No description has been provided for this image

1 - Bibliotecas

In [2]:
```python
import trimesh
import numpy as np
import matplotlib.pyplot as plt
import open3d as o3d
import pandas as pd
```

```
Jupyter environment detected. Enabling Open3D WebVisualizer.
[Open3D INFO] WebRTC GUI backend enabled.
[Open3D INFO] WebRTCWindowSystem: HTTP handshake server disabled.
```

2 - Preparação dos Dados

In [3]:
```python
nuvens = [None] * 30
print(len(nuvens))
for i in range(len(nuvens)):
    path_nuvem = f'./KITTI-Sequence/{i:06d}/{i:06d}_points.obj'
    try:
        nuvens[i] = trimesh.load(path_nuvem).vertices
        print(f'nuvem: {i} / shape: {nuvens[i].shape}')
    except Exception as e:
        print(f"erro na nuvem {i}: {e}")

# vai servir como referencia
primeira_nuvem = nuvens[0]
```

```
30
nuvem: 0 / shape: (62553, 3)
nuvem: 1 / shape: (62340, 3)
nuvem: 2 / shape: (62138, 3)
nuvem: 3 / shape: (61833, 3)
nuvem: 4 / shape: (61542, 3)
nuvem: 5 / shape: (61375, 3)
nuvem: 6 / shape: (61159, 3)
nuvem: 7 / shape: (60197, 3)
nuvem: 8 / shape: (59560, 3)
nuvem: 9 / shape: (58954, 3)
nuvem: 10 / shape: (59038, 3)
nuvem: 11 / shape: (60849, 3)
nuvem: 12 / shape: (61935, 3)
nuvem: 13 / shape: (62687, 3)
nuvem: 14 / shape: (63122, 3)
nuvem: 15 / shape: (63202, 3)
nuvem: 16 / shape: (62748, 3)
nuvem: 17 / shape: (63098, 3)
nuvem: 18 / shape: (62854, 3)
nuvem: 19 / shape: (62570, 3)
nuvem: 20 / shape: (62128, 3)
nuvem: 21 / shape: (61925, 3)
nuvem: 22 / shape: (61574, 3)
nuvem: 23 / shape: (61391, 3)
nuvem: 24 / shape: (61335, 3)
nuvem: 25 / shape: (61300, 3)
nuvem: 26 / shape: (61155, 3)
nuvem: 27 / shape: (61145, 3)
nuvem: 28 / shape: (62211, 3)
nuvem: 29 / shape: (61793, 3)
```

3 - Funções

3.0 - encontrar_pontos_correspondentes

In [4]:
```python
def encontrar_pontos_correspondentes(origem, alvo, indices, distancias, raio=0.5):
    distancia = np.sqrt(np.sum((origem - alvo)**2, axis=1))

    indice_minimo = np.argmin(distancia)
    distancia_minima = distancia[indice_minimo]

    distancias_correspondentes = []
    indices_correspondentes = []

    indices.append(indice_minimo)
    distancias.append(distancia_minima)

    for i in range(len(distancias)):
        if distancias[i] < raio:
            distancias_correspondentes.append(distancias[i])
            indices_correspondentes.append(indices[i])

    distancias = np.array(distancias_correspondentes)
    indices = np.array(indices_correspondentes)

    return distancias, indices
```

3.1 - transformacao_rigida

In [5]:
```python
def transformacao_rigida(origem, alvo):

    centroide_origem = np.mean(origem, axis=0)
    centroide_alvo = np.mean(alvo, axis=0)

    matriz_covariancia = (origem - centroide_origem).T @ (alvo - centroide_alvo)

    matriz_rotacao_origem, _, matriz_rotacao_alvo_transposta = np.linalg.svd(matriz_covariancia)
    matriz_rotacao = matriz_rotacao_alvo_transposta.T @ matriz_rotacao_origem.T

    vetor_translacao = centroide_alvo - (matriz_rotacao @ centroide_origem)

    matriz_transformacao = np.identity(4)
    matriz_transformacao[:3, :3] = matriz_rotacao
    matriz_transformacao[:3, 3] = vetor_translacao

    return matriz_transformacao
```

3.2 - ICP

In [6]:
```python
def icp(origem, alvo, limite_pontos, max_iter=200, tolerancia=0.01):

    origem = origem[:limite_pontos, :]
    alvo = alvo[:limite_pontos, :]

    matriz_transformacao_total = np.identity(4)

    for i in range(max_iter):
        distancias, indices = encontrar_pontos_correspondentes(origem, alvo, [], [])

        matriz_transformacao_iterativa = transformacao_rigida(origem[indices], alvo[indices])

        matriz_transformacao_total = matriz_transformacao_iterativa @ matriz_transformacao_total

        origem = (matriz_transformacao_iterativa[:3, :3] @ origem.T).T + matriz_transformacao_iterativa[:3, 3]

        if np.sum(distancias) < tolerancia:
            break

    return matriz_transformacao_total
```

3.3 - aplicar_icp

In [7]:
```python
def aplicar_icp(nuvens):
    trajetoria = np.zeros((len(nuvens), 4, 4))
    trajetoria[0] = np.identity(4)

    nuvens_transformadas = []
    matrizes_transformacao_icp = []

    for i in range(1, len(nuvens)):

        limite_pontos = min(len(nuvens[i-1]), len(nuvens[i]))
        matriz_transformacao = icp(nuvens[i-1], nuvens[i], limite_pontos)

        matrizes_transformacao_icp.append(matriz_transformacao)

        trajetoria[i] = matriz_transformacao @ trajetoria[i-1]

        nuvem_transformada = (matriz_transformacao @ np.vstack([nuvens[i].T, np.ones((1, nuvens[i].shape[0]))])).T

        nuvem_transformada = nuvem_transformada[:3].T

        nuvens_transformadas.append(nuvem_transformada)

    return trajetoria, nuvens_transformadas, matrizes_transformacao_icp
```

3.4 - visualizar_nuvens 2D

In [9]:
```python
def visualizar_nuvens(nuvens_originais, nuvens_transformadas, titulo):
    fig = plt.figure(figsize=(5, 5))
    ax = fig.add_subplot(111, projection='3d')

    for i in range(len(nuvens_transformadas)):
        ax.scatter(nuvens_transformadas[i][:, 0], nuvens_transformadas[i][:, 1], nuvens_transformadas[i][:, 2], s=10, color='red')
        ax.scatter(nuvens_originais[i][:, 0], nuvens_originais[i][:, 1], nuvens_originais[i][:, 2], s=1, color='blue')

    plt.title(titulo)

    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    ax.set_zlabel('Z')

    plt.show()
```

3.5 - visualizar_nuvens 3D

In [10]:
```python
def visualizar_nuvens_open3d(nuvens_originais, nuvens_transformadas):
    vis = o3d.visualization.Visualizer()
    vis.create_window()

    for nuvem in nuvens_originais:
        ponto_cloud = o3d.geometry.PointCloud()
        ponto_cloud.points = o3d.utility.Vector3dVector(nuvem)
        vis.add_geometry(ponto_cloud)

    for nuvem in nuvens_transformadas:
        ponto_cloud_transformada = o3d.geometry.PointCloud()
        ponto_cloud_transformada.points = o3d.utility.Vector3dVector(nuvem)
        ponto_cloud_transformada.paint_uniform_color([1, 0, 0])
        vis.add_geometry(ponto_cloud_transformada)

    vis.run()
    vis.destroy_window()
```

4 - Aplicando ICP nas Nuvens de Pontos

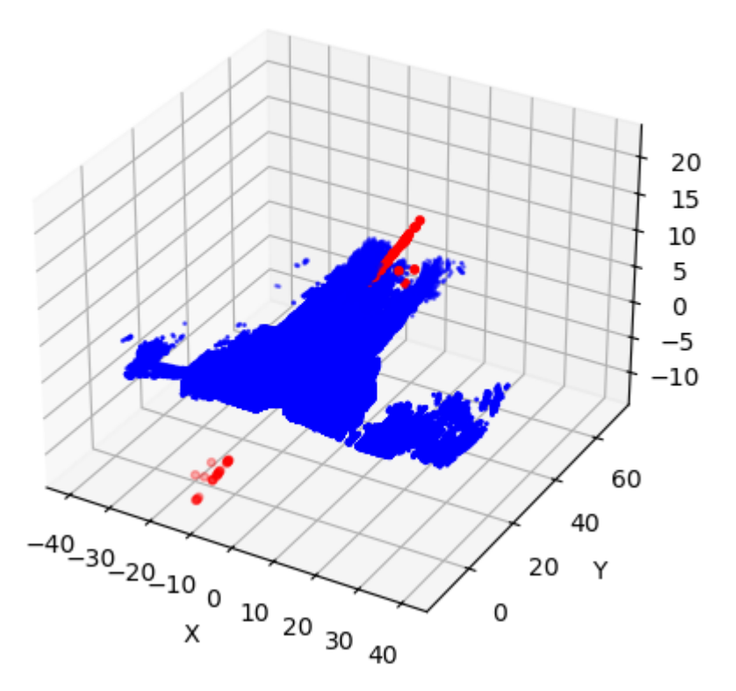
In [11]:
```python
trajetoria, nuvens_transformadas_icp, matrizes_transformacao_icp = aplicar_icp(nuvens)
```

5 - Visualização dos Resultados

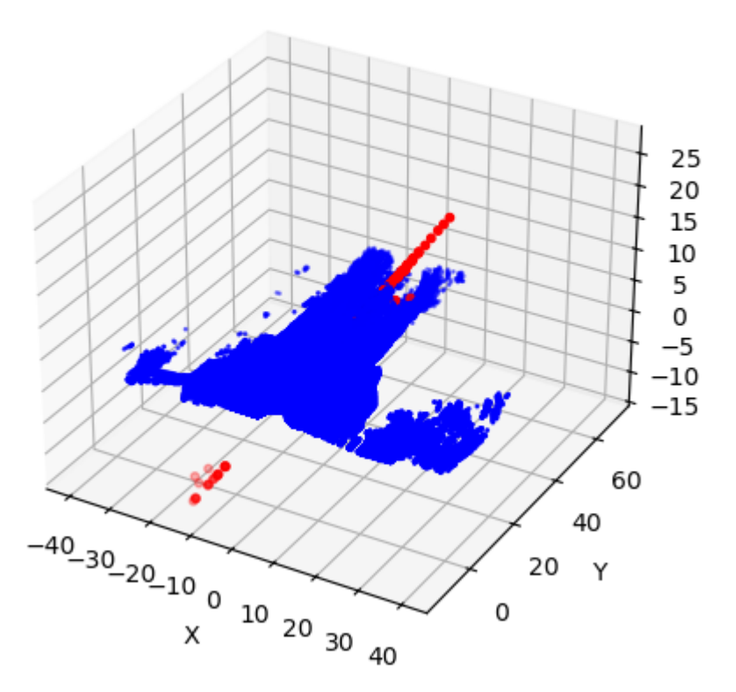
5.0 - Visualização 2D - ICP x Ground Truth

In [12]:
```python
titulo_icp = 'nuvem ICP'
titulo_gt = 'nuvem GT'

ground_truth = np.load('ground_truth.npy')
nuvens_transformadas_ground_truth = []

for i in range(len(nuvens)):
    nuvem_transformada = (ground_truth[i] @ np.vstack([nuvens[i].T, np.ones((1, nuvens[i].shape[0]))])).T
    nuvem_transformada = nuvem_transformada[:3].T
    nuvens_transformadas_ground_truth.append(nuvem_transformada)

visualizar_nuvens(nuvens, nuvens_transformadas_icp, titulo_icp)
visualizar_nuvens(nuvens, nuvens_transformadas_ground_truth, titulo_gt)
```

nuvem ICP



nuvem GT



5.1 - Visualização 3D - ICP x Ground Truth

In [13]:
```python
visualizar_nuvens_open3d(nuvens, nuvens_transformadas_icp)
visualizar_nuvens_open3d(nuvens, nuvens_transformadas_ground_truth)
```

6 - Análise dos resultados

6.0 - Criação de Dataframe com Métricas

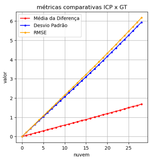
In [14]:
```python
metricas = []

for i in range(len(matrizes_transformacao_icp)):
    diferenca = abs(matrizes_transformacao_icp[i] - ground_truth[i])

    media = np.mean(diferenca)
    desvio_padrao = np.std(diferenca)
    maximo = np.max(diferenca)
    minimo = np.min(diferenca)
    rmse = np.sqrt(np.mean(diferenca**2))

    metricas.append({
        'Nuvem': i,
        'Média da Diferença': media,
        'Desvio Padrão': desvio_padrao,
        'RMSE': rmse,
        'Máximo': maximo,
        'Mínimo': minimo

    })

metricas = pd.DataFrame(metricas)
metricas
```

Out[14]:

	Nuvem	Média da Diferença	Desvio Padrão	RMSE	Máximo	Mínimo
0	0	6.283944e-09	2.419751e-08	2.500014e-08	1.000000e-07	0.0
1	1	5.886626e-02	2.069366e-01	2.151465e-01	8.588181e-01	0.0
2	2	1.177696e-01	4.135117e-01	4.299553e-01	1.716115e+00	0.0
3	3	1.764277e-01	6.204171e-01	6.450149e-01	2.574811e+00	0.0
4	4	2.350673e-01	8.271641e-01	8.599169e-01	3.432779e+00	0.0
5	5	2.940480e-01	1.033986e+00	1.074984e+00	4.291181e+00	0.0
6	6	3.528794e-01	1.240716e+00	1.289923e+00	5.149150e+00	0.0
7	7	4.109289e-01	1.447474e+00	1.504674e+00	6.006995e+00	0.0
8	8	4.681691e-01	1.652525e+00	1.717562e+00	6.857631e+00	0.0
9	9	5.402588e-01	1.863267e+00	1.940011e+00	7.737267e+00	0.0
10	10	5.881968e-01	2.068122e+00	2.150140e+00	8.583008e+00	0.0
11	11	6.470679e-01	2.274685e+00	2.364929e+00	9.440330e+00	0.0
12	12	7.038655e-01	2.481765e+00	2.579648e+00	1.029881e+01	0.0
13	13	7.646040e-01	2.688175e+00	2.794800e+00	1.115648e+01	0.0
14	14	8.304709e-01	2.895746e+00	3.012478e+00	1.202001e+01	0.0
15	15	8.758746e-01	3.100467e+00	3.221808e+00	1.286427e+01	0.0
16	16	9.472006e-01	3.308800e+00	3.441707e+00	1.373431e+01	0.0
17	17	1.007681e+00	3.515596e+00	3.657162e+00	1.459362e+01	0.0
18	18	1.062283e+00	3.729055e+00	3.877409e+00	1.547734e+01	0.0
19	19	1.135280e+00	3.943311e+00	4.103481e+00	1.637246e+01	0.0
20	20	1.184343e+00	4.160692e+00	4.325971e+00	1.726779e+01	0.0
21	21	1.247068e+00	4.379433e+00	4.553528e+00	1.817544e+01	0.0
22	22	1.308899e+00	4.597919e+00	4.780593e+00	1.908167e+01	0.0
23	23	1.362649e+00	4.821768e+00	5.010614e+00	2.000408e+01	0.0
24	24	1.434054e+00	5.039511e+00	5.239578e+00	2.091366e+01	0.0
25	25	1.496796e+00	5.262620e+00	5.471341e+00	2.183882e+01	0.0
26	26	1.560465e+00	5.488653e+00	5.706168e+00	2.277596e+01	0.0
27	27	1.601788e+00	5.719266e+00	5.939337e+00	2.371598e+01	0.0
28	28	1.686817e+00	5.940892e+00	6.175723e+00	2.465033e+01	0.0

6.1 - Visualização das Métricas

In [15]:
```python
# Definir o tamanho da figura
plt.figure(figsize=(5, 5))

plt.plot(metricas['Nuvem'], metricas['Média da Diferença'], marker='.', label='Média da Diferença', color='red')
plt.plot(metricas['Nuvem'], metricas['Desvio Padrão'], marker='.', label='Desvio Padrão', color='blue')
plt.plot(metricas['Nuvem'], metricas['RMSE'], marker='.', label='RMSE', color='orange')

plt.title('métricas comparativas ICP x GT')
plt.xlabel('nuvem')
plt.ylabel('valor')
plt.legend()
plt.grid()

plt.show()
```


métricas comparativas ICP x GT
Média da Diferença
Desvio Padrão
RMSE
valor
nuvem
0
1
2
3
4
5
6
0
5
10
15
20
25

[ ]