Roteiro para Processamento de Nuvens de Pontos utilizando a Open3D
Nível: Básico

Natalia Amorim

Grupo OpenCVismo Brasil
Junho de 2022

1 Instalação da Biblioteca

Instalar a open3d é simples:

```
pip install scipy numpy Pillow distutils
pip install open3d
```

Lembre-se que utilizaremos a versão 3.8 do python, então garanta que as libs sejam instaladas para esta versão.

Caso esteja usando o ubuntu 22.04 LTS será necessário fazer a instalação da versão 3.8 do python para que possamos utilizar a open3d. Prossiga com os passos disponíveis neste link para a instalação: `https://www.linuxcapable.com/how-to-install-python-3-8-on-ubuntu-22-04-lts/` e depois use os seguintes comandos:

```
python3.8 -m pip install scipy numpy Pillow distutils
python3.8 -m pip install open3d
```

Se depois de seguir estes passos, erros ainda ocorrerem na importação da biblioteca, utilize os seguintes comandos:

```
python3.8 -m pip install numpy --upgrade
python3.8 -m pip install scipy --upgrade
python3.8 -m pip install Pillow --upgrade
```

2 Leitura e Visualização de Nuvens de Pontos

Para fazer este passo-a-passo, baixe os arquivos disponibilizados no canal arquivos-palestras do servidor OpenCVismo Brasil do Discord. Faça download dos arquivos: room.pcd, cat.pcd e table_scene.pcd. Depois crie um arquivo python e coloque os arquivos pcd no mesmo diretório do seu arquivo python.

Comece importando as bibliotecas necessárias:

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

A leitura de arquivos de nuvens de pontos com a open3d pode ser feita da seguite forma:

```
cloud = o3d.io.read_point_cloud("cat.pcd")
#print(type(cloud))
print(np.asarray(cloud.points))
```

Podemos utilizar um método simples para visualizar nossa nuvem de pontos, para isso basta adicionarmos os seguintes comandos em nosso arquivo python:

```
o3d.visualization.draw_geometries([cloud],
                                  zoom=0.3412,
                                  front=[0.4257, -0.2125, -0.8795],
                                  lookat=[-0.9, -0.68, -1.18],
                                  up=[-0.0694, -0.9768, 0.2024])
```

Outra forma de visualizarmos nuvens de pontos com a open3d é utilizando a classe Visualizer. Esta classe nos permite criar visualizações mais personalizadas para nossas nuvens de pontos. Adicione esta função ao seu código fonte e depois faça a chamada passando a sua nuvem de pontos:

```
def view_cloud(point_cloud):
    vis = o3d.visualization.Visualizer()
    vis.create_window()
    vis.add_geometry(point_cloud)
    ctr = vis.get_view_control()
    vis.run()
    vis.destroy_window()
```

Podemos também adicionar uma animação de rotação à nossa visualização. Para isso, basta criarmos uma função da seguinte forma:

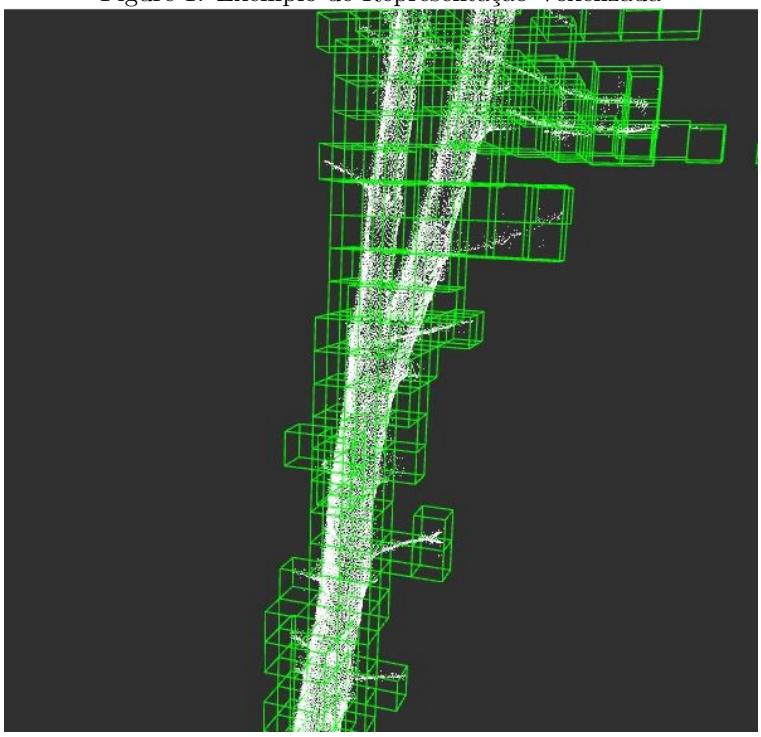
```
def custom_draw_geometry_with_rotation(point_cloud):

    def rotate_view(vis):
        ctr = vis.get_view_control()
        ctr.rotate(10.0, 0.0)
        return False

    o3d.visualization.draw_geometries_with_animation_callback([point_cloud],
                                                              rotate_view)
```

3 Voxelização

As nuvens de pontos em sua forma bruta consistem em um conjunto de pontos desordenados que representam (de forma descontínua) uma superfície. Desta forma, relações como vizinhança entre pontos, são desconhecidas a priori.

Uma abordagem que resolve este problema é a Voxelização. A voxelização consiste no processo de criar um grid regular 3D que é dividido em cubos que possuem seis faces, oito vertices doze arestas. Cada um destes cubos recebe o nome de *Voxel*.

Figure 1: Exemplo de Representação Voxelizada



Fonte: https://medium.com/@ghimire.aiesecer/voxel-fitting-on-lidar-point-cloud-713b2c7b1f3d

A voxelização também é facilmente implementada utilizando a open3d. Para criarmos uma representação voxelizada podemos usar o método *voxel_down_sample* da seguinte forma:

```
voxel_rep=cloud.voxel_down_sample(voxel_size=0.05)
```

Atenção: O que estamos fazendo quando chamamos este método é reamostrar a nossa nuvem de pontos através da voxelização! Criamos uma estrutura voxelizada onde cada voxel (cubo) mede 5 cm, verifica-se q quantidade de cada ponto dentro de cada voxel e os pontos que aparecem na visualização são a média do conjunto de pontos que está dentro de cada voxel!

Agora visualize a representação:

```
o3d.visualization.draw_geometries([voxel_rep],
                                  zoom=0.3412,
                                  front=[0.4257, -0.2125, -0.8795],
                                  lookat=[5.0, 2.8, -1.18],
                                  up=[-0.0694, -0.9768, 0.2024])
```