深蓝学院
shenlanxueyuan.com

三维点云处理
第一节作业讲解

主讲人 贾清源

纲要

深蓝学院 shenlanxueyuan.com

- 第一部分：Kdtree & Octree
- 第二部分：benchmark

Kdtree & Octree

- 黎老师github实现:https://github.com/lijx10/NN-Trees
- [作业1]Kdtree的构建&自适应切割维度

```python
# 作业1
# 屏蔽开始
middle_left_idx = math.ceil(point_indices_sorted.shape[0] / 2) - 1
middle_left_point_idx = point_indices_sorted[middle_left_idx]
middle_left_point_value = db[middle_left_point_idx, axis]

middle_right_idx = middle_left_idx + 1
middle_right_point_idx = point_indices_sorted[middle_right_idx]
middle_right_point_value = db[middle_right_point_idx, axis]

root.value = (middle_left_point_value + middle_right_point_value) * 0.5
# === get the split position ===
root.left = kdtree_recursive_build(root.left,
                                   db,
                                   point_indices_sorted[0:middle_right_idx],
                                   axis_round_robin(axis, dim=db.shape[1]),
                                   leaf_size)
root.right = kdtree_recursive_build(root.right,
                                    db,
                                    point_indices_sorted[middle_right_idx:],
                                    axis_round_robin(axis, dim=db.shape[1]),
                                    leaf_size)
```

```python
#自适应切割维度
def axis_round_robin_adaptive(db, point_indices):
    data = db[point_indices, :]
    var = np.max(data, axis=0) - np.min(data, axis=0)
    axis_var_max = max(var[0], var[1], var[2])
    if axis_var_max == var[0]:
        return 0
    elif axis_var_max == var[1]:
        return 1
    elif axis_var_max == var[2]:
        return 2
```

kdtree和octree的实现代码在黎老师的github上都可以找到，作业代码跟里面的实现对比一下就可以了，建议大家至少看一遍实现逻辑；自适应切割维度这块是我单独实现了一个函数，求取当前数据的范围，取范围最大的一个维度作为下一轮的切分维度。

输入参数db是原始的数据，然后point_indices是递归步骤传入的索引，也就是point_indices_sorted[0:middle_right_idx]
和point_indices_sorted[middle_right_idx:]

Kdtree & Octree

●[作业4]Octree的构建

莫顿码对点的归属进行编码

|或运算，&与运算

000, 001 -> x

000, 010 -> y

000, 100 -> z

从而确定点所对应的象限

```python
# 作业4
# 屏蔽开始
root.is_leaf = False
children_point_indices = [[] for i in range(8)]
for point_idx in point_indices:
    point_db = db[point_idx]
    morton_code = 0
    if point_db[0] > center[0]:
        morton_code = morton_code | 1
    if point_db[1] > center[1]:
        morton_code = morton_code | 2
    if point_db[2] > center[2]:
        morton_code = morton_code | 4
    children_point_indices[morton_code].append(point_idx)
# create children
factor = [-0.5, 0.5]
for i in range(8):
    child_center_x = center[0] + factor[(i & 1) > 0] * extent
    child_center_y = center[1] + factor[(i & 2) > 0] * extent
    child_center_z = center[2] + factor[(i & 4) > 0] * extent
    child_extent = 0.5 * extent
    child_center = np.asarray([child_center_x, child_center_y, child_center_z])
    root.children[i] = octree_recursive_build(root.children[i],
                                              db,
                                              child_center,
                                              child_extent,
                                              children_point_indices[i],
                                              leaf_size,
                                              min_extent)
```

Octree这块有一个莫顿编码的一个操作，它用来确定点属于哪这个立方体网格中所属哪个象限。或运算就是有一个数是1的话，他就取到1，如果没有1的话，就是0。然后与运算就是只有当两个数字都为1，然后它才可以取成1。比如这里就是跟X轴，就是跟001进行位运算。然后Y轴就是跟2也就是二进制010进行位运算。然后Z轴和4也就是100进行位运算

Kdtree & Octree

深蓝学院
shenlanxueyuan.com

- [作业5]Octree fast radius search

如果当前节点被包含在worstDist当中，就需要将每个点都压入result_set.

如果该节点没有被完全包住，同时它是叶子节点，就需要将该节点的点都压入rseult_set.

```
Radius search normal:
Search takes 8587.214ms

Radius search fast:
Search takes 5818.956ms
```

```python
# 作业5
# 提示：尽量利用上面的inside、overlaps、contains等函数
# 屏蔽开始
if contains(query, result_set.worstDist(), root):
    # compare the contents of the octant
    leaf_points = db[root.point_indices, :]
    diff = np.linalg.norm(np.expand_dims(query, 0) - leaf_points, axis=1)
    for i in range(diff.shape[0]):
        result_set.add_point(diff[i], root.point_indices[i])
    # don't need to check any child
    return False

if root.is_leaf and len(root.point_indices) > 0:
    # compare the contents of a leaf
    leaf_points = db[root.point_indices, :]
    diff = np.linalg.norm(np.expand_dims(query, 0) - leaf_points, axis=1)
    for i in range(diff.shape[0]):
        result_set.add_point(diff[i], root.point_indices[i])
    # check whether we can stop search now
    return inside(query, result_set.worstDist(), root)

# no need to go to most relevant child first, because anyway we will go through all children
for c, child in enumerate(root.children):
    if child is None:
        continue
    if False == overlaps(query, result_set.worstDist(), child):
        continue
    if octree_radius_search_fast(child, db, result_set, query):
        return True
```

在fast radius search里头就是做了一个提前剪枝操作。如果当前节点被完全包含在worstdist当中，就需要将它的每个点都压入这个result set。
如果这个节点没有被完全包住，然后它还是叶子节点的话，将该节点的点都压入result set

benchmark

- 数据范围
- 调整min_extent

```python
def main():
    # configuration
    leaf_size = 32
    min_extent = 1
    k = 8
    radius = 1

    root_dir = '000000.bin' # 数据集路径
    db_np = read_velodyne_bin(root_dir) # (124668, 3)
    print(db_np.shape)
    print(np.min(db_np,axis=0))
    print(np.max(db_np,axis=0))
```

```
(124668, 3)
[-78.087395 -55.72341   -11.556541]
[77.96733    44.878613   2.8253412]
```

在这里我大概算了一下这个数据范围，它的是有12万多个点，然后它的最大和最小范围是以米来计算，大概x方向150m，y方向100m，z方向14m，因此我调整了一下octree的min extent值，不要太小

benchmark

● 搜索时间为0？

修改query点的索引，0点可能会被直接找到。

```python
print("kdtree --------------")
construction_time_sum = 0
knn_time_sum = 0
radius_time_sum = 0
brute_time_sum = 0
# for i in range(iteration_num):
#     filename = os.path.join(root_dir, cat[i])
#     db_np = read_velodyne_bin(filename)

begin_t = time.time()
root = kdtree.kdtree_construction(db_np, leaf_size)
construction_time_sum += time.time() - begin_t

query = db_np[10000,:]
```

然后有些同学是有搜索时间为0的情况。在KDtree里头，第0个点有可能被一下子就找到这块，建议就是改一下这个query的索引值

benchmark

●运行结果

- Kdtree建树比Octree快
- 搜索速度与参数相关，Kdtree一般较快
- Kdtree和Octree的knn是暴力算法耗时的1/4以下即可

```
octree --------------
Octree: build 5105.153, knn 1.996, radius 4.110, brute 10.108
kdtree --------------
Kdtree: build 229.542, knn 0.997, radius 4.986, brute 11.095
```

这里就是一个运行结果的一个截图，一般结论就是Kdtree建树要比octree要快很多。然后搜索速的话，其实跟一些参数有关系，跟K，跟leaf size，还有radius半径，以及octree的extent都相关，但是他们一般都是比暴力搜索要快很多，基本是暴力搜索的4倍以上

深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening