ikd-Tree: An Incremental k-d tree for robotic applications

USER MANUAL

Yixi Cai, yixicai@connect.hku.hk

1 Introduction:

1.1 What is ikd-Tree?

ikd-Tree is an incremental k-d tree designed for robotic applications. The ikd-Tree incrementally updates a k-d tree with new coming points only, leading to much lower computation time than existing static k-d trees. Besides point-wise operations, the ikd-Tree supports several features such as box-wise operations and down-sampling that are practically useful in robotic applications.

1.2 What can ikd-Tree do?

1) **Build Tree**: The ikd-tree can build a balanced k-d tree from input points.
2) **Incremental Updates**: Dynamically insert points to or delete points from the k-d tree.
3) **Box-wise Delete**: Delete points inside axis-aligned bounding boxes.
4) **Box-wise Recover**: Recover deleted points inside given axis-aligned bounding boxes.
5) **kNN Search**: Search k nearest neighbors within given range limitation.
6) **Box-wise Search**: Acquire points inside an axis-aligned bounding box on the k-d tree.
7) **Radius Search**: Acquire points inside a ball with given radius on the k-d tree.

2 Using ikd-Tree (C++ API)

2.1 Initialization & Parameter Setup

1) `KD_TREE`

```
KD_TREE<PointType>::KD_TREE(float delete_param, float balance_param, float
box_length)
```

Description: Constructor of ikd-Tree.

delete_param: The delete criterion parameter to trigger rebuilding process. If the number of invalid nodes on the (sub-)tree exceeds **delete_param** of the total number of nodes, rebuilding process is triggered to remove these invalid nodes. The default value is 0.5.

balance_param: The balance criterion parameter to trigger rebuilding process. If the number of nodes on the left sub-tree is larger than **balance_param** of the total number of nodes, rebuilding process is triggered to maintain the balance property of the k-d tree. The default value is 0.6.

box_length: The size of the downsampling box on the ikd-Tree (Unit: meter). The default value is 0.2.

2) `Set_delete_criterion_param`

```
void KD_TREE<PointType>::Set_delete_criterion_param(float delete_param)
```

Description: Set the delete criterion parameter.

delete_param: The delete criterion parameter to trigger rebuilding process.

3) `Set_balance_criterion_param`

```
void KD_TREE<PointType>::Set_balance_criterion_param(float balance_param)
```

Description: Set the balance criterion parameter.

balance_param: The balance criterion parameter to trigger rebuilding process.

4) `set_downsample_param`

```
void KD_TREE<PointType>::set_downsample_param(float downsample_param)
```

Description: Set the size of downsampling box on the ikd-Tree.

downsample_param: The size of the downsampling box on the k-d tree (Unit: meter).

5) `InitializeKDTree`

```
void KD_TREE<PointType>::InitializeKDTree(float delete_param, float
balance_param, float box_length)
```

Description: Initialize the parameters of ikd-Tree.

delete_param: The delete criterion parameter to trigger rebuilding process. The default value is 0.5.

balance_param: The balance criterion parameter to trigger rebuilding process. The default value is 0.6.

box_length: The size of the downsampling box on the ikd-Tree (Unit: meter). The default value is 0.2.

2.2 Tree Information

1) `size`

```
int KD_TREE<PointType>::size()
```

Description: Return the total number of nodes on the ikd-Tree, including both valid nodes and invalid nodes.

2) `tree_range`

```
BoxPointType KD_TREE<PointType>::tree_range()
```

Description: Return the axis-aligned bounding box of all points on the ikd-Tree. The return type is composed by two 1x3 arrays which represent the minimal and maximal value on each coordinate axis.

3) `validnum`

```
int KD_TREE<PointType>::validnum()
```

Description: Return the total number of valid nodes on the ikd-Tree.

4) `root_alpha`

```
void KD_TREE<PointType>::root_alpha(float &alpha_bal, float &alpha_del)
```

Description: Return the balance criterion value and balance criterion value of the ikd-Tree.

2.3 Tree Functions

1) `Build`

```
void KD_TREE<PointType>::Build(PointVector point_cloud)
```

Description: Build an ikd-Tree from point cloud input.

point_cloud: Point cloud input is stored in a vector of points with the type of **PointType**.

2) `Add_Points`

```
int KD_TREE<PointType>::Add_Points(PointVector & PointToAdd, bool
downsample_on)
```

Description: Insert new points into the ikd-Tree.

PointToAdd: A vector of new points in the type of **PointType**.

downsample_on: The bool parameter to determine whether downsampling is required for these new points when inserted to the ikd-Tree.

3) `Add_Point_Boxes`

```
void KD_TREE<PointType>::Add_Point_Boxes(vector<BoxPointType> & BoxPoints)
```

Description: Recover the deleted points (invalid nodes) inside given axis-aligned bounding boxes on the ikd-Tree.

BoxPoints: A vector of axis-aligned bounding boxes in the type of **BoxPointType**.

4) `Delete_Points`

```
void KD_TREE<PointType>::Delete_Points(PointVector & PointToDel)
```

Description: Delete points from the ikd-Tree.

PointToAdd: A vector of points to be deleted in the type of **PointType**.

5) `Delete_Point_Boxes`

```
int KD_TREE<PointType>::Delete_Point_Boxes(vector<BoxPointType> & BoxPoints)
```

Description: Delete points inside given axis-aligned bounding boxes from the ikd-Tree.

BoxPoints: A vector of axis-aligned bounding boxes in the type of **BoxPointType**.

6) `Nearest_Search`

```
void KD_TREE<PointType>::Nearest_Search(PointType point, int k_nearest,
PointVector& Nearest_Points, vector<float> & Point_Distance, double max_dist)
```

Description: Search k nearest neighbors of the target point on the ikd-Tree.

point: The target point to find nearest neighbors of.

k_nearest: The number of nearest neighbors to search.

Nearest_Points: Return the nearest neighbor points of the target point.

Point_Distance: Return the distance from the nearest neighbor points to the target point (squared distance, Unit: m^2).

max_dist: The range limitation to find nearest neighbor (Unit: meter).

7) `Box_Search`

```
void KD_TREE<PointType>::Box_Search(const BoxPointType &Box_of_Point,
PointVector &Storage)
```

Description: Return the points inside the given axis-aligned bounding box.

Box_of_Point: The target axis-aligned bounding box.

Storage: Return points inside the bounding box.

`8)  Radius_Search`

```
void KD_TREE<PointType>::Radius_Search(PointType point, const float radius,
PointVector &Storage)
```

Description: Return the points inside a ball with the given center point and the radius.

point: The center of the target ball.

radius: The radius of the target ball.

Storage: Return points inside the ball.

`9)  flatten`

```
void KD_TREE<PointType>::flatten(KD_TREE_NODE * root, PointVector &Storage,
delete_point_storage_set storage_type)
```

Description: Return the valid points rooted at a given node on the ikd-Tree. This function is set public to retrieve valid points on the ikd-Tree by providing its root node.

root: The root node of the (sub-)tree to retrieve points.

Storage: Return points on the (sub-)tree.

storage_type: The parameter to determine whether the deleted points should be recorded in a buffer before being removed.

`10) acquire_removed_points`

```
void KD_TREE<PointType>::acquire_removed_points(PointVector & removed_points)
```

Description: Return the removed points in the buffer and clear the buffer.

removed_points: Return the removed points.

3 Reference

[1] Cai, Y., Xu, W., & Zhang, F. (2021). ikd-Tree: An Incremental KD Tree for Robotic Applications. arXiv preprint arXiv:2102.10808.

[2] Xu, W., Cai, Y., He, D., Lin, J., & Zhang, F. (2022). Fast-lio2: Fast direct lidar-inertial odometry. IEEE Transactions on Robotics.