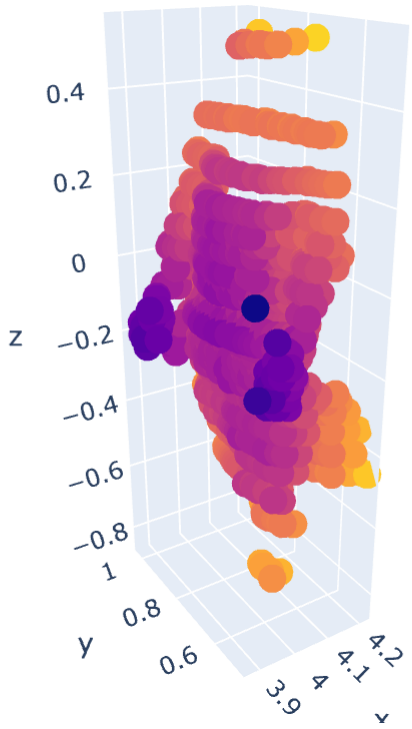
In [41]:

```
# Importing Libraries

import pandas as pd          # For converting the data into dataframe
import numpy as np           # For numeric computation
import plotly.express as px # For plotting the 3D scatter plot
import cvxpy as cp           # For Optimization
```

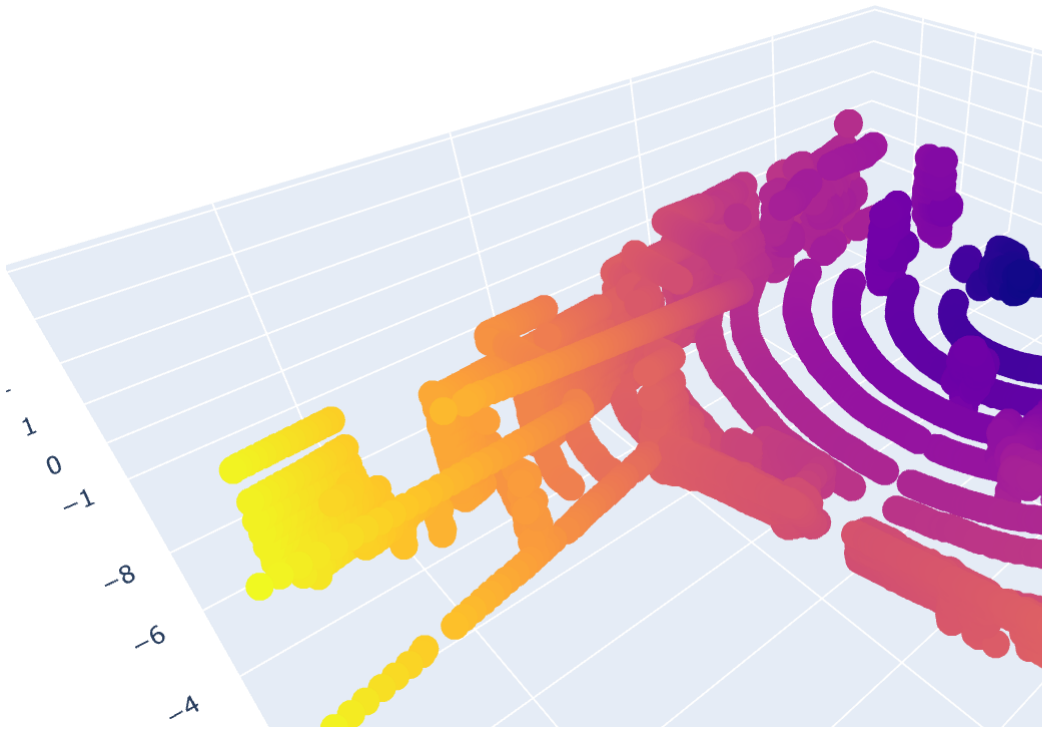
In [42]:

```
# Importing the objective file for constrains

df_d = pd.read_csv('object.txt', delimiter = ' ')
df_d.columns = ['x','y','z','a','b','c']
df_d =df_d.drop(['a','b','c'], axis = 1)
df_d['d'] = np.sqrt(df_d['x']**2 + df_d['y']**2 + df_d['z']**2)
data_d = np.asarray(df_d)
fig = px.scatter_3d( x=data_d[:,0], y=data_d[:,1], z=data_d[:,2], color =df_d['d'])
fig.show()
```



In [43]:

```
# Importing the objective file for scene

df_D = pd.read_csv('scene1.txt', delimiter = ' ')
df_D.columns = ['x','y','z','a','b','c']
df_D =df_D.drop(['a','b','c'], axis = 1)
df_D['d'] = np.sqrt(df_D['x']**2 + df_D['y']**2 + df_D['z']**2)
data_D = np.asarray(df_D)
fig = px.scatter_3d( x= data_D[:,0], y=data_D[:,1], z=data_D[:,2], color = df_D['d'], s
fig.show()
```



In [44]:

```
# Taking dimention of the segmented object to localize

d_dimention  = [[max(data_d[:,0]),min(data_d[:,0])],[max(data_d[:,1]),min(data_d[:,1])]
d_actual=np.asmatrix(d_dimention)
D_dimention  = [[max(data_D[:,0]),min(data_D[:,0])],[max(data_D[:,1]),min(data_D[:,1])]
dimention = [d_dimention, D_dimention]
D=np.asmatrix(D_dimention)

#fromat xmax, xmin, ymax, ymin, zmax, zmin
print(d_actual)
print(d_dimention[0])
```

```
[[ 4.22202492  3.86629534]
 [ 1.03018069  0.48565832]
 [ 0.5225758  -0.83467525]]
[4.22202492, 3.86629534]
```

In [45]:

```
# Calculating objective position for relative error

d_position = [np.mean(data_d[:,0]), np.mean(data_d[:,1]), np.mean(data_d[:,2])]
d_place = np.sqrt(d_position[0]**2 + d_position[1]**2 + d_position[2]**2)
```

In [46]:

```
# Generating optimization variables

x=cp.Variable((2,1))#x[1]=x_min,x[0]=x_max
y=cp.Variable((2,1))
z=cp.Variable((2,1))

```

In [47]:

```


d_est=cp.Variable((3,2))
```

In [48]:

```
#Generating Constrains

constraint1=[D_dimention[0][1]<=d_est[0,1],d_est[0,1]<=d_est[0,0],d_est[0,0]<=D_dimenti
constraint2=[D_dimention[1][1]<=d_est[1,1],d_est[1,1]<=d_est[1,0],d_est[1,0]<=D_dimenti
constraint3=[D_dimention[2][1]<=d_est[2,1],d_est[2,1]<=d_est[2,0],d_est[2,0]<=D_dimenti
# D_dimention[1][1]<=d_est[1,1]<=d_est[1,0]<=D_dimention[1][0], D_dimention[2][1]<=d_es
constraint=constraint1+constraint2+constraint3
```

In [49]:

```
#d_est=

objective_function=cp.Minimize(cp.norm((d_est-d_actual),2))

problem=cp.Problem(objective_function,constraint)

problem.solve()

print("Estimated position of the object is: ",d_est.value)
print("\n Actual position of the object is: ",d_actual)
print(np.linalg.norm(d_est.value-d_actual,2))
```

```
Estimated position of the object is:  [[ 4.22202523  3.86629503]
 [ 1.03017877  0.48566024]
 [ 0.52257838 -0.83467783]]

 Actual position of the object is:  [[ 4.22202492  3.86629534]
 [ 1.03018069  0.48565832]
 [ 0.5225758  -0.83467525]]
4.567199244556122e-06
```

In [ ]:

```
1
```