중간 보고서

전기컴퓨터공학부 정보컴퓨터공학전공

201646104 기태욱

201733111 서준오

201924549 이해성

팀명 : 점쟁이

분과 : A분과

과제명 : 포인트 클라우드를 이용한 3D 공간 생성/복원

지도교수 : 이 명 호

목차

1. 요구조건 및 제약 사항 분석에 대한 수정사항

1-1. 요구조건

PointCloud 모델에서 천장, 벽, 바닥을 탐색하고, 실내의 작은 물체, 노이즈 등 복원이 필요 없는 물체들을 검출하여 제거하고 제거된 영역을 다시 채우는 방법을 연구한다.

- 천장, 벽, 바닥을 추출하기 위한 알고리즘 구현
- 작은 물체, 가구들을 분류 및 처리 기능 구현
- 텍스쳐 처리 및 제거된 영역 복원 기능 구현

1-2 제약사항 분석에 대한 수정사항

pointcloud.ply

cropped2.ply

pointcloud.ply boundary point

cropped2.ply boundary point

<그림1> - 전체 데이터와 구역별로 분리 된 후 Boundary Point를 검출한 결과

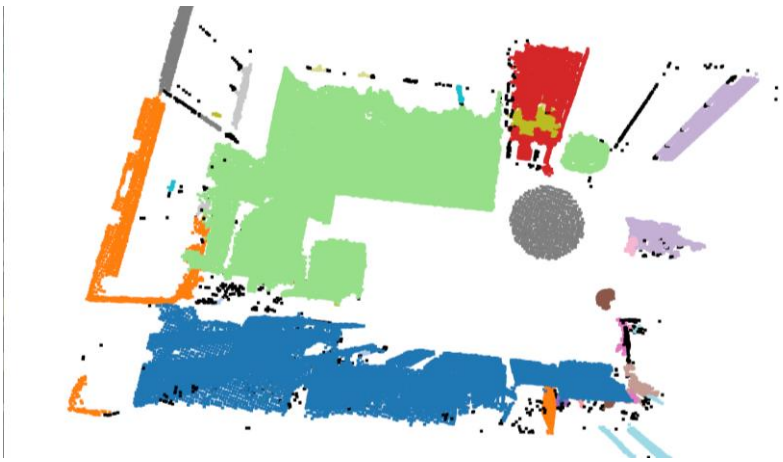
1. 처리되는 공간 스케일 한정

<그림1>과 같이 3D데이터 모델 전체를 기준으로 처리했을 때 정확도 및 처리 결과의 퀄리티가 낮으므로, 공간을 구역 별로 분리해 높은 정확도와 퀄리티를 얻는다.

<그림2> - 유리창에 반사되어 생성된 노이즈 Point Cloud Data

2. 수집된 데이터의 한계점

<그림2>의 화장실의 거울에 반사된 공간과, 베란다의 유리창의 애매모호한 경계면 등 불분명한 외부의 공간 처리에 어려움이 있다. 따라서, 데이터가 분명한 섹터에 중점을 둘 예정이다.



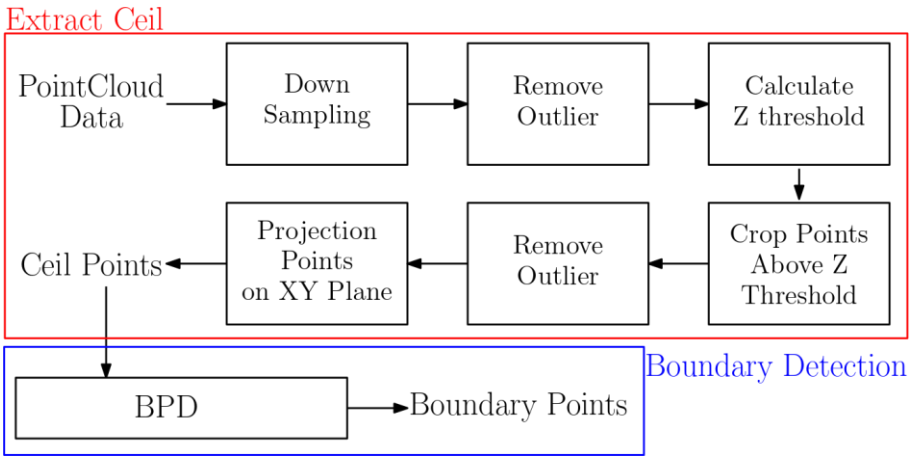
<그림3> - 천장, 바닥, 벽 제거 후 남은 가구들을 DB Scan을 통해 분류한 화면

3. 인접한 가구들이 하나의 가구로 인식됨

open3d에서 제공하는 DBScan메서드를 활용하여 가구를 분류하면 인접한 정도가 큰 가구들은 하나의 객체로 취급하여 <그림3>과 같이 복수의 가구가 하나의 덩어리로 인식되어 버린다.(초록색으로 표시된 그룹은 침대, 책상, 의자를 포함한다) 때문에 DBScan 대신 OWL을 이용하여 가구가 가진 정보를 분석하여 이 가구가 무엇인지 판별하는 식으로 가구 분류와 복원을 진행할 예정이다.

2.설계 상세화 및 변경 내역

3D 포인트 클라우드 데이터에서 천장면을 도출해내 평면에 투영어떤 공간의 가구를 제외한 모든 천장, 바닥, 벽을 제거하기 위해 <그림4>와 같은 일련의 방법을 사용하였다.



<그림4> - 공간의 벽, 천장, 바닥을 제거하기 위한 알고리즘의 흐름도

2-1 공간내 주요 평면 추출

가장 큰 면인 천장을 먼저 탐색하고자 Open3d의 down sampling, 통계적 outlier 제거기능을 통해 노이즈를 제거한다음 데이터의 전체 z범위에 대해 Z threshold를 구하여, 해당 값보다 높은 z값을 가지는 point들을 추출하여 천장면의 point데이터를 검출해 다운 샘플링 처리 후 평면에 투사한다. 다음, BPD 알고리즘[1]을 이용해 추출한 천장면의 boundary points를 구한다. 이 과정을 구현한 코드는 부록의 <코드1>, <코드2>와 같다.

2-2 공간내 주요 평면 제거

천장 및 바닥을 제거하기 위해, 두개의 Z threshold 값(z_high, z_low)을 사용하여 z 값이 두 threshold 사이인 모든 points를 구한다. 다음, 벽을 제거하기 위해 위에서 구한 boundary points를 기준으로 해당 공간의 모든 점에 대해 z값에 상관없이 해당 점으로부터 반지름이 r인 원기둥(또는 길이가 n인 사각기둥)의 범위만큼 x,y 좌표가 겹치는 점들을 제거한다. 이를 부록의 <코드4>에 나타내었다.

2-3 DBSCAN을 통한 내부 가구 그룹화

마지막으로, open3d의 DBSCAN 기능을 사용하여 내부 가구들을 여러 그룹으로 분할하였다. 이를 main문인 부록의 <코드5>에 나타내었다.

---

[1] Mineo, C.; Pierce, S.G.; Summan, R. Novel algorithms for 3D surface point cloud boundary detection and edge reconstruction. *J. Comput. Des. Eng.* **2019**, *6*, 81–91

2-4 이후 계획

1. 천장 및 바닥 검출시 정확한 z 값을 구하기 위해 공간을 z threshold에 맞게 자른 다음 open3D의 plane segmentation 기능을 사용하는 방법을 구현할 것이다.

2. 벽 제거 방법을 원 또는 사각기둥을 사용한 방식이 아닌 점을 시계방향으로 정렬한다음 edge를 연결하여 해당 edge의 x,y 값에 맞게 일정 범위 만큼 벽을 지우는 방식을 구현할 것이다.

3. 정확도를 높이기 위해, 분리된 각 그룹에 대해 Martin Günther외 3명이 작성한 논문 Model-based furniture recognition for building semantic object maps[2] 에 제시된 Ontology Web Language (OWL)를 사용하여 그룹에 존재하는 가구들을 분리한 다음, bounding box의 크기에 따른 분류작업 수행을 시도해볼 것이다.

3. 갱신된 과제 추진 계획

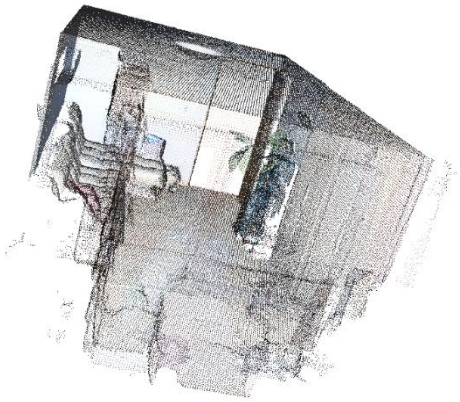
5월			6월				7월				8월				9월			
2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Point cloud 관련 스터디																		
			벽, 천장, 바닥, 사물 구별 기능 구현															
								사물 검출 기준 설정										
									중간보고서									
											사물 검출 및 제거 실행							
														제거된 영역 메쉬화 연구				
																	최종보고서	
																		발표

4. 구성원별 진척도

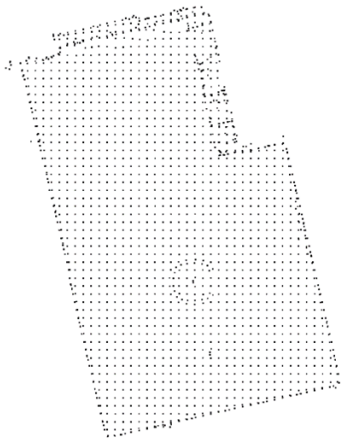
[2] Günther, M.; Wiemann, T.; Albrecht, S.; Hertzberg, J. Model-based furniture recognition for building semantic object maps. *Artif. Intell.* **2017**, *247*, 336–351.

기태욱	● bounding box의 크기에 따른 분류작업 수행 ● 사물 검출 최적 기준 탐색 및 수립
서준오	● 모델 테스트 및 성능 평가 ● 천장, 바닥의 정확한 Z값 검출을 위한 테스트
이해성	● 가구 분류 기준 및 알고리즘 선택 ● 가구 추출 알고리즘 개선

5. 보고 시점까지의 과제 수행 내용 및 중간 결과

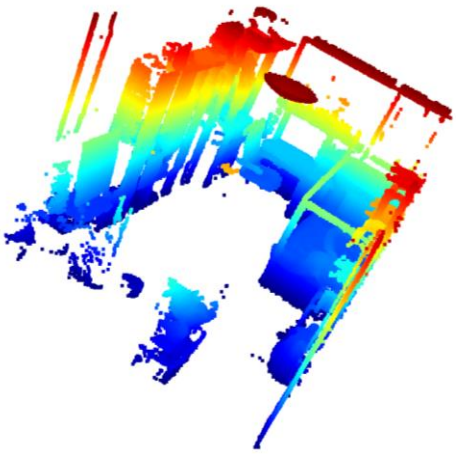


- 원본 point cloud data(cropped1 방)



- extract ceil(xy평면에 투영되어 검출된 cropped1 방의 천장)

- Boundary Point (cropped1의 검출된 천장 데이터에 외곽선 검출을 적용한 결과)



- 외곽선을 이용해 벽을 제거하고, 천장과 바닥도 제거하여 가구들만 남은 모습

- DBSCAN을 사용해 여러 그룹으로 분류된 가구들의 모습

-

6. 부록

```
import numpy as np
import open3d as o3d

def display_outlier(points, ind):
    inlier = points.select_by_index(ind)
    outlier = points.select_by_index(ind, invert=True)
    inlier.paint_uniform_color([0.7, 0.7, 0.7])
    outlier.paint_uniform_color([1, 0, 0])
    o3d.visualization.draw_geometries([inlier, outlier])

def extract_ceil(pointcloud, show_progress=False, voxel_size=0.1,
nb_neighbors=30, std_ratio=2.0, z_ratio=0.25,
                 save_filename=None):
    if show_progress:
        o3d.visualization.draw_geometries([pointcloud])

    # remove outlier
    pc_down = pointcloud.voxel_down_sample(voxel_size=voxel_size)
    cl, ind = pc_down.remove_statistical_outlier(nb_neighbors=nb_neighbors,
std_ratio=std_ratio)
    if show_progress:
        display_outlier(pc_down, ind)

    pc_down = pc_down.select_by_index(ind)

    # extract ceil
    high_z = -10000
    min_z = 1000000
    high_ind = 0
    low_ind = 0
    points = pc_down.points
    for i in range(len(points)):
        if points[i][2] > high_z:
            high_z = points[i][2]
            high_ind = i

        if points[i][2] < min_z:
            min_z = points[i][2]
            low_ind = i
    if show_progress:
        print(f'{high_z} , {min_z}')
        high_point = pc_down.select_by_index([high_ind])
        low_point = pc_down.select_by_index([low_ind])
        temp = pc_down.select_by_index([high_ind, low_ind], invert=True)
        temp.paint_uniform_color([0.7, 0.7, 0.7])
        high_point.paint_uniform_color([1, 0, 0])
        low_point.paint_uniform_color([0, 1, 0])

        o3d.visualization.draw_geometries([low_point, high_point, temp])

    # Parameter
    z_threshold = abs(high_z - min_z) * z_ratio
```

```
    inliers = []
    for i in range(len(points)) :
        if points[i][2] >= z_threshold :
            inliers.append(i)

    # remove outlier in ceil
    segments = pc_down.select_by_index(inliers)
    if show_progress :
        o3d.visualization.draw_geometries([segments])

    cl, ind = segments.remove_statistical_outlier(nb_neighbors=nb_neighbors,
std_ratio=std_ratio)
    if show_progress:
        display_outlier(segments, ind)
    segments = segments.select_by_index(ind)

    # project to highest Z value
    points = np.array(segments.points)
    points[:, 2] = high_z
    segments.points = o3d.utility.Vector3dVector(points)

    if show_progress:
        o3d.visualization.draw_geometries([segments])

    points = np.array(segments.points)

    if save_filename != None:
        np.savetxt(save_filename, points[:, :2])

    return points[:, :2], high_z, min_z
```

<코드1> - 천장 추출을 위해 사용된 코드 plane_extractor.py

```
import numpy as np
import Circle as C
import open3d as o3d
import polar
from sklearn.neighbors import NearestNeighbors
from itertools import combinations

def cal_boundary(points, k=30, save_filename=None) :
    neighbors = NearestNeighbors(n_neighbors=k).fit(points)
    distances, indices = neighbors.kneighbors(points)

    max_b = 0
    min_b = 1000000000

    boundary = []
    for i in range(points.shape[0]) :
        is_boundary = False
        p_neighbor, p_distance = indices[i], np.round(distances[i], 5)
        neighbor_points = points[p_neighbor[1:]]
```

```
        p_mean = np.mean(p_distance)
        p_std = np.std(p_distance)
        local_resol = round(p_mean + 2 * p_std, 5)

        if local_resol > max_b :
            max_b = local_resol
        if min_b > local_resol :
            min_b = local_resol

        pairs = list(combinations(p_neighbor[1:], 2))
        print(i)
        for j in range(len(pairs)) :
            count = 0
            p1 = points[i]
            p2 = points[pairs[j][0]]
            p3 = points[pairs[j][1]]
            c = C.Circle(p1, p2, p3)
            if c.radius == None :
                continue

            if c.radius >= local_resol :
                cn_distance = np.linalg.norm((neighbor_points - c.center),
axis=1)
                cn_distance = np.round(cn_distance, 5)

                for k in range(len(cn_distance)) :
                    if cn_distance[k] <= c.radius :
                        count += 1
                        if count > 3 :
                            break

            if count == 3 :
                boundary.append(points[i])
                is_boundary = True
                break

    print(f'len : {len(boundary)}')
    print(f'{min_b}')
    print(f'{max_b}')
    pc = o3d.geometry.PointCloud()
    pc.points = o3d.utility.Vector3dVector(np.pad(np.array(boundary), (0, 1),
'constant', constant_values=0))

    o3d.visualization.draw_geometries([pc])
    points = np.array(pc.points)
    if save_filename != None :
        np.savetxt(save_filename, points[:, :2])

    return points
```

<코드2> - BPD 알고리즘을 구현한 코드 BPD.py

```python
import math

class Circle :
    def __init__(self, p1, p2, p3) :
        v1 = np.dot(p2, p2.transpose())
        v2 = (np.dot(p1, p1.transpose()) - v1) / 2
        v3 = (v1 - (p3[0]**2) - (p3[1]**2)) / 2
        det = (p1[0] - p2[0]) * (p2[1] - p3[1]) - (p2[0] - p3[0]) * (p1[1] -
p2[1])

        if abs(det) < 1.0e-10:
            self.radius = None
            self.center = None
            return

        # Center of circle
        cx = round((v2 * (p2[1] - p3[1]) - v3 * (p1[1] - p2[1])) / det, 5)
        cy = round(((p1[0] - p2[0]) * v3 - (p2[0] - p3[0]) * v2) / det, 5)

        self.center = np.array([cx,cy])
        self.radius = round(np.linalg.norm(p1-self.center), 5)

    def print(self) :
        print(f'R : {self.radius} C : {self.center}')

    def to_array(self) :
        arr = [[self.radius * math.sin(i), self.radius * math.cos(i)] for i in
range(0, 360, 10)]
        return np.array(arr)
```

<코드3> - <코드2>에 사용된 원을 나타내는 클래스 Circle.py

```python
import numpy as np
import open3d as o3d

"""
@Params : points : numpy array
          boundary : numpy array
          raidus : float
          min_z  : float
          high_z : float
@Return  : pointcloud data
"""
def remove_cylinder(points, boundary, radius, min_z, high_z) :
    pc = o3d.geometry.PointCloud()
    pc.points = o3d.utility.Vector3dVector(points)

    for i in range(boundary.shape[0]) :
        z = np.arange(min_z, high_z, radius * 0.7)
        pillar_points = np.tile(boundary[i], (z.shape[0], 1))

        pillar_points = np.hstack((pillar_points, z.reshape(z.shape[0], 1)))
        pillar_pc = o3d.geometry.PointCloud()
        pillar_pc.points = o3d.utility.Vector3dVector(pillar_points)
```

```
        dists = np.asarray(pc.compute_point_cloud_distance(pillar_pc))
        ind = np.where(dists > radius)[0]
        pc = pc.select_by_index(ind)
        print(i)
    return pc

"""
@Params : points : numpy array
          boundary : numpy array
@Return  : numpy array
"""
def remove_wall(points, boundary, method="edge", radius=0.5, box_size=0.5,
min_z=None, high_z=None) :
    if method == "edge" :
        #use edge algorithm
        pass
    if method == "box" :
        #use box algorithm
        pass
    if method == "cylinder" :
        result = remove_cylinder(points, boundary, radius=radius, min_z=min_z,
high_z=high_z)

    return result

"""
@Params : pointcloud : point cloud object
          z_high : float
          z_low  : float
@Return  : point cloud object
"""
def remove_ceil_and_floor(pointcloud, z_high, z_low) :
    points = pointcloud.points
    inliers = []
    for i in range(len(points)):
        if points[i][2] <= z_high - 0.08 and points[i][2] >= z_low + 0.04:
            inliers.append(i)

    return pointcloud.select_by_index(inliers)
```

<코드4> - 벽을 제거하는 알고리즘을 구현한 remove_wall.py

```python
import plane_extractor
import BPD
import numpy as np
import remove_wall
import open3d as o3d
import matplotlib.pyplot as plt

pc = o3d.io.read_point_cloud("cropped_2.ply")
o3d.visualization.draw_geometries([pc])

# 면 뜯어내기(다운샘플 + z 축 편집)
plane, high_z, min_z = plane_extractor.extract_ceil(pc)
# BPD, boundary point 도출 후, txt 파일로 저장
boundary = BPD.cal_boundary(plane, save_filename="ceil_boundary3.txt")
boundary = np.loadtxt("ceil_boundary3.txt")

# boundary point 를 사용하여 벽 제거
removed = remove_wall.remove_wall(pc.points, boundary, method="cylinder",
radius=0.053, min_z=min_z , high_z=high_z)
# 천장, 바닥 제거
removed = remove_wall.remove_ceil_and_floor(removed, high_z, min_z + 0.04)

# DBSCAN 을 통한 내부 가구 그룹화
with o3d.utility.VerbosityContextManager(
        o3d.utility.VerbosityLevel.Debug) as cm:
    labels = np.array(
        removed.cluster_dbscan(eps=0.05, min_points=15, print_progress=True))

max_label = labels.max()
print(f"point cloud has {max_label + 1} clusters")
colors = plt.get_cmap("tab20")(labels / (max_label if max_label > 0 else 1))
colors[labels < 0] = 0
removed.colors = o3d.utility.Vector3dVector(colors[:, :3])
o3d.visualization.draw_geometries([removed])
```

<코드5> - main.py