python-kdtree Documentation

Release 0.15

Stefan Kögl

Apr 24, 2017

Contents

The `kdtree` package can construct, modify and search kd-trees.

CHAPTER 1

Example Usage

```
>>> import kdtree

# Create an empty tree by specifying the number of
# dimensions its points will have
>>> emptyTree = kdtree.create(dimensions=3)

# A kd-tree can contain different kinds of points, for example tuples
>>> point1 = (2, 3, 4)

# Lists can also be used as points
>>> point2 = [4, 5, 6]

# Other objects that support indexing can be used, too
>>> import collections
>>> Point = collections.namedtuple('Point', 'x y z')
>>> point3 = Point(5, 3, 2)

# A tree is created from a list of points
>>> tree = kdtree.create([point1, point2, point3])

# Each (sub)tree is represented by its root node
>>> tree
<KDNode - [4, 5, 6]>

# Adds a tuple to the tree
>>> tree.add( (5, 4, 3) )

# Removes the previously added point and returns the new root
>>> tree = tree.remove( (5, 4, 3) )

# Retrieving the Tree in inorder
>>> list(tree.inorder())
[<KDNode - (2, 3, 4)>, <KDNode - [4, 5, 6]>, <KDNode - Point(x=5, y=3, z=2)>]

# Retrieving the Tree in level order
```

```
>>> list(kdtree.level_order(tree))
[<KDNode - [4, 5, 6]>, <KDNode - (2, 3, 4)>, <KDNode - Point(x=5, y=3, z=2)>]

# Find the nearest node to the location (1, 2, 3)
>>> tree.search_nn( (1, 2, 3) )
<KDNode - (2, 3, 4)>

# Add a point to make the tree more interesting
>>> tree.add( (10, 2, 1) )

# Visualize the Tree
>>> kdtree.visualize(tree)


                     [4, 5, 6]

           (2, 3, 4)       Point(x=5, y=3, z=2)

                            (10, 2, 1)

# Take the right subtree of the root
>>> subtree = tree.right

# and detatch it
>>> tree.right = None
>>> kdtree.visualize(tree)

           [4, 5, 6]

      (2, 3, 4)

>>> kdtree.visualize(subtree)

      Point(x=5, y=3, z=2)

      (10, 2, 1)

# and re-attach it
>>> tree.right = subtree
>>> kdtree.visualize(tree)

                     [4, 5, 6]

           (2, 3, 4)       Point(x=5, y=3, z=2)

                            (10, 2, 1)

# Add a node to make the tree unbalanced
>>> tree.is_balanced
True
>>> tree.add( (6, 1, 5) )
>>> tree.is_balanced
False
>>> kdtree.visualize(tree)

                                   [4, 5, 6]

               (2, 3, 4)                           Point(x=5, y=3, z=2)
```

```
                                               (10, 2, 1)
                                                       (6, 1, 5)
# rebalance the tree
>>> tree = tree.rebalance()
>>> tree.is_balanced
True
>>> kdtree.visualize(tree)

                Point(x=5, y=3, z=2)

           [4, 5, 6]            (6, 1, 5)

      (2, 3, 4)
```

CHAPTER 2

Indices and tables

- genindex
- modindex
- search