**COMPUTER PROGRAMMING LABORATORY**

**Experiment # 7:**

**Libs**

**QUESTIONS**

In this question, you will write a python program to calculate means of voxels, which describe the point cloud data. The program must include the following steps:

1. The program must include ***getParameters*** function. In the function, prompt the length of cube (i.e l) and step size (i.e ss) and return these values.
2. The program must include ***generatePointCloud*** function. The function receives the length of cube and returns a **numpy** array. In the function, use **numpy** library to generate 200 points with x, y, and z coordinates, which will be integer values between 0 and the length of cube. An example
3. w output for 15 points is given in Figure 1. Assume that the length of cube is 12.

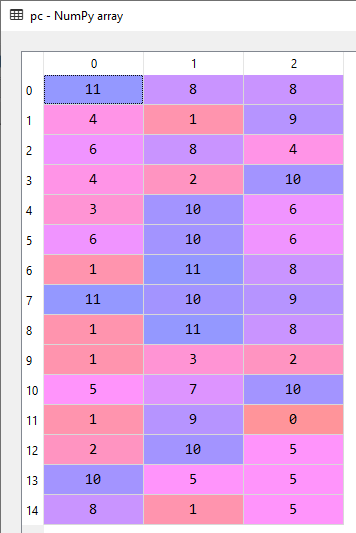


Fig. 1 An example numpy array for 15 points

1. The program must include ***writePCToDataFrameAndFile*** function. The function receives the numpy array. In the function, convert the numpy array to a DataFrame (**Hint: You can define Series for each column and add the columns to the DataFrame**). The column headers of the DataFrame must be axis\_1, axis\_2, and axis\_3 for x, y, and z axes. Write the DataFrame to a file with .csv extension. The values in the file must be separated with the slash character (“/”). Also, the file does not include index values. An example output for 15 points is given in Figure 2.

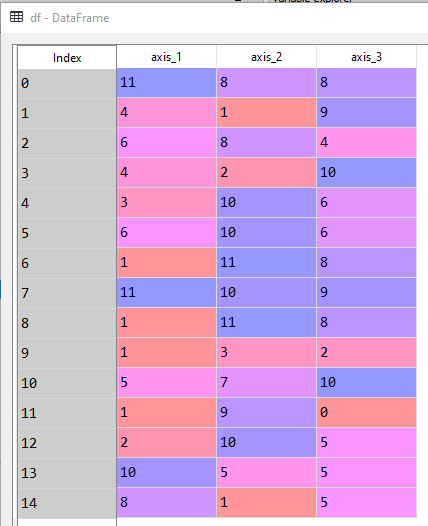


Fig. 2 An example DataFrame for 15 points

1. The program must include ***findPointsInAVoxel*** function. The function receives the numpy array, the length of cube and step size. In the function, first define an empty dictionary (i.e voxels). Assume that the length of cube 6 and the step size is 3 (Figure 3(a)). In this case, you will have 2 voxels for each axis. To reach each voxel use 3 nested for statement. Then, find the points in each voxel (Figure 3(b)). Assign these points into dictionary as values and the key of these values must be the lower-right corner of the voxel. An example dictionary is given in Figure 4. Notice that, voxels can contain more than 1 point and the values are in list type. **To implement conditions use & instead of and keyword.** The function must return the dictionary.

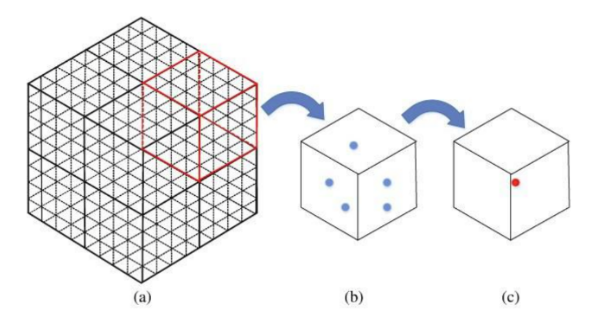


Fig. 3 An example to calculate mean of a voxel [1]

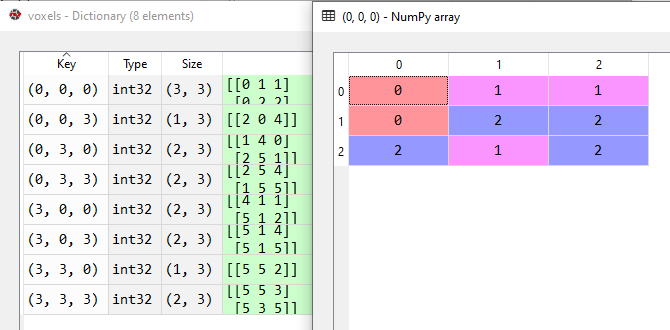


Fig. 4 An example dictionary. The right figure shows the values corresponding to (0,0,0) key

1. The program must include ***calculateVoxelMeans*** function. The function receives the dictionary, the length of cube, and step size. In the function, first define a numpy array with the size of dictionary and initialize the array with 0. Then, calculate means of the points in each voxel (Figure 3(c)). **In this step, you have to use for statement with the dictionary.** Then, assign the mean points to the numpy array. Write the numpy array to a file with .csv extension. The values in the file must be separated with the comma character (“,”). An example output for the numpy array is given in Figure 5.

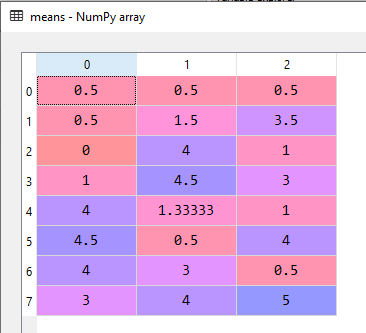


Fig. 5 An example output for the numpy array

1. The program must include ***plotFilteredPoints*** function. The function receives the filenames and plots the point cloud and mean points via matplotlib.pyplot library. In the function, read the point cloud file to a DataFrame and the mean points to a numpy array. Then, add the following statements to your function. Use **scatter** method to plot the points. An example output is given Figure 6. In the figure, point cloud points and mean points are shown with red and green colors, respectively.

from mpl\_toolkits.mplot3d import Axes3D // add the lib to your code

fig = plt.figure() // add the statement to your code

ax = fig.add\_subplot(111, projection='3d') // add the statement to your code

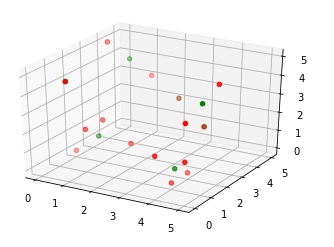


Fig. 6 An example output for plotFilteredPoints function

1. The program must include ***main*** function. In the function, call the functions ***getParameters, generatePointCloud, writePCToDataFrameAndFile, findPointsInAVoxel, calculateVoxelMeans, plotFilteredPoints***.