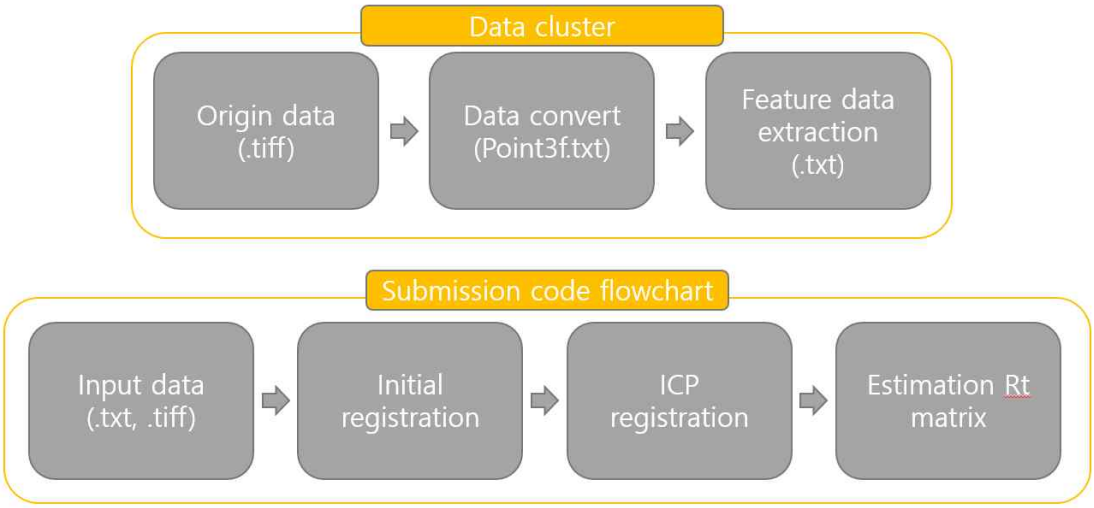
Coding Test - 3D Part

1. 개요
   - 목적 : real data와 master data를 이용하여, merged 3D 하기 위한 transformation matrix를 구하시오.
   - 주요 기능 : PCL를 이용하여 point cloud간 ICP
   - 사용 기술 : opencv, PCL, CloudCompare(tool)

2. 시스템 구성



   - 노이즈 제거 및 notch 영역을 추출하기 위해 DBSCAN 방법을 시도했으나, Rt결과가 정밀하지 않음.
   - 원본 데이터 용량이 큰 관계로 CloudCompare를 이용하여 real - master 데이터간 매칭될 수 있는 notch 영역 데이터를 따로 추출.
   - 해당 데이터들을 기반으로 ICP를 이용하여 정합.
   - Transformation matrix 추정.

3. 단계별 설명
   - Notch영역 및 실측 데이터 로드

```
/// TOP Notch data File read
// Notch 데이터 경로
string segment_realTopPC_path = "../Data/NotchSeg_realTopPC.txt";
string segment_idealTopPC_path = "../Data/NotchSeg_idealTopPC.txt";
pcl::PointCloud<pcl::PointXYZ>::Ptr realTopCloud = ReadTxtFiletoCloud(segment_realTopPC_path);      // real top point cloud
pcl::PointCloud<pcl::PointXYZ>::Ptr idealTopCloud = ReadTxtFiletoCloud(segment_idealTopPC_path);    // ideal top point cloud

/* ... */

/// Real data File read
// Real data 경로
string Real_Top_path = "../Data/REAL_TOP.tiff";            // 8227350
vector<Point3f> realTopPC;
int realTopPC_size = ReadRealFile(Real_Top_path, &realTopPC);
```

- 초기 정렬 적용

```
// 초기 정렬 행렬
Eigen::Affine3f TOP_initial = Eigen::Affine3f::Identity();
TOP_initial.translation() << 1.3, -165.0, 1.0;
Eigen::Matrix4f T_ideal_top;
T_ideal_top << -1, 0, 0, 7, 0, -1, 0, 537, 0, 0, 1, 0, 0, 0, 0, 1;
```

- ICP를 이용한 두 데이터간 정합

```
void ICPRegistration(
    pcl::PointCloud<pcl::PointXYZ>::Ptr notchRealCloud,
    pcl::PointCloud<pcl::PointXYZ>::Ptr notchIdealCloud,
    const vector<Point3f>& realPC,
    const Eigen::Affine3f& T_initial,
    const Eigen::Matrix4f& T_ideal,
    const string& outputFileName)
{
    // 초기 정렬 적용
    pcl::transformPointCloud(*notchRealCloud, *notchRealCloud, T_initial);


    pcl::IterativeClosestPoint<pcl::PointXYZ, pcl::PointXYZ> icp;
    icp.setInputSource(notchRealCloud);
    icp.setInputTarget(notchIdealCloud);

    pcl::PointCloud<pcl::PointXYZ> Final;
    icp.align(Final);


    if (!icp.hasConverged())
    {
        cout << "failed" << endl;
        return;
    }

    cout << "ICP Transformation Matrix : " << endl << icp.getFinalTransformation() << endl;
```

- 변환 행렬을 적용하여 real data -> master data 변환

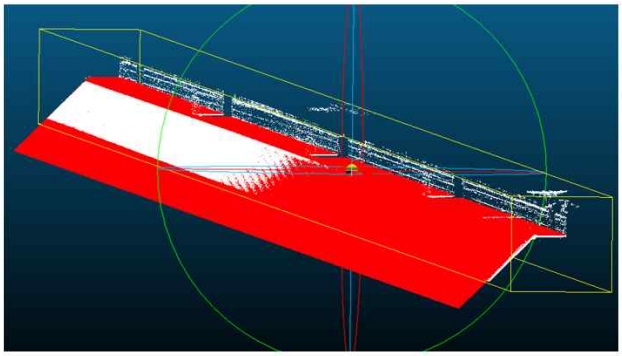
```
    Eigen::Matrix4f RT = icp.getFinalTransformation();
    Eigen::Matrix4f T_final = RT * T_initial.matrix();
    Eigen::Matrix4f T_merged = T_ideal * T_final; // transformation matrix to merged 3D(real data -> ideal data -> origin frame)


    cout << "merged 3D Transformation Matrix(Top) : " << endl << T_merged << endl;


    vector<Point3f> transformedPC;
    transformedPC.reserve(realPC.size());

    for (const auto& pt : realPC)
    {
        Eigen::Vector4f pt_h(pt.x, pt.y, pt.z, 1.0f);
        Eigen::Vector4f pt_transformed = T_final * pt_h;
        transformedPC.push_back(Point3f(pt_transformed[0], pt_transformed[1], pt_transformed[2]));
    }
```

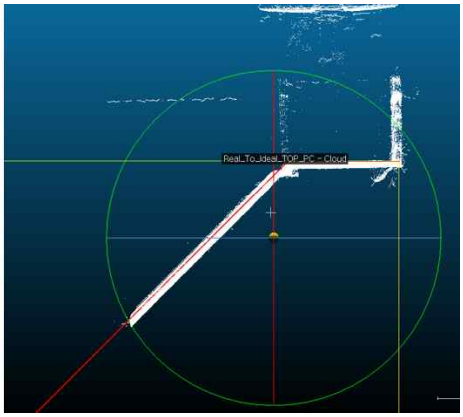
- 변환된 point cloud 저장 및 3D merged transformation matrix 출력

```
    // saved read -> ideal Point Cloud
    savePointCloudTotxt(transformedPC, outputFileName);
    cout << "Real to Ideal Transformed Point Cloud saved : " << outputFileName << endl;
}
```

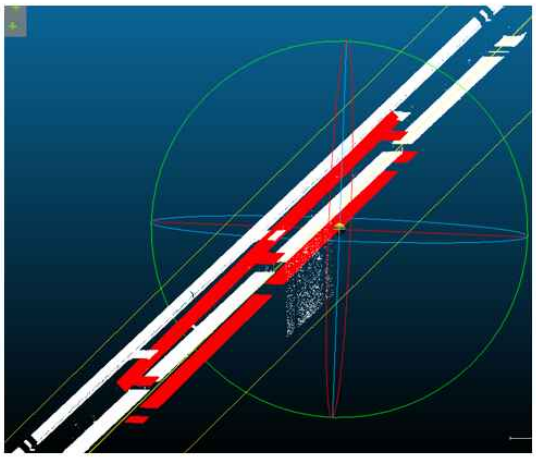
4. 결과 분석

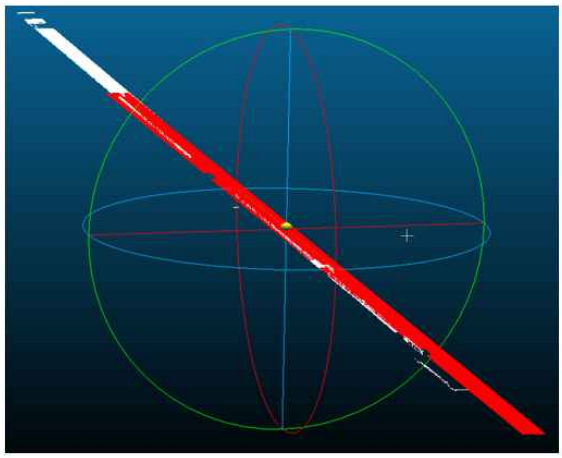
- Top view 기준 정합 결과





- Side & Bottom 정합 결과





Side, Bottom

5. 한계점

- 하나의 프로젝트에서 매칭을 위한 feature extration 기능 구현x
- notch(feature) 및 최초 정렬을 위한 Rt가 정밀하지 못함
- 다른 알고리즘이나 파라미터 수정을 진행하여 정밀한 Rt을 구하지 못함(top view 제외 나머지 view에 대한 정합 실패)
- file road에 대한 Hardcoded path