码龄7年 暂无认证
54 原创 | 5万+ 周排名 | 124万+ 总排名 | 9万+ 访问 | 等级
1294 积分 | 49 粉丝 | 71 获赞 | 27 评论 | 559 收藏
私信 关注

搜博主文章

热门文章
STM32定时器使用（三）外部脉冲计数 16426
STM32定时器使用（二）PWM方波输出 11121
STM32定时器使用（一）时钟定时 7386
ubuntu更换清华源 6846
ROS环境搭建 4255

最新评论
STM32定时器使用（三）外部脉冲计数
one coward: 能分享一下源码吗，大佬。2565135701@qq.com
ubuntu下u盘只读不能写的问题解决
踏实写代码，认真搞学术的小研: 我感谢你18代祖宗阿，太好了
ROS引用同工作空间其他功能包的自定…
学海菜鸡: 你好,我在使用的时候提示被引用的功能包`A_packge`没有编译，在我的5…
ROS引用同工作空间其他功能包的自定…
wuli罗里: 不同工作空间下的自定义消息头文件怎么导入呢
ROS引用同工作空间其他功能包的自定…
wqm-story027: A的1）不需要修改，改了之后编译会报错

您愿意向朋友推荐“博客详情页”吗？
强烈不推荐 不推荐 一般般 推荐 强烈推荐

最新文章
Eigen3实现欧拉角、四元数、旋转矩阵之间的变换
ubuntu更换清华源
深入理解B样条曲线
2022年 2篇 2021年 9篇
2020年 12篇 2019年 31篇

原创 guijiaqing 于 2022-01-18 12:18:22 发布 673 收藏 2 版权
分类专栏： ROS学习之路 文章标签： c++

ROS学习之路 专栏收录该内容 3 订阅 28 篇文章 订阅专栏

分类专栏
ROS学习之路 28篇
机器人学 10篇
Linux 6篇
STM32 9篇
python 2篇

1，OcTreeKey adjustKeyAtDepth (const OcTreeKey &key, unsigned int depth) const

Adjusts a 3D key from the lowest level to correspond to a higher depth (by shifting the key values)
从最低级别调整3D关键点以对应更高的深度（通过移动关键点值）

2 size_t calcNumNodes () const

Traverses the tree to calculate the total number of nodes.
遍历树以计算节点总数。

3 virtual bool castRay (const point3d &origin, const point3d &direction, point3d &end, bool ignoreUnknownCells=false, double maxRange=-1.0) const

Performs raycasting in 3d, similar to computeRay().Can be called in parallel e.g. with OpenMP for a speedup.A ray is cast from 'origin' with a given direction, the first non-free cell is returned in 'end' (as center coordinate). This could also be the origin node if it is occupied or unknown. castRay() returns true if an occupied node was hit by the raycast. If the raycast returns false you can search() the node at 'end' and see whether it's unknown space.
重要函数：
有点类似于computeRay()作用是在3D空间里进行光线投射，可以和OpenMP并发使用来提高速度。一条光束从原点沿着指定的方向出发，遇到的第一个non-free格子被记录在end里（它会作为这个格子的中心坐标）。原始节点也有可能是被占或者未知的格子。如果在光束中间遇到一个被已经填充了的格子，则castRay()返回的是true，相反如果返回的是false的话，说明中间没有被占用的，那么我们可以查看end来区分，最后一个是否是未知空间。

4 changedKeysBegin () const

Iterator to traverse all keys of changed nodes.
you need to enableChangeDetection() first. Here, an OcTreeKey always refers to a node at the lowest tree level (its size is the minimum tree resolution)
遍历已更改节点的所有键的迭代器。
您需要首先启用ChangeDetection（）。在这里，OcTreeKey总是指最低树级别的节点（其大小是最小树分辨率）

5 void clear ()

Deletes the complete tree structure.
删除完整的树结构

6 void clearKeyRays ()

Clear KeyRay vector to minimize unneeded memory.
This is only useful for the StaticMemberInitializer classes, don't call it for an octree that is actually used.
清除KeyRay vector以最小化不必要的内存。
这只对StaticMemberInitializer类有用，不要为实际使用的八叉树调用它。

7 void computeDiscreteUpdate (const Pointcloud &scan, const octomap::point3d &origin, KeySet &free_cells, KeySet &occupied_cells, double maxrange)

Helper for insertPointCloud().Computes all octree nodes affected by the point cloud integration at once. Here, occupied nodes have a preference over free ones. This function first discretizes the scan with the octree grid, which results in fewer raycasts (=speedup) but a slightly different result than computeUpdate().
insertPointCloud（）的帮助程序。
一次计算受点云集成影响的所有八叉树节点。在这里，被占用的节点比空闲的节点具有更高的优先级。此函数首先使用八叉树网格对扫描进行 离散化，这会减少光线投射（=加速），但结果与computeUpdate（）略有不同。

8 bool computeRay (const point3d &origin, const point3d &end, std::vector< point3d > &ray)

Traces a ray from origin to end (excluding), returning the coordinates of all nodes traversed by the beam.You still need to check if a node at that coordinate exists (e.g. with search()).
从原点到端点（不包括）跟踪光线，返回光束穿过的所有节点的坐标。
您仍然需要检查该坐标处是否存在节点（例如，使用search（））。
KeyRay structure that holds the keys of all nodes traversed by the ray, excluding "end"
KeyRay结构，包含射线穿过的所有节点的键，不包括"end"
返回false通常意味着其中一个坐标超出八叉树的范围

9 bool computeRayKeys (const point3d &origin, const point3d &end, KeyRay &ray) const

Traces a ray from origin to end (excluding), returning an OcTreeKey of all nodes traversed by the beam.
You still need to check if a node at that coordinate exists (e.g. with search()).
从原点到终点（不包括）跟踪光线，返回光束所经过的所有节点的八叉键。
您仍然需要检查该坐标处是否存在节点（例如，使用search（））。

10 void computeUpdate (const Pointcloud &scan, const octomap::point3d &origin, KeySet &free_cells, KeySet &occupied_cells, double maxrange)

Helper for insertPointCloud().
Computes all octree nodes affected by the point cloud integration at once. Here, occupied nodes have a preference over free ones.
insertPointCloud（）的帮助程序。
一次计算受点云集成影响的所有八叉树节点。在这里，被占用的节点比空闲的节点具有更高的优先级。

11 coordToKey

OcTreeKey coordToKey (const point3d &coord) const
OcTreeKey coordToKey (const point3d &coord, unsigned depth) const
key_type coordToKey (double coordinate) const
key_type coordToKey (double coordinate, unsigned depth) const
OcTreeKey coordToKey (double x, double y, double z) const
OcTreeKey coordToKey (double x, double y, double z, unsigned depth) const

Converts from a 3D coordinate into a 3D addressing key.
Converts from a 3D coordinate into a 3D addressing key at a given depth.
Converts from a single coordinate into a discrete key.
Converts from a single coordinate into a discrete key at a given depth.
Converts from a 3D coordinate into a 3D addressing key.
Converts from a 3D coordinate into a 3D addressing key at a given depth.
将三维坐标转换为三维寻址键。
将三维坐标转换为给定深度的三维寻址键。
将单个坐标转换为离散关键点。
将单个坐标转换为给定深度处的离散关键点。
将三维坐标转换为三维寻址键。
将三维坐标转换为给定深度的三维寻址键。

12 coordToKeyChecked

bool coordToKeyChecked (const point3d &coord, OcTreeKey &key) const
bool coordToKeyChecked (const point3d &coord, unsigned depth, OcTreeKey &key) const
bool coordToKeyChecked (double coordinate, key_type &key) const
bool coordToKeyChecked (double coordinate, unsigned depth, key_type &key) const
bool coordToKeyChecked (double x, double y, double z, OcTreeKey &key) const
bool coordToKeyChecked (double x, double y, double z, unsigned depth, OcTreeKey &key) const

Converts a 3D coordinate into a 3D OcTreeKey, with boundary checking.
Converts a 3D coordinate into a 3D OcTreeKey at a certain depth, with boundary checking.
Converts a single coordinate into a discrete addressing key, with boundary checking.
Converts a single coordinate into a discrete addressing key, with boundary checking.
Converts a 3D coordinate into a 3D OcTreeKey, with boundary checking.
Converts a 3D coordinate into a 3D OcTreeKey at a certain depth, with boundary checking.

通过边界检查将三维坐标转换为三维八叉键。
通过边界检查将三维坐标转换为特定深度的三维八叉键。
通过边界检查将单个坐标转换为离散寻址键。
通过边界检查将单个坐标转换为离散寻址键。
通过边界检查将三维坐标转换为三维八叉键。
通过边界检查将三维坐标转换为特定深度的三维八叉键。

13 OcTreeNode * createNodeChild (OcTreeNode *node, unsigned int childIdx)

Creates (allocates) the i-th child of the node.
创建（分配）节点的第i个子节点。

14 deleteNode

bool deleteNode (const OcTreeKey &key, unsigned int depth=0)
bool deleteNode (const point3d &value, unsigned int depth=0)
bool deleteNode (double x, double y, double z, unsigned int depth=0)
Delete a node (if exists) given an addressing key.Will always delete at the lowest level unless depth !=0, and expand pruned inner nodes as needed. Pruned nodes at level “depth” will directly be deleted as a whole.
Delete a node (if exists) given a 3d point.Will always delete at the lowest level unless depth !=0, and expand pruned inner nodes as needed. Pruned nodes at level “depth” will directly be deleted as a whole.
Delete a node (if exists) given a 3d point.Will always delete at the lowest level unless depth !=0, and expand pruned inner nodes as needed. Pruned nodes at level “depth” will directly be deleted as a whole.
删除给定寻址密钥的节点（如果存在）。将始终在最低级别删除，除非深度=0，并根据需要展开修剪的内部节点。“深度”级别的修剪节点将作为一个整体直接删除。
删除给定三维点的节点（如果存在）。将始终在最低级别删除，除非深度=0，并根据需要展开修剪的内部节点。“深度”级别的修剪节点将作为一个整体直接删除。
删除给定三维点的节点（如果存在）。将始终在最低级别删除，除非深度=0，并根据需要展开修剪的内部节点。“深度”级别的修剪节点将作为一个整体直接删除。
删除节点的第i个子节点。

void deleteNodeChild (OcTreeNode *node, unsigned int childIdx)
Deletes the i-th child of the node.
删除节点的第i个子节点。

15 void enableChangeDetection (bool enable)

track or ignore changes while inserting scans (default: ignore)
插入扫描时跟踪或忽略更改（默认值：忽略）

16 void getMetricMax (double &x, double &y, double &z) const

maximum value of the bounding box of all known space in x, y, z
x、y、z中所有已知空间的边界框的最大值

17 void getMetricMin (double &x, double &y, double &z) const

minimum value of the bounding box of all known space in x, y, z
x、y、z中所有已知空间的边界框的最小值

18 void getMetricSize (double &x, double &y, double &z)

Size of OcTree (all known space) in meters for x, y and z dimension.
x、y和z维八叉树（所有已知空间）的大小，单位为米。

19 OcTreeNode * getNodeChild (OcTreeNode *node, unsigned int childIdx)

返回 const ptr to child number childIdx of node
const ptr到节点的子编号childIdx

20 double getNodeSize (unsigned depth) const

21 bool getNormals (const point3d &point, std::vector< point3d > &normals, bool unknownStatus=true) const

Performs a step of the marching cubes surface reconstruction algorithm to retrieve the normal of the triangles that fall in the cube formed by the voxels located at the vertex of a given voxel.
执行marching cubes曲面重建算法的步骤，以检索由位于给定体素顶点的体素形成的立方体中三角形的法线。

22 size_t getNumLeafNodes () const

Traverses the tree to calculate the total number of leaf nodes.
遍历树以计算叶节点的总数

23 double getResolution ()

24 OcTreeNode * getRoot () const

25 unsigned int getTreeDepth () const

26 std::string getTreeType () const

returns actual class name as string for identification
将实际类名作为字符串返回以进行标识

27 void getUnknownLeafCenters (point3d_list &node_centers, point3d pmin, point3d pmax, unsigned int depth=0) constI

28 insertPointCloud

void insertPointCloud (const Pointcloud &scan, const octomap::point3d &sensor_origin, double maxrange=-1., bool lazy_eval=false, bool discretize=false)
void insertPointCloud (const Pointcloud &scan, const point3d &sensor_origin, const pose6d &frame_origin, double maxrange=-1., bool lazy_eval=false, bool discretize=false)
void insertPointCloud (const ScanNode &scan, double maxrange=-1., bool lazy_eval=false, bool discretize=false)
lazy_eval whether update of inner nodes is omitted after the update (default: false). This speeds up the insertion, but you need to call updateInnerOccupancy() when done. 更新后是否省略内部节点的更新（默认值：false）。这会加快插入速度，但完成后需要调用updateInnerOccupancy（）。
discretize whether the scan is discretized first into octree key cells (default: false). This reduces the number of raycasts using computeDiscreteUpdate(), resulting in a potential speedup.* 是否首先将扫描离散化为八叉树关键单元（默认值：false）。这将使用

computeDiscreteUpdate（）减少光线投射的数量，从而产生潜在的加速*

Special care is taken that each voxel in the map is updated only once, and occupied nodes have a preference over free ones. This avoids holes in the floor from mutual deletion and is more efficient than the plain ray insertion in insertPointCloudRays().
需要特别注意的是，贴图中的每个体素只更新一次，并且占用的节点优先于自由节点。这避免了地板上的孔被相互删除，并且比insertPointCloudRays（）中的普通光线插入更有效。

29 void insertPointCloudRays (const Pointcloud &scan, const point3d &sensor_origin, double maxrange=-1., bool lazy_eval=false)

This function simply inserts all rays of the point clouds as batch operation. Discretization effects can lead to the deletion of occupied space, it is usually recommended to use insertPointCloud() instead.此函数仅将点云的所有光线作为 批处理 操作插入。离散化效果可能导致删除占用的空间，通常建议使用insertPointCloud（）。

30 bool insertRay (const point3d &origin, const point3d &end, double maxrange=-1.0, bool lazy_eval=false)

Insert one ray between origin and end into the tree. 在树的原点和端点之间插入一条光线。

31 bool isNodeAtThreshold (const OcTreeNode &occupancyNode) const

queries whether a node is at the clamping threshold according to the tree's parameter 根据树的参数查询节点是否处于钳制阈值

32 bool isNodeCollapsible (const OcTreeNode *node) const

A node is collapsible if all children exist, don't have children of their own and have the same occupancy value. 如果所有子节点都存在，且没有自己的子节点且占用率值相同，则节点是可折叠的。

33 bool isNodeOccupied (const OcTreeNode &occupancyNode) const

queries whether a node is occupied according to the tree's parameter for "occupancy" 根据树的"占用"参数查询节点是否被占用

34 keyToCoord

point3d keyToCoord (const OcTreeKey &key) const
point3d keyToCoord (const OcTreeKey &key, unsigned depth) const
double keyToCoord (key_type key) const
double keyToCoord (key_type key, unsigned depth) const

converts from an addressing key at the lowest tree level into a coordinate corresponding to the key's center
converts from an addressing key at a given depth into a coordinate corresponding to the key's center
converts from a discrete key at the lowest tree level into a coordinate corresponding to the key's center
converts from a discrete key at a given depth into a coordinate corresponding to the key's center
将最低树级别的寻址键转换为对应于键中心的坐标
将给定深度的寻址键转换为对应于键中心的坐标
将最低树级别的离散关键点转换为对应于关键点中心的坐标
将给定深度处的离散关键点转换为对应于关键点中心的坐标

35 void prune ()

Lossless compression of the octree: A node will replace all of its eight children if they have identical values.
You usually don't have to call prune() after a regular occupancy update, updateNode() incrementally prunes all affected nodes.
八叉树的无损压缩：一个节点将替换其所有八个子节点，如果它们具有相同的值。您通常不必在常规占用率更新后调用prune（），updateNode（）会增量修剪所有受影响的节点。

36 bool pruneNode (OcTreeNode *node)

Prunes a node when it is collapsible. 修剪可折叠的节点。

37 readBinary

bool readBinary (const std::string &filename)
bool readBinary (std::istream &s)
std::istream & readBinaryData (std::istream &s)
std::istream & readBinaryNode (std::istream &s, OcTreeNode *node)
std::istream & readData (std::istream &s)

Reads OcTree from a binary file.
Reads an OcTree from an input stream.
（Existing nodes of the tree are deleted before the tree is read. ）
从二进制文件中读取八叉树。
从输入流读取八叉树。
在读取树之前，将删除树的现有节点。

Reads the actual data, implemented in OccupancyOcTreeBase::readBinaryData()
读取实际数据，在OccupncyOctreebase:：readBinaryData（）中实现

Read node from binary stream (max-likelihood value), recursively continue with all children.This will set the log_odds_occupancy value of all leaves to either free or occupied.
从二进制流读取节点（最大似然值），递归地继续所有子节点。这会将所有叶片的对数占空比值设置为空闲或占用。

Read all nodes from the input stream (without file header), for this the tree needs to be already created.For general file IO, you should probably use AbstractOcTree::read() instead.
从输入流中读取所有节点（不带文件头），为此，需要已经创建树。对于常规文件IO，您可能应该改用AbstractOcTree:：read（）。

38 void resetChangeDetection ()

Reset the set of changed keys. Call this after you obtained all changed nodes.
重置已更改的密钥集。在获得所有更改的节点后调用此函数。

39 search

OcTreeNode * search (const OcTreeKey &key, unsigned int depth=0) const
OcTreeNode * search (const point3d &value, unsigned int depth=0) const
OcTreeNode * search (double x, double y, double z, unsigned int depth=0) const

Search a node at specified depth given an addressing key (depth=0: search full tree depth) You need to check if the returned node is NULL, since it can be in unknown space. 给定一个寻址键（depth=0:Search full tree depth），在指定深度搜索节点。您需要检查返回的节点是否为NULL，因为它可能位于未知空间中。
Search node at specified depth given a 3d point (depth=0: search full tree depth) You need to check if the returned node is NULL, since it can be in unknown space. 在指定深度搜索节点给定一个3d点（深度=0：搜索完整树深度），您需要检查返回的节点是否为空，因为它可能位于未知空间中。
Search node at specified depth given a 3d point (depth=0: search full tree depth).You need to check if the returned node is NULL, since it can be in unknown space. 在给定3d点的指定深度处搜索节点（深度=0：搜索完整树深度）。您需要检查返回的节点是否为NULL，因为它可能位于未知空间中。

40 setNodeValue

OcTreeNode * setNodeValue (const OcTreeKey &key, float log_odds_value, bool lazy_eval=false)
OcTreeNode * setNodeValue (const point3d &value, float log_odds_value, bool lazy_eval=false)
OcTreeNode * setNodeValue (double x, double y, double z, float log_odds_value, bool lazy_eval=false)
Set log_odds value of voxel to log_odds_value.This only works if key is at the lowest octree level将体素的对数赔率值设置为对数赔率值。这仅在键处于最低八叉树级别时有效
Set log_odds value of voxel to log_odds_value.Looks up the OcTreeKey corresponding to the coordinate and then calls setNodeValue() with it.将体素的对数赔率值设置为对数赔率值。查找对应于坐标的OcTreeKey，然后使用它调用setNodeValue（）。

Set log_odds value of voxel to log_odds_value.Looks up the OcTreeKey corresponding to the coordinate and then calls setNodeValue() with it.将体素的对数赔率值设置为对数赔率值。查找对应于坐标的OcTreeKey，然后使用它调用setNodeValue（）。

41 void updateInnerOccupancy ()

Updates the occupancy of all inner nodes to reflect their children's occupancy.If you performed batch-updates with lazy evaluation enabled, you must call this before any queries to ensure correct multi-resolution behavior.
更新所有内部节点的占用率以反映其子节点的占用率。如果在启用延迟评估的情况下执行批处理更新，则必须在任何查询之前调用此函数，以确保正确的多分辨率行为。

42 updataNode

OcTreeNode * updateNode (const OcTreeKey &key, bool occupied, bool lazy_eval=false)
OcTreeNode * updateNode (const OcTreeKey &key, float log_odds_update, bool lazy_eval=false)
OcTreeNode * updateNode (const point3d &value, bool occupied, bool lazy_eval=false)
OcTreeNode * updateNode (const point3d &value, float log_odds_update, bool lazy_eval=false)
OcTreeNode * updateNode (double x, double y, double z, bool occupied, bool lazy_eval=false)
OcTreeNode * updateNode (double x, double y, double z, float log_odds_update, bool lazy_eval=false)
Integrate occupancy measurement. 整合占用率测量。
Manipulate log_odds value of a voxel by changing it by log_odds_update (relative).This only works if key is at the lowest octree level 通过log_Lobbits_update（相对）更改体素的log_Lobbits值，从而操纵体素的log_Lobbits值。这仅在键处于最低八叉树级别时有效
Integrate occupancy measurement.Looks up the OcTreeKey corresponding to the coordinate and then calls udpateNode() with it.整合入住率测量。查找对应于该坐标的OcTreeKey，然后使用它调用udpateNode（）。
Manipulate log_odds value of a voxel by changing it by log_odds_update (relative).Looks up the OcTreeKey corresponding to the coordinate and then calls updateNode() with it.通过log_Lobbits_update（相对）更改体素的log_Lobbits值，从而操纵体素的log_Lobbits值。查找对应于坐标的OcTreeKey，然后使用它调用updateNode（）。
Integrate occupancy measurement.Looks up the OcTreeKey corresponding to the coordinate and then calls udpateNode() with it.整合入住率测量。查找对应于该坐标的OcTreeKey，然后使用它调用udpateNode（）。
Manipulate log_odds value of a voxel by changing it by log_odds_update (relative).Looks up the OcTreeKey corresponding to the coordinate and then calls updateNode() with it.通过log_Lobbits_update（相对）更改体素的log_Lobbits值，从而操纵体素的log_Lobbits值。查找对应于坐标的OcTreeKey，然后使用它调用updateNode（）。

43 void updateNodeLogOdds (OcTreeNode *occupancyNode, const float &update) const

update logodds value of node by adding to the current value.
通过添加到当前值来更新节点的logodds值。

44 write

bool write (const std::string &filename) const
bool write (std::ostream &s) const
bool writeBinary (const std::string &filename)
bool writeBinary (std::ostream &s)
bool writeBinaryConst (const std::string &filename) const
bool writeBinaryConst (std::ostream &s) const
std::ostream & writeBinaryData (std::ostream &s) const
std::ostream & writeBinaryNode (std::ostream &s, const OcTreeNode *node) const
std::ostream & writeData (std::ostream &s) const

Write file header and complete tree to file (serialization) 将文件头写入并完成树到文件（序列化）
Write file header and complete tree to stream (serialization) 将文件头和完整的树写入流（序列化）

Writes OcTree to a binary file using writeBinary().The OcTree is first converted to the maximum likelihood estimate and pruned. 使用writeBinary（）将八叉树写入二进制文件。首先将八叉树转换为最大似然估计并进行修剪。
Writes compressed maximum likelihood OcTree to a binary stream.The OcTree is first converted to the maximum likelihood estimate and pruned for maximum compression. 将压缩的最大似然八叉树写入二进制流。八叉树首先转换为最大似然估计，并进行修剪以获得最大压缩。

Writes OcTree to a binary file using writeBinaryConst().The OcTree is not changed, in particular not pruned first. Files will be smaller when the tree is pruned first or by using writeBinary() instead. 使用writeBinaryConst（）将八叉树写入二进制文件。八叉树不会改变，特别是不会先修剪。首先修剪树或改用writeBinary（）修剪树时，文件将变小。
Writes the maximum likelihood OcTree to a binary stream (const variant).Files will be smaller when the tree is pruned first or by using writeBinary() instead. 将最大似然八叉树写入二进制流（常量变量）。首先修剪树或改用writeBinary（）修剪树时，文件将变小。

Writes the actual data, implemented in OccupancyOcTreeBase::writeBinaryData()写入实际数据，在OccupncyOctreebase::writeBinaryData（）中实现

Write node to binary stream (max-likelihood value), recursively continue with all children.This will discard the log_odds_occupancy value, writing all leaves as either free or occupied.将节点写入二进制流（最大似然值），递归地继续所有子节点。这将丢弃log_Lobbits_Occupation值，将所有叶子写入空闲或已占用状态。

Write complete state of tree to stream (without file header) unmodified. Pruning the tree first produces smaller files (lossless compression) 未修改将树的完整状态写入流（无文件头）。修剪树首先生成较小的文件（无损压缩）

【原文】http://octomap.github.io/octomap/doc/classoctomap_1_1OcTree.html