :  
(classname, score, count) = (REG_no_overtaking, 0.9304105606, 2614)  
(classname, score, count) = (REG_no_bike, 0.0695894394, 195)  
(sum_frame, n_max_frame) : ( 36, 18)  
(sum_frame, n_max_frame) : ( 1, 1), sum_frame < 3, n_max_frame < 2  
list_detection_status : ( true, false)
```



2. 클래스네임으로 분류된 각각의 클러스터들에 대하여, 해당 클러스터들과 연관된 프레임 인덱스들을 비교하여 중복되는 프레임 인덱스를 가지고 있는지 확인한다. 만약 있다면, 해당 프레임에 2개의 서로 다른 클래스네임 정보를 모두 포함하고 있기 때문에 디텍션이 정상적으로 된것으로 판단한다.

▼ ex 1. 중복되는 프레임 인덱스를 가지고 있을때

```

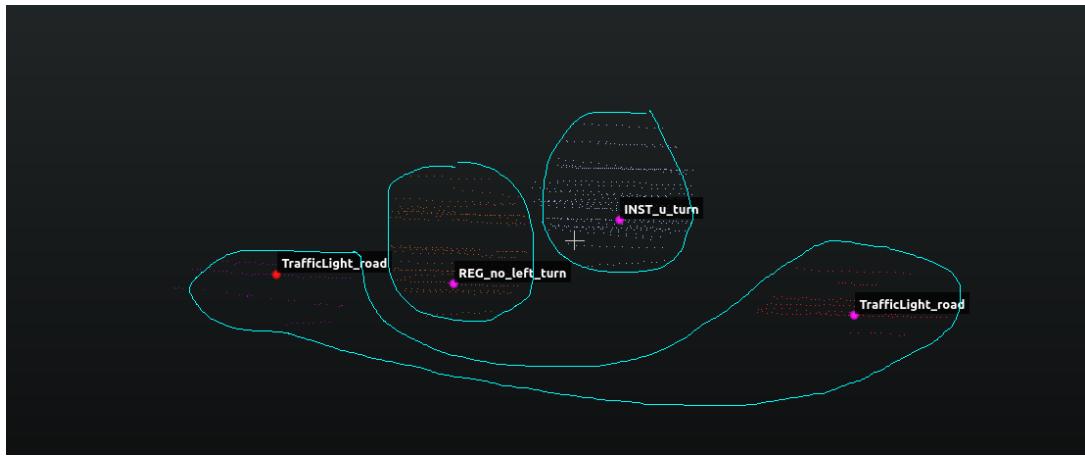
289th cluster info :
(classname, score, count) = (INST_u_turn,  0.4587815593,   593)
(classname, score, count) = (REG_no_left_turn,  0.3453949098,   446)
(classname, score, count) = (TrafficLight_road,  0.1958235310,   268)
    (sum_frame, n_max_frame) : ( 32,   14)
    (sum_frame, n_max_frame) : ( 25,   12)
    (sum_frame, n_max_frame) : (  9,    7)

// a 와 b 비교
cam_num : 0, (n_low, n_high, n_intersection) = (13, 12, 9) break

// a 와 c 비교
cam_num : 0, (n_low, n_high, n_intersection) = (13, 7, 4) break

// b 와 c 비교
cam_num : 0, (n_low, n_high, n_intersection) = (12, 7, 5) break
list_detection_status : ( true,  true,  true)

```



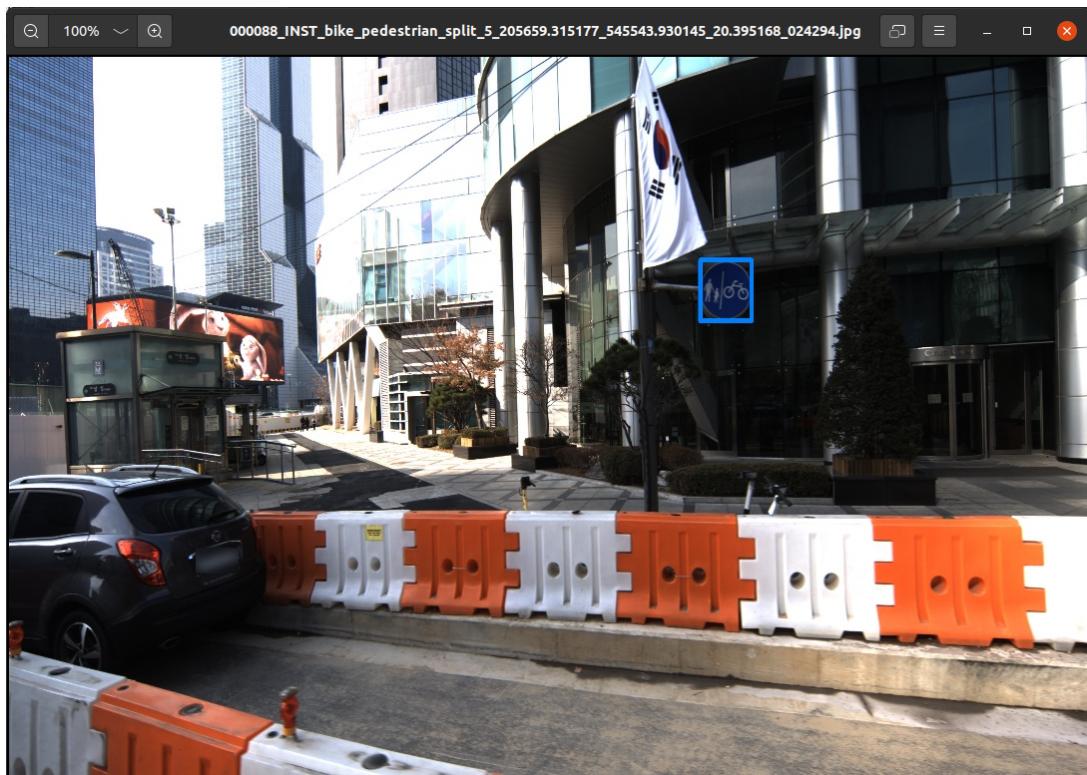
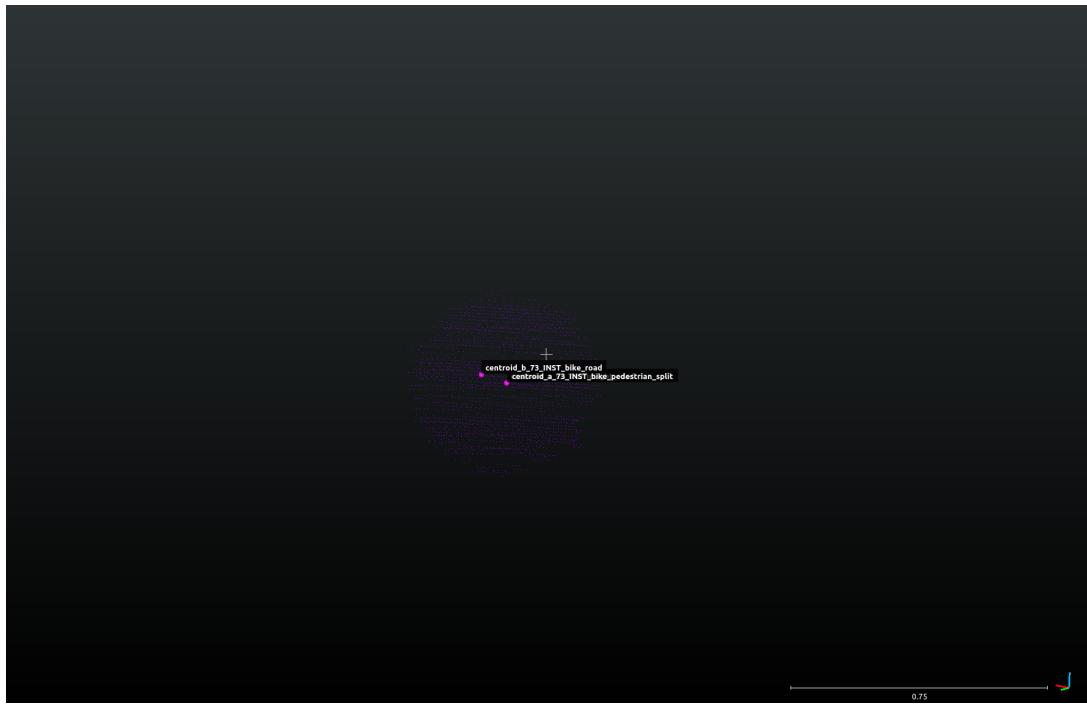
3. 클래스네임으로 분류된 두 개의 클러스터를 비교하였을 때 중복되는 프레임 인덱스가 없다면, 마지막으로 **클러스터간의 무게중심을 비교한다.** 두 포인트의 거리가 일정거리 이하이면, 두 클러스터들은 같은 위치에 있으므로 디텍션이 잘못된 것으로 판단한다.

▼ ex 1. $\text{dist}_{\text{betw_clusters}} < \text{cluster_tolerance} = 0.25$

```

73th cluster info :
(classname, score, count) = (INST_bike_pedestrian_split, 0.9001958833, 3117)
(classname, score, count) = (INST_bike_road, 0.0998041167, 349)
    (sum_frame, n_max_frame) : ( 47, 28)
    (sum_frame, n_max_frame) : ( 9, 5)
cam_num : 0, (n_low, n_high) = (7, 5)
cam_num : 1, (n_low, n_high) = (0, 0)
cam_num : 2, (n_low, n_high) = (0, 0)
cam_num : 3, (n_low, n_high) = (12, 2)
cam_num : 4, (n_low, n_high) = (28, 2)
dist_clusters : 0.0776454366
list_detection_status : ( true, false)

```



▼ ex 2. dist_betw_clusters > cluster_tolerance = 0.25

```

139th cluster info :
  (classname, score, count) = (REG_slow,    0.4307286796,    955)
  (classname, score, count) = (REG_high_speed_limit,   0.3481044308,    754)
  (classname, score, count) = (CAUT_left_bend_road,   0.2211668895,    484)
    (sum_frame, n_max_frame) : ( 10,     9)
    (sum_frame, n_max_frame) : ( 16,    11)
    (sum_frame, n_max_frame) : (  9,     9)

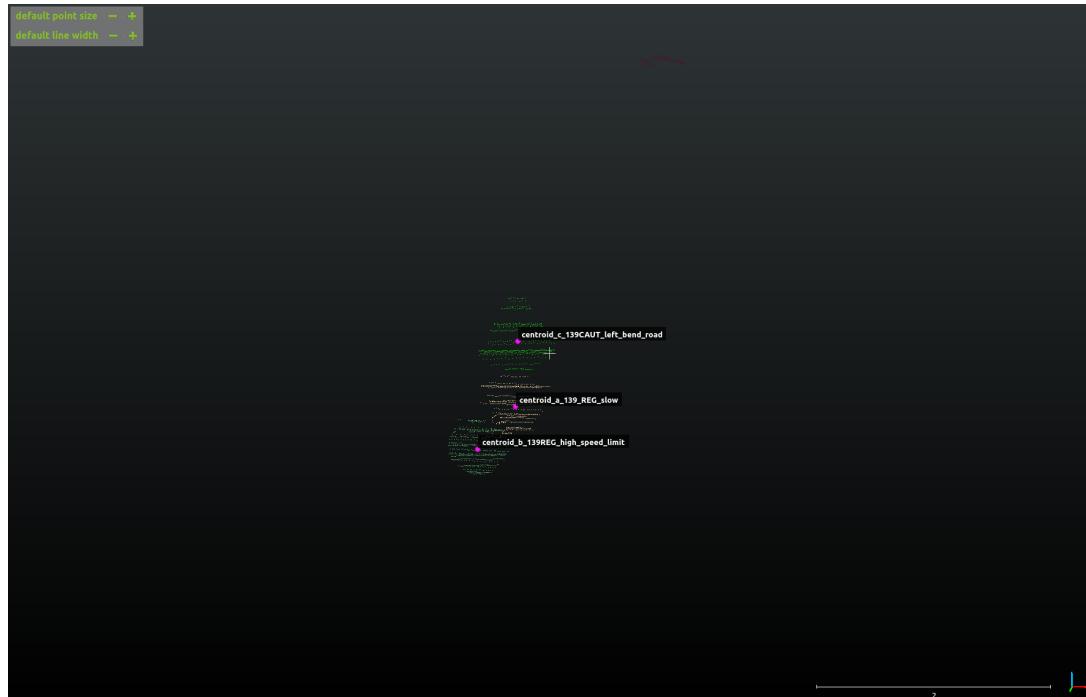
  // a 와 b 비교
  cam_num : 0, (n_low, n_high) = (9, 0)
  cam_num : 1, (n_low, n_high) = (0, 0)
  cam_num : 2, (n_low, n_high) = (0, 5)
  cam_num : 3, (n_low, n_high) = (0, 11)
  cam_num : 4, (n_low, n_high) = (1, 0)
  dist_clusters : 0.8685356314

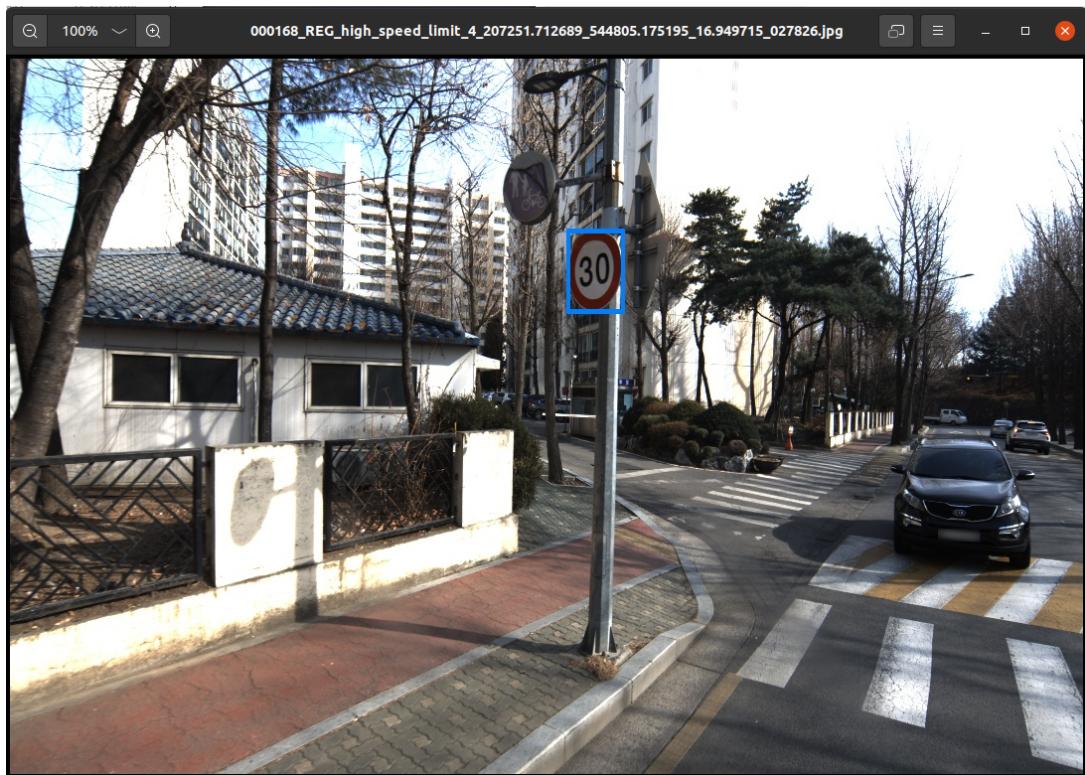
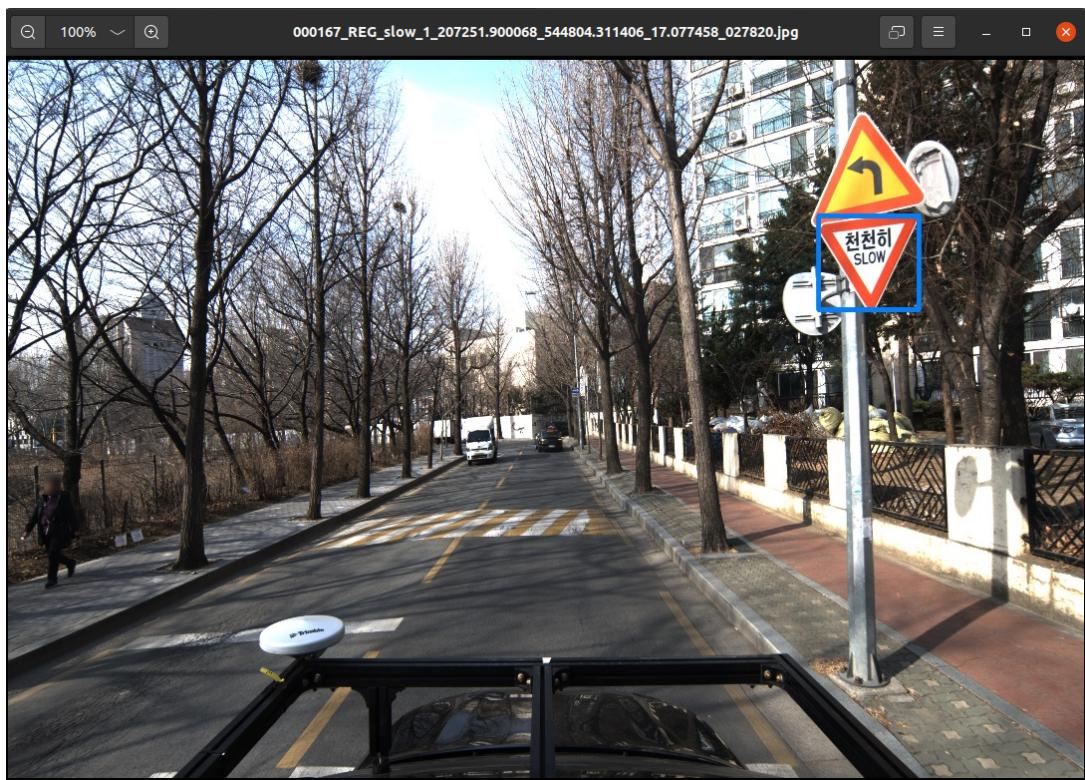
  // a 와 c 비교
  cam_num : 0, (n_low, n_high, n_intersection) = (9, 9, 7) break

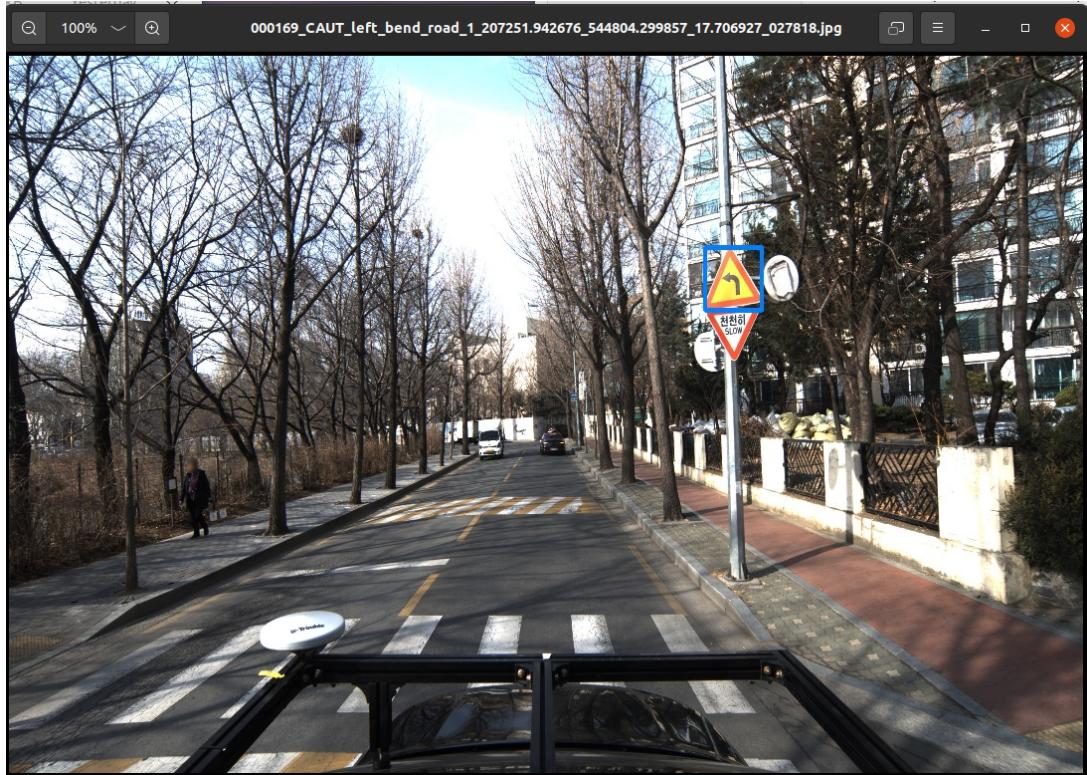
  // b 와 c 비교
  cam_num : 0, (n_low, n_high) = (0, 9)
  cam_num : 1, (n_low, n_high) = (0, 0)
  cam_num : 2, (n_low, n_high) = (5, 0)
  cam_num : 3, (n_low, n_high) = (11, 0)
  cam_num : 4, (n_low, n_high) = (0, 0)
  dist_clusters : 1.1311788053

  list_detection_status : ( true,  true,  true)

```

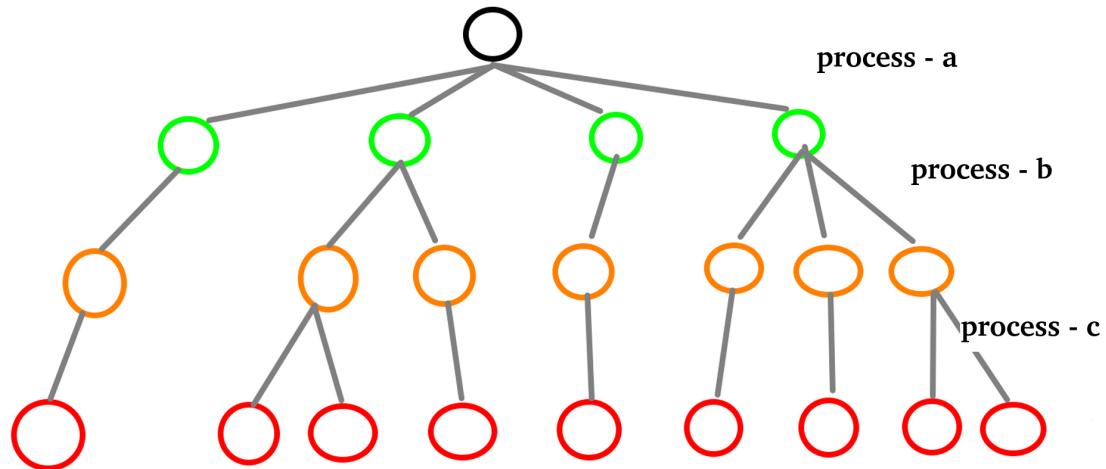
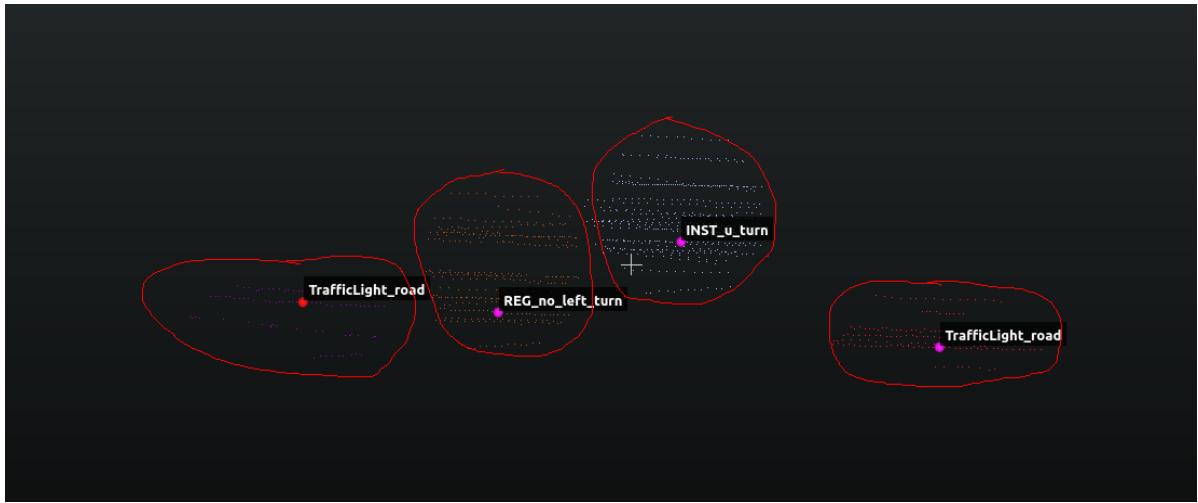






c. 클래스네임별로 분리된 클러스터들 각각에 대하여 Euclidean Clustering을 적용한다. 이때 생성된 클러스터들이 최종 시설물 객체가 된다. 그리고 해당 클러스터를 대표하는 포인트 및 이미지를 찾는다.





d. 기존 시설물 객체 리스트(A)와 취득데이터를 통해 얻은 새로운 객체 리스트(B)를 비교

- B에 속하는 임의의 객체 b에 대하여, 같은 클래스네임을 가지고 가장 가까이에 있는 A에 속하는 객체 a를 찾는다.
 - 만약 일정거리(threshold) 안에 있는 객체 a가 없다면, 기존에 없었는데 새로 생긴 객체로 간주한다
 - 만약 일정거리(threshold) 안에 있는 객체 a가 여러개라면, 그 중에서 가장 가까운 것 (a_0)과 매칭하여, 키값을 제외한 기존 a_0 의 정보를 b의 정보로 교체한다.
- B에 속하는 모든 객체 b에 대하여 이를 반복하여 객체들을 세가지 범주로 분류할 수 있다
 - 매칭된 객체

- 기존 리스트에만 있는 객체
- 새로운 리스트에만 있는 객체 (새로운 키값 부여)

이를 종합하여 기존 시설을 객체 리스트를 업데이트한다

- 문제점 : i 단계를 수행할 때 어떤 객체를 먼저 선택하는지에 따라서 결과값이 달라질 수 있다

b. 프로그램 입력방법 변경 (.pcd files → .pcap file)

1. 추진배경

- a. 기존에는 시설을 갱신을 위해서 .pcd 파일들이 반드시 필요했다. 한번 취득할때마다 여러 pcd 파일들이 생겨 데이터 보관 및 유지에 다소 불편함이 있었다.
- b. 주행차에서 데이터를 취득할 때 raw data로 pcap 파일을 생성하는데, 여기에 누적된 라이다포인트 정보가 담겨있다. 이를 바로 시설물갱신 프로그램의 입력값으로 활용하면 시간 및 저장공간을 절약할 수 있다.

2. 결과

- a. pcap 파일에서 라이다포인트 정보를 가져올 수 있도록 라이브러리화된 소스코드를 MMS 팀에서 받았다. 이를 활용하여 pcap 파일을 입력으로 받아 시설물 갱신이 가능해졌다.
- b. 이번에 작업한 레플리카 시설물갱신 프로그램에 pcap파일 입력 옵션 기능을 추가로 구현 하려고 했는데, 기존 프로그램과 mms팀 코드의 타입 및 구조가 달라 결국 pcap 버전과 .pcd 버전의 프로그램을 각각 만들었다.

3. 향후계획

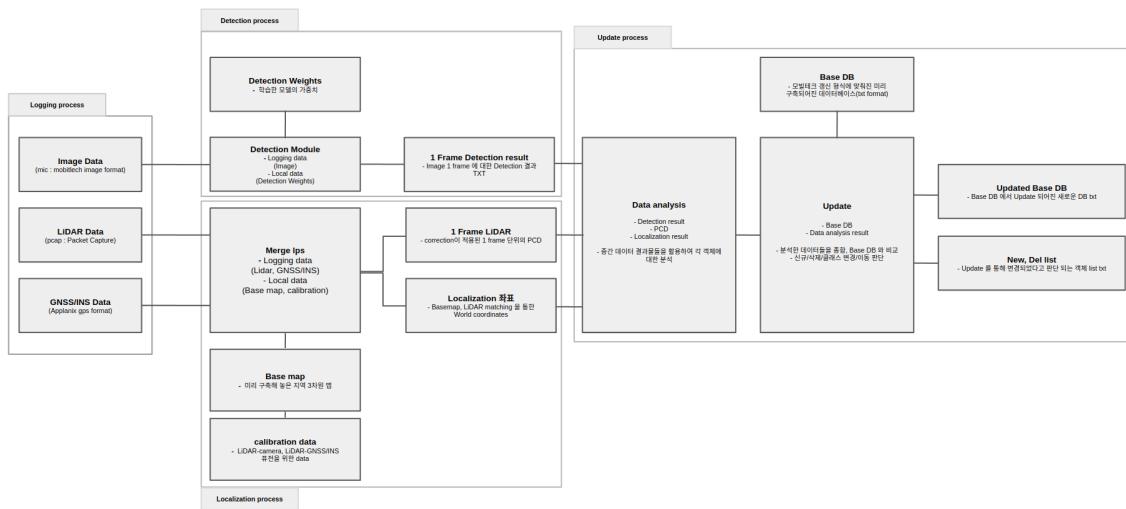
- a. 해당 프로그램은 포항 과제에서 포항 전역에 시설물 정보 리스트를 만드는데 사용될 예정이다.

2. ITSK 과제

a. 요약

1. 변화탐지 소프트웨어 기능
 - a. 도로를 주행하여 얻은 로우 데이터(라이더점군, 이미지, gps정보)를 입력으로 받아 시설물 객체 정보를 나타내는 프로그램.
 - b. 기존 시설물 정보(.txt파일)와 취득한 데이터를 통해 얻은 시설물 정보를 비교하여 새로운 시설물 정보를 최신화함

2. 프로그램 프로세스

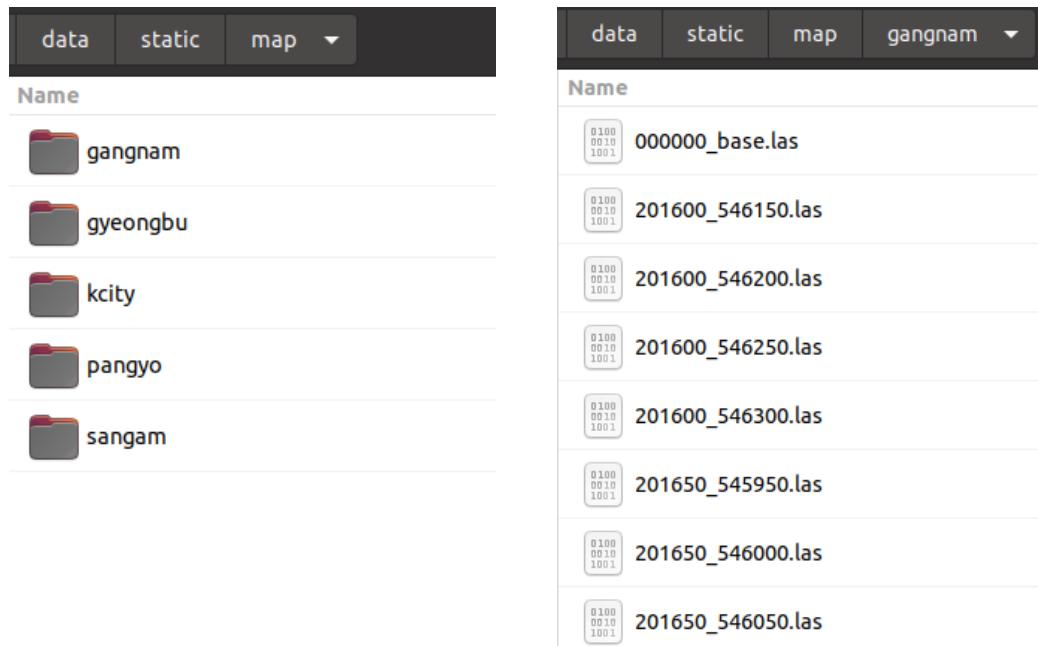


3. ITSK 중간 요구사항 (2022.12.26)

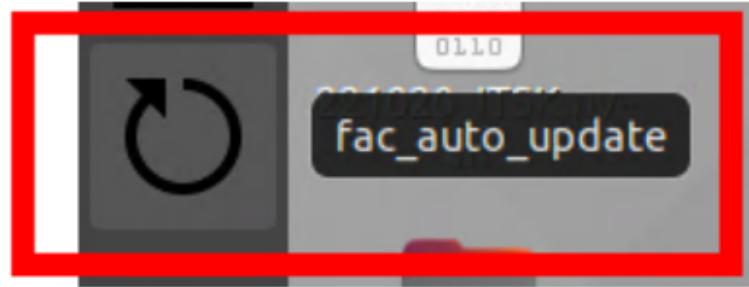
- a. 5개의 지역(강남, 상암, 경부고속도로, 판교, kc-city)에서 프로그램을 돌려야 하는데, 프로그램을 실행할 때마다 지역명을 입력하는게 번거롭다.
- b. 리눅스 터미널에서 명령어를 입력하는 방법 대신 간단하게 프로그램 더블클릭으로 실행할 수 있으면 좋겠다. 유저 입장에서는 너무 불편하다.

4. 해결방법

- a. 변화탐지 소프트웨어에 내장된 5개 지역 기준맵과 주행자동차의 궤적을 활용하여 맵 지역을 활용하여 주행지역을 자동으로 선택
 - i. 주행차의 궤적은 여러개의 포인트(x, y)로 이루어져 있음. 이 중에서 랜덤으로 100 개를 뽑는다.
 - ii. 뽑은 각 점이 5개 각각의 지역에 속하는지 확인하고, 맞으면 카운트한다.
 - 201600_546150.las 파일이 있으므로 $201600 \leq x < 201600 + 50$ & $546150 \leq y < 546150 + 50$ 에 속하는 점 (x,y)는 강남 지역 안에 있음을 알 수 있음
 - iii. 가장 많이 카운트된 지역을 정답으로 선택한다 (이후 해당지역의 기준맵을 적용하여 로컬라이저 모듈을 실행한다).



- b. 시작메뉴에 프로그램 바로가기 아이콘을 추가함



1. 저장공간에 남아있는 취득데이터를 취득시간 순서대로 정렬한다.
2. 현재 저장된 시설물 객체 DB의 생성날짜의 마지막 업데이트 시간을 고려하여 그 시간 이후의 취득데이터들을 입력값으로 하여 프로그램을 순차적으로 실행한다.

3. To do

- a. 과제 : 포항, UOK, 무한
- b. 시설물 간신 프로그램 고도화
- c. 연구주제 탐색