原创 你东东哥哥 于 2018-03-08 18:40:18 发布 6364 收藏 44
分类专栏： 学习笔记 文章标签： octomap

学习笔记 专栏收录该内容 0 订阅 5 篇文章 订阅专栏

分类专栏
学习笔记 5篇

octomap库的一点总结

前些天突然想起高博《视觉SLAM十四讲》提到的octomap八叉树地图，在网上百度又只有高博写的那么一片博文，于是就想自己看看这个库，了解其几个基本的类和接口，以下是我这几天来的总结，如果有错误请及时指出，或者联系我，我也会及时改正。
（yuanliudongdong@163.com）

一，官方的 Introduction部分翻译(http://octomap.github.io/octomap/doc/index.html)

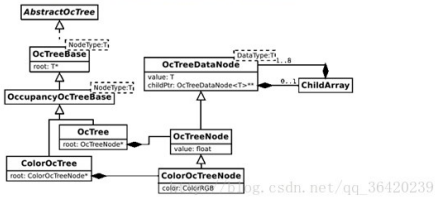
Introduction

The OctoMap library implements a 3D occupancy grid mapping approach. It provides data structures and mapping algorithms. The map is implemented using an Octree. It is designed to meet the following requirements:

Octomap库实现了一种填充3D栅格的构图方式，并提供了相应的数据结构和构图算法。整个地图利用一个Octree的类来实现。

Getting Started

Jump right in and have a look at the main class octomap::OcTree OcTree and the examples in src/octomap/simple_example.cpp. To integrate single measurements into the 3D map have a look at OcTree::insertRay(...), to insert full 3D scans (pointclouds) please have a look at OcTree::insertPointCloud(...). Queries can be performed e.g. with OcTree::search(...) or OcTree::castRay(...). The preferred way to batch-access or process nodes in an Octree is with the iterators leaf_iterator, tree_iterator, or leaf_bbx_iterator.



直接查看本库主要的类octomap::OcTree，以及其示例程序src/octomap/simple_example.cpp。如果你想要将单组测量数据整合到3D地图中，请查看OcTree::insertRay(…)函数；如果你想要将整个3D扫描数据（点云）整合到地图中，请查看OcTree::insertPointCloud(…)函数。同时可以通过OcTree::search(...) 和OcTree::castRay(...)函数来进行地图空间占用的查询。推荐使用迭代器的方式对Octree的节点等进行查询，例如leaf_iterator，tree_iterator和leaf_bbx_iterator。

二，示例代码分析(Github源码地址：https://github.com/OctoMap/octomap)

在使用该库时，主要使用了基本的OcTree类和带有颜色信息的ColorOcTree类，所以只简单分析了这两个类的示例代码。

1，octomap/octomap/src/simple_example.cpp

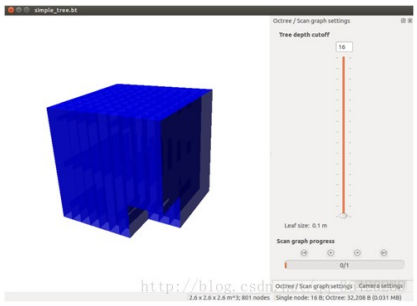
```
1.  #include <octomap/octomap.h>
2.  #include <octomap/OcTree.h>
3.  using namespace std;
4.  using namespace octomap;
5.
6.  //查询信息输出函数，输入为octomap命名空间下的3D点位置和用OcTree::search(...)得到的返回值
7.  void print_query_info(point3d query, OcTreeNode* node)
8.  {
9.      if (node != NULL)
10.     {
11.        cout << "occupancy probability at " << query << ":\t " << node->getOccupancy() << endl;
12.     }
13.     else
14.       cout << "occupancy probability at " << query << ":\t is unknown" << endl;
15. }
16.  // 主函数入口
17. int main(int argc, char** argv)
18. {
19.   cout << endl;
20.   cout << "generating example map" << endl;
21.   // 建立一个空的OcTree对象地图，分辨率为0.1
22.   OcTree tree (0.1);  // create empty tree with resolution 0.1
23.
24.   // insert some measurements of occupied cells
25.   // 向地图中插入一些测量数据，这些数据点在地图中表现为已占用状态
26.   for (int x=-20; x<20; x++)
27.   {
28.     for (int y=-20; y<20; y++)
29.     {
30.       for (int z=-20; z<20; z++)
31.       {
32.         //建立空间点对象，并向地图中更新节点
33.         point3d endpoint ((float) x*0.05f, (float) y*0.05f, (float) z*0.05f);
34.         tree.updateNode(endpoint, true); // integrate 'occupied' measurement
35.       }
36.     }
37.   }
38.
39.   // insert some measurements of free cells
40. // 向地图中插入一些测量数据，这些数据点在地图中表现为已未占用状态
41.   for (int x=-30; x<30; x++)
42.   {
43.     for (int y=-30; y<30; y++)
44.     {
45.       for (int z=-30; z<30; z++)
46.       {
47.         //建立空间点对象，并向地图中更新节点
48.         point3d endpoint ((float) x*0.02f-1.0f, (float) y*0.02f-1.0f, (float) z*0.02f-1.0f);
49.         tree.updateNode(endpoint, false);  // integrate 'free' measurement
50.       }
51.     }
52.   }
53. //上面已经完成了对地图的创建，之后的程序是对地图节点数据的访问
54.   cout << endl;
55.   cout << "performing some queries:" << endl;
56.  // 节点 (0,0,0)
57.   point3d query (0., 0., 0.);
58.   OcTreeNode* result = tree.search (query);
59.   print_query_info(query, result);
60. // 节点 (-1,-1,-1)
61.   query = point3d(-1.,-1.,-1.);
62.   result = tree.search (query);
63.   print_query_info(query, result);
64. // 节点 (1,1,1)
65.   query = point3d(1.,1.,1.);
66.   result = tree.search (query);
67.   print_query_info(query, result);
68.   cout << endl;
```

```
69. // 将建好的地图保存
70.   tree.writeBinary("simple_tree.bt");
71.   cout << "wrote example file simple_tree.bt" << endl << endl;
72.   cout << "now you can use octovis to visualize: octovis simple_tree.bt"  << endl;
73.   cout << "Hint: hit 'F'-key in viewer to see the freespace" << endl  << endl;
74. }
```

这是执行该代码后的运行结果：

图表 1

这是用octovis查看所生成的地图simple_tree.bt：



图表 2

下面针对一些关键语句进行分析：

(1)OcTree tree(0.1);建立OcTree地图对象。这个函数的声明是这样的：

resolution译为“分辨率”，即访问地图的最小单位；

(2) point3d endpoint((float)x*0.02f-1.0f, (float)y*0.02f-1.0f, (float)z*0.02f-1.0f);

tree.updateNode(endpoint, false);

建立一个三维空间点，并将其添加到地图当中；point3d构造函数的三个参数分别是三维点的x/y/z坐标；使用.updateNode()函数将空间点信息更新到地图当中，第二个参数表示该空间点对应的节点未被占用(false);

(3) OcTreeNode* result = tree.search (query);节点信息查询函数；函数声明是这样的

传入参数为一个三维点对象，如果在这个地图tree中，这个三维点对应的位置有节点（不管占用与否），那么将返回该位置的节点指针，否则将返回一个空指针；

(4) cout << "occupancy probability at " << query<< ":\t " << node->getOccupancy() << endl；如果查询到有该位置上节点的信息，则使用getOccupancy()函数输出该节点占用情况，那为什么这个Occupancy是个小数呢？这是因为Octomap在描述一个栅格是否被占用时，并不是单一的只描述为占用和被占用，而是用一个概率（Occupancy probability）来描述它，即这个栅格被占用的概率是多少，通过这个概率来确定这个栅格被占用的可能性。

2，Occupancy probability

关于这个占用概率值，高博在一篇博客中做出了和通俗易懂的解释，我就不再复制粘贴了，下面附上连接(https://www.cnblogs.com/gaoxiang12/p/5041142.html)。

我要说的是，对于一个最小单位的栅格，如何判断updateNode()函数对其做出了贡献或者说更新。就比如说

```
1.          point3d endpoint( 4.05f, 4.05f, 4.05f );
2.
3.          tree.updateNode( endpoint, true );
```

这两行代码，由于分辨率是0.1，并不能精确到0.01，所以要把这个（4.05,4.05,4.05）归到（4.0,4.0,4.0）这个节点呢，还是（4.1,4.1,4.1）这个节点或者其他节点呢？经过我的多次测试，应当归到（4.0,4.0,4.0）节点当中，也就是说，对于节点（4.0,4.0,4.0）来说，凡是同时满足x=[4.0,4.1)，y=[4.0,4.1)，z=[4.0,4.1)的point，如果使用updateNode()函数将这个point更新到地图当中，那么必然会影响到节点（4.0,4.0,4.0）的Occupancy probability。

总结就是，对于一个点point(x,y,z)，使用updateNode()函数将其更新到地图当中，那么Occupancy probability受到影响的节点将是

$$node(x_0,y_0,z_0),(x_0 \le x, x_0 \ge x-resolution, y_0 \le y, y_0 \ge y-resolution, z_0 \le z, z_0 \ge z-resolution)$$

即小于point坐标值的最大节点。同时当我们用search ()函数对point进行查询时，返回的信息也将是小于point坐标值的最大节点信息。

另外，在对某个节点进行不同次数的更新之后，发现Occupancy probability最大值为0.971，最小值为0.1192，这也验证了高博那篇博文中提到的最大值和最小值限制。

3，octomap/src/testing/test_color_tree.cpp

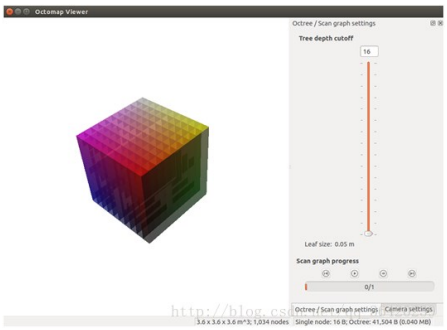
带有色彩的八叉树地图ColorOcTree与基本的OcTree类似，需要知道如何向节点当中添加颜色信息就好了。下面是test_color_tree.cpp的部分代码：

```
1.  int main(int argc, char** argv)
2.
3.  {
4.
```

```
3.    //分辨率
4.    double res = 0.05;  // create empty tree with resolution 0.05(different from default 0.1 for test)
5.    //建立彩色地图对象
6.    ColorOcTree tree (res);
7.    // insert some measurements of occupied cells
8.    for (int x=-20; x<20; x++)
9.    {
10.     for (int y=-20; y<20; y++)
11.     {
12.       for (int z=-20; z<20; z++)
13.       {
14.         point3d endpoint ((float)x*0.05f+0.01f, (float) y*0.05f+0.01f, (float) z*0.05f+0.01f);
15.         ColorOcTreeNode* n =tree.updateNode(endpoint, true);
16.         //设置节点颜色信息的函数，每个地图节点差五个像素大小，渐变
17.         n->setColor(z*5+100,x*5+100,y*5+100);
18.       }
19.     }
20.   }
21.
22.   // insert some measurements of free cells
23.   for (int x=-30; x<30; x++)
24.   {
25.     for (int y=-30; y<30; y++)
26.     {
27.       for (int z=-30; z<30; z++)
28.       {
29.         point3d endpoint ((float) x*0.02f+2.0f,(float) y*0.02f+2.0f, (float) z*0.02f+2.0f);
30.         ColorOcTreeNode* n =tree.updateNode(endpoint, false);
31.         //不被占用的节点设置为黄色
32.         n->setColor(255,255,0); // set colorto yellow
33.       }
34.     }
35.   }
36.   // set inner node colors
37.   tree.updateInnerOccupancy();
38. }
```

这是执行该代码后的运行结果：

这是用octovis查看所生成的地图simple_color_tree.ot：



三，安装和编译octomap库时的一个小问题

第一次编译安装octomap库时，直接按照网上的教程进行的操作：

```
1.    git clone https://github.com/OctoMap/octomap
2.
3.    cd octomap
4.
5.    mkdir build
6.
```

```
4.  cd build

5.  cmake ..

6.  make

7.  sudo make install
```

可是在执行make指令时却出现了undefined reference to的错误

在网上百度了好久，找到了关于这个问题的帖子（https://github.com/OctoMap/octomap/issues/171）

1，　我首先用的方法是 sudo make，确实，编译通过，也安装成功了，但是我在编译自己写的octomap_test程序时，仍然会出现类似的undefined reference to的错误，于是在编译时还是需要sudo；

2，　后来我觉得有一个对方法1的评价特别对：

于是我尝试了第二种做法：

打开查询了这个/opt/ros/indigo/share/octpmap/octomap-config-version文件，我的版本是1.6.9，在重新下载了octomap源码之后，校对了版本，具体指令如下：

```
1.  git clone https://github.com/OctoMap/octomap

2.  cd octomap

3.  git tag   //列出所有版本，查看是否有自己的版本

4.  git checkout v1.6.9   //校验为自己的版本

5.  mkdir build

6.  cd build

7.  cmake ..

8.  make

9.  sudo make instal
```


感觉第二种做法才是真正的解决方案。

6	54万+	122万+	1万+	
原创	周排名	总排名	访问	等级

268	4	15	8	66
积分	粉丝	获赞	评论	收藏

私信 关注

看我是怎么学SLAM(二)——3D自主探索建图源码解读_周小枫的博客-CSDN博 ... 1-25
这份源码能实现Kobuki自主探索房间并建立房间的三维地图,地图表现形式是八叉树。 源码下载与配置 步骤: 安装octomap_ros和rviz插件 sudo apt-getinst…

octomap的入门与学习 qq_42424625的博客 1万+
octomap的入门与学习octomap简介八叉树的表达八叉树的更新安装 octomap简介 octomap是一种基于八叉树的三维地图创建工具, 可以显示包含无障碍区…

【Octomap】入门与自定义地图的实现 weixin_46737655的博客 278
Octomap浅入门：涉及Octomap的入门使用教程以及利用Octomap自建.bt的个性化地图

octomap(3):搭建一个完整的octomap建图工作空间 biter0088的博客 918
说明：下面的步骤将octomap、octomap_server、octomap_ros、octomap_msgs等功能包的源码（cpp、.h等）进行了集中，使其在一个工作空间中，这…

采用Cartographer、LIO-SAM构建三维点云地图，采用Octomap构建八叉树地图（三维栅格地图） weixin_44570248的博客 2873
采用Cartographer、LIO-SAM构建三维点云地图，采用Octomap构建八叉树地图（三维栅格地图） 采用Cartographer构建三维点云地图 采用的数据集是…

octomap_server使用－－生成二维占据栅格地图和三维概率地图 sru_alo的博客 1万+
octomap_server是ROS中的一个基于octomap的功能包。我在查阅资料的时候，发现所有的介绍、博客等资料都是在介绍其将点云地图转化为基于Oc…

Octomap Yubao 的专栏 764
Octomap Github: https://github.com/OctoMap/octomap.git Official: http://octomap.github.io/ API 文档：http://octomap.github.io/octomap/doc/ SLAM拾萃…

octomap学习 朝闻雨的博客 1047
#include<octomap/octomap.h> octomap::OcTree tree(o.o5); //参数…

Octomap原理 qq_57061492的博客 127
Octomap原理简述

octomap入门学习 Li03 2722
从今天开始学习octomap，对于octomap还是一无所知的状态，一起来学习一下。 1 What is octomap? The OctoMap library implements a 3D occupanc…

用octomap生成的二维栅格地图进行move_base路径规划仿真 heirenlop的博客 2192
1.map_server保存二维栅格地图 octomap生成的二维栅格地图可以用map_server保存 map_server是个功能包,这个功能包可以单独下载,也可以直接下载n…

他们喜欢把T265和D435i结合起来用（T265是定位，D435i是建图，两者合起来就是同时定位和建图，不就是SLA… TYINY的博客 3529
我这篇博文之前说了 https://blog.csdn.net/sinat_16643223/article/details/107784885 今天又看到一些 https://www.youtube.com/channel/UC78JXTJ-Y3Zp…

【SLAM】----Ubuntu16.04配置环境tensorflow-gpu1.5.0+cuda9.0+cudnn7.0+opencv2.4.11+Anaconda3+pcl1.7+octomap rick的博客 148
提示：文章写完后，目录可以自动生成，如何生成可参考右边的帮助文档 文章目录前言一、CUDA9.0安装二、安装cudnn7.0.52.读入数据总结 前言 提示…

【SLAM】之建图Bag->Pcd->OctoMap 热门推荐 littlethunder的专栏 2万+
上篇中我们得到了3D激光雷达获得的点云图，存在.bag文件中，接下来我们再用上上篇末尾的做法跑loam_velodyne算法，在RVIZ中的显示效果如下： 这…