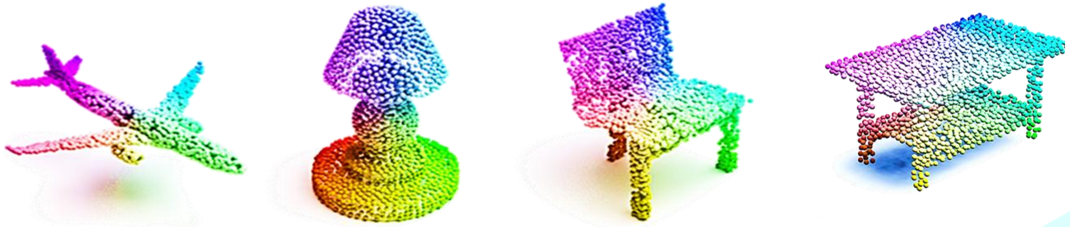
Open3D 활용한
Point Cloud Data 전처리 및 가시화

Point Cloud Data Preprocessing & Visualization via Open 3D

한국원자력연구원 서호건



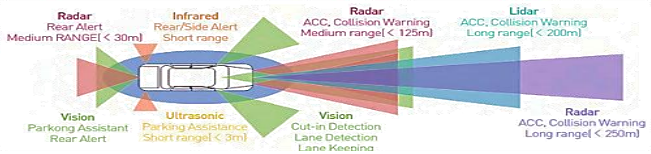
CONTENTS

01 3D 공간 정보 처리 기술 개요

» Digital Twin의 가상환경 구축 및 실시간 상호작용의 핵심 기술

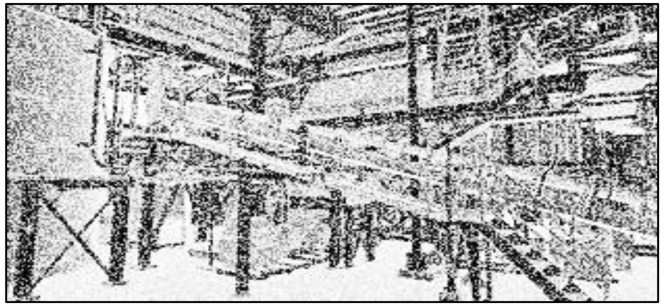
- 실시간으로 실제 현장을 가상환경으로 재구성하는 것이 필요
- 감지된 대상이 구체적으로 무엇인지를 판단하는 것이 유용
  ex) 1 m 앞에 무언가(?) 있음 → 1 m 앞에 보행자 있음
- Point Cloud 데이터 분석을 통해, 높은 정밀도와 정확도로 3D 공간을 재구성하고 어떤 형상인지까지도 판단하고자 함

〈 자율주행 위한 3D 공간 재구성〉

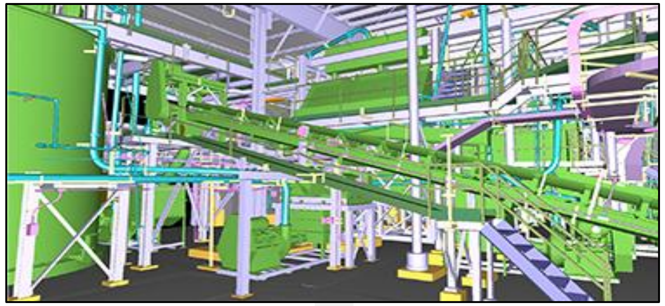


〈 실제 현장 (일상, 산업, 사고) 〉

〈 포인트 클라우드 수집 〉



〈 3D Digital Twin 환경 재구성 〉

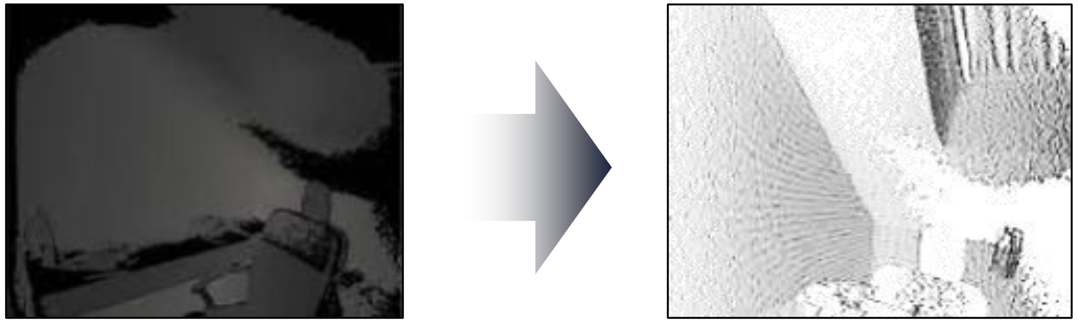


실제 현장의 변화

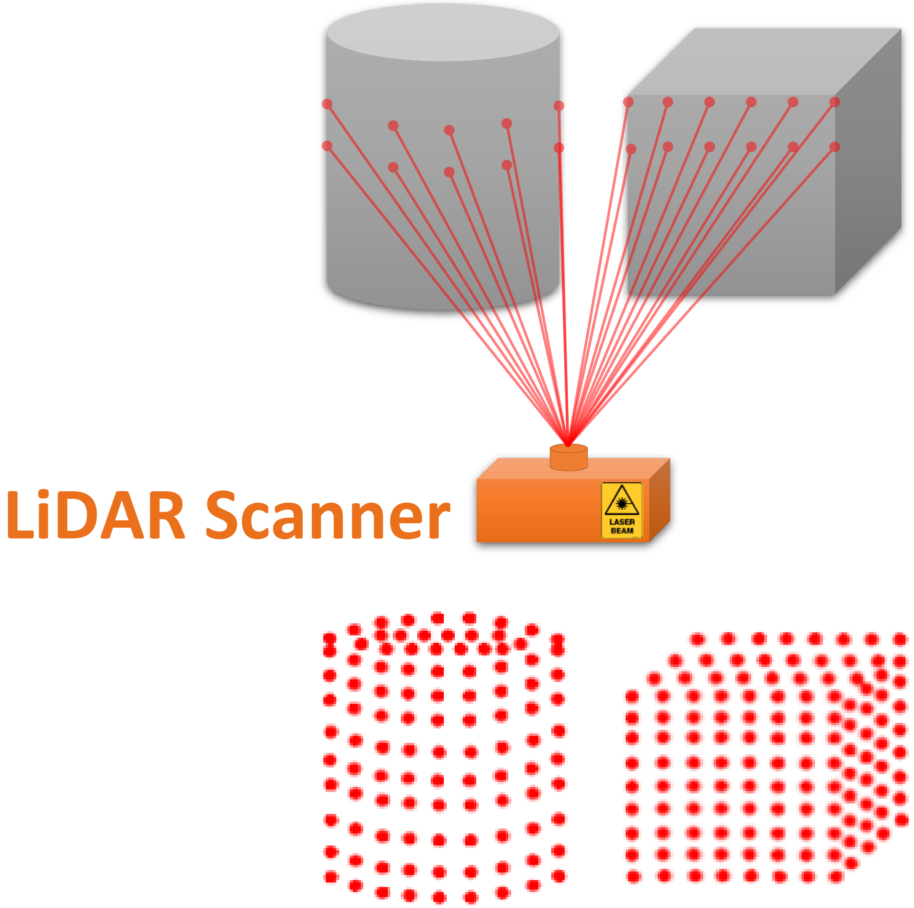
02 Point Cloud 소개

» Point Cloud 란?

- Point Cloud (점 구름): 점들의 집합
- 하나의 점은 3D 공간 좌표계에서
  x, y, z 축 상의 좌표 값으로 정의될 수 있음
- LiDAR (Light Detection and Ranging)
  : 광학 펄스를 비춘 후 반사광으로 거리 측정
- Point Cloud 생성 방식
  1. LiDAR를 활용한 측정
  2. Depth Image로부터 변환

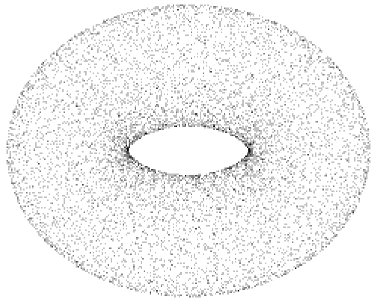


〈 LiDAR를 활용한 Point Cloud 측정 〉



02 Point Cloud 소개

» Artificial Intelligence for Point Cloud



< x, y, z >

100, 105, 103

235, 32, 535

33, 322, 252

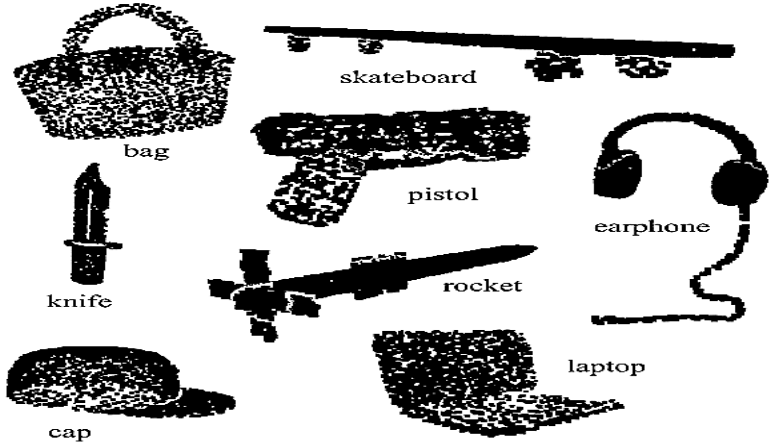
:

도넛?

반지?

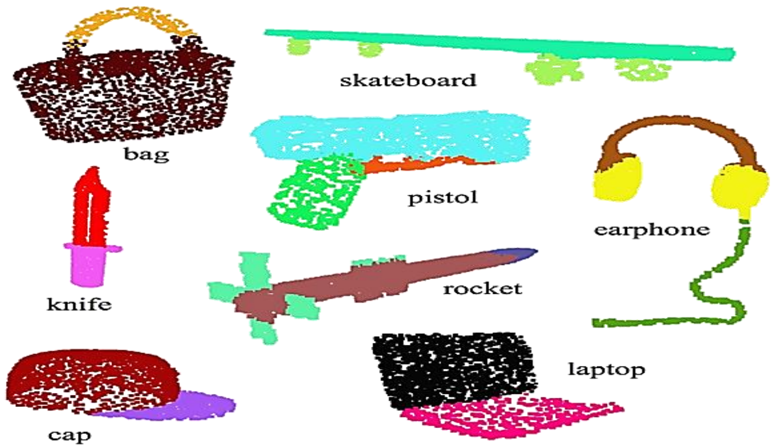
» 형상 분류 (Shape Classification)

- 좌표 집합에 부합하는 대상을 추정



» 형상 분할 (Shape Segmentation)

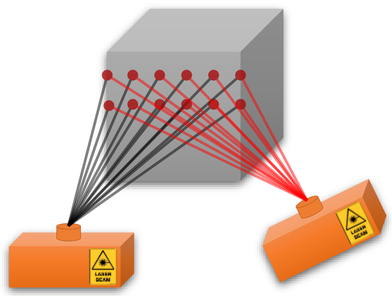
- 각 좌표에 부합하는 대상을 추정

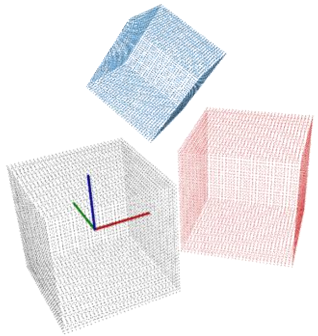


02 Point Cloud 소개

» Rigid Transformation Invariance (rotation, translation, reflection)

- 동일 대상이지만 크기, 좌표가 상이할 수 있음





» Permutation Invariance

- 동일 대상이지만 점들의 순서가 상이할 수 있음

< x, y, z >		< x, y, z >
33, 322, 252	=	235, 32, 535
235, 32, 535		100, 105, 103
100, 105, 103		33, 322, 252
:		:

» Sampling

- 실제 스캔 시, 취득되는 점(좌표)의 수가 가변적임

» Partial View

- 특정 각도에서 국부 형상에 대해서만 취득될 수 있음

03 Open3D를 활용한 Point Cloud Data 전처리 및 가시화

OPEN3D :: A Modern Library for 3D Data Processing

- Open3D Frontend

  : a set of carefully selected data structures and algorithms in C++ and Python

- Open3D Backend

  :highly optimized parallelization and GPU acceleration for 3D operations

- Open3D Core Features

  : 3D Data Structures, 3D Data Processing Algorithms, 3D Visualization, Scene Reconstruction, Surface Alignment, Physically based Rendering (PBR), 3D Machine Learning support with PyTorch and TensorFlow

03 Open3D를 활용한 Point Cloud Data 전처리 및 가시화

» TXT file (3 columns: x, y, z) ➔ 3D Point Cloud Visualization

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

xyz = np.loadtxt('guitar.txt')

pcd = o3d.geometry.PointCloud()

pcd.points = o3d.utility.Vector3dVector(xyz)

pcd.paint_uniform_color([1, 0.706, 0])

o3d.visualization.draw_geometries([pcd])
```

```
pcd = {PointCloud} PointCloud with 2048 points.
>  HalfEdgeTriangleMesh = {Type} Type.HalfEdgeTriangleMesh
>  Image = {Type} Type.Image
>  LineSet = {Type} Type.LineSet
>  PointCloud = {Type} Type.PointCloud
>  RGBDImage = {Type} Type.RGBDImage
>  TetraMesh = {Type} Type.TetraMesh
>  TriangleMesh = {Type} Type.TriangleMesh
>  Type = {pybind11_type} <class 'open3d.cpu.pybind.geometry.Geometry.Type'>
>  Unspecified = {Type} Type.Unspecified
>  VoxelGrid = {Type} Type.VoxelGrid
>  colors = {Vector3dVector: 0} std::vector<Eigen::Vector3d> with 0 elements.₩nUse numpy.asarray() to access data.
>  normals = {Vector3dVector: 0} std::vector<Eigen::Vector3d> with 0 elements.₩nUse numpy.asarray() to access data.
>  points = {Vector3dVector: 2048} std::vector<Eigen::Vector3d> with 2048 elements.₩nUse numpy.asarray() to access data.
```

guitar.txt - Windows 메모장

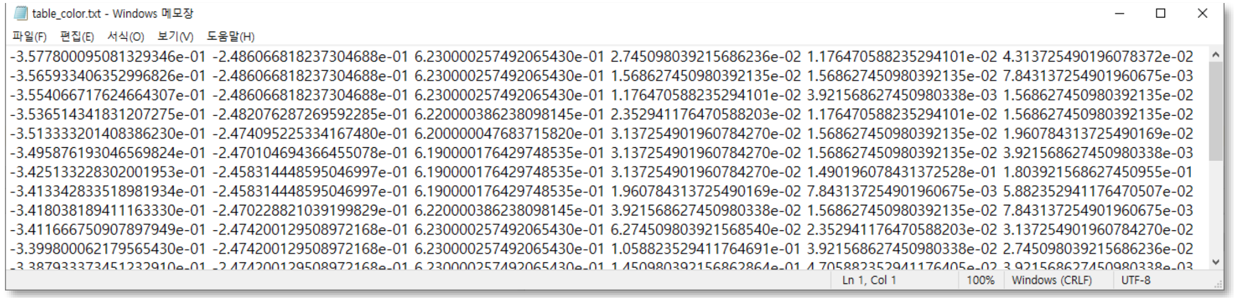
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
7.979658842086791992e-01 -1.555308233946561813e-02 8.455718308687210083e-02
-4.889959394931793213e-01 -8.291828446090221405e-03 -7.515705074183642864e-04
1.446241736412048340e-01 -1.635682769119739532e-02 1.853975951671600342e-01
-9.702072292566299438e-02 3.939257469028234482e-03 -1.647530645132064819e-01
4.402242898941040039e-01 1.147810090333223343e-02 -3.955466672778129578e-02
-2.013589143753051758e-01 -1.335617527365684509e-02 1.901605427265167236e-01
-3.570475876331329346e-01 9.230587631464004517e-03 -2.353041470050811768e-01
1.985868066549301147e-01 2.124213241040706635e-02 -4.389340430498123169e-02
9.989024996757507324e-01 -3.418660536408424377e-02 -3.201581910252571106e-02
-2.672214806079864502e-01 2.866190299391746521e-02 -2.183428779244422913e-02
-2.237648330628871918e-02 -1.575401984155178070e-02 4.363309592008590698e-02
-4.162122011184692383e-01 5.301161669194698334e-03 2.001329362392425537e-01
6.375886797904968262e-01 1.973138563334941864e-02 -3.082508593797683716e-02
8.173844963312149048e-02 -2.870261436328291893e-03 -1.829117685556411743e-01
1.446241736412048340e-01 -1.635682769119739532e-02 1.853975951671600342e-01
-9.702072292566299438e-02 3.939257469028234482e-03 -1.647530645132064819e-01
4.402242898941040039e-01 1.147810090333223343e-02 -3.955466672778129578e-02
-2.013589143753051758e-01 -1.335617527365684509e-02 1.901605427265167236e-01
-3.570475876331329346e-01 9.230587631464004517e-03 -2.353041470050811768e-01
1.985868066549301147e-01 2.124213241040706635e-02 -4.389340430498123169e-02
9.989024996757507324e-01 -3.418660536408424377e-02 -3.201581910252571106e-02
-2.672214806079864502e-01 2.866190299391746521e-02 -2.183428779244422913e-02
-2.237648330628871918e-02 -1.575401984155178070e-02 4.363309592008590698e-02
-4.162122011184692383e-01 5.301161669194698334e-03 2.001329362392425537e-01
6.375886797904968262e-01 1.973138563334941864e-02 -3.082508593797683716e-02
8.173844963312149048e-02 -2.870261436328291893e-03 -1.829117685556411743e-01
8.165711164474487305e-01 -2.431094273924827576e-02 -9.204377233982086182e-02
3.159337937831878662e-01 1.698525436222553253e-02 3.622820600867271423e-02
-4.628741741180419922e-01 9.081746451556682587e-03 -1.404173970222473145e-01
-2.469045668840408325e-01 -1.575401984155178070e-02 -1.523247510194778442e-01
-3.122461438179016113e-01 1.316745858639478683e-02 1.089672744274139404e-01
-1.652647405862808228e-01 -1.731462590396404266e-02 5.587533861398696899e-02
9.474031329154968262e-01 -3.161907941102981567e-02 9.363655000925064087e-02
1.297346595674753189e-02 1.550345215946435928e-03 1.701412945985794067e-01
6.618446111679077148e-02 1.796760968863964081e-02 -4.208497703075408936e-02
-3.734945952892303467e-01 -1.731462590396404266e-02 -6.251353770494461060e-02
-1.264914125204086304e-01 2.677905187010765076e-02 -5.100805312395095825e-02
5.328039526939392090e-01 9.587808512151241302e-03 3.118991106748580933e-02
-1.000719815492630005e-01 1.295163575559854507e-02 1.449870169162750244e-01
```

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

03 Open3D를 활용한 Point Cloud Data 전처리 및 가시화

» TXT file (6 columns: x, y, z, r, g, b) ➔ 3D Point Cloud Visualization


table_color.txt - Windows 메모장

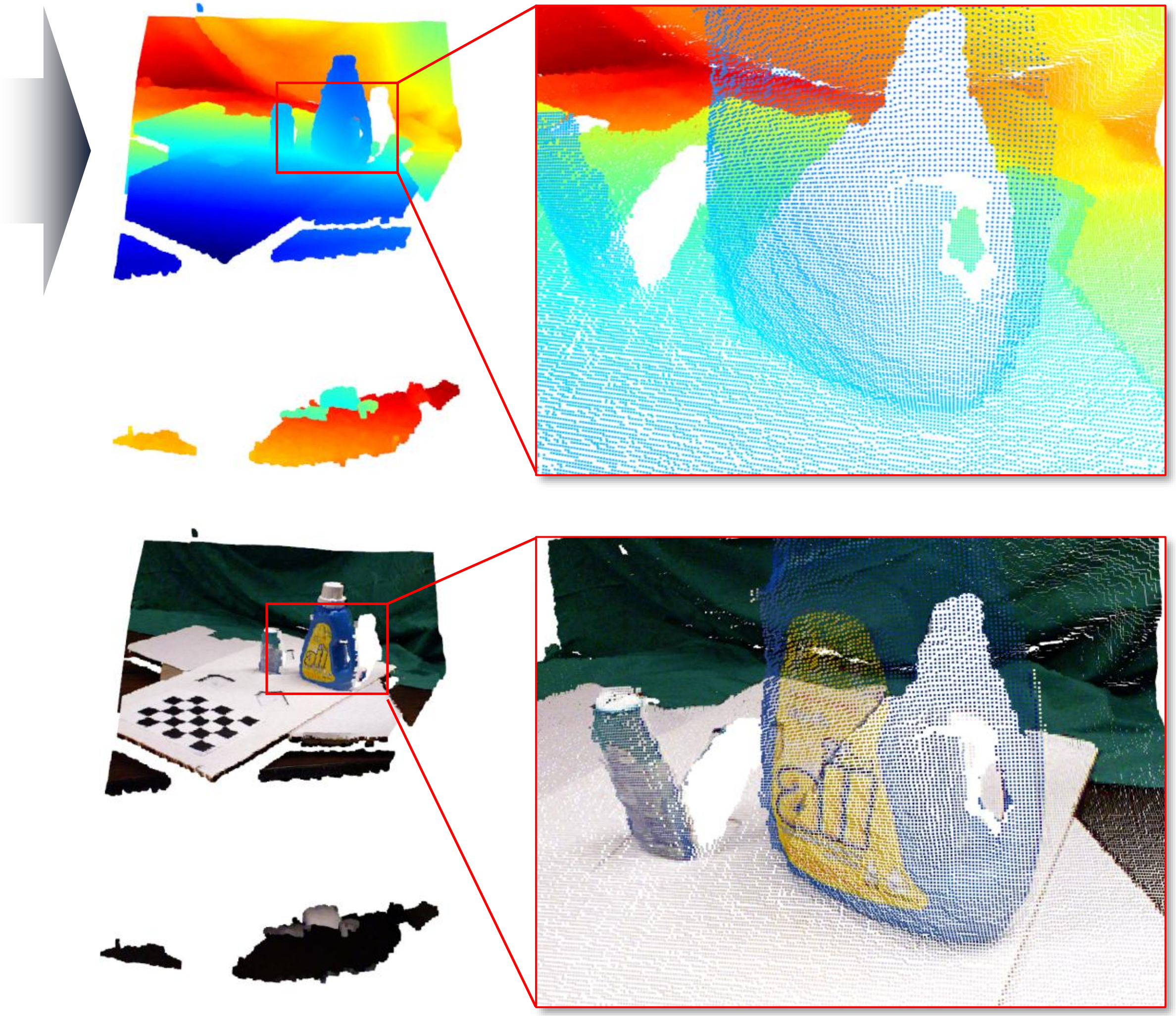
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
-3.577800095081329346e-01 -2.486066818237304688e-01 6.230000257492065430e-01 2.745098039215686236e-02 1.176470588235294101e-02 4.313725490196078372e-02
-3.565933406352996826e-01 -2.486066818237304688e-01 6.230000257492065430e-01 1.568627450980392135e-02 1.568627450980392135e-02 7.843137254901960675e-03
-3.554066717624664307e-01 -2.486066818237304688e-01 6.230000257492065430e-01 1.176470588235294101e-02 3.921568627450980338e-03 1.568627450980392135e-02
-3.536514341831207275e-01 -2.482076287269592285e-01 6.220000386238098145e-01 2.352941176470588203e-02 1.176470588235294101e-02 1.568627450980392135e-02
-3.513333201408386230e-01 -2.474095225334167480e-01 6.200000047683715820e-01 3.137254901960784270e-02 1.568627450980392135e-02 1.960784313725490169e-02
-3.495876193046569824e-01 -2.470104694366455078e-01 6.190000176429748535e-01 3.137254901960784270e-02 1.568627450980392135e-02 3.921568627450980338e-03
-3.425133228302001953e-01 -2.458314448595046997e-01 6.190000176429748535e-01 3.137254901960784270e-02 1.490196078431372528e-01 1.803921568627450955e-01
-3.413342833518981934e-01 -2.458314448595046997e-01 6.190000176429748535e-01 1.960784313725490169e-02 7.843137254901960675e-03 5.882352941176470507e-02
-3.418038189411163330e-01 -2.470228821039199829e-01 6.220000386238098145e-01 3.921568627450980338e-02 1.568627450980392135e-02 7.843137254901960675e-03
-3.411666750907897949e-01 -2.474200129508972168e-01 6.230000257492065430e-01 6.274509803921568540e-02 2.352941176470588203e-02 3.137254901960784270e-02
-3.399800062179565430e-01 -2.474200129508972168e-01 6.230000257492065430e-01 1.058823529411764691e-01 3.921568627450980338e-02 2.745098039215686236e-02
-3.387933373451232910e-01 -2.474200129508972168e-01 6.230000257492065430e-01 1.450980392156862864e-01 4.705882352941176405e-02 3.921568627450980338e-03
```

Ln 1, Col 1 100% Windows (CRLF) UTF-8

```python
xyz_color = np.loadtxt('table_color.txt')
pcd = o3d.geometry.PointCloud()
pcd.points = o3d.utility.Vector3dVector(xyz_color[:,:3])
pcd.colors = o3d.utility.Vector3dVector(xyz_color[:,3:])
o3d.visualization.draw_geometries([pcd])
```

```python
pcd = o3d.io.read_point_cloud('table_color.pcd’)
pcd = o3d.io.read_point_cloud('table_color.ply’)
```



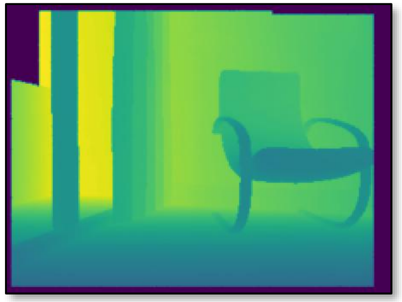
03 Open3D를 활용한 Point Cloud Data 전처리 및 가시화

» RGBD image (2 images: rgb, depth) ➔ 3D Point Cloud Visualization

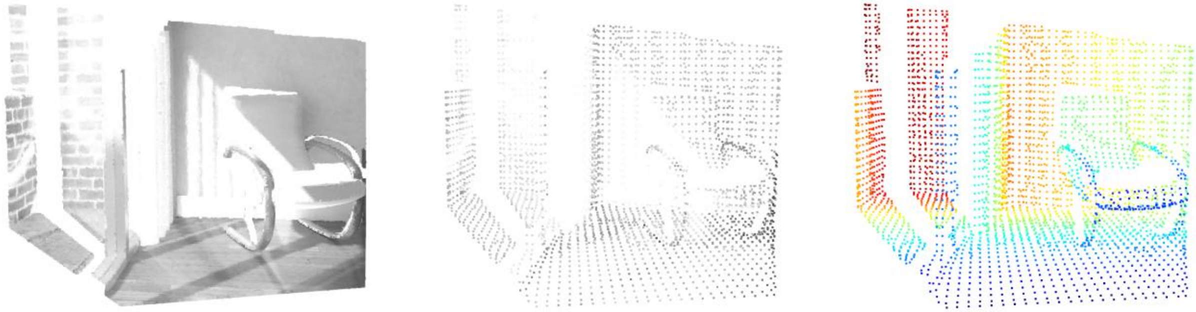
```
color_raw, depth_raw = o3d.io.read_image('img_rgb.jpg’), o3d.io.read_image('img_depth.png')
rgbd_image = o3d.geometry.RGBDImage.create_from_color_and_depth(color_raw, depth_raw)
pcd = o3d.geometry.PointCloud.create_from_rgbd_image(
    rgbd_image,
    o3d.camera.PinholeCameraIntrinsic(o3d.camera.PinholeCameraIntrinsicParameters.PrimeSenseDefault))
o3d.visualization.draw_geometries([pcd])
```

〈 img_rgb.jpg 〉

〈 img_depth.png 〉



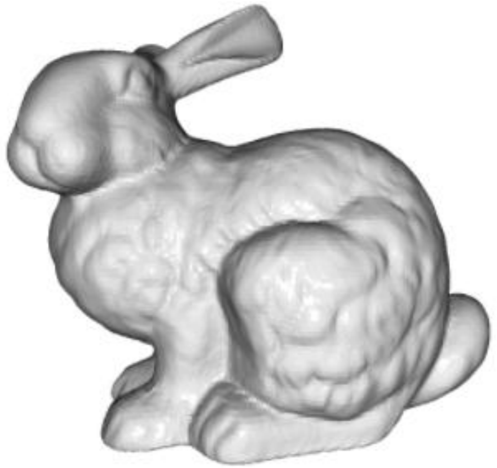
〈 Point Cloud including Color 〉



03 Open3D를 활용한 Point Cloud Data 전처리 및 가시화

» Mesh → 3D Point Cloud via Uniform Sampling

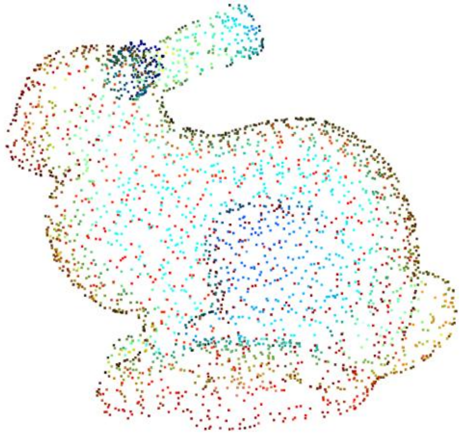
〈 Mesh 〉



```
mesh = o3d.io.read_triangle_mesh('bunny.ply')
mesh.compute_vertex_normals()
o3d.visualization.draw_geometries([mesh])
```

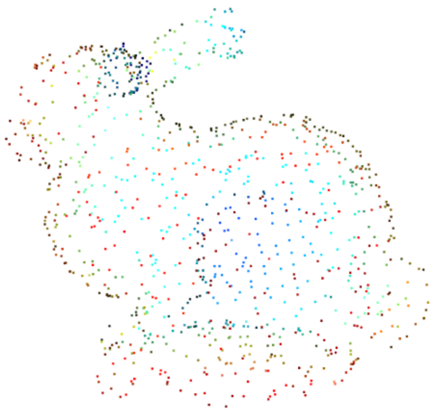
```
pcd = mesh.sample_points_uniformly(number_of_points=500)
o3d.visualization.draw_geometries([pcd])
```

number_of_points=3000

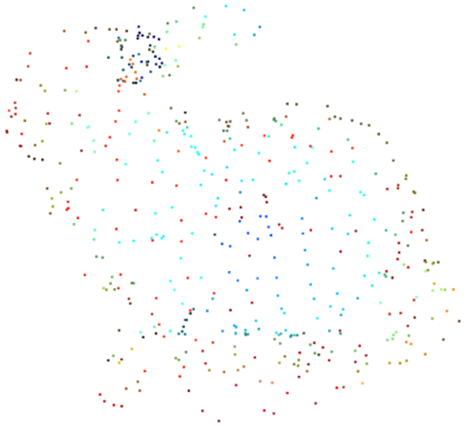


number_of_points=2000

number_of_points=1000



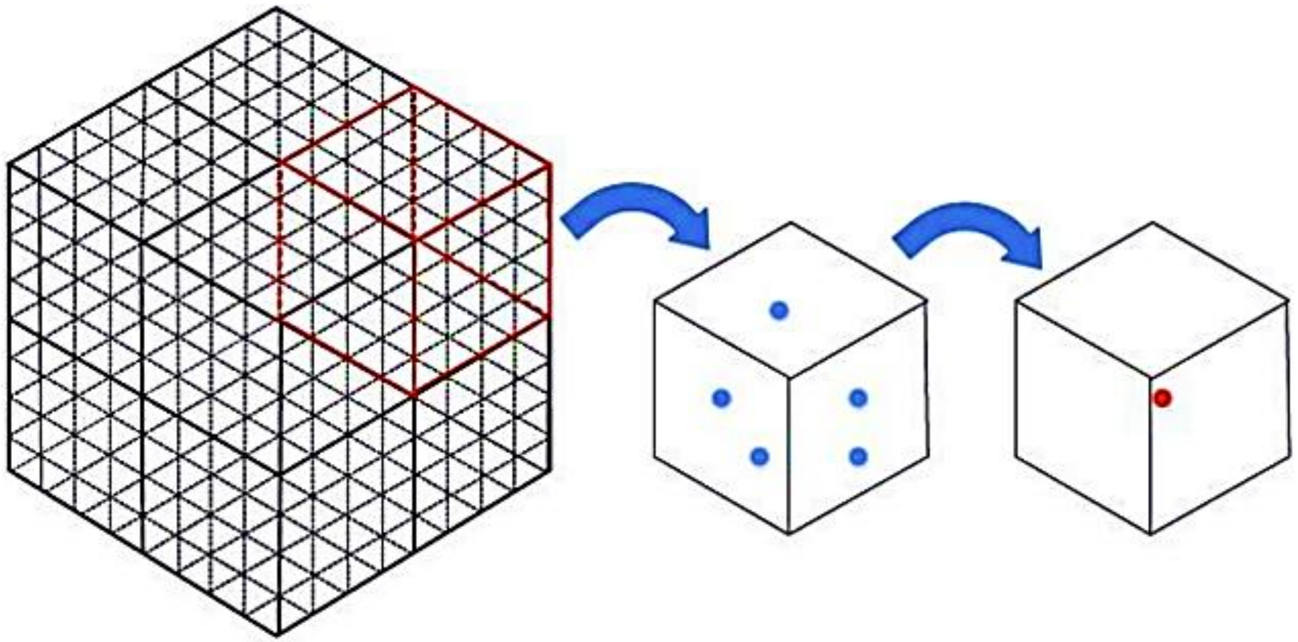
number_of_points=500



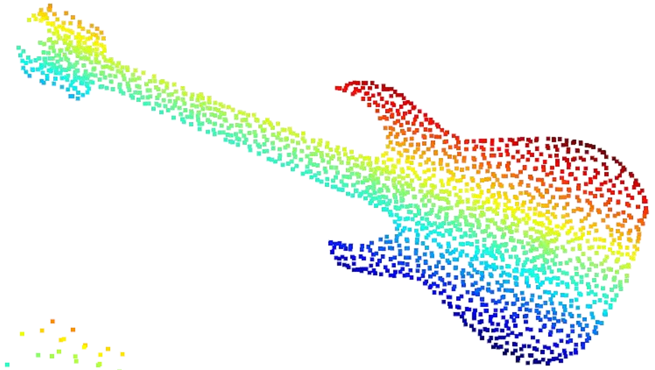
03 Open3D를 활용한 Point Cloud Data 전처리 및 가시화

» 3D Point Cloud ➔ Down Sampling based Voxel

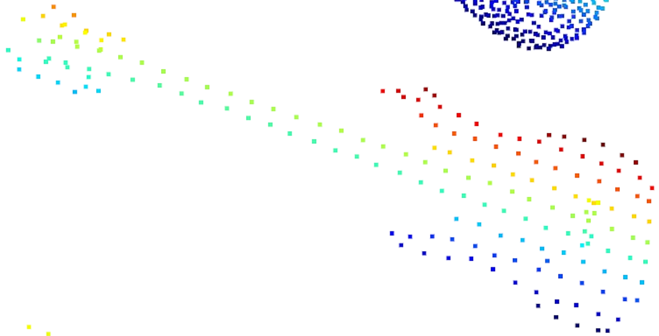
```
pcd_downsampled = pcd.voxel_down_sample(voxel_size=0.05)
o3d.visualization.draw_geometries([pcd_downsampled])
```



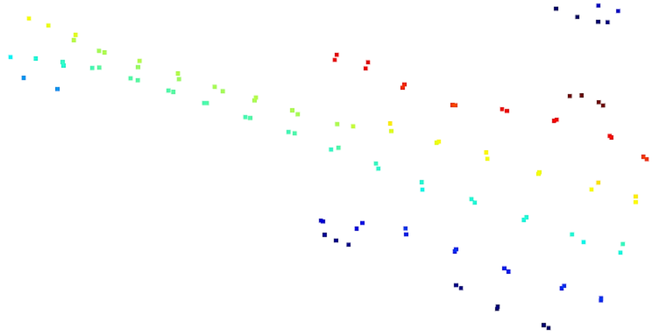
voxel_size=0.01



voxel_size=0.05



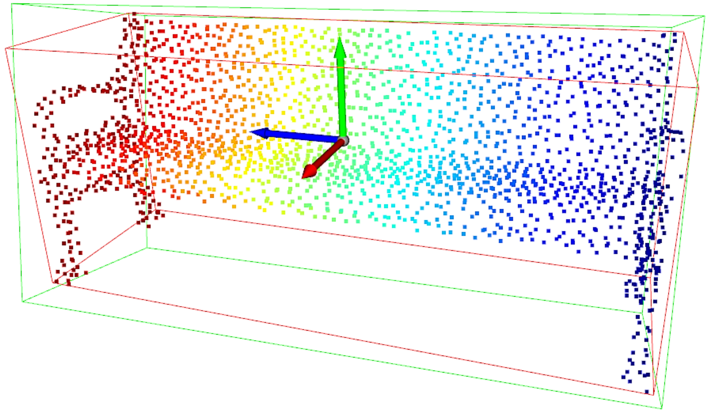
voxel_size=0.1



03 Open3D를 활용한 Point Cloud Data 전처리 및 가시화

» 3D Point Cloud ➔ 3D Bounding Box (axis_aligned, oriented)

```
xyz = np.loadtxt('bench.txt')
pcd = o3d.geometry.PointCloud()
pcd.points = o3d.utility.Vector3dVector(xyz)
aabb = pcd.get_axis_aligned_bounding_box()
aabb.color = (1,0,0)
print(np.asarray(aabb.get_box_points()))
obb = pcd.get_oriented_bounding_box()
obb.color = (0,1,0)
print(np.asarray(obb.get_box_points()))
mesh_frame = o3d.geometry.TriangleMesh.create_coordinate_frame(size=0.3, origin=[0, 0, 0])
o3d.visualization.draw_geometries([pcd, aabb, obb, mesh_frame])
```



```
print(np.asarray(aabb.get_box_points()))
[[-0.25250256 -0.43938023 -0.86245656]
 [ 0.30719468 -0.43938023 -0.86245656]
 [-0.25250256  0.29203793 -0.86245656]
 [-0.25250256 -0.43938023  0.85810483]
 [ 0.30719468  0.29203793  0.85810483]
 [-0.25250256  0.29203793  0.85810483]
 [ 0.30719468 -0.43938023  0.85810483]
 [ 0.30719468  0.29203793 -0.86245656]]
```

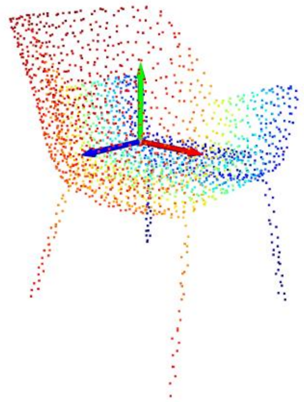
```
print(np.asarray(obb.get_box_points()))
[[-0.43925416  0.26731042 -0.84257911]
 [-0.37473499  0.26093893  0.88099575]
 [-0.25017401 -0.56199767 -0.85272269]
 [ 0.19008377  0.41107953 -0.86560586]
 [ 0.44368309 -0.42460005  0.84782541]
 [ 0.37916392 -0.41822856 -0.87574944]
 [ 0.25460294  0.40470804  0.85796899]
 [-0.18565484 -0.56836916  0.87085216]]
```

03 Open3D를 활용한 Point Cloud Data 전처리 및 가시화

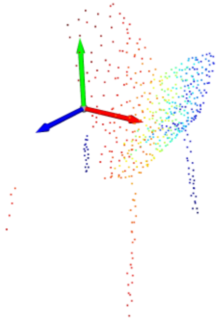
» Full-view 3D Point Cloud → Partial-view 3D Point Cloud Sampling

```
xyz = np.loadtxt('chair.txt')
pcd = o3d.geometry.PointCloud()
pcd.points = o3d.utility.Vector3dVector(xyz)
diameter = np.linalg.norm(np.asarray(pcd.get_max_bound()) - np.asarray(pcd.get_min_bound()))
camera = [1, 0, 0]
radius = diameter * 100
_, pt_map = pcd.hidden_point_removal(camera, radius)
pcd = pcd.select_by_index(pt_map)
mesh_frame = o3d.geometry.TriangleMesh.create_coordinate_frame(size=0.3)
o3d.visualization.draw_geometries([pcd, mesh_frame])
```

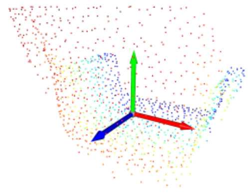
Full View



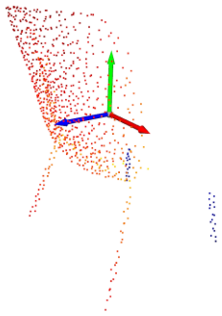
[1, 0, 0]



[0, 1, 0]



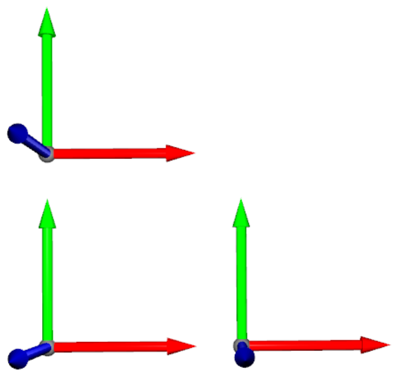
[0, 0, 1]



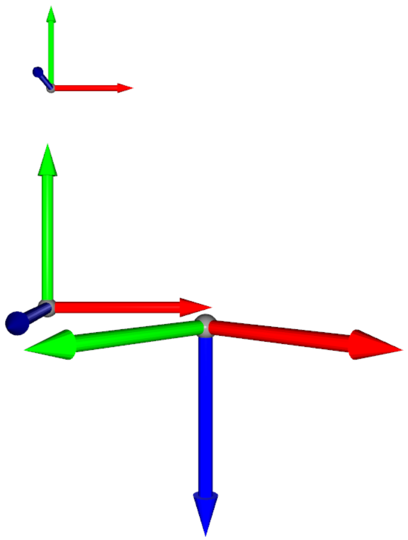
03 Open3D를 활용한 Point Cloud Data 전처리 및 가시화

» 3D Transformation :: Translate, Rotation, Scale

```python
mesh_list = [o3d.geometry.TriangleMesh.create_coordinate_frame() for i in range(3)]
mesh_list[1] = mesh_list[1].translate((1.3, 0, 0))
mesh_list[2] = mesh_list[2].translate((0, 1.3, 0))
o3d.visualization.draw_geometries(mesh_list)
```



```python
R = mesh_list[1].get_rotation_matrix_from_xyz((np.pi / 2, 0, np.pi / 4))
mesh_list[1].rotate(R, center=(0, 0, 0))
mesh_list[2].scale(0.5, center=mesh_list[2].get_center())
o3d.visualization.draw_geometries(mesh_list)
```

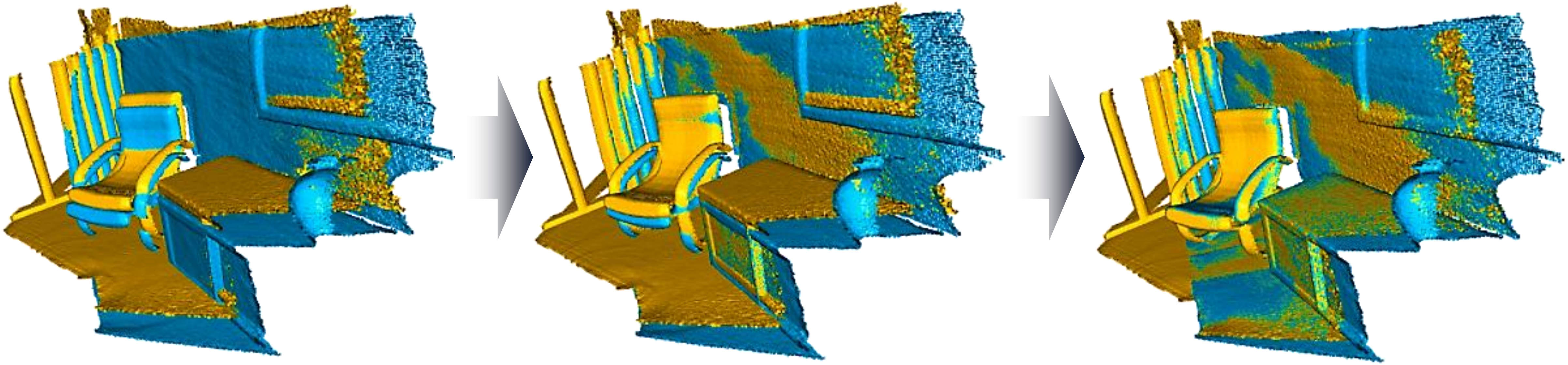


03 Open3D를 활용한 Point Cloud Data 전처리 및 가시화

» ICP (Iterative Closest Point) Registration Algorithm

- Point-to-point ICP $E(\mathbf{T}) = \sum_{(\mathbf{p},\mathbf{q})\in\mathcal{K}} \|\mathbf{p} - \mathbf{T}\mathbf{q}\|^2$
- Point-to-plane ICP $E(\mathbf{T}) = \sum_{(\mathbf{p},\mathbf{q})\in\mathcal{K}} ((\mathbf{p} - \mathbf{T}\mathbf{q}) \cdot \mathbf{n}_{\mathbf{p}})^2,$

```
reg_p2p = o3d.pipelines.registration.registration_icp(
    source, target, threshold, trans_init,
    o3d.pipelines.registration.TransformationEstimationPointToPoint(),
    o3d.pipelines.registration.ICPConvergenceCriteria(max_iteration=2000))
```



감사합니다

Open3D 활용한 Point Cloud Data 전처리 및 가시화