CS 548: Assignment 02

Programming Assignments (95%)

python/Assign02.py

Add the following code:

```python
import open3d as o3d
import numpy as np
from collections import deque

class OctoNode:
    indices = None
    children = None
    minP = None
    maxP = None

def traverse_octree(node):
    if node.indices is not None:
        return node.indices
    else:
        index_list = []
        for child in node.children:
            index_list.append(list(traverse_octree(child)))
        return index_list
```

OctoNode is a single node of an octree. Based on the values of *indices* and *children*:

- ***LEAF***
    - *children* is None
    - *indices* is a list of which points are inside the node (although it may be an empty list)
- ***NON-LEAF***
    - *children* is a list of the 8 OctoNodes that are subdivisions of this node
    - *indices* is None

minP and *maxP* represent the minimum and maximum corners (in x, y, z) of the AABB the node covers.

traverse_octree will return a nested list of point indices, indicating the structure of the octree.

In addition, you will add the following functions:

- **def get_best_AABB(cloud, margin=(0,0,0))**
    - This will return the min and max bounds (minP and maxP) of the best-fitting axis-aligned bounding box (AABB). It will also add on a margin to both the minimum and maximum points to make it slightly larger.
    - You can assume the cloud has at least one point.
    - minP and maxP should be numpy arrays, each with (x,y,z) coordinates.
    - I would recommend using the np.minimum and np.maximum functions.
    - You should return minP, maxP.
- **def split_box_octree(minP, maxP)**
    - Given a box defined by minP and maxP, split the box into octants and return a list of these boxes.
    - You should loop through Z, then Y, then X.
    - Each box is a list/tuple containing its boxMinP and boxMaxP.
    - Return the list of boxes.
- **def get_distance_from_box(minP, maxP, point)**
    - Given a box defined by minP and maxP, get the distance of point from that box.
    - If the point is INSIDE the box (conceptually: minP <= point <= maxP), then the distance is zero.
    - Otherwise, use the logic on Slide 75 of the (02) Nearest Neighbors slide deck to compute the distance.
    - Return the distance.
- **def get_indices_inside(points, indices, minP, maxP)**
    - Given a list of points, a list of indices, and a box defined by minP and maxP, produce a list of indices inside and a list of indices outside this box.
    - ONLY consider the points identified by indices.
    - If the distance is less than or equal to 1e-6, you can consider a point inside the box.
    - Return the inside list and the outside list.
- **def build_octree(cloud, margin, max_depth)**
    - Based on the points in a cloud, produce and return an octree.
    - Create a root node of type OctoNode.
    - Based on the cloud points and the margin, get the best AABB and set the corresponding minP and maxP values for the root.
    - I would recommend converting the list of cloud points to a numpy array for convenience.
    - Get the full list of indices.
    - You will need a queue (or at least a list) of nodes left to process.
        - I would recommend using deque from the collections module and then popleft() to remove and append() to add.

        - For each node, you will need (the node itself), (the indices at that node), (the current depth). I would advise adding all three items as a tuple to the queue.
    - While the queue isn't empty:
        - Get the next item off the queue.
        - If we are at max_depth OR the number of indices is 1 or less:
            - Set the indices for this node.
        - Otherwise:
            - Split the box for this node.
            - For each sub-box:
                - Create a new OctoNode with this sub-box.
                - Append this node as a child of the original node.
                - Get the list of indices inside and outside of this sub-box.
                    - NOTE: To avoid ambiguity, points will be assigned to a sub-box on a first-come, first-serve basis. In other words, a point that is inside box 2 will be removed from consideration, even if it was also inside box 5.
                    - Set the current indices equal to the points outside (i.e., points left to consider).
                    - **Do NOT set the indices of the child node yet!**
                - Append this child node (along with the current nodes inside as well as the depth+1) to the queue.
    - Return the root node of the tree.
- **def do_radius_search_octree(cloud, tree, query_point, radius)**
    - Given a cloud and an octree built from that cloud, get the list of indices that are at radius distance or less from the query_point.
    - You will need a queue or a list to keep track of which nodes still need to be processed.
        - *Note:* we are doing a radius search, so a more complicated priority queue on distance or min-heap is unnecessary.
    - Add the root to the queue.
    - While the queue isn't empty:
        - Get the next node from the queue.
        - If the node's indices are not None:
            - This is a leaf node.
            - Loop through the node's indices and check the Euclidean distance of each point from the query_point.
            - If the dist is less than or equal to the radius, add to the list of indices to keep.
        - Otherwise:
            - Loop through the children of this node:

        - If the distance of the child box from the query_point is less than or equal to the radius, add this child node to the queue.
    - Return the indices.

While optional, you may find it helpful to have the following main() function for visualization purposes:

```
def main():
    # Set filename
    filename = "data/assign02/bunny.pcd"

    # Load cloud
    cloud = o3d.io.read_point_cloud(filename)

    # Get points and colors
    points = np.asarray(cloud.points)
    colors = np.asarray(cloud.colors)

    # Build tree
    tree = build_octree(cloud, margin=(0.1,0.1,0.1), max_depth=4)

    # Do radius search
    query_point = np.array((6.79621601, 5.66879749, 8.07549858))
    radius = 1.7
    search_indices = do_radius_search_octree(cloud, tree, np.array(query_point), 
radius)

    # Set colors based on indices
    for i in range(len(colors)):
        colors[i] = (0,0,0)

    for index in search_indices:
        colors[index] = (255,0,0)

    # Visualize models
    o3d.visualization.draw_geometries([cloud])

if __name__ == "__main__":
    main()
```

Testing Screenshot (5%)

I have provided several files for testing:

- data/assign02
    - bunny.pcd
    - medium_cube.pcd
    - small_bunny.pcd
    - small_cube.pcd
    - tiny_cube.pcd
- python/
    - Test_Assign02.py – the test program for the Python code

Run the testing program through the testing section of Visual Code.

You MUST run the tests and send a screenshot of the test results! Even if your program(s) do not pass all the tests, you MUST send this screenshot!

None of the tests check the main functions; I will test this manually.

Python Tests

You may have to do "Command Palette" → "Python: Configure Tests" → pytest → python (directory)

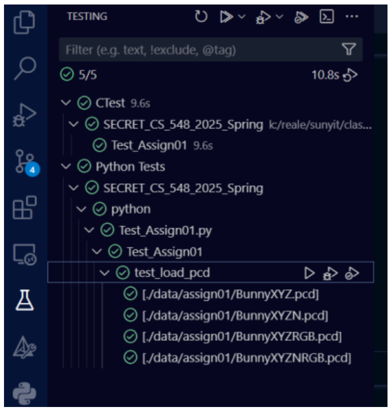
You should then be able to run the Python tests in your testing window in Visual Code.

ALTERNATIVELY: open a terminal and enter: **pytest python/Test_Assign02.py**

…then take a screenshot of the terminal output.

Screenshot Example

This screenshot should show clearly the testing view in Visual Code:



```
TESTING
Filter (e.g. text, !exclude, @tag)
5/5                                              10.8s
v CTest 9.6s
  v SECRET_CS_548_2025_Spring k:/reale/sunyit/clas...
      Test_Assign01 9.6s
v Python Tests
  v SECRET_CS_548_2025_Spring
    v python
      v Test_Assign01.py
        v Test_Assign01
          v test_load_pcd
              [./data/assign01/BunnyXYZ.pcd]
              [./data/assign01/BunnyXYZN.pcd]
              [./data/assign01/BunnyXYZRGB.pcd]
              [./data/assign01/BunnyXYZNRGB.pcd]
```

Grading

Your OVERALL assignment grade is weighted as follows:

- 5% - Testing results screenshot
- 95% - Programming assignments

I reserve the right to take points off for not meeting the specifications in this assignment description. In general, these are things that will be penalized:

- **Code that is not syntactically correct (up to 60 points off!)**
- Sloppy or poor coding style
- Bad coding design principles
- Code that crashes, does not run, or takes a VERY long time to complete
- Using code from ANY source other than the course materials
- Collaboration on code of ANY kind; this is an INDIVIDUAL PROJECT
- Sharing code with other people in this class or using code from this or any other related class
- Output that is incorrect
- Algorithms/implementations that are incorrect
- Submitting improper files
- Failing to submit ALL required files