project_name

ver:1.0

制作者 Doxygen 1.9.1

Chapter 1

命名空间索引

1.1 命名空间列表

这里列出了所有文档化的命名空间定义,附带简要说明:

Chapter 2

继承关系索引

2.1 类继承关系

此继承关系列表按字典顺序粗略的排序:

Chapter 3

结构体索引

3.1 结构体

这里列出了所有结构体，并附带简要说明:

Chapter 4

命名空间文档

4.1 costmap_2d 命名空间参考

结构体

- struct MapLocation
- class Costmap2D

  A 2D costmap provides a mapping between points in the world and their associated "costs".
- class Costmap2DPublisher

  A tool to periodically publish visualization data from a Costmap2D
- class Costmap2DROS

  A ROS wrapper for a 2D Costmap. Handles subscribing to topics that provide observations about obstacles in either the form of PointCloud or LaserScan messages.
- class CostmapLayer
- class CellData

  Storage for cell information used during obstacle inflation
- class InflationLayer
- class Layer
- class LayeredCostmap

  Instantiates different layer plugins and aggregates them into one score
- class Observation

  Stores an observation in terms of a point cloud and the origin of the source
- class ObservationBuffer

  Takes in point clouds from sensors, transforms them to the desired frame, and stores them
- class ObstacleLayer
- class StaticLayer
- class VoxelLayer

函数

- std::vector< std::vector< float > > parseVVF (const std::string &input, std::string &error_return)

  Parse a vector of vectors of floats from a string.
- void calculateMinAndMaxDistances (const std::vector< geometry_msgs::Point > &footprint, double &min_dist, double &max_dist)

  Calculate the extreme distances for the footprint
- geometry_msgs::Point toPoint (geometry_msgs::Point32 pt)

Convert Point32 to Point

- geometry_msgs::Point32 toPoint32 (geometry_msgs::Point pt)

  Convert Point to Point32

- geometry_msgs::Polygon toPolygon (std::vector< geometry_msgs::Point > pts)

  Convert vector of Points to Polygon msg

- std::vector< geometry_msgs::Point > toPointVector (geometry_msgs::Polygon polygon)

  Convert Polygon msg to vector of Points.

- void transformFootprint (double x, double y, double theta, const std::vector< geometry_msgs::Point > &footprint_spec, std::vector< geometry_msgs::Point > &oriented_footprint)

  Given a pose and base footprint, build the oriented footprint of the robot (list of Points)

- void transformFootprint (double x, double y, double theta, const std::vector< geometry_msgs::Point > &footprint_spec, geometry_msgs::PolygonStamped &oriented_footprint)

  Given a pose and base footprint, build the oriented footprint of the robot (PolygonStamped)

- void padFootprint (std::vector< geometry_msgs::Point > &footprint, double padding)

  Adds the specified amount of padding to the footprint (in place)

- std::vector< geometry_msgs::Point > makeFootprintFromRadius (double radius)

  Create a circular footprint from a given radius

- bool makeFootprintFromString (const std::string &footprint_string, std::vector< geometry_msgs::Point > &footprint)

  Make the footprint from the given string.

- std::vector< geometry_msgs::Point > makeFootprintFromParams (ros::NodeHandle &nh)

  Read the ros-params "footprint" and/or "robot_radius" from the given NodeHandle using searchParam() to go up the tree.

- std::vector< geometry_msgs::Point > makeFootprintFromXMLRPC (XmlRpc::XmlRpcValue &footprint_↩ xmlrpc, const std::string &full_param_name)

  Create the footprint from the given XmlRpcValue.

- void writeFootprintToParam (ros::NodeHandle &nh, const std::vector< geometry_msgs::Point > &footprint)

  Write the current unpadded_footprint_ to the "footprint" parameter of the given NodeHandle so that dynamic_↩ reconfigure will see the new value.

4.1.1 详细描述

Provides a mapping for often used cost values

4.1.2 函数说明

4.1.2.1 calculateMinAndMaxDistances()

```
void costmap_2d::calculateMinAndMaxDistances (
            const std::vector< geometry_msgs::Point > & footprint,
            double & min_dist,
            double & max_dist )
```

Calculate the extreme distances for the footprint

参数

footprint	The footprint to examine
min_dist	Output parameter of the minimum distance
max_dist	Output parameter of the maximum distance

4.1.2.2 makeFootprintFromString()

```
bool costmap_2d::makeFootprintFromString (
            const std::string & footprint_string,
            std::vector< geometry_msgs::Point > & footprint )
```

Make the footprint from the given string.

Format should be bracketed array of arrays of floats, like so: [[1.0, 2.2], [3.3, 4.2], ...]

4.1.2.3 makeFootprintFromXMLRPC()

```
std::vector<geometry_msgs::Point> costmap_2d::makeFootprintFromXMLRPC (
            XmlRpc::XmlRpcValue & footprint_xmlrpc,
            const std::string & full_param_name )
```

Create the footprint from the given XmlRpcValue.

参数

footprint_xmlrpc	should be an array of arrays, where the top-level array should have 3 or more elements, and the sub-arrays should all have exactly 2 elements (x and y coordinates).
full_param_name	this is the full name of the rosparam from which the footprint_xmlrpc value came. It is used only for reporting errors.

4.1.2.4 parseVVF()

```
std::vector<std::vector<float> > costmap_2d::parseVVF (
            const std::string & input,
            std::string & error_return )
```

Parse a vector of vectors of floats from a string.

参数

error_return	If no error, error_return is set to ””. If error, error_return will describe the error. Syntax is [[1.0, 2.0], [3.3, 4.4, 5.5], ...]

On error, error_return is set and the return value could be anything, like part of a successful parse.

4.1.2.5 transformFootprint() [1/2]

```
void costmap_2d::transformFootprint (
            double x,
            double y,
            double theta,
            const std::vector< geometry_msgs::Point > & footprint_spec,
            geometry_msgs::PolygonStamped & oriented_footprint )
```

Given a pose and base footprint, build the oriented footprint of the robot (PolygonStamped)

参数

x	The x position of the robot
y	The y position of the robot
theta	The orientation of the robot
footprint_spec	Basic shape of the footprint
oriented_footprint	Will be filled with the points in the oriented footprint of the robot

4.1.2.6 transformFootprint() [2/2]

```
void costmap_2d::transformFootprint (
            double x,
            double y,
            double theta,
            const std::vector< geometry_msgs::Point > & footprint_spec,
            std::vector< geometry_msgs::Point > & oriented_footprint )
```

Given a pose and base footprint, build the oriented footprint of the robot (list of Points)

参数

x	The x position of the robot
y	The y position of the robot
theta	The orientation of the robot
footprint_spec	Basic shape of the footprint
oriented_footprint	Will be filled with the points in the oriented footprint of the robot

Chapter 5

结构体说明

5.1 _pf_sample_set_t结构体 参考

成员变量

- int **sample_count**
- pf_sample_t ∗ **samples**
- pf_kdtree_t ∗ **kdtree**
- int **cluster_count**
- int **cluster_max_count**
- pf_cluster_t ∗ **clusters**
- pf_vector_t **mean**
- pf_matrix_t **cov**
- int **converged**
- double **n_effective**

该结构体的文档由以下文件生成:

- amcl/include/amcl/pf/pf.h

5.2 _pf_t结构体 参考

成员变量

- int **min_samples**
- int **max_samples**
- double **pop_err**
- double **pop_z**
- int ∗ **limit_cache**
- int **current_set**
- pf_sample_set_t **sets** [2]
- double **w_slow**
- double **w_fast**
- double **alpha_slow**
- double **alpha_fast**

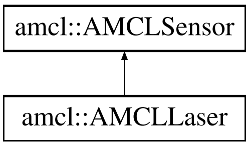
- pf_init_model_fn_t **random_pose_fn**
- void * **random_pose_data**
- double **dist_threshold**
- int **converged**
- int **selective_resampling**

该结构体的文档由以下文件生成:

- amcl/include/amcl/pf/pf.h

5.3 amcl::AMCLLaser类 参考

类 amcl::AMCLLaser 继承关系图:



Public 成员函数

- **AMCLLaser** (size_t max_beams, map_t *map)
- void **SetModelBeam** (double z_hit, double z_short, double z_max, double z_rand, double sigma_hit, double lambda_short, double chi_outlier)
- void **SetModelLikelihoodField** (double z_hit, double z_rand, double sigma_hit, double max_occ_dist)
- void **SetModelLikelihoodFieldProb** (double z_hit, double z_rand, double sigma_hit, double max_occ_↩dist, bool do_beamskip, double beam_skip_distance, double beam_skip_threshold, double beam_skip_error_↩threshold)
- virtual bool **UpdateSensor** (pf_t *pf, AMCLSensorData *data)
- void **SetLaserPose** (pf_vector_t &laser_pose)

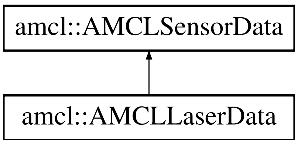
额外继承的成员函数

该类的文档由以下文件生成:

- amcl/include/amcl/sensors/amcl_laser.h

5.4 amcl::AMCLLaserData类 参考

类 amcl::AMCLLaserData 继承关系图:



成员变量

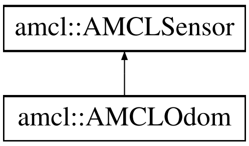
- int **range_count**
- double **range_max**
- double(∗ **ranges**)[2]

该类的文档由以下文件生成:

- amcl/include/amcl/sensors/amcl_laser.h

5.5 amcl::AMCLOdom类 参考

类 amcl::AMCLOdom 继承关系图:



Public 成员函数

- void **SetModelDiff** (double alpha1, double alpha2, double alpha3, double alpha4)
- void **SetModelOmni** (double alpha1, double alpha2, double alpha3, double alpha4, double alpha5)
- void **SetModel** (odom_model_t type, double alpha1, double alpha2, double alpha3, double alpha4, double alpha5=0)
- virtual bool **UpdateAction** (pf_t ∗pf, AMCLSensorData ∗data)

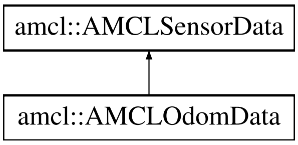
额外继承的成员函数

该类的文档由以下文件生成:

- amcl/include/amcl/sensors/amcl_odom.h

5.6 amcl::AMCLOdomData类 参考

类 amcl::AMCLOdomData 继承关系图:



成员变量

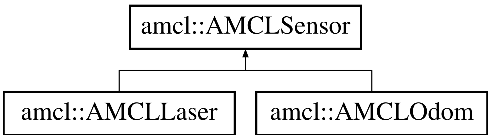
- pf_vector_t **pose**
- pf_vector_t **delta**

该类的文档由以下文件生成:

- amcl/include/amcl/sensors/amcl_odom.h

5.7 amcl::AMCLSensor类 参考

类 amcl::AMCLSensor 继承关系图:



Public 成员函数

- virtual bool **UpdateAction** (pf_t *pf, AMCLSensorData *data)
- virtual bool **InitSensor** (pf_t *pf, AMCLSensorData *data)
- virtual bool **UpdateSensor** (pf_t *pf, AMCLSensorData *data)

成员变量

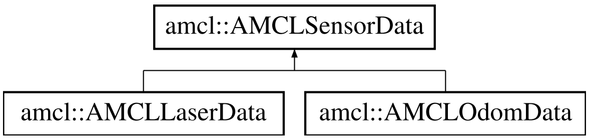
- bool **is_action**
- pf_vector_t **pose**

该类的文档由以下文件生成:

- amcl/include/amcl/sensors/amcl_sensor.h

5.8 amcl::AMCLSensorData类 参考

类 amcl::AMCLSensorData 继承关系图:



成员变量

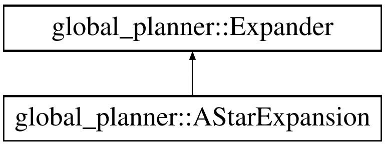
- AMCLSensor ∗ **sensor**
- double **time**

该类的文档由以下文件生成:

- amcl/include/amcl/sensors/amcl_sensor.h

5.9 global_planner::AStarExpansion类 参考

类 global_planner::AStarExpansion 继承关系图:



Public 成员函数

- **AStarExpansion** (PotentialCalculator ∗p_calc, int nx, int ny)
- bool **calculatePotentials** (unsigned char ∗costs, double start_x, double start_y, double end_x, double end_y, int cycles, float ∗potential)

额外继承的成员函数

该类的文档由以下文件生成:

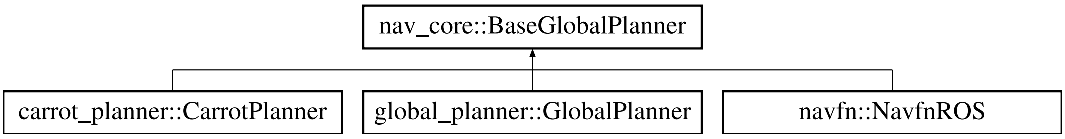
- global_planner/include/global_planner/astar.h

5.10 nav_core::BaseGlobalPlanner类 参考

Provides an interface for global planners used in navigation. All global planners written as plugins for the navigation stack must adhere to this interface.

```
#include <base_global_planner.h>
```

类 nav_core::BaseGlobalPlanner 继承关系图:


nav_core::BaseGlobalPlanner
carrot_planner::CarrotPlanner
global_planner::GlobalPlanner
navfn::NavfnROS

Public 成员函数

- virtual bool makePlan (const geometry_msgs::PoseStamped &start, const geometry_msgs::PoseStamped &goal, std::vector< geometry_msgs::PoseStamped > &plan)=0

  Given a goal pose in the world, compute a plan
- virtual bool makePlan (const geometry_msgs::PoseStamped &start, const geometry_msgs::PoseStamped &goal, std::vector< geometry_msgs::PoseStamped > &plan, double &cost)

  Given a goal pose in the world, compute a plan
- virtual void initialize (std::string name, costmap_2d::Costmap2DROS ∗costmap_ros)=0

  Initialization function for the BaseGlobalPlanner
- virtual ∼BaseGlobalPlanner ()

  Virtual destructor for the interface

5.10.1 详细描述

Provides an interface for global planners used in navigation. All global planners written as plugins for the navigation stack must adhere to this interface.

5.10.2 成员函数说明

5.10.2.1 initialize()

```
virtual void nav_core::BaseGlobalPlanner::initialize (
            std::string name,
            costmap_2d::Costmap2DROS ∗ costmap_ros )  [pure virtual]
```

Initialization function for the BaseGlobalPlanner

参数

name	The name of this planner
costmap_ros	A pointer to the ROS wrapper of the costmap to use for planning

在 navfn::NavfnROS, global_planner::GlobalPlanner , 以及 carrot_planner::CarrotPlanner 内被实现.

5.10.2.2 makePlan() [1/2]

```
virtual bool nav_core::BaseGlobalPlanner::makePlan (
            const geometry_msgs::PoseStamped & start,
            const geometry_msgs::PoseStamped & goal,
            std::vector< geometry_msgs::PoseStamped > & plan )  [pure virtual]
```

Given a goal pose in the world, compute a plan

参数

start	The start pose
goal	The goal pose
plan	The plan... filled by the planner

返回

True if a valid plan was found, false otherwise

在 navfn::NavfnROS, global_planner::GlobalPlanner , 以及 carrot_planner::CarrotPlanner 内被实现.

5.10.2.3 makePlan() [2/2]

```
virtual bool nav_core::BaseGlobalPlanner::makePlan (
             const geometry_msgs::PoseStamped & start,
             const geometry_msgs::PoseStamped & goal,
             std::vector< geometry_msgs::PoseStamped > & plan,
             double & cost )  [inline], [virtual]
```

Given a goal pose in the world, compute a plan

参数

start	The start pose
goal	The goal pose
plan	The plan... filled by the planner
cost	The plans calculated cost

返回

True if a valid plan was found, false otherwise

该类的文档由以下文件生成:

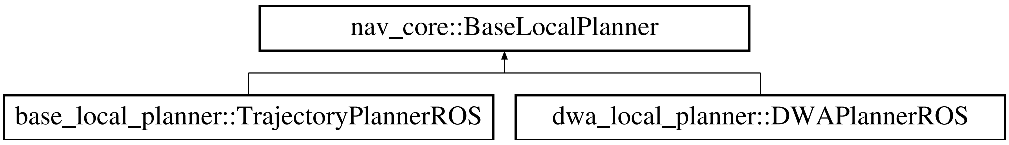
- nav_core/include/nav_core/base_global_planner.h

5.11 nav_core::BaseLocalPlanner类 参考

Provides an interface for local planners used in navigation. All local planners written as plugins for the navigation stack must adhere to this interface.

```
#include <base_local_planner.h>
```

类 nav_core::BaseLocalPlanner 继承关系图:


nav_core::BaseLocalPlanner
base_local_planner::TrajectoryPlannerROS
dwa_local_planner::DWAPlannerROS

Public 成员函数

- virtual bool computeVelocityCommands (geometry_msgs::Twist &cmd_vel)=0

  Given the current position, orientation, and velocity of the robot, compute velocity commands to send to the base
- virtual bool isGoalReached ()=0

  Check if the goal pose has been achieved by the local planner
- virtual bool setPlan (const std::vector< geometry_msgs::PoseStamped > &plan)=0

  Set the plan that the local planner is following
- virtual void initialize (std::string name, tf2_ros::Buffer ∗tf, costmap_2d::Costmap2DROS ∗costmap_ros)=0

  Constructs the local planner
- virtual ∼BaseLocalPlanner ()

  Virtual destructor for the interface

5.11.1 详细描述

Provides an interface for local planners used in navigation. All local planners written as plugins for the navigation stack must adhere to this interface.

5.11.2 成员函数说明

5.11.2.1 computeVelocityCommands()

```
virtual bool nav_core::BaseLocalPlanner::computeVelocityCommands (
                geometry_msgs::Twist & cmd_vel ) [pure virtual]
```

Given the current position, orientation, and velocity of the robot, compute velocity commands to send to the base

参数

cmd_vel	Will be filled with the velocity command to be passed to the robot base

返回

True if a valid velocity command was found, false otherwise

在 dwa_local_planner::DWAPlannerROS , 以及 base_local_planner::TrajectoryPlannerROS 内被实现.

5.11.2.2 initialize()

```
virtual void nav_core::BaseLocalPlanner::initialize (
             std::string name,
             tf2_ros::Buffer * tf,
             costmap_2d::Costmap2DROS * costmap_ros )  [pure virtual]
```

Constructs the local planner

参数

name	The name to give this instance of the local planner
tf	A pointer to a transform listener
costmap_ros	The cost map to use for assigning costs to local plans

在 dwa_local_planner::DWAPlannerROS , 以及 base_local_planner::TrajectoryPlannerROS 内被实现.

5.11.2.3 isGoalReached()

```
virtual bool nav_core::BaseLocalPlanner::isGoalReached ( )  [pure virtual]
```

Check if the goal pose has been achieved by the local planner

返回

True if achieved, false otherwise

在 dwa_local_planner::DWAPlannerROS , 以及 base_local_planner::TrajectoryPlannerROS 内被实现.

5.11.2.4 setPlan()

```
virtual bool nav_core::BaseLocalPlanner::setPlan (
             const std::vector< geometry_msgs::PoseStamped > & plan )  [pure virtual]
```

Set the plan that the local planner is following

参数

plan	The plan to pass to the local planner

返回

True if the plan was updated successfully, false otherwise 全局路径规划出的结果利用该函数存取到局部规划器中

在 dwa_local_planner::DWAPlannerROS , 以及 base_local_planner::TrajectoryPlannerROS 内被实现.

该类的文档由以下文件生成:

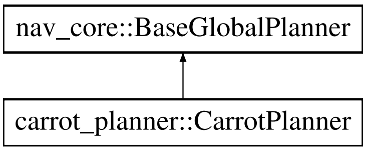
- nav_core/include/nav_core/base_local_planner.h

5.12 carrot_planner::CarrotPlanner类 参考

Provides a simple global planner that will compute a valid goal point for the local planner by walking back along the vector between the robot and the user-specified goal point until a valid cost is found.

```
#include <carrot_planner.h>
```

类 carrot_planner::CarrotPlanner 继承关系图:



Public 成员函数

- CarrotPlanner ()

  Constructor for the CarrotPlanner
- CarrotPlanner (std::string name, costmap_2d::Costmap2DROS ∗costmap_ros)

  Constructor for the CarrotPlanner
- void initialize (std::string name, costmap_2d::Costmap2DROS ∗costmap_ros)

  Initialization function for the CarrotPlanner
- bool makePlan (const geometry_msgs::PoseStamped &start, const geometry_msgs::PoseStamped &goal, std::vector< geometry_msgs::PoseStamped > &plan)

  Given a goal pose in the world, compute a plan

5.12.1 详细描述

Provides a simple global planner that will compute a valid goal point for the local planner by walking back along the vector between the robot and the user-specified goal point until a valid cost is found.

5.12.2 构造及析构函数说明

5.12.2.1 CarrotPlanner()

```
carrot_planner::CarrotPlanner::CarrotPlanner (
            std::string name,
            costmap_2d::Costmap2DROS * costmap_ros )
```

Constructor for the CarrotPlanner

参数

name	The name of this planner
costmap_ros	A pointer to the ROS wrapper of the costmap to use for planning

5.12.3 成员函数说明

5.12.3.1 initialize()

```
void carrot_planner::CarrotPlanner::initialize (
            std::string name,
            costmap_2d::Costmap2DROS * costmap_ros )  [virtual]
```

Initialization function for the CarrotPlanner

参数

name	The name of this planner
costmap_ros	A pointer to the ROS wrapper of the costmap to use for planning

实现了 nav_core::BaseGlobalPlanner.

5.12.3.2 makePlan()

```
bool carrot_planner::CarrotPlanner::makePlan (
            const geometry_msgs::PoseStamped & start,
            const geometry_msgs::PoseStamped & goal,
            std::vector< geometry_msgs::PoseStamped > & plan )  [virtual]
```

Given a goal pose in the world, compute a plan

参数

start	The start pose
goal	The goal pose
plan	The plan... filled by the planner

返回

True if a valid plan was found, false otherwise

实现了 nav_core::BaseGlobalPlanner.

该类的文档由以下文件生成:

- carrot_planner/include/carrot_planner/carrot_planner.h

5.13 costmap_2d::CellData类 参考

Storage for cell information used during obstacle inflation

```
#include <inflation_layer.h>
```

Public 成员函数

- CellData (double i, unsigned int x, unsigned int y, unsigned int sx, unsigned int sy)

  Constructor for a CellData objects

成员变量

- unsigned int **index_**
- unsigned int **x_**
- unsigned int **y_**
- unsigned int **src_x_**
- unsigned int **src_y_**

5.13.1 详细描述

Storage for cell information used during obstacle inflation

5.13.2 构造及析构函数说明

5.13.2.1 CellData()

```
costmap_2d::CellData::CellData (
            double i,
            unsigned int x,
            unsigned int y,
            unsigned int sx,
            unsigned int sy )  [inline]
```

Constructor for a CellData objects

参数

i	The index of the cell in the cost map
x	The x coordinate of the cell in the cost map
y	The y coordinate of the cell in the cost map
sx	The x coordinate of the closest obstacle cell in the costmap
sy	The y coordinate of the closest obstacle cell in the costmap

返回

该类的文档由以下文件生成:

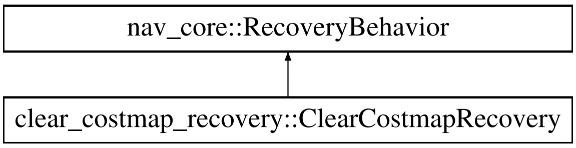
- costmap_2d/include/costmap_2d/inflation_layer.h

5.14 clear_costmap_recovery::ClearCostmapRecovery类 参考

A recovery behavior that reverts the navigation stack's costmaps to the static map outside of a user-specified region.

```
#include <clear_costmap_recovery.h>
```

类 clear_costmap_recovery::ClearCostmapRecovery 继承关系图:


nav_core::RecoveryBehavior
clear_costmap_recovery::ClearCostmapRecovery

Public 成员函数

- ClearCostmapRecovery ()

  Constructor, make sure to call initialize in addition to actually initialize the object
- void initialize (std::string name, tf2_ros::Buffer *tf, costmap_2d::Costmap2DROS *global_costmap, costmap_2d::Costmap2DROS *local_costmap)

  Initialization function for the ClearCostmapRecovery recovery behavior
- void runBehavior ()

  Run the ClearCostmapRecovery recovery behavior. Reverts the costmap to the static map outside of a user-specified window and clears unknown space around the robot.

5.14.1 详细描述

A recovery behavior that reverts the navigation stack's costmaps to the static map outside of a user-specified region.

5.14.2 构造及析构函数说明

5.14.2.1 ClearCostmapRecovery()

```
clear_costmap_recovery::ClearCostmapRecovery::ClearCostmapRecovery ( )
```

Constructor, make sure to call initialize in addition to actually initialize the object

参数

5.14.3 成员函数说明

5.14.3.1 initialize()

```
void clear_costmap_recovery::ClearCostmapRecovery::initialize (
            std::string name,
            tf2_ros::Buffer * tf,
            costmap_2d::Costmap2DROS * global_costmap,
            costmap_2d::Costmap2DROS * local_costmap )  [virtual]
```

Initialization function for the ClearCostmapRecovery recovery behavior

参数

tf	A pointer to a transform listener
global_costmap	A pointer to the global_costmap used by the navigation stack
local_costmap	A pointer to the local_costmap used by the navigation stack

实现了 nav_core::RecoveryBehavior.

该类的文档由以下文件生成:

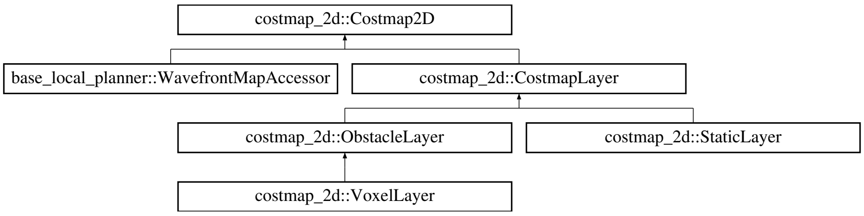
- clear_costmap_recovery/include/clear_costmap_recovery/clear_costmap_recovery.h

5.15 costmap_2d::Costmap2D类 参考

A 2D costmap provides a mapping between points in the world and their associated ”costs”.

`#include <costmap_2d.h>`

类 costmap_2d::Costmap2D 继承关系图:



结构体

- class MarkCell
- class PolygonOutlineCells

Public 类型

- typedef boost::recursive_mutex **mutex_t**

Public 成员函数

- Costmap2D (unsigned int cells_size_x, unsigned int cells_size_y, double resolution, double origin_x, double origin_y, unsigned char default_value=0)

    Constructor for a costmap
- Costmap2D (const Costmap2D &map)

    Copy constructor for a costmap, creates a copy efficiently
- Costmap2D & operator= (const Costmap2D &map)

    Overloaded assignment operator
- bool copyCostmapWindow (const Costmap2D &map, double win_origin_x, double win_origin_y, double win_←size_x, double win_size_y)

    Turn this costmap into a copy of a window of a costmap passed in
- Costmap2D ()

    Default constructor
- virtual ∼Costmap2D ()

    Destructor
- unsigned char getCost (unsigned int mx, unsigned int my) const

    Get the cost of a cell in the costmap
- void setCost (unsigned int mx, unsigned int my, unsigned char cost)

    Set the cost of a cell in the costmap
- void mapToWorld (unsigned int mx, unsigned int my, double &wx, double &wy) const

    Convert from map coordinates to world coordinates
- bool worldToMap (double wx, double wy, unsigned int &mx, unsigned int &my) const

    Convert from world coordinates to map coordinates
- void worldToMapNoBounds (double wx, double wy, int &mx, int &my) const

    Convert from world coordinates to map coordinates without checking for legal bounds
- void worldToMapEnforceBounds (double wx, double wy, int &mx, int &my) const

    Convert from world coordinates to map coordinates, constraining results to legal bounds.
- unsigned int getIndex (unsigned int mx, unsigned int my) const

    Given two map coordinates... compute the associated index
- void indexToCells (unsigned int index, unsigned int &mx, unsigned int &my) const

    Given an index... compute the associated map coordinates
- unsigned char ∗ getCharMap () const

    Will return a pointer to the underlying unsigned char array used as the costmap
- unsigned int getSizeInCellsX () const

    Accessor for the x size of the costmap in cells
- unsigned int getSizeInCellsY () const

    Accessor for the y size of the costmap in cells
- double getSizeInMetersX () const

    Accessor for the x size of the costmap in meters
- double getSizeInMetersY () const

Accessor for the y size of the costmap in meters

- double getOriginX () const

  Accessor for the x origin of the costmap
- double getOriginY () const

  Accessor for the y origin of the costmap
- double getResolution () const

  Accessor for the resolution of the costmap
- void **setDefaultValue** (unsigned char c)
- unsigned char **getDefaultValue** ()
- bool setConvexPolygonCost (const std::vector< geometry_msgs::Point > &polygon, unsigned char cost_value)

  Sets the cost of a convex polygon to a desired value
- void polygonOutlineCells (const std::vector< MapLocation > &polygon, std::vector< MapLocation > &polygon_cells)

  Get the map cells that make up the outline of a polygon
- void convexFillCells (const std::vector< MapLocation > &polygon, std::vector< MapLocation > &polygon_cells)

  Get the map cells that fill a convex polygon
- virtual void updateOrigin (double new_origin_x, double new_origin_y)

  Move the origin of the costmap to a new location.... keeping data when it can
- bool saveMap (std::string file_name)

  Save the costmap out to a pgm file
- void **resizeMap** (unsigned int size_x, unsigned int size_y, double resolution, double origin_x, double origin_y)
- void **resetMap** (unsigned int x0, unsigned int y0, unsigned int xn, unsigned int yn)
- unsigned int cellDistance (double world_dist)

  Given distance in the world... convert it to cells
- mutex_t ∗ **getMutex** ()

Protected 成员函数

- template<typename data_type >
  void copyMapRegion (data_type ∗source_map, unsigned int sm_lower_left_x, unsigned int sm_lower_left_y, unsigned int sm_size_x, data_type ∗dest_map, unsigned int dm_lower_left_x, unsigned int dm_lower_left_y, unsigned int dm_size_x, unsigned int region_size_x, unsigned int region_size_y)

  Copy a region of a source map into a destination map
- virtual void deleteMaps ()

  Deletes the costmap, static_map, and markers data structures
- virtual void resetMaps ()

  Resets the costmap and static_map to be unknown space
- virtual void initMaps (unsigned int size_x, unsigned int size_y)

  Initializes the costmap, static_map, and markers data structures
- template<class ActionType >
  void raytraceLine (ActionType at, unsigned int x0, unsigned int y0, unsigned int x1, unsigned int y1, unsigned int max_length=UINT_MAX)

  Raytrace a line and apply some action at each step

Protected 属性

- unsigned int size_x_

  Constructor for a costmap
- unsigned int **size_y_**
- double **resolution_**
- double **origin_x_**
- double **origin_y_**
- unsigned char ∗ **costmap_**
- unsigned char **default_value_**

友元

- class **CostmapTester**

5.15.1 详细描述

A 2D costmap provides a mapping between points in the world and their associated ”costs”.

5.15.2 构造及析构函数说明

5.15.2.1 Costmap2D() [1/2]

```
costmap_2d::Costmap2D::Costmap2D (
            unsigned int cells_size_x,
            unsigned int cells_size_y,
            double resolution,
            double origin_x,
            double origin_y,
            unsigned char default_value = 0 )
```

Constructor for a costmap

参数

cells_size_x	The x size of the map in cells
cells_size_y	The y size of the map in cells
resolution	The resolution of the map in meters/cell
origin_x	The x origin of the map
origin_y	The y origin of the map
default_value	Default Value

5.15.2.2 Costmap2D() [2/2]

```
costmap_2d::Costmap2D::Costmap2D (
            const Costmap2D & map )
```

Copy constructor for a costmap, creates a copy efficiently

参数

map	The costmap to copy

5.15.3 成员函数说明

5.15.3.1 cellDistance()

```
unsigned int costmap_2d::Costmap2D::cellDistance (
            double world_dist )
```

Given distance in the world... convert it to cells

参数

world_dist	The world distance

返回

The equivalent cell distance

5.15.3.2 convexFillCells()

```
void costmap_2d::Costmap2D::convexFillCells (
            const std::vector< MapLocation > & polygon,
            std::vector< MapLocation > & polygon_cells )
```

Get the map cells that fill a convex polygon

参数

polygon	The polygon in map coordinates to rasterize
polygon_cells	Will be set to the cells that fill the polygon

5.15.3.3 copyCostmapWindow()

```
bool costmap_2d::Costmap2D::copyCostmapWindow (
            const Costmap2D & map,
            double win_origin_x,
            double win_origin_y,
            double win_size_x,
            double win_size_y )
```

Turn this costmap into a copy of a window of a costmap passed in

参数

map	The costmap to copy
win_↩ origin_x	The x origin (lower left corner) for the window to copy, in meters
win_↩ origin_y	The y origin (lower left corner) for the window to copy, in meters
win_size_x	The x size of the window, in meters
win_size_y	The y size of the window, in meters

5.15.3.4 copyMapRegion()

```
template<typename data_type >
void costmap_2d::Costmap2D::copyMapRegion (
             data_type * source_map,
             unsigned int sm_lower_left_x,
             unsigned int sm_lower_left_y,
             unsigned int sm_size_x,
             data_type * dest_map,
             unsigned int dm_lower_left_x,
             unsigned int dm_lower_left_y,
             unsigned int dm_size_x,
             unsigned int region_size_x,
             unsigned int region_size_y )  [inline], [protected]
```

Copy a region of a source map into a destination map

参数

source_map	The source map
sm_lower_↩ left_x	The lower left x point of the source map to start the copy
sm_lower_↩ left_y	The lower left y point of the source map to start the copy
sm_size_x	The x size of the source map
dest_map	The destination map
dm_lower_↩ left_x	The lower left x point of the destination map to start the copy
dm_lower_↩ left_y	The lower left y point of the destination map to start the copy
dm_size_x	The x size of the destination map
region_size_x	The x size of the region to copy
region_size_y	The y size of the region to copy

5.15.3.5 getCharMap()

```
unsigned char* costmap_2d::Costmap2D::getCharMap ( ) const
```

Will return a pointer to the underlying unsigned char array used as the costmap

返回

A pointer to the underlying unsigned char array storing cost values

5.15.3.6 getCost()

```
unsigned char costmap_2d::Costmap2D::getCost (
            unsigned int mx,
            unsigned int my ) const
```

Get the cost of a cell in the costmap

参数

mx	The x coordinate of the cell
my	The y coordinate of the cell

返回

The cost of the cell

5.15.3.7 getIndex()

```
unsigned int costmap_2d::Costmap2D::getIndex (
            unsigned int mx,
            unsigned int my ) const  [inline]
```

Given two map coordinates... compute the associated index

参数

mx	The x coordinate
my	The y coordinate

返回

The associated index

5.15.3.8 getOriginX()

```
double costmap_2d::Costmap2D::getOriginX ( ) const
```

Accessor for the x origin of the costmap

返回

The x origin of the costmap

5.15.3.9 getOriginY()

```
double costmap_2d::Costmap2D::getOriginY ( ) const
```

Accessor for the y origin of the costmap

返回

The y origin of the costmap

5.15.3.10 getResolution()

```
double costmap_2d::Costmap2D::getResolution ( ) const
```

Accessor for the resolution of the costmap

返回

The resolution of the costmap

5.15.3.11 getSizeInCellsX()

```
unsigned int costmap_2d::Costmap2D::getSizeInCellsX ( ) const
```

Accessor for the x size of the costmap in cells

返回

The x size of the costmap

5.15.3.12 getSizeInCellsY()

```
unsigned int costmap_2d::Costmap2D::getSizeInCellsY ( ) const
```

Accessor for the y size of the costmap in cells

返回

The y size of the costmap

5.15.3.13 getSizeInMetersX()

```
double costmap_2d::Costmap2D::getSizeInMetersX ( ) const
```

Accessor for the x size of the costmap in meters

返回

The x size of the costmap (returns the centerpoint of the last legal cell in the map)

5.15.3.14 getSizeInMetersY()

```
double costmap_2d::Costmap2D::getSizeInMetersY ( ) const
```

Accessor for the y size of the costmap in meters

返回

The y size of the costmap (returns the centerpoint of the last legal cell in the map)

5.15.3.15 indexToCells()

```
void costmap_2d::Costmap2D::indexToCells (
             unsigned int index,
             unsigned int & mx,
             unsigned int & my ) const  [inline]
```

Given an index... compute the associated map coordinates

参数

index	The index
mx	Will be set to the x coordinate
my	Will be set to the y coordinate

5.15.3.16 initMaps()

```
virtual void costmap_2d::Costmap2D::initMaps (
             unsigned int size_x,
             unsigned int size_y )  [protected], [virtual]
```

Initializes the costmap, static_map, and markers data structures

参数

size←↩ _x	The x size to use for map initialization
size←↩ _y	The y size to use for map initialization

5.15.3.17 mapToWorld()

```
void costmap_2d::Costmap2D::mapToWorld (
             unsigned int mx,
             unsigned int my,
             double & wx,
             double & wy ) const
```

Convert from map coordinates to world coordinates

参数

mx	The x map coordinate
my	The y map coordinate
wx	Will be set to the associated world x coordinate
wy	Will be set to the associated world y coordinate

5.15.3.18 operator=()

```
Costmap2D& costmap_2d::Costmap2D::operator= (
             const Costmap2D & map )
```

Overloaded assignment operator

参数

map	The costmap to copy

返回

A reference to the map after the copy has finished

5.15.3.19 polygonOutlineCells()

```
void costmap_2d::Costmap2D::polygonOutlineCells (
            const std::vector< MapLocation > & polygon,
            std::vector< MapLocation > & polygon_cells )
```

Get the map cells that make up the outline of a polygon

参数

polygon	The polygon in map coordinates to rasterize
polygon_cells	Will be set to the cells contained in the outline of the polygon

5.15.3.20 raytraceLine()

```
template<class ActionType >
void costmap_2d::Costmap2D::raytraceLine (
            ActionType at,
            unsigned int x0,
            unsigned int y0,
            unsigned int x1,
            unsigned int y1,
            unsigned int max_length = UINT_MAX )  [inline], [protected]
```

Raytrace a line and apply some action at each step

参数

at	The action to take... a functor
x0	The starting x coordinate
y0	The starting y coordinate
x1	The ending x coordinate
y1	The ending y coordinate
max_length	The maximum desired length of the segment... allows you to not go all the way to the endpoint

5.15.3.21 saveMap()

```
bool costmap_2d::Costmap2D::saveMap (
            std::string file_name )
```

Save the costmap out to a pgm file

参数

file_name	The name of the file to save

5.15.3.22 setConvexPolygonCost()

```
bool costmap_2d::Costmap2D::setConvexPolygonCost (
            const std::vector< geometry_msgs::Point > & polygon,
            unsigned char cost_value )
```

Sets the cost of a convex polygon to a desired value

参数

polygon	The polygon to perform the operation on
cost_value	The value to set costs to

返回

True if the polygon was filled... false if it could not be filled

5.15.3.23 setCost()

```
void costmap_2d::Costmap2D::setCost (
            unsigned int mx,
            unsigned int my,
            unsigned char cost )
```

Set the cost of a cell in the costmap

参数

mx	The x coordinate of the cell
my	The y coordinate of the cell
cost	The cost to set the cell to

5.15.3.24 updateOrigin()

```
virtual void costmap_2d::Costmap2D::updateOrigin (
            double new_origin_x,
            double new_origin_y )  [virtual]
```

Move the origin of the costmap to a new location.... keeping data when it can

参数

new_↩ origin_x	The x coordinate of the new origin
new_↩ origin_y	The y coordinate of the new origin

被 costmap_2d::VoxelLayer 重载.

5.15.3.25 worldToMap()

```
bool costmap_2d::Costmap2D::worldToMap (
            double wx,
            double wy,
            unsigned int & mx,
            unsigned int & my ) const
```

Convert from world coordinates to map coordinates

参数

wx	The x world coordinate
wy	The y world coordinate
mx	Will be set to the associated map x coordinate
my	Will be set to the associated map y coordinate

返回

True if the conversion was successful (legal bounds) false otherwise

5.15.3.26 worldToMapEnforceBounds()

```
void costmap_2d::Costmap2D::worldToMapEnforceBounds (
            double wx,
            double wy,
            int & mx,
            int & my ) const
```

Convert from world coordinates to map coordinates, constraining results to legal bounds.

参数

wx	The x world coordinate
wy	The y world coordinate
mx	Will be set to the associated map x coordinate
my	Will be set to the associated map y coordinate

注解

The returned map coordinates are guaranteed to lie within the map.

5.15.3.27 worldToMapNoBounds()

```
void costmap_2d::Costmap2D::worldToMapNoBounds (
            double wx,
            double wy,
            int & mx,
            int & my ) const
```

Convert from world coordinates to map coordinates without checking for legal bounds

参数

wx	The x world coordinate
wy	The y world coordinate
mx	Will be set to the associated map x coordinate
my	Will be set to the associated map y coordinate

注解

The returned map coordinates **are not guaranteed to lie within the map.**

5.15.4 结构体成员变量说明

5.15.4.1 size_x_

```
unsigned int costmap_2d::Costmap2D::size_x_  [protected]
```

Constructor for a costmap

参数

cells_size_x	The x size of the map in cells

参数

cells_size_y	The y size of the map in cells
resolution	The resolution of the map in meters/cell
origin_x	The x origin of the map
origin_y	The y origin of the map
default_value	Default Value

该类的文档由以下文件生成:

- costmap_2d/include/costmap_2d/costmap_2d.h

5.16 costmap_2d::Costmap2DPublisher类 参考

A tool to periodically publish visualization data from a Costmap2D

```
#include <costmap_2d_publisher.h>
```

Public 成员函数

- Costmap2DPublisher (ros::NodeHandle *ros_node, Costmap2D *costmap, std::string global_frame, std::string topic_name, bool always_send_full_costmap=false)

  Constructor for the Costmap2DPublisher
- ∼Costmap2DPublisher ()

  Destructor
- void updateBounds (unsigned int x0, unsigned int xn, unsigned int y0, unsigned int yn)

  Include the given bounds in the changed-rectangle.
- void publishCostmap ()

  Publishes the visualization data over ROS
- bool active ()

  Check if the publisher is active

5.16.1 详细描述

A tool to periodically publish visualization data from a Costmap2D

5.16.2 成员函数说明

5.16.2.1 active()

```
bool costmap_2d::Costmap2DPublisher::active ( )  [inline]
```

Check if the publisher is active

返回

True if the frequency for the publisher is non-zero, false otherwise

该类的文档由以下文件生成:

- costmap_2d/include/costmap_2d/costmap_2d_publisher.h

5.17 costmap_2d::Costmap2DROS类 参考

A ROS wrapper for a 2D Costmap. Handles subscribing to topics that provide observations about obstacles in either the form of PointCloud or LaserScan messages.

```
#include <costmap_2d_ros.h>
```

Public 成员函数

- Costmap2DROS (const std::string &name, tf2_ros::Buffer &tf)

  Constructor for the wrapper
- void start ()

  Subscribes to sensor topics if necessary and starts costmap updates, can be called to restart the costmap after calls to either stop() or pause()
- void stop ()

  Stops costmap updates and unsubscribes from sensor topics
- void pause ()

  Stops the costmap from updating, but sensor data still comes in over the wire
- void resume ()

  Resumes costmap updates
- void **updateMap** ()
- void resetLayers ()

  Reset each individual layer
- bool isCurrent ()

  Same as getLayeredCostmap()->isCurrent().
- bool getRobotPose (geometry_msgs::PoseStamped &global_pose) const

  Get the pose of the robot in the global frame of the costmap
- std::string getName () const

  Returns costmap name
- double getTransformTolerance () const

  Returns the delay in transform (tf) data that is tolerable in seconds
- Costmap2D ∗ getCostmap ()

  Return a pointer to the "master" costmap which receives updates from all the layers.
- std::string getGlobalFrameID ()

  Returns the global frame of the costmap

- std::string getBaseFrameID ()

  Returns the local frame of the costmap
- LayeredCostmap ∗ **getLayeredCostmap** ()
- geometry_msgs::Polygon getRobotFootprintPolygon ()

  Returns the current padded footprint as a geometry_msgs::Polygon.
- std::vector< geometry_msgs::Point > getRobotFootprint ()

  Return the current footprint of the robot as a vector of points.
- std::vector< geometry_msgs::Point > getUnpaddedRobotFootprint ()

  Return the current unpadded footprint of the robot as a vector of points.
- void getOrientedFootprint (std::vector< geometry_msgs::Point > &oriented_footprint) const

  Build the oriented footprint of the robot at the robot's current pose
- void setUnpaddedRobotFootprint (const std::vector< geometry_msgs::Point > &points)

  Set the footprint of the robot to be the given set of points, padded by footprint_padding.
- void setUnpaddedRobotFootprintPolygon (const geometry_msgs::Polygon &footprint)

  Set the footprint of the robot to be the given polygon, padded by footprint_padding.

Protected 属性

- LayeredCostmap ∗ **layered_costmap_**
- std::string **name_**
- tf2_ros::Buffer & tf_

  Used for transforming point clouds
- std::string global_frame_

  The global frame for the costmap
- std::string robot_base_frame_

  The frame_id of the robot base
- double transform_tolerance_

  timeout before transform errors

5.17.1 详细描述

A ROS wrapper for a 2D Costmap. Handles subscribing to topics that provide observations about obstacles in either the form of PointCloud or LaserScan messages.

5.17.2 构造及析构函数说明

5.17.2.1 Costmap2DROS()

```
costmap_2d::Costmap2DROS::Costmap2DROS (
            const std::string & name,
            tf2_ros::Buffer & tf )
```

Constructor for the wrapper

参数

name	The name for this costmap
tf	A reference to a TransformListener

5.17.3 成员函数说明

5.17.3.1 getBaseFrameID()

```
std::string costmap_2d::Costmap2DROS::getBaseFrameID ( )  [inline]
```

Returns the local frame of the costmap

返回

The local frame of the costmap

5.17.3.2 getCostmap()

```
Costmap2D* costmap_2d::Costmap2DROS::getCostmap ( )  [inline]
```

Return a pointer to the "master" costmap which receives updates from all the layers.

Same as calling getLayeredCostmap()->getCostmap().

5.17.3.3 getGlobalFrameID()

```
std::string costmap_2d::Costmap2DROS::getGlobalFrameID ( )  [inline]
```

Returns the global frame of the costmap

返回

The global frame of the costmap

5.17.3.4 getOrientedFootprint()

```
void costmap_2d::Costmap2DROS::getOrientedFootprint (
            std::vector< geometry_msgs::Point > & oriented_footprint ) const
```

Build the oriented footprint of the robot at the robot's current pose

参数

oriented_footprint	Will be filled with the points in the oriented footprint of the robot

5.17.3.5 getRobotFootprint()

```
std::vector<geometry_msgs::Point> costmap_2d::Costmap2DROS::getRobotFootprint ( )  [inline]
```

Return the current footprint of the robot as a vector of points.

This version of the footprint is padded by the footprint_padding_ distance, set in the rosparam ”footprint_padding”.

The footprint initially comes from the rosparam ”footprint” but can be overwritten by dynamic reconfigure or by messages received on the ”footprint” topic.

5.17.3.6 getRobotPose()

```
bool costmap_2d::Costmap2DROS::getRobotPose (
            geometry_msgs::PoseStamped & global_pose ) const
```

Get the pose of the robot in the global frame of the costmap

参数

global_pose	Will be set to the pose of the robot in the global frame of the costmap

返回

True if the pose was set successfully, false otherwise

5.17.3.7 getUnpaddedRobotFootprint()

```
std::vector<geometry_msgs::Point> costmap_2d::Costmap2DROS::getUnpaddedRobotFootprint ( )
[inline]
```

Return the current unpadded footprint of the robot as a vector of points.

This is the raw version of the footprint without padding.

The footprint initially comes from the rosparam ”footprint” but can be overwritten by dynamic reconfigure or by messages received on the ”footprint” topic.

5.17.3.8 setUnpaddedRobotFootprint()

```
void costmap_2d::Costmap2DROS::setUnpaddedRobotFootprint (
            const std::vector< geometry_msgs::Point > & points )
```

Set the footprint of the robot to be the given set of points, padded by footprint_padding.

Should be a convex polygon, though this is not enforced.

First expands the given polygon by footprint_padding_ and then sets padded_footprint_ and calls layered_costmap_->setFootprint(). Also saves the unpadded footprint, which is available from getUnpaddedRobotFootprint().

5.17.3.9 setUnpaddedRobotFootprintPolygon()

```
void costmap_2d::Costmap2DROS::setUnpaddedRobotFootprintPolygon (
            const geometry_msgs::Polygon & footprint )
```

Set the footprint of the robot to be the given polygon, padded by footprint_padding.

Should be a convex polygon, though this is not enforced.

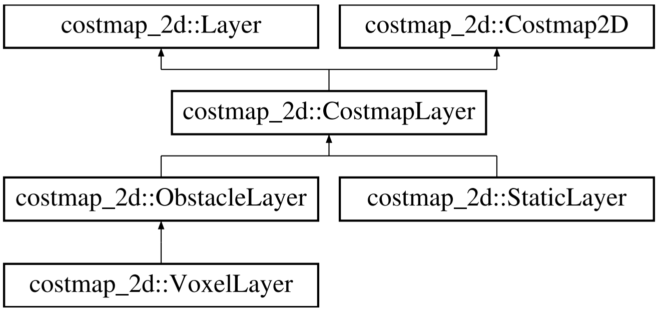
First expands the given polygon by footprint_padding_ and then sets padded_footprint_ and calls layered_costmap_->setFootprint(). Also saves the unpadded footprint, which is available from getUnpaddedRobotFootprint().

该类的文档由以下文件生成:

- costmap_2d/include/costmap_2d/costmap_2d_ros.h

5.18 costmap_2d::CostmapLayer类 参考

类 costmap_2d::CostmapLayer 继承关系图:



Public 成员函数

- bool **isDiscretized** ()
- virtual void matchSize ()

  Implement this to make this layer match the size of the parent costmap.
- virtual void **clearArea** (int start_x, int start_y, int end_x, int end_y)
- void addExtraBounds (double mx0, double my0, double mx1, double my1)

Protected 成员函数

- void **updateWithTrueOverwrite** (costmap_2d::Costmap2D &master_grid, int min_i, int min_j, int max_i, int max_j)
- void **updateWithOverwrite** (costmap_2d::Costmap2D &master_grid, int min_i, int min_j, int max_i, int max_j)
- void **updateWithMax** (costmap_2d::Costmap2D &master_grid, int min_i, int min_j, int max_i, int max_j)
- void **updateWithAddition** (costmap_2d::Costmap2D &master_grid, int min_i, int min_j, int max_i, int max_j)
- void touch (double x, double y, double ∗min_x, double ∗min_y, double ∗max_x, double ∗max_y)
- void **useExtraBounds** (double ∗min_x, double ∗min_y, double ∗max_x, double ∗max_y)

Protected 属性

- bool **has_extra_bounds_**

额外继承的成员函数

5.18.1 成员函数说明

5.18.1.1 addExtraBounds()

```
void costmap_2d::CostmapLayer::addExtraBounds (
            double mx0,
            double my0,
            double mx1,
            double my1 )
```

If an external source changes values in the costmap, it should call this method with the area that it changed to ensure that the costmap includes this region as well.

参数

mx0	Minimum x value of the bounding box
my0	Minimum y value of the bounding box
mx1	Maximum x value of the bounding box
my1	Maximum y value of the bounding box

5.18.1.2 touch()

```
void costmap_2d::CostmapLayer::touch (
            double x,
            double y,
            double ∗ min_x,
            double ∗ min_y,
            double * max_x,
            double * max_y )  [protected]
```

Updates the bounding box specified in the parameters to include the location (x,y)

参数

x	x-coordinate to include
y	y-coordinate to include
min_x	bounding box
min_y	bounding box
max_x	bounding box
max_y	bounding box

该类的文档由以下文件生成:

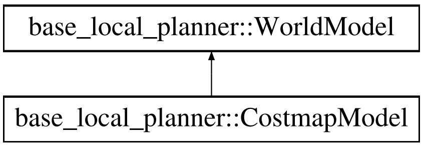
- costmap_2d/include/costmap_2d/costmap_layer.h

5.19 base_local_planner::CostmapModel类 参考

A class that implements the WorldModel interface to provide grid based collision checks for the trajectory controller using the costmap.

```
#include <costmap_model.h>
```

类 base_local_planner::CostmapModel 继承关系图:


base_local_planner::WorldModel
base_local_planner::CostmapModel

Public 成员函数

- CostmapModel (const costmap_2d::Costmap2D &costmap)

  Constructor for the CostmapModel
- virtual ∼CostmapModel ()

  Destructor for the world model
- virtual double footprintCost (const geometry_msgs::Point &position, const std::vector< geometry_msgs::Point > &footprint, double inscribed_radius, double circumscribed_radius)

  Checks if any obstacles in the costmap lie inside a convex footprint that is rasterized into the grid
- double lineCost (int x0, int x1, int y0, int y1) const

  Rasterizes a line in the costmap grid and checks for collisions
- double pointCost (int x, int y) const

Checks the cost of a point in the costmap

- virtual double footprintCost (const geometry_msgs::Point &position, const std::vector< geometry_msgs::Point > &footprint, double inscribed_radius, double circumscribed_radius)=0

  Subclass will implement this method to check a footprint at a given position and orientation for legality in the world
- double **footprintCost** (double x, double y, double theta, const std::vector< geometry_msgs::Point > &footprint_spec, double inscribed_radius=0.0, double circumscribed_radius=0.0)
- double footprintCost (const geometry_msgs::Point &position, const std::vector< geometry_msgs::Point > &footprint, double inscribed_radius, double circumscribed_radius, double extra)

  Checks if any obstacles in the costmap lie inside a convex footprint that is rasterized into the grid

5.19.1 详细描述

A class that implements the WorldModel interface to provide grid based collision checks for the trajectory controller using the costmap.

5.19.2 构造及析构函数说明

5.19.2.1 CostmapModel()

```
base_local_planner::CostmapModel::CostmapModel (
            const costmap_2d::Costmap2D & costmap )
```

Constructor for the CostmapModel

参数

costmap	The costmap that should be used

返回

5.19.3 成员函数说明

5.19.3.1 footprintCost() [1/3]

```
virtual double base_local_planner::CostmapModel::footprintCost (
            const geometry_msgs::Point & position,
            const std::vector< geometry_msgs::Point > & footprint,
            double inscribed_radius,
            double circumscribed_radius ) [virtual]
```

Checks if any obstacles in the costmap lie inside a convex footprint that is rasterized into the grid

参数

position	The position of the robot in world coordinates
footprint	The specification of the footprint of the robot in world coordinates
inscribed_radius	The radius of the inscribed circle of the robot
circumscribed_radius	The radius of the circumscribed circle of the robot

返回

> Positive if all the points lie outside the footprint, negative otherwise: -1 if footprint covers at least a lethal obstacle cell, or -2 if footprint covers at least a no-information cell, or -3 if footprint is [partially] outside of the map

实现了 base_local_planner::WorldModel.

5.19.3.2 footprintCost() [2/3]

```
virtual double base_local_planner::WorldModel::footprintCost
```

Subclass will implement this method to check a footprint at a given position and orientation for legality in the world

参数

position	The position of the robot in world coordinates
footprint	The specification of the footprint of the robot in world coordinates
inscribed_radius	The radius of the inscribed circle of the robot
circumscribed_radius	The radius of the circumscribed circle of the robot

返回

> Positive if all the points lie outside the footprint, negative otherwise: -1 if footprint covers at least a lethal obstacle cell, or -2 if footprint covers at least a no-information cell, or -3 if footprint is partially or totally outside of the map

5.19.3.3 footprintCost() [3/3]

```
double base_local_planner::WorldModel::footprintCost  [inline]
```

Checks if any obstacles in the costmap lie inside a convex footprint that is rasterized into the grid

参数

position	The position of the robot in world coordinates
footprint	The specification of the footprint of the robot in world coordinates
inscribed_radius	The radius of the inscribed circle of the robot
circumscribed_radius	The radius of the circumscribed circle of the robot

返回

Positive if all the points lie outside the footprint, negative otherwise

5.19.3.4 lineCost()

```
double base_local_planner::CostmapModel::lineCost (
             int x0,
             int x1,
             int y0,
             int y1 ) const
```

Rasterizes a line in the costmap grid and checks for collisions

参数

x0	The x position of the first cell in grid coordinates
y0	The y position of the first cell in grid coordinates
x1	The x position of the second cell in grid coordinates
y1	The y position of the second cell in grid coordinates

返回

A positive cost for a legal line... negative otherwise

5.19.3.5 pointCost()

```
double base_local_planner::CostmapModel::pointCost (
             int x,
             int y ) const
```

Checks the cost of a point in the costmap

参数

x	The x position of the point in cell coordinates
y	The y position of the point in cell coordinates

返回

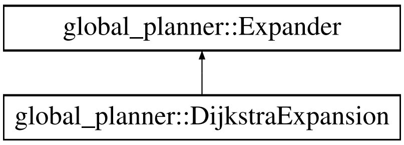
A positive cost for a legal point... negative otherwise

该类的文档由以下文件生成:

- base_local_planner/include/base_local_planner/costmap_model.h

5.20 global_planner::DijkstraExpansion类 参考

类 global_planner::DijkstraExpansion 继承关系图:



Public 成员函数

- **DijkstraExpansion** (PotentialCalculator ∗p_calc, int nx, int ny)
- bool **calculatePotentials** (unsigned char ∗costs, double start_x, double start_y, double end_x, double end_y, int cycles, float ∗potential)
- void setSize (int nx, int ny)

  Sets or resets the size of the map
- void **setNeutralCost** (unsigned char neutral_cost)
- void **setPreciseStart** (bool precise)

额外继承的成员函数

5.20.1 成员函数说明

5.20.1.1 setSize()

```
void global_planner::DijkstraExpansion::setSize (
            int nx,
            int ny )  [virtual]
```

Sets or resets the size of the map

参数

nx	The x size of the map
ny	The y size of the map sets or resets the size of the map

重载 global_planner::Expander .

该类的文档由以下文件生成:

- global_planner/include/global_planner/dijkstra.h

5.21 dwa_local_planner::DWAPlanner类 参考

A class implementing a local planner using the Dynamic Window Approach

```
#include <dwa_planner.h>
```

Public 成员函数

- DWAPlanner (std::string name, base_local_planner::LocalPlannerUtil ∗planner_util)

  Constructor for the planner
- void reconfigure (DWAPlannerConfig &cfg)

  Reconfigures the trajectory planner
- bool checkTrajectory (const Eigen::Vector3f pos, const Eigen::Vector3f vel, const Eigen::Vector3f vel_samples)

  Check if a trajectory is legal for a position/velocity pair
- base_local_planner::Trajectory findBestPath (const geometry_msgs::PoseStamped &global_pose, const geometry_msgs::PoseStamped &global_vel, geometry_msgs::PoseStamped &drive_velocities)

  Given the current position and velocity of the robot, find the best trajectory to exectue
- void updatePlanAndLocalCosts (const geometry_msgs::PoseStamped &global_pose, const std::vector< geometry_msgs::PoseStamped > &new_plan, const std::vector< geometry_msgs::Point > &footprint_spec)

  Update the cost functions before planning
- double getSimPeriod ()

  Get the period at which the local planner is expected to run
- bool getCellCosts (int cx, int cy, float &path_cost, float &goal_cost, float &occ_cost, float &total_cost)

  Compute the components and total cost for a map grid cell
- bool setPlan (const std::vector< geometry_msgs::PoseStamped > &orig_global_plan)

5.21.1 详细描述

A class implementing a local planner using the Dynamic Window Approach

5.21.2 构造及析构函数说明

5.21.2.1 DWAPlanner()

```
dwa_local_planner::DWAPlanner::DWAPlanner (
            std::string name,
            base_local_planner::LocalPlannerUtil ∗ planner_util )
```

Constructor for the planner

参数

name	The name of the planner
costmap_ros	A pointer to the costmap instance the planner should use
global_frame	the frame id of the tf frame to use

5.21.3 成员函数说明

5.21.3.1 checkTrajectory()

```
bool dwa_local_planner::DWAPlanner::checkTrajectory (
            const Eigen::Vector3f pos,
            const Eigen::Vector3f vel,
            const Eigen::Vector3f vel_samples )
```

Check if a trajectory is legal for a position/velocity pair

参数

pos	The robot's position
vel	The robot's velocity
vel_samples	The desired velocity

返回

True if the trajectory is valid, false otherwise

5.21.3.2 findBestPath()

```
base_local_planner::Trajectory dwa_local_planner::DWAPlanner::findBestPath (
            const geometry_msgs::PoseStamped & global_pose,
            const geometry_msgs::PoseStamped & global_vel,
            geometry_msgs::PoseStamped & drive_velocities )
```

Given the current position and velocity of the robot, find the best trajectory to exectue

参数

global_pose	The current position of the robot
global_vel	The current velocity of the robot
drive_velocities	The velocities to send to the robot base

返回

The highest scoring trajectory. A cost >= 0 means the trajectory is legal to execute.

5.21.3.3 getCellCosts()

```
bool dwa_local_planner::DWAPlanner::getCellCosts (
            int cx,
            int cy,
            float & path_cost,
            float & goal_cost,
            float & occ_cost,
            float & total_cost )
```

Compute the components and total cost for a map grid cell

参数

cx	The x coordinate of the cell in the map grid
cy	The y coordinate of the cell in the map grid
path_cost	Will be set to the path distance component of the cost function
goal_cost	Will be set to the goal distance component of the cost function
occ_cost	Will be set to the costmap value of the cell
total_cost	Will be set to the value of the overall cost function, taking into account the scaling parameters

返回

True if the cell is traversible and therefore a legal location for the robot to move to

5.21.3.4 getSimPeriod()

```
double dwa_local_planner::DWAPlanner::getSimPeriod ( )  [inline]
```

Get the period at which the local planner is expected to run

返回

The simulation period

5.21.3.5 setPlan()

```
bool dwa_local_planner::DWAPlanner::setPlan (
            const std::vector< geometry_msgs::PoseStamped > & orig_global_plan )
```

sets new plan and resets state

5.21.3.6 updatePlanAndLocalCosts()

```
void dwa_local_planner::DWAPlanner::updatePlanAndLocalCosts (
            const geometry_msgs::PoseStamped & global_pose,
            const std::vector< geometry_msgs::PoseStamped > & new_plan,
            const std::vector< geometry_msgs::Point > & footprint_spec )
```

Update the cost functions before planning

参数

global_pose	The robot's current pose
new_plan	The new global plan
footprint_spec	The robot's footprint

The obstacle cost function gets the footprint. The path and goal cost functions get the global_plan The alignment cost functions get a version of the global plan that is modified based on the global_pose

该类的文档由以下文件生成:

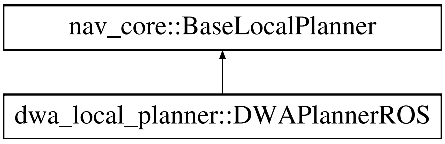
- dwa_local_planner/include/dwa_local_planner/dwa_planner.h

5.22 dwa_local_planner::DWAPlannerROS类 参考

ROS Wrapper for the DWAPlanner that adheres to the BaseLocalPlanner interface and can be used as a plugin for move_base.

```
#include <dwa_planner_ros.h>
```

类 dwa_local_planner::DWAPlannerROS 继承关系图:



Public 成员函数

- DWAPlannerROS ()

  Constructor for DWAPlannerROS wrapper
- void initialize (std::string name, tf2_ros::Buffer ∗tf, costmap_2d::Costmap2DROS ∗costmap_ros)

  Constructs the ros wrapper
- ∼DWAPlannerROS ()

  Destructor for the wrapper
- bool computeVelocityCommands (geometry_msgs::Twist &cmd_vel)

  Given the current position, orientation, and velocity of the robot, compute velocity commands to send to the base
- bool dwaComputeVelocityCommands (geometry_msgs::PoseStamped &global_pose, geometry_msgs::Twist &cmd_vel)

  Given the current position, orientation, and velocity of the robot, compute velocity commands to send to the base, using dynamic window approach
- bool setPlan (const std::vector< geometry_msgs::PoseStamped > &orig_global_plan)

  Set the plan that the controller is following
- bool isGoalReached ()

  Check if the goal pose has been achieved
- bool **isInitialized** ()

5.22.1 详细描述

ROS Wrapper for the DWAPlanner that adheres to the BaseLocalPlanner interface and can be used as a plugin for move_base.

5.22.2 成员函数说明

5.22.2.1 computeVelocityCommands()

```
bool dwa_local_planner::DWAPlannerROS::computeVelocityCommands (
            geometry_msgs::Twist & cmd_vel )  [virtual]
```

Given the current position, orientation, and velocity of the robot, compute velocity commands to send to the base

参数

cmd_vel	Will be filled with the velocity command to be passed to the robot base

返回

True if a valid trajectory was found, false otherwise

实现了 nav_core::BaseLocalPlanner.

5.22.2.2 dwaComputeVelocityCommands()

```
bool dwa_local_planner::DWAPlannerROS::dwaComputeVelocityCommands (
            geometry_msgs::PoseStamped & global_pose,
            geometry_msgs::Twist & cmd_vel )
```

Given the current position, orientation, and velocity of the robot, compute velocity commands to send to the base, using dynamic window approach

参数

cmd_vel	Will be filled with the velocity command to be passed to the robot base

返回

True if a valid trajectory was found, false otherwise

5.22.2.3 initialize()

```
void dwa_local_planner::DWAPlannerROS::initialize (
            std::string name,
            tf2_ros::Buffer * tf,
            costmap_2d::Costmap2DROS * costmap_ros )  [virtual]
```

Constructs the ros wrapper

参数

name	The name to give this instance of the trajectory planner
tf	A pointer to a transform listener
costmap	The cost map to use for assigning costs to trajectories

实现了 nav_core::BaseLocalPlanner.

5.22.2.4 isGoalReached()

```
bool dwa_local_planner::DWAPlannerROS::isGoalReached ( )  [virtual]
```

Check if the goal pose has been achieved

返回

True if achieved, false otherwise

实现了 nav_core::BaseLocalPlanner.

5.22.2.5 setPlan()

```
bool dwa_local_planner::DWAPlannerROS::setPlan (
            const std::vector< geometry_msgs::PoseStamped > & orig_global_plan )  [virtual]
```

Set the plan that the controller is following

参数

orig_global_plan	The plan to pass to the controller

返回

True if the plan was updated successfully, false otherwise

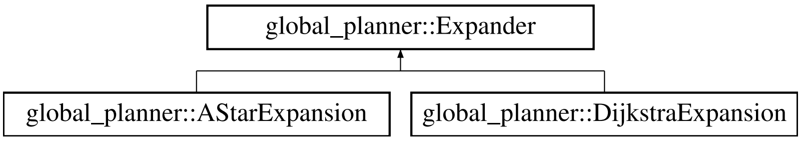
实现了 nav_core::BaseLocalPlanner.

该类的文档由以下文件生成:

- dwa_local_planner/include/dwa_local_planner/dwa_planner_ros.h

5.23 global_planner::Expander类 参考

类 global_planner::Expander 继承关系图:


global_planner::Expander
global_planner::AStarExpansion
global_planner::DijkstraExpansion

Public 成员函数

- **Expander** (PotentialCalculator ∗p_calc, int nx, int ny)
- virtual bool **calculatePotentials** (unsigned char ∗costs, double start_x, double start_y, double end_x, double end_y, int cycles, float ∗potential)=0
- virtual void setSize (int nx, int ny)

  Sets or resets the size of the map
- void **setLethalCost** (unsigned char lethal_cost)
- void **setNeutralCost** (unsigned char neutral_cost)
- void **setFactor** (float factor)
- void **setHasUnknown** (bool unknown)
- void **clearEndpoint** (unsigned char ∗costs, float ∗potential, int gx, int gy, int s)

Protected 成员函数

- int **toIndex** (int x, int y)

Protected 属性

- int **nx_**
- int **ny_**
- int ns_
- bool **unknown_**
- unsigned char **lethal_cost_**
- unsigned char **neutral_cost_**
- int **cells_visited_**
- float **factor_**
- PotentialCalculator ∗ **p_calc_**

5.23.1 成员函数说明

5.23.1.1 setSize()

```
virtual void global_planner::Expander::setSize (
            int nx,
            int ny )  [inline], [virtual]
```

Sets or resets the size of the map

参数

nx	The x size of the map
ny	The y size of the map sets or resets the size of the map

被 global_planner::DijkstraExpansion 重载.

5.23.2 结构体成员变量说明

5.23.2.1 ns_

```
int global_planner::Expander::ns_ [protected]
```

size of grid, in pixels

该类的文档由以下文件生成:

- global_planner/include/global_planner/expander.h

5.24 base_local_planner::FootprintHelper类 参考

Public 成员函数

- std::vector< base_local_planner::Position2DInt > getFootprintCells (Eigen::Vector3f pos, std::vector< geometry_msgs::Point > footprint_spec, const costmap_2d::Costmap2D &, bool fill)

  Used to get the cells that make up the footprint of the robot
- void getLineCells (int x0, int x1, int y0, int y1, std::vector< base_local_planner::Position2DInt > &pts)

  Use Bresenham's algorithm to trace a line between two points in a grid
- void getFillCells (std::vector< base_local_planner::Position2DInt > &footprint)

  Fill the outline of a polygon, in this case the robot footprint, in a grid

5.24.1 成员函数说明

5.24.1.1 getFillCells()

```
void base_local_planner::FootprintHelper::getFillCells (
             std::vector< base_local_planner::Position2DInt > & footprint )
```

Fill the outline of a polygon, in this case the robot footprint, in a grid

参数

footprint	The list of cells making up the footprint in the grid, will be modified to include all cells inside the footprint

5.24.1.2 getFootprintCells()

```
std::vector<base_local_planner::Position2DInt> base_local_planner::FootprintHelper::getFootprint↩
Cells (
            Eigen::Vector3f pos,
            std::vector< geometry_msgs::Point > footprint_spec,
            const costmap_2d::Costmap2D & ,
            bool fill )
```

Used to get the cells that make up the footprint of the robot

参数

x_i	The x position of the robot
y_i	The y position of the robot
theta↩ _i	The orientation of the robot
fill	If true: returns all cells in the footprint of the robot. If false: returns only the cells that make up the outline of the footprint.

返回

The cells that make up either the outline or entire footprint of the robot depending on fill

5.24.1.3 getLineCells()

```
void base_local_planner::FootprintHelper::getLineCells (
            int x0,
            int x1,
            int y0,
            int y1,
            std::vector< base_local_planner::Position2DInt > & pts )
```

Use Bresenham's algorithm to trace a line between two points in a grid

参数

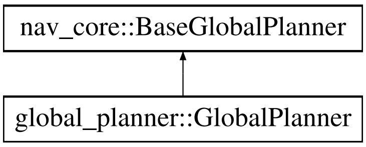
x0	The x coordinate of the first point
x1	The x coordinate of the second point
y0	The y coordinate of the first point
y1	The y coordinate of the second point
pts	Will be filled with the cells that lie on the line in the grid

该类的文档由以下文件生成:

- base_local_planner/include/base_local_planner/footprint_helper.h

5.25 global_planner::GlobalPlanner类 参考

类 global_planner::GlobalPlanner 继承关系图:



Public 成员函数

- GlobalPlanner ()

  Default constructor for the PlannerCore object
- GlobalPlanner (std::string name, costmap_2d::Costmap2D ∗costmap, std::string frame_id)

  Constructor for the PlannerCore object
- ∼GlobalPlanner ()

  Default deconstructor for the PlannerCore object
- void initialize (std::string name, costmap_2d::Costmap2DROS ∗costmap_ros)

  Initialization function for the PlannerCore object
- void **initialize** (std::string name, costmap_2d::Costmap2D ∗costmap, std::string frame_id)
- bool makePlan (const geometry_msgs::PoseStamped &start, const geometry_msgs::PoseStamped &goal, std::vector< geometry_msgs::PoseStamped > &plan)

  Given a goal pose in the world, compute a plan
- bool makePlan (const geometry_msgs::PoseStamped &start, const geometry_msgs::PoseStamped &goal, double tolerance, std::vector< geometry_msgs::PoseStamped > &plan)

  Given a goal pose in the world, compute a plan
- bool computePotential (const geometry_msgs::Point &world_point)

  Computes the full navigation function for the map given a point in the world to start from
- bool getPlanFromPotential (double start_x, double start_y, double end_x, double end_y, const geometry_msgs::PoseStamped &goal, std::vector< geometry_msgs::PoseStamped > &plan)

  Compute a plan to a goal after the potential for a start point has already been computed (Note: You should call computePotential first)
- double getPointPotential (const geometry_msgs::Point &world_point)

  Get the potential, or naviagation cost, at a given point in the world (Note: You should call computePotential first)
- bool validPointPotential (const geometry_msgs::Point &world_point)

  Check for a valid potential value at a given point in the world (Note: You should call computePotential first)
- bool validPointPotential (const geometry_msgs::Point &world_point, double tolerance)

  Check for a valid potential value at a given point in the world (Note: You should call computePotential first)
- void publishPlan (const std::vector< geometry_msgs::PoseStamped > &path)

  Publish a path for visualization purposes
- bool **makePlanService** (nav_msgs::GetPlan::Request &req, nav_msgs::GetPlan::Response &resp)

Protected 属性

- costmap_2d::Costmap2D ∗ costmap_

  Store a copy of the current costmap in costmap. Called by makePlan.
- std::string **frame_id_**
- ros::Publisher **plan_pub_**
- bool **initialized_**
- bool **allow_unknown_**

5.25.1 构造及析构函数说明

5.25.1.1 GlobalPlanner()

```
global_planner::GlobalPlanner::GlobalPlanner (
            std::string name,
            costmap_2d::Costmap2D * costmap,
            std::string frame_id )
```

Constructor for the PlannerCore object

参数

name	The name of this planner
costmap	A pointer to the costmap to use
frame↵_id	Frame of the costmap

5.25.2 成员函数说明

5.25.2.1 computePotential()

```
bool global_planner::GlobalPlanner::computePotential (
            const geometry_msgs::Point & world_point )
```

Computes the full navigation function for the map given a point in the world to start from

参数

world_point	The point to use for seeding the navigation function

返回

True if the navigation function was computed successfully, false otherwise

5.25.2.2 getPlanFromPotential()

```
bool global_planner::GlobalPlanner::getPlanFromPotential (
             double start_x,
             double start_y,
             double end_x,
             double end_y,
             const geometry_msgs::PoseStamped & goal,
             std::vector< geometry_msgs::PoseStamped > & plan )
```

Compute a plan to a goal after the potential for a start point has already been computed (Note: You should call computePotential first)

参数

start↩ _x	
start↩ _y	
end↩ _x	
end↩ _y	
goal	The goal pose to create a plan to
plan	The plan... filled by the planner

返回

True if a valid plan was found, false otherwise

5.25.2.3 getPointPotential()

```
double global_planner::GlobalPlanner::getPointPotential (
             const geometry_msgs::Point & world_point )
```

Get the potential, or naviagation cost, at a given point in the world (Note: You should call computePotential first)

参数

world_point	The point to get the potential for

返回

The navigation function's value at that point in the world

5.25.2.4 initialize()

```
void global_planner::GlobalPlanner::initialize (
            std::string name,
            costmap_2d::Costmap2DROS * costmap_ros )  [virtual]
```

Initialization function for the PlannerCore object

参数

name	The name of this planner
costmap_ros	A pointer to the ROS wrapper of the costmap to use for planning

实现了 nav_core::BaseGlobalPlanner.

5.25.2.5 makePlan() [1/2]

```
bool global_planner::GlobalPlanner::makePlan (
            const geometry_msgs::PoseStamped & start,
            const geometry_msgs::PoseStamped & goal,
            double tolerance,
            std::vector< geometry_msgs::PoseStamped > & plan )
```

Given a goal pose in the world, compute a plan

参数

start	The start pose
goal	The goal pose
tolerance	The tolerance on the goal point for the planner
plan	The plan... filled by the planner

返回

True if a valid plan was found, false otherwise

5.25.2.6 makePlan() [2/2]

```
bool global_planner::GlobalPlanner::makePlan (
            const geometry_msgs::PoseStamped & start,
            const geometry_msgs::PoseStamped & goal,
            std::vector< geometry_msgs::PoseStamped > & plan )  [virtual]
```

Given a goal pose in the world, compute a plan

参数

start	The start pose
goal	The goal pose
plan	The plan... filled by the planner

返回

True if a valid plan was found, false otherwise

实现了 nav_core::BaseGlobalPlanner.

5.25.2.7 validPointPotential() [1/2]

```
bool global_planner::GlobalPlanner::validPointPotential (
            const geometry_msgs::Point & world_point )
```

Check for a valid potential value at a given point in the world (Note: You should call computePotential first)

参数

world_point	The point to get the potential for

返回

True if the navigation function is valid at that point in the world, false otherwise

5.25.2.8 validPointPotential() [2/2]

```
bool global_planner::GlobalPlanner::validPointPotential (
            const geometry_msgs::Point & world_point,
            double tolerance )
```

Check for a valid potential value at a given point in the world (Note: You should call computePotential first)

参数

world_point	The point to get the potential for
tolerance	The tolerance on searching around the world_point specified

返回

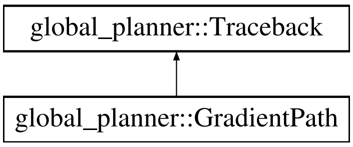
True if the navigation function is valid at that point in the world, false otherwise

该类的文档由以下文件生成:

- global_planner/include/global_planner/planner_core.h

5.26 global_planner::GradientPath类 参考

类 global_planner::GradientPath 继承关系图:



Public 成员函数

- **GradientPath** (PotentialCalculator ∗p_calc)
- void **setSize** (int xs, int ys)
- bool **getPath** (float ∗potential, double start_x, double start_y, double end_x, double end_y, std::vector< std::pair< float, float > > &path)

额外继承的成员函数

该类的文档由以下文件生成:

- global_planner/include/global_planner/gradient_path.h

5.27 global_planner::greater1结构体 参考

Public 成员函数

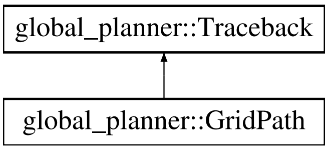
- bool **operator()** (const Index &a, const Index &b) const

该结构体的文档由以下文件生成:

- global_planner/include/global_planner/astar.h

5.28 global_planner::GridPath类 参考

类 global_planner::GridPath 继承关系图:



Public 成员函数

- **GridPath** (PotentialCalculator *p_calc)
- bool **getPath** (float *potential, double start_x, double start_y, double end_x, double end_y, std::vector< std↩ ::pair< float, float > > &path)

额外继承的成员函数

该类的文档由以下文件生成:

- global_planner/include/global_planner/grid_path.h

5.29 global_planner::Index类 参考

Public 成员函数

- **Index** (int a, float b)

成员变量

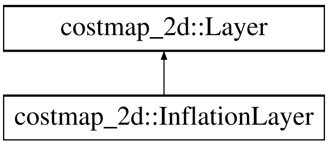
- int **i**
- float **cost**

该类的文档由以下文件生成:

- global_planner/include/global_planner/astar.h

5.30 costmap_2d::InflationLayer类 参考

类 costmap_2d::InflationLayer 继承关系图:



Public 成员函数

- virtual void onInitialize ()

  This is called at the end of initialize(). Override to implement subclass-specific initialization.
- virtual void updateBounds (double robot_x, double robot_y, double robot_yaw, double ∗min_x, double ∗min_y, double ∗max_x, double ∗max_y)

  This is called by the LayeredCostmap to poll this plugin as to how much of the costmap it needs to update. Each layer can increase the size of this bounds.
- virtual void updateCosts (costmap_2d::Costmap2D &master_grid, int min_i, int min_j, int max_i, int max_j)

  Actually update the underlying costmap, only within the bounds calculated during UpdateBounds().
- virtual bool **isDiscretized** ()
- virtual void matchSize ()

  Implement this to make this layer match the size of the parent costmap.
- virtual void **reset** ()
- virtual unsigned char computeCost (double distance) const

  Given a distance, compute a cost.
- void setInflationParameters (double inflation_radius, double cost_scaling_factor)

  Change the values of the inflation radius parameters

Protected 成员函数

- virtual void onFootprintChanged ()

  LayeredCostmap calls this whenever the footprint there changes (via LayeredCostmap::setFootprint()). Override to be notified of changes to the robot's footprint.

Protected 属性

- boost::recursive_mutex ∗ **inflation_access_**
- double **resolution_**
- double **inflation_radius_**
- double **inscribed_radius_**
- double **weight_**
- bool **inflate_unknown_**

5.30.1 成员函数说明

5.30.1.1 computeCost()

```
virtual unsigned char costmap_2d::InflationLayer::computeCost (
            double distance ) const  [inline], [virtual]
```

Given a distance, compute a cost.

参数

distance	The distance from an obstacle in cells

返回

A cost value for the distance

5.30.1.2 onInitialize()

```
virtual void costmap_2d::InflationLayer::onInitialize ( )  [virtual]
```

This is called at the end of initialize(). Override to implement subclass-specific initialization.

tf_, name_, and layered_costmap_ will all be set already when this is called.

重载 costmap_2d::Layer .

5.30.1.3 setInflationParameters()

```
void costmap_2d::InflationLayer::setInflationParameters (
             double inflation_radius,
             double cost_scaling_factor )
```

Change the values of the inflation radius parameters

参数

inflation_radius	The new inflation radius
cost_scaling_factor	The new weight

5.30.1.4 updateBounds()

```
virtual void costmap_2d::InflationLayer::updateBounds (
             double robot_x,
             double robot_y,
             double robot_yaw,
             double * min_x,
             double * min_y,
             double * max_x,
             double * max_y )  [virtual]
```

This is called by the LayeredCostmap to poll this plugin as to how much of the costmap it needs to update. Each layer can increase the size of this bounds.

For more details, see ”Layered Costmaps for Context-Sensitive Navigation”, by Lu et. Al, IROS 2014.

重载 costmap_2d::Layer .

该类的文档由以下文件生成:

- costmap_2d/include/costmap_2d/inflation_layer.h

5.31 base_local_planner::LatchedStopRotateController类 参考

Public 成员函数

- **LatchedStopRotateController** (const std::string &name="")
- bool **isPositionReached** (LocalPlannerUtil *planner_util, const geometry_msgs::PoseStamped &global_pose)
- bool **isGoalReached** (LocalPlannerUtil *planner_util, OdometryHelperRos &odom_helper, const geometry_msgs::PoseStamped &global_pose)
- void **resetLatching** ()
- bool stopWithAccLimits (const geometry_msgs::PoseStamped &global_pose, const geometry_msgs::PoseStamped &robot_vel, geometry_msgs::Twist &cmd_vel, Eigen::Vector3f acc_lim, double sim_period, boost::function< bool(Eigen::Vector3f pos, Eigen::Vector3f vel, Eigen::Vector3f vel_samples)> obstacle_check)

  Stop the robot taking into account acceleration limits
- bool rotateToGoal (const geometry_msgs::PoseStamped &global_pose, const geometry_msgs::PoseStamped &robot_vel, double goal_th, geometry_msgs::Twist &cmd_vel, Eigen::Vector3f acc_lim, double sim_period, base_local_planner::LocalPlannerLimits &limits, boost::function< bool(Eigen::Vector3f pos, Eigen::Vector3f vel, Eigen::Vector3f vel_samples)> obstacle_check)

  Once a goal position is reached... rotate to the goal orientation
- bool **computeVelocityCommandsStopRotate** (geometry_msgs::Twist &cmd_vel, Eigen::Vector3f acc_lim, double sim_period, LocalPlannerUtil *planner_util, OdometryHelperRos &odom_helper, const geometry_msgs::PoseStamped &global_pose, boost::function< bool(Eigen::Vector3f pos, Eigen::Vector3f vel, Eigen::Vector3f vel_samples)> obstacle_check)

5.31.1 成员函数说明

5.31.1.1 rotateToGoal()

```
bool base_local_planner::LatchedStopRotateController::rotateToGoal (
             const geometry_msgs::PoseStamped & global_pose,
             const geometry_msgs::PoseStamped & robot_vel,
             double goal_th,
             geometry_msgs::Twist & cmd_vel,
             Eigen::Vector3f acc_lim,
             double sim_period,
             base_local_planner::LocalPlannerLimits & limits,
             boost::function< bool(Eigen::Vector3f pos, Eigen::Vector3f vel, Eigen::Vector3f
vel_samples)> obstacle_check )
```

Once a goal position is reached... rotate to the goal orientation

参数

global_pose	The pose of the robot in the global frame
robot_vel	The velocity of the robot
goal_th	The desired th value for the goal
cmd_vel	The velocity commands to be filled

返回

True if a valid trajectory was found, false otherwise

5.31.1.2 stopWithAccLimits()

```
bool base_local_planner::LatchedStopRotateController::stopWithAccLimits (
             const geometry_msgs::PoseStamped & global_pose,
             const geometry_msgs::PoseStamped & robot_vel,
             geometry_msgs::Twist & cmd_vel,
             Eigen::Vector3f acc_lim,
             double sim_period,
             boost::function< bool(Eigen::Vector3f pos, Eigen::Vector3f vel, Eigen::Vector3f
vel_samples)> obstacle_check )
```

Stop the robot taking into account acceleration limits

参数

global_pose	The pose of the robot in the global frame
robot_vel	The velocity of the robot
cmd_vel	The velocity commands to be filled

返回

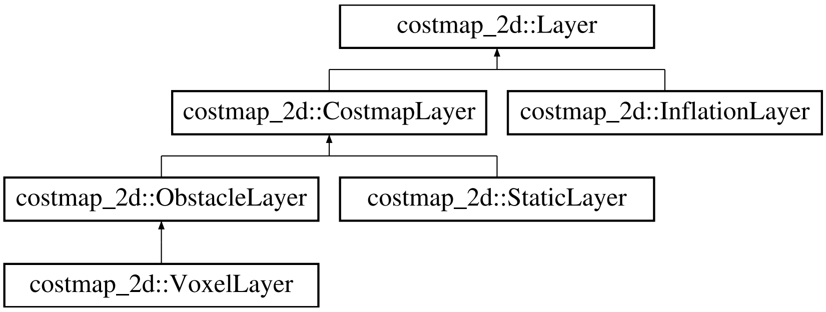
True if a valid trajectory was found, false otherwise

该类的文档由以下文件生成:

- base_local_planner/include/base_local_planner/latched_stop_rotate_controller.h

5.32 costmap_2d::Layer类 参考

类 costmap_2d::Layer 继承关系图:



Public 成员函数

- void **initialize** (LayeredCostmap ∗parent, std::string name, tf2_ros::Buffer ∗tf)
- virtual void updateBounds (double robot_x, double robot_y, double robot_yaw, double ∗min_x, double ∗min_y, double ∗max_x, double ∗max_y)

  This is called by the LayeredCostmap to poll this plugin as to how much of the costmap it needs to update. Each layer can increase the size of this bounds.
- virtual void updateCosts (Costmap2D &master_grid, int min_i, int min_j, int max_i, int max_j)

  Actually update the underlying costmap, only within the bounds calculated during UpdateBounds().
- virtual void deactivate ()

  Stop publishers.
- virtual void activate ()

  Restart publishers if they've been stopped.
- virtual void **reset** ()
- bool isCurrent () const

  Check to make sure all the data in the layer is up to date. If the layer is not up to date, then it may be unsafe to plan using the data from this layer, and the planner may need to know.
- virtual void matchSize ()

  Implement this to make this layer match the size of the parent costmap.
- std::string **getName** () const
- const std::vector< geometry_msgs::Point > & getFootprint () const

  Convenience function for layered_costmap_->getFootprint().
- virtual void onFootprintChanged ()

  LayeredCostmap calls this whenever the footprint there changes (via LayeredCostmap::setFootprint()). Override to be notified of changes to the robot's footprint.

Protected 成员函数

- virtual void onInitialize ()

  This is called at the end of initialize(). Override to implement subclass-specific initialization.

Protected 属性

- LayeredCostmap ∗ **layered_costmap_**
- bool **current_**
- bool enabled_

  Currently this var is managed by subclasses. TODO: make this managed by this class and/or container class.
- std::string **name_**
- tf2_ros::Buffer ∗ **tf_**

5.32.1 成员函数说明

5.32.1.1 isCurrent()

```
bool costmap_2d::Layer::isCurrent ( ) const  [inline]
```

Check to make sure all the data in the layer is up to date. If the layer is not up to date, then it may be unsafe to plan using the data from this layer, and the planner may need to know.

A layer's current state should be managed by the protected variable current_.

返回

Whether the data in the layer is up to date.

5.32.1.2 onInitialize()

```
virtual void costmap_2d::Layer::onInitialize ( )  [inline], [protected], [virtual]
```

This is called at the end of initialize(). Override to implement subclass-specific initialization.

tf_, name_, and layered_costmap_ will all be set already when this is called.

被 costmap_2d::VoxelLayer, costmap_2d::StaticLayer, costmap_2d::ObstacleLayer , 以及 costmap_2d::InflationLayer 重载.

5.32.1.3 updateBounds()

```
virtual void costmap_2d::Layer::updateBounds (
             double robot_x,
             double robot_y,
             double robot_yaw,
             double * min_x,
             double * min_y,
             double * max_x,
             double * max_y )  [inline], [virtual]
```

This is called by the LayeredCostmap to poll this plugin as to how much of the costmap it needs to update. Each layer can increase the size of this bounds.

For more details, see ”Layered Costmaps for Context-Sensitive Navigation”, by Lu et. Al, IROS 2014.

被 costmap_2d::VoxelLayer, costmap_2d::StaticLayer, costmap_2d::ObstacleLayer , 以及 costmap_2d::InflationLayer 重载.

该类的文档由以下文件生成:

- costmap_2d/include/costmap_2d/layer.h

5.33 costmap_2d::LayeredCostmap类 参考

Instantiates different layer plugins and aggregates them into one score

```
#include <layered_costmap.h>
```

Public 成员函数

- LayeredCostmap (std::string global_frame, bool rolling_window, bool track_unknown)

  Constructor for a costmap
- ~LayeredCostmap ()

  Destructor
- void updateMap (double robot_x, double robot_y, double robot_yaw)

  Update the underlying costmap with new data. If you want to update the map outside of the update loop that runs, you can call this.
- std::string **getGlobalFrameID** () const
- void **resizeMap** (unsigned int size_x, unsigned int size_y, double resolution, double origin_x, double origin_y, bool size_locked=false)
- void **getUpdatedBounds** (double &minx, double &miny, double &maxx, double &maxy)
- bool **isCurrent** ()
- Costmap2D ∗ **getCostmap** ()
- bool **isRolling** ()
- bool **isTrackingUnknown** ()
- std::vector< boost::shared_ptr< Layer > > ∗ **getPlugins** ()
- void **addPlugin** (boost::shared_ptr< Layer > plugin)
- bool **isSizeLocked** ()
- void **getBounds** (unsigned int ∗x0, unsigned int ∗xn, unsigned int ∗y0, unsigned int ∗yn)
- bool **isInitialized** ()
- void setFootprint (const std::vector< geometry_msgs::Point > &footprint_spec)

  Updates the stored footprint, updates the circumscribed（外接圆半径） and inscribed radii（内切圆）, and calls onFootprintChanged() in all layers.
- const std::vector< geometry_msgs::Point > & getFootprint ()

  Returns the latest footprint stored with setFootprint().
- double getCircumscribedRadius ()

  The radius of a circle centered at the origin of the robot which just surrounds all points on the robot's footprint.
- double getInscribedRadius ()

  The radius of a circle centered at the origin of the robot which is just within all points and edges of the robot's footprint.

5.33.1 详细描述

Instantiates different layer plugins and aggregates them into one score

5.33.2 成员函数说明

5.33.2.1 getCircumscribedRadius()

```
double costmap_2d::LayeredCostmap::getCircumscribedRadius ( )  [inline]
```

The radius of a circle centered at the origin of the robot which just surrounds all points on the robot's footprint.

This is updated by setFootprint().

5.33.2.2 getInscribedRadius()

```
double costmap_2d::LayeredCostmap::getInscribedRadius ( )  [inline]
```

The radius of a circle centered at the origin of the robot which is just within all points and edges of the robot's footprint.

This is updated by setFootprint().

该类的文档由以下文件生成:

- costmap_2d/include/costmap_2d/layered_costmap.h

5.34 base_local_planner::LineIterator类 参考

```
#include <line_iterator.h>
```

Public 成员函数

- **LineIterator** (int x0, int y0, int x1, int y1)
- bool **isValid** () const
- void **advance** ()
- int **getX** () const
- int **getY** () const
- int **getX0** () const
- int **getY0** () const
- int **getX1** () const
- int **getY1** () const

5.34.1 详细描述

An iterator implementing Bresenham Ray-Tracing.

该类的文档由以下文件生成:

- base_local_planner/include/base_local_planner/line_iterator.h

5.35 base_local_planner::LocalPlannerLimits类 参考

Public 成员函数

- **LocalPlannerLimits** (double nmax_vel_trans, double nmin_vel_trans, double nmax_vel_x, double nmin_vel_x, double nmax_vel_y, double nmin_vel_y, double nmax_vel_theta, double nmin_vel_theta, double nacc_lim_x, double nacc_lim_y, double nacc_lim_theta, double nacc_lim_trans, double nxy_goal_tolerance, double nyaw_goal_tolerance, bool nprune_plan=true, double ntrans_stopped_vel=0.1, double ntheta_stopped_vel=0.1)
- Eigen::Vector3f getAccLimits ()

  Get the acceleration limits of the robot

成员变量

- double **max_vel_trans**
- double **min_vel_trans**
- double **max_vel_x**
- double **min_vel_x**
- double **max_vel_y**
- double **min_vel_y**
- double **max_vel_theta**
- double **min_vel_theta**
- double **acc_lim_x**
- double **acc_lim_y**
- double **acc_lim_theta**
- double **acc_lim_trans**
- bool **prune_plan**
- double **xy_goal_tolerance**
- double **yaw_goal_tolerance**
- double **trans_stopped_vel**
- double **theta_stopped_vel**
- bool **restore_defaults**

5.35.1 成员函数说明

5.35.1.1 getAccLimits()

```
Eigen::Vector3f base_local_planner::LocalPlannerLimits::getAccLimits ( )  [inline]
```

Get the acceleration limits of the robot

返回

The acceleration limits of the robot

该类的文档由以下文件生成:

- base_local_planner/include/base_local_planner/local_planner_limits.h

5.36 base_local_planner::LocalPlannerUtil类 参考

Helper class implementing infrastructure code many local planner implementations may need.

`#include <local_planner_util.h>`

Public 成员函数

- void reconfigureCB (LocalPlannerLimits &config, bool restore_defaults)

  Callback to update the local planner's parameters
- void **initialize** (tf2_ros::Buffer *tf, costmap_2d::Costmap2D *costmap, std::string global_frame)
- bool **getGoal** (geometry_msgs::PoseStamped &goal_pose)
- bool **setPlan** (const std::vector< geometry_msgs::PoseStamped > &orig_global_plan)
- bool **getLocalPlan** (const geometry_msgs::PoseStamped &global_pose, std::vector< geometry_msgs::↩ PoseStamped > &transformed_plan)
- costmap_2d::Costmap2D * **getCostmap** ()
- LocalPlannerLimits **getCurrentLimits** ()
- std::string **getGlobalFrame** ()

5.36.1 详细描述

Helper class implementing infrastructure code many local planner implementations may need.

该类的文档由以下文件生成:

- base_local_planner/include/base_local_planner/local_planner_util.h

5.37 map_cell_t结构体 参考

成员变量

- int **occ_state**
- double **occ_dist**

该结构体的文档由以下文件生成:

- amcl/include/amcl/map/map.h

5.38 map_t结构体 参考

成员变量

- double **origin_x**
- double **origin_y**
- double **scale**
- int **size_x**
- int **size_y**
- map_cell_t * **cells**
- double **max_occ_dist**

该结构体的文档由以下文件生成:

- amcl/include/amcl/map/map.h

5.39 base_local_planner::MapCell类 参考

Stores path distance and goal distance information used for scoring trajectories

```
#include <map_cell.h>
```

Public 成员函数

- MapCell ()

  Default constructor
- MapCell (const MapCell &mc)

  Copy constructor

成员变量

- unsigned int **cx**
- unsigned int cy

  Cell index in the grid map
- double target_dist

  Distance to planner's path
- bool target_mark

  Marks for computing path/goal distances
- bool within_robot

  Mark for cells within the robot footprint

5.39.1 详细描述

Stores path distance and goal distance information used for scoring trajectories

5.39.2 构造及析构函数说明

5.39.2.1 MapCell()

```
base_local_planner::MapCell::MapCell (
                const MapCell & mc )
```

Copy constructor

参数

mc	The MapCell to be copied

该类的文档由以下文件生成:

- base_local_planner/include/base_local_planner/map_cell.h

5.40 base_local_planner::MapGrid类 参考

A grid of MapCell cells that is used to propagate path and goal distances for the trajectory controller.

```
#include <map_grid.h>
```

Public 成员函数

- MapGrid ()

  Creates a 0x0 map by default
- MapGrid (unsigned int size_x, unsigned int size_y)

  Creates a map of size_x by size_y
- MapCell & operator() (unsigned int x, unsigned int y)

  Returns a map cell accessed by (col, row)
- MapCell operator() (unsigned int x, unsigned int y) const

  Returns a map cell accessed by (col, row)
- MapCell & **getCell** (unsigned int x, unsigned int y)
- ∼MapGrid ()

  Destructor for a MapGrid
- MapGrid (const MapGrid &mg)

  Copy constructor for a MapGrid
- MapGrid & operator= (const MapGrid &mg)

  Assignment operator for a MapGrid
- void resetPathDist ()

  reset path distance fields for all cells
- void sizeCheck (unsigned int size_x, unsigned int size_y)

  check if we need to resize
- void commonInit ()

  Utility to share initialization code across constructors
- size_t getIndex (int x, int y)

  Returns a 1D index into the MapCell array for a 2D index
- double obstacleCosts ()
- double unreachableCellCosts ()
- bool updatePathCell (MapCell ∗current_cell, MapCell ∗check_cell, const costmap_2d::Costmap2D &costmap)

  Used to update the distance of a cell in path distance computation
- void computeTargetDistance (std::queue< MapCell ∗ > &dist_queue, const costmap_2d::Costmap2D &costmap)

  Compute the distance from each cell in the local map grid to the planned path
- void computeGoalDistance (std::queue< MapCell ∗ > &dist_queue, const costmap_2d::Costmap2D &costmap)

  Compute the distance from each cell in the local map grid to the local goal point
- void setTargetCells (const costmap_2d::Costmap2D &costmap, const std::vector< geometry_msgs::PoseStamped > &global_plan)

  Update what cells are considered path based on the global plan
- void setLocalGoal (const costmap_2d::Costmap2D &costmap, const std::vector< geometry_msgs::PoseStamped > &global_plan)

  Update what cell is considered the next local goal

静态 Public 成员函数

- static void adjustPlanResolution (const std::vector< geometry_msgs::PoseStamped > &global_plan_in, std::vector< geometry_msgs::PoseStamped > &global_plan_out, double resolution)

成员变量

- double **goal_x_**
- double goal_y_

  The goal distance was last computed from
- unsigned int **size_x_**
- unsigned int size_y_

  The dimensions of the grid

5.40.1 详细描述

A grid of MapCell cells that is used to propagate path and goal distances for the trajectory controller.

5.40.2 构造及析构函数说明

5.40.2.1 MapGrid() [1/2]

```
base_local_planner::MapGrid::MapGrid (
              unsigned int size_x,
              unsigned int size_y )
```

Creates a map of size_x by size_y

参数

size_x	The width of the map
size_y	The height of the map

5.40.2.2 MapGrid() [2/2]

```
base_local_planner::MapGrid::MapGrid (
              const MapGrid & mg )
```

Copy constructor for a MapGrid

参数

mg	The MapGrid to copy

5.40.3 成员函数说明

5.40.3.1 adjustPlanResolution()

```
static void base_local_planner::MapGrid::adjustPlanResolution (
            const std::vector< geometry_msgs::PoseStamped > & global_plan_in,
            std::vector< geometry_msgs::PoseStamped > & global_plan_out,
            double resolution )  [static]
```

increase global plan resolution to match that of the costmap by adding points linearly between global plan points This is necessary where global planners produce plans with few points.

参数

global_plan_in	input
global_plan_output	output
resolution	desired distance between waypoints

5.40.3.2 computeGoalDistance()

```
void base_local_planner::MapGrid::computeGoalDistance (
            std::queue< MapCell * > & dist_queue,
            const costmap_2d::Costmap2D & costmap )
```

Compute the distance from each cell in the local map grid to the local goal point

参数

goal_queue	A queue containing the local goal cell

5.40.3.3 computeTargetDistance()

```
void base_local_planner::MapGrid::computeTargetDistance (
            std::queue< MapCell * > & dist_queue,
            const costmap_2d::Costmap2D & costmap )
```

Compute the distance from each cell in the local map grid to the planned path

参数

dist_queue	A queue of the initial cells on the path

5.40.3.4 getIndex()

```
size_t base_local_planner::MapGrid::getIndex (
            int x,
            int y )
```

Returns a 1D index into the MapCell array for a 2D index

参数

x	The desired x coordinate
y	The desired y coordinate

返回

The associated 1D index

5.40.3.5 obstacleCosts()

```
double base_local_planner::MapGrid::obstacleCosts ( ) [inline]
```

return a value that indicates cell is in obstacle

5.40.3.6 operator()() [1/2]

```
MapCell& base_local_planner::MapGrid::operator() (
            unsigned int x,
            unsigned int y ) [inline]
```

Returns a map cell accessed by (col, row)

参数

x	The x coordinate of the cell
y	The y coordinate of the cell

返回

A reference to the desired cell

5.40.3.7 operator()() [2/2]

```
MapCell base_local_planner::MapGrid::operator() (
             unsigned int x,
             unsigned int y ) const  [inline]
```

Returns a map cell accessed by (col, row)

参数

x	The x coordinate of the cell
y	The y coordinate of the cell

返回

A copy of the desired cell

5.40.3.8 operator=()

```
MapGrid& base_local_planner::MapGrid::operator= (
             const MapGrid & mg )
```

Assignment operator for a MapGrid

参数

mg	The MapGrid to assign from

5.40.3.9 sizeCheck()

```
void base_local_planner::MapGrid::sizeCheck (
             unsigned int size_x,
             unsigned int size_y )
```

check if we need to resize

参数

size↩_x	The desired width
size↩_y	The desired height

5.40.3.10 unreachableCellCosts()

```
double base_local_planner::MapGrid::unreachableCellCosts ( )  [inline]
```

returns a value indicating cell was not reached by wavefront propagation of set cells. (is behind walls, regarding the region covered by grid)

5.40.3.11 updatePathCell()

```
bool base_local_planner::MapGrid::updatePathCell (
            MapCell * current_cell,
            MapCell * check_cell,
            const costmap_2d::Costmap2D & costmap )  [inline]
```

Used to update the distance of a cell in path distance computation

参数

current_cell	The cell we're currently in
check_cell	The cell to be updated

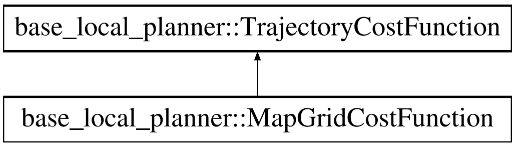
该类的文档由以下文件生成:

- base_local_planner/include/base_local_planner/map_grid.h

5.41 base_local_planner::MapGridCostFunction类 参考

```
#include <map_grid_cost_function.h>
```

类 base_local_planner::MapGridCostFunction 继承关系图:


base_local_planner::TrajectoryCostFunction
base_local_planner::MapGridCostFunction

Public 成员函数

- **MapGridCostFunction** (costmap_2d::Costmap2D ∗costmap, double xshift=0.0, double yshift=0.0, bool is←_local_goal_function=false, CostAggregationType aggregationType=Last)
- void setTargetPoses (std::vector< geometry_msgs::PoseStamped > target_poses)
- void **setXShift** (double xshift)
- void **setYShift** (double yshift)
- void setStopOnFailure (bool stop_on_failure)

  If true, failures along the path cause the entire path to be rejected.
- bool prepare ()
- double scoreTrajectory (Trajectory &traj)
- double obstacleCosts ()
- double unreachableCellCosts ()
- double **getCellCosts** (unsigned int cx, unsigned int cy)

额外继承的成员函数

5.41.1 详细描述

This class provides cost based on a map_grid of a small area of the world. The map_grid covers a the costmap, the costmap containing the information about sensed obstacles. The map_grid is used by setting certain cells to distance 0, and then propagating distances around them, filling up the area reachable around them.

The approach using grid_maps is used for computational efficiency, allowing to score hundreds of trajectories very quickly.

This can be used to favor trajectories which stay on a given path, or which approach a given goal.

参数

costmap_ros	Reference to object giving updates of obstacles around robot
xshift	where the scoring point is with respect to robot center pose
yshift	where the scoring point is with respect to robot center pose
is_local_goal_function,scores	for local goal rather than whole path
aggregationType	how to combine costs along trajectory

5.41.2 成员函数说明

5.41.2.1 obstacleCosts()

```
double base_local_planner::MapGridCostFunction::obstacleCosts ( )  [inline]
```

return a value that indicates cell is in obstacle

5.41.2.2 prepare()

```
bool base_local_planner::MapGridCostFunction::prepare ( )  [virtual]
```

propagate distances

实现了 base_local_planner::TrajectoryCostFunction.

5.41.2.3 scoreTrajectory()

```
double base_local_planner::MapGridCostFunction::scoreTrajectory (
             Trajectory & traj )  [virtual]
```

return a score for trajectory traj

实现了 base_local_planner::TrajectoryCostFunction.

5.41.2.4 setStopOnFailure()

```
void base_local_planner::MapGridCostFunction::setStopOnFailure (
             bool stop_on_failure )  [inline]
```

If true, failures along the path cause the entire path to be rejected.

Default is true.

5.41.2.5 setTargetPoses()

```
void base_local_planner::MapGridCostFunction::setTargetPoses (
             std::vector< geometry_msgs::PoseStamped > target_poses )
```

set line segments on the grid with distance 0, resets the grid

5.41.2.6 unreachableCellCosts()

```
double base_local_planner::MapGridCostFunction::unreachableCellCosts ( )  [inline]
```

returns a value indicating cell was not reached by wavefront propagation of set cells. (is behind walls, regarding the region covered by grid)

该类的文档由以下文件生成:

- base_local_planner/include/base_local_planner/map_grid_cost_function.h

5.42 base_local_planner::MapGridVisualizer类 参考

Public 成员函数

- MapGridVisualizer ()

  Default constructor
- void initialize (const std::string &name, std::string frame, boost::function< bool(int cx, int cy, float &path_cost, float &goal_cost, float &occ_cost, float &total_cost)> cost_function)

  Initializes the MapGridVisualizer
- void publishCostCloud (const costmap_2d::Costmap2D ∗costmap_p_)

  Build and publish a PointCloud if the publish_cost_grid_pc parameter was true. Only include points for which the cost_function at (cx,cy) returns true.

5.42.1 成员函数说明

5.42.1.1 initialize()

```
void base_local_planner::MapGridVisualizer::initialize (
             const std::string & name,
             std::string frame,
             boost::function< bool(int cx, int cy, float &path_cost, float &goal_cost, float
&occ_cost, float &total_cost)> cost_function )
```

Initializes the MapGridVisualizer

参数

name	The name to be appended to ∼/ in order to get the proper namespace for parameters
costmap	The costmap instance to use to get the size of the map to generate a point cloud for
cost_function	The function to use to compute the cost values to be inserted into each point in the output PointCloud as well as whether to include a given point or not

该类的文档由以下文件生成:

- base_local_planner/include/base_local_planner/map_grid_visualizer.h

5.43 costmap_2d::MapLocation结构体 参考

成员变量

- unsigned int **x**
- unsigned int **y**

该结构体的文档由以下文件生成:

- costmap_2d/include/costmap_2d/costmap_2d.h

5.44 costmap_2d::Costmap2D::MarkCell类 参考

Public 成员函数

- **MarkCell** (unsigned char ∗costmap, unsigned char value)
- void **operator()** (unsigned int offset)

该类的文档由以下文件生成:

- costmap_2d/include/costmap_2d/costmap_2d.h

5.45 move_base::MoveBase类 参考

A class that uses the actionlib::ActionServer interface that moves the robot base to a goal location.

```
#include <move_base.h>
```

Public 成员函数

- MoveBase (tf2_ros::Buffer &tf)

  Constructor for the actions
- virtual ∼MoveBase ()

  Destructor - Cleans up
- bool executeCycle (geometry_msgs::PoseStamped &goal, std::vector< geometry_msgs::PoseStamped > &global_plan)

  Performs a control cycle

5.45.1 详细描述

A class that uses the actionlib::ActionServer interface that moves the robot base to a goal location.

5.45.2 构造及析构函数说明

5.45.2.1 MoveBase()

```
move_base::MoveBase::MoveBase (
            tf2_ros::Buffer & tf )
```

Constructor for the actions

参数

name	The name olocal_plannerf the action
tf	A reference to a TransformListener

5.45.3 成员函数说明

5.45.3.1 executeCycle()

```
bool move_base::MoveBase::executeCycle (
            geometry_msgs::PoseStamped & goal,
            std::vector< geometry_msgs::PoseStamped > & global_plan )
```

Performs a control cycle

参数

goal	A reference to the goal to pursue
global_plan	A reference to the global plan being used

返回

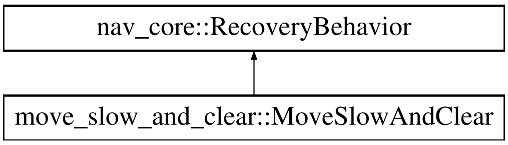
True if processing of the goal is done, false otherwise

该类的文档由以下文件生成:

- move_base/include/move_base/move_base.h

5.46 move_slow_and_clear::MoveSlowAndClear类 参考

类 move_slow_and_clear::MoveSlowAndClear 继承关系图:


nav_core::RecoveryBehavior
move_slow_and_clear::MoveSlowAndClear

Public 成员函数

- void initialize (std::string n, tf2_ros::Buffer ∗tf, costmap_2d::Costmap2DROS ∗global_costmap, costmap_2d::Costmap2DROS ∗local_costmap)

    Initialize the parameters of the behavior
- void runBehavior ()

    Run the behavior

该类的文档由以下文件生成:

- move_slow_and_clear/include/move_slow_and_clear/move_slow_and_clear.h

5.47 navfn::NavFn类 参考

Navigation function class. Holds buffers for costmap, navfn map. Maps are pixel-based. Origin is upper left, x is right, y is down.

```
#include <navfn.h>
```

Public 成员函数

- NavFn (int nx, int ny)

    Constructs the planner
- void setNavArr (int nx, int ny)

    Sets or resets the size of the map
- void setCostmap (const COSTTYPE ∗cmap, bool isROS=true, bool allow_unknown=true)

    Set up the cost array for the planner, usually from ROS
- bool calcNavFnAstar ()

    Calculates a plan using the A∗ heuristic, returns true if one is found
- bool calcNavFnDijkstra (bool atStart=false)

    Caclulates the full navigation function using Dijkstra
- float ∗ getPathX ()

    Accessor for the x-coordinates of a path
- float ∗ getPathY ()

    Accessor for the y-coordinates of a path
- int getPathLen ()

    Accessor for the length of a path
- float getLastPathCost ()

    Gets the cost of the path found the last time a navigation function was computed
- void setGoal (int ∗goal)

    Sets the goal position for the planner. Note: the navigation cost field computed gives the cost to get to a given point from the goal, not from the start.
- void setStart (int ∗start)

    Sets the start position for the planner. Note: the navigation cost field computed gives the cost to get to a given point from the goal, not from the start.
- void initCost (int k, float v)

    Initialize cell k with cost v for propagation
- void updateCell (int n)

Updates the cell at index n

- void updateCellAstar (int n)

    Updates the cell at index n using the A∗ heuristic
- void setupNavFn (bool keepit=false)
- bool propNavFnDijkstra (int cycles, bool atStart=false)

    Run propagation for <cycles> iterations, or until start is reached using breadth-first Dijkstra method
- bool propNavFnAstar (int cycles)

    Run propagation for <cycles> iterations, or until start is reached using the best-first A∗ method with Euclidean distance heuristic
- int calcPath (int n, int ∗st=NULL)

    Calculates the path for at mose <n> cycles
- float gradCell (int n)
- void display (void fn(NavFn ∗nav), int n=100)
- void savemap (const char ∗fname)

成员变量

- int **nx**
- int **ny**
- int ns
- COSTTYPE ∗ costarr
- float ∗ potarr
- bool ∗ pending
- int nobs
- int ∗ pb1
- int ∗ **pb2**
- int ∗ pb3
- int ∗ **curP**
- int ∗ **nextP**
- int ∗ overP
- int **curPe**
- int **nextPe**
- int overPe
- float curT
- float priInc
- int **goal** [2]
- int **start** [2]
- float ∗ gradx
- float ∗ grady
- float ∗ **pathx**
- float ∗ pathy
- int npath
- int npathbuf
- float last_path_cost_
- float pathStep
- int displayInt
- void(∗ displayFn)(NavFn ∗nav)

5.47.1 详细描述

Navigation function class. Holds buffers for costmap, navfn map. Maps are pixel-based. Origin is upper left, x is right, y is down.

5.47.2 构造及析构函数说明

5.47.2.1 NavFn()

```
navfn::NavFn::NavFn (
             int nx,
             int ny )
```

Constructs the planner

参数

nx	The x size of the map
ny	The y size of the map

5.47.3 成员函数说明

5.47.3.1 calcNavFnAstar()

```
bool navfn::NavFn::calcNavFnAstar ( )
```

Calculates a plan using the A∗ heuristic, returns true if one is found

返回

True if a plan is found, false otherwise calculates a plan, returns true if found

5.47.3.2 calcNavFnDijkstra()

```
bool navfn::NavFn::calcNavFnDijkstra (
             bool atStart = false )
```

Caclulates the full navigation function using Dijkstra

calculates the full navigation function

5.47.3.3 calcPath()

```
int navfn::NavFn::calcPath (
             int n,
             int * st = NULL )
```

Calculates the path for at mose <n> cycles

参数

n	The maximum number of cycles to run for

返回

The length of the path found calculates path for at most <n> cycles, returns path length, 0 if none

5.47.3.4 display()

```
void navfn::NavFn::display (
             void  fnNavFn *nav,
             int n = 100 )
```

display callback <n> is the number of cycles between updates

5.47.3.5 getLastPathCost()

```
float navfn::NavFn::getLastPathCost ( )
```

Gets the cost of the path found the last time a navigation function was computed

返回

The cost of the last path found Return cost of path found the last time A∗ was called

5.47.3.6 getPathLen()

```
int navfn::NavFn::getPathLen ( )
```

Accessor for the length of a path

返回

The length of a path length of path, 0 if not found

5.47.3.7 getPathX()

```
float* navfn::NavFn::getPathX ( )
```

Accessor for the x-coordinates of a path

返回

The x-coordinates of a path x-coordinates of path

5.47.3.8 getPathY()

```
float* navfn::NavFn::getPathY ( )
```

Accessor for the y-coordinates of a path

返回

The y-coordinates of a path y-coordinates of path

5.47.3.9 gradCell()

```
float navfn::NavFn::gradCell (
             int n )
```

calculates gradient at cell <n>, returns norm

5.47.3.10 initCost()

```
void navfn::NavFn::initCost (
             int k,
             float v )
```

Initialize cell k with cost v for propagation

参数

k	the cell to initialize
v	the cost to give to the cell initialize cell <k> with cost <v>, for propagation

5.47.3.11 propNavFnAstar()

```
bool navfn::NavFn::propNavFnAstar (
            int cycles )
```

Run propagation for <cycles> iterations, or until start is reached using the best-first A∗ method with Euclidean distance heuristic

参数

cycles	The maximum number of iterations to run for

返回

true if the start point is reached returns true if start point found

5.47.3.12 propNavFnDijkstra()

```
bool navfn::NavFn::propNavFnDijkstra (
            int cycles,
            bool atStart = false )
```

Run propagation for <cycles> iterations, or until start is reached using breadth-first Dijkstra method

参数

cycles	The maximum number of iterations to run for
atStart	Whether or not to stop when the start point is reached

返回

true if the start point is reached returns true if start point found or full prop

5.47.3.13 savemap()

```
void navfn::NavFn::savemap (
            const char * fname )
```

save costmap write out costmap and start/goal states as fname.pgm and fname.txt

5.47.3.14 setCostmap()

```
void navfn::NavFn::setCostmap (
             const COSTTYPE * cmap,
             bool isROS = true,
             bool allow_unknown = true )
```

Set up the cost array for the planner, usually from ROS

参数

cmap	The costmap
isROS	Whether or not the costmap is coming in in ROS format
allow_unknown	Whether or not the planner should be allowed to plan through unknown space sets up the cost map

5.47.3.15 setGoal()

```
void navfn::NavFn::setGoal (
            int * goal )
```

Sets the goal position for the planner. Note: the navigation cost field computed gives the cost to get to a given point from the goal, not from the start.

goal and start positions

参数

goal	the goal position

5.47.3.16 setNavArr()

```
void navfn::NavFn::setNavArr (
            int nx,
            int ny )
```

Sets or resets the size of the map

参数

nx	The x size of the map
ny	The y size of the map sets or resets the size of the map

5.47.3.17 setStart()

```
void navfn::NavFn::setStart (
            int * start )
```

Sets the start position for the planner. Note: the navigation cost field computed gives the cost to get to a given point from the goal, not from the start.

参数

start	the start position

5.47.3.18 setupNavFn()

```
void navfn::NavFn::setupNavFn (
            bool keepit = false )
```

resets all nav fn arrays for propagation

5.47.3.19 updateCell()

```
void navfn::NavFn::updateCell (
            int n )
```

Updates the cell at index n

propagation

参数

n	The index to update updates the cell at index <n>

5.47.3.20 updateCellAstar()

```
void navfn::NavFn::updateCellAstar (
            int n )
```

Updates the cell at index n using the A∗ heuristic

参数

n	The index to update updates the cell at index <n>, uses A∗ heuristic

5.47.4 结构体成员变量说明

5.47.4.1 costarr

```
COSTTYPE* navfn::NavFn::costarr
```

cell arrays cost array in 2D configuration space

5.47.4.2 curT

```
float navfn::NavFn::curT
```

block priority thresholds current threshold

5.47.4.3 displayFn

```
void(* navfn::NavFn::displayFn) (NavFn *nav)
```

display function itself

5.47.4.4 displayInt

```
int navfn::NavFn::displayInt
```

save second argument of display() above

5.47.4.5 gradx

```
float* navfn::NavFn::gradx
```

gradient and paths

5.47.4.6 grady

```
float * navfn::NavFn::grady
```

gradient arrays, size of potential array

5.47.4.7 last_path_cost_

```
float navfn::NavFn::last_path_cost_
```

Holds the cost of the path found the last time A* was called

5.47.4.8 nobs

```
int navfn::NavFn::nobs
```

number of obstacle cells

5.47.4.9 npath

```
int navfn::NavFn::npath
```

number of path points

5.47.4.10 npathbuf

```
int navfn::NavFn::npathbuf
```

size of pathx, pathy buffers

5.47.4.11 ns

```
int navfn::NavFn::ns
```

size of grid, in pixels

5.47.4.12 overP

```
int * navfn::NavFn::overP
```

priority buffer block ptrs

5.47.4.13 overPe

```
int navfn::NavFn::overPe
```

end points of arrays

5.47.4.14 pathStep

```
float navfn::NavFn::pathStep
```

step size for following gradient

5.47.4.15 pathy

```
float * navfn::NavFn::pathy
```

path points, as subpixel cell coordinates

5.47.4.16 pb1

```
int* navfn::NavFn::pb1
```

block priority buffers

5.47.4.17 pb3

```
int * navfn::NavFn::pb3
```

storage buffers for priority blocks

5.47.4.18 pending

```
bool* navfn::NavFn::pending
```

pending cells during propagation

5.47.4.19 potarr

```
float* navfn::NavFn::potarr
```

potential array, navigation function potential

5.47.4.20 priInc

```
float navfn::NavFn::priInc
```

priority threshold increment

该类的文档由以下文件生成:

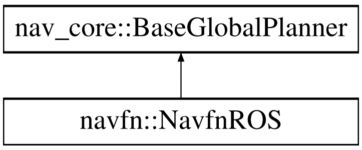
- navfn/include/navfn/navfn.h

5.48 navfn::NavfnROS类 参考

Provides a ROS wrapper for the navfn planner which runs a fast, interpolated navigation function on a costmap.

```
#include <navfn_ros.h>
```

类 navfn::NavfnROS 继承关系图:



Public 成员函数

- NavfnROS ()

    Default constructor for the NavFnROS object
- NavfnROS (std::string name, costmap_2d::Costmap2DROS ∗costmap_ros)

    Constructor for the NavFnROS object
- NavfnROS (std::string name, costmap_2d::Costmap2D ∗costmap, std::string global_frame)

    Constructor for the NavFnROS object
- void initialize (std::string name, costmap_2d::Costmap2DROS ∗costmap_ros)

    Initialization function for the NavFnROS object
- void initialize (std::string name, costmap_2d::Costmap2D ∗costmap, std::string global_frame)

    Initialization function for the NavFnROS object
- bool makePlan (const geometry_msgs::PoseStamped &start, const geometry_msgs::PoseStamped &goal, std::vector< geometry_msgs::PoseStamped > &plan)

    Given a goal pose in the world, compute a plan
- bool makePlan (const geometry_msgs::PoseStamped &start, const geometry_msgs::PoseStamped &goal, double tolerance, std::vector< geometry_msgs::PoseStamped > &plan)

    Given a goal pose in the world, compute a plan
- bool computePotential (const geometry_msgs::Point &world_point)

    Computes the full navigation function for the map given a point in the world to start from
- bool getPlanFromPotential (const geometry_msgs::PoseStamped &goal, std::vector< geometry_msgs::↩ PoseStamped > &plan)

    Compute a plan to a goal after the potential for a start point has already been computed (Note: You should call computePotential first)
- double getPointPotential (const geometry_msgs::Point &world_point)

    Get the potential, or naviagation cost, at a given point in the world (Note: You should call computePotential first)
- bool validPointPotential (const geometry_msgs::Point &world_point)

    Check for a valid potential value at a given point in the world (Note: You should call computePotential first)
- bool validPointPotential (const geometry_msgs::Point &world_point, double tolerance)

    Check for a valid potential value at a given point in the world (Note: You should call computePotential first)
- void publishPlan (const std::vector< geometry_msgs::PoseStamped > &path, double r, double g, double b, double a)

    Publish a path for visualization purposes
- bool **makePlanService** (nav_msgs::GetPlan::Request &req, nav_msgs::GetPlan::Response &resp)

Protected 属性

- costmap_2d::Costmap2D ∗ costmap_

    Store a copy of the current costmap in costmap. Called by makePlan.
- boost::shared_ptr< NavFn > **planner_**
- ros::Publisher **plan_pub_**
- ros::Publisher **potarr_pub_**
- bool **initialized_**
- bool **allow_unknown_**
- bool **visualize_potential_**

5.48.1 详细描述

Provides a ROS wrapper for the navfn planner which runs a fast, interpolated navigation function on a costmap.

5.48.2 构造及析构函数说明

5.48.2.1 NavfnROS() [1/2]

```
navfn::NavfnROS::NavfnROS (
            std::string name,
            costmap_2d::Costmap2DROS * costmap_ros )
```

Constructor for the NavFnROS object

参数

name	The name of this planner
costmap	A pointer to the ROS wrapper of the costmap to use

5.48.2.2 NavfnROS() [2/2]

```
navfn::NavfnROS::NavfnROS (
            std::string name,
            costmap_2d::Costmap2D * costmap,
            std::string global_frame )
```

Constructor for the NavFnROS object

参数

name	The name of this planner
costmap	A pointer to the costmap to use
global_frame	The global frame of the costmap

5.48.3 成员函数说明

5.48.3.1 computePotential()

```
bool navfn::NavfnROS::computePotential (
            const geometry_msgs::Point & world_point )
```

Computes the full navigation function for the map given a point in the world to start from

参数

world_point	The point to use for seeding the navigation function

返回

True if the navigation function was computed successfully, false otherwise

5.48.3.2 getPlanFromPotential()

```
bool navfn::NavfnROS::getPlanFromPotential (
             const geometry_msgs::PoseStamped & goal,
             std::vector< geometry_msgs::PoseStamped > & plan )
```

Compute a plan to a goal after the potential for a start point has already been computed (Note: You should call computePotential first)

参数

goal	The goal pose to create a plan to
plan	The plan... filled by the planner

返回

True if a valid plan was found, false otherwise

5.48.3.3 getPointPotential()

```
double navfn::NavfnROS::getPointPotential (
             const geometry_msgs::Point & world_point )
```

Get the potential, or naviagation cost, at a given point in the world (Note: You should call computePotential first)

参数

world_point	The point to get the potential for

返回

The navigation function's value at that point in the world

5.48.3.4 initialize() [1/2]

```
void navfn::NavfnROS::initialize (
             std::string name,
             costmap_2d::Costmap2D * costmap,
             std::string global_frame )
```

Initialization function for the NavFnROS object

参数

name	The name of this planner
costmap	A pointer to the costmap to use for planning
global_frame	The global frame of the costmap

5.48.3.5 initialize() [2/2]

```
void navfn::NavfnROS::initialize (
             std::string name,
             costmap_2d::Costmap2DROS * costmap_ros )   [virtual]
```

Initialization function for the NavFnROS object

参数

name	The name of this planner
costmap	A pointer to the ROS wrapper of the costmap to use for planning

实现了 nav_core::BaseGlobalPlanner.

5.48.3.6 makePlan() [1/2]

```
bool navfn::NavfnROS::makePlan (
             const geometry_msgs::PoseStamped & start,
             const geometry_msgs::PoseStamped & goal,
             double tolerance,
             std::vector< geometry_msgs::PoseStamped > & plan )
```

Given a goal pose in the world, compute a plan

参数

start	The start pose
goal	The goal pose
tolerance	The tolerance on the goal point for the planner
plan	The plan... filled by the planner

返回

True if a valid plan was found, false otherwise

5.48.3.7 makePlan() [2/2]

```
bool navfn::NavfnROS::makePlan (
            const geometry_msgs::PoseStamped & start,
            const geometry_msgs::PoseStamped & goal,
            std::vector< geometry_msgs::PoseStamped > & plan )  [virtual]
```

Given a goal pose in the world, compute a plan

参数

start	The start pose
goal	The goal pose
plan	The plan... filled by the planner

返回

True if a valid plan was found, false otherwise

实现了 nav_core::BaseGlobalPlanner.

5.48.3.8 validPointPotential() [1/2]

```
bool navfn::NavfnROS::validPointPotential (
            const geometry_msgs::Point & world_point )
```

Check for a valid potential value at a given point in the world (Note: You should call computePotential first)

参数

world_point	The point to get the potential for

返回

True if the navigation function is valid at that point in the world, false otherwise

5.48.3.9 validPointPotential() [2/2]

```
bool navfn::NavfnROS::validPointPotential (
            const geometry_msgs::Point & world_point,
            double tolerance )
```

Check for a valid potential value at a given point in the world (Note: You should call computePotential first)

参数

world_point	The point to get the potential for
tolerance	The tolerance on searching around the world_point specified

返回

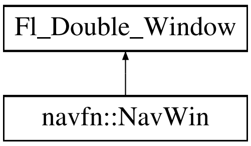
True if the navigation function is valid at that point in the world, false otherwise

该类的文档由以下文件生成:

- navfn/include/navfn/navfn_ros.h

5.49 navfn::NavWin类 参考

类 navfn::NavWin 继承关系图:



Public 成员函数

- **NavWin** (int w, int h, const char ∗name)
- void **drawPot** (NavFn ∗nav)
- void **drawOverlay** ()
- void **draw** ()

成员变量

- int **nw**
- int **nh**
- int **pw**
- int **ph**
- int **dec**
- int **inc**
- float **maxval**
- uchar ∗ **im**

- int ∗ **pc**
- int ∗ **pn**
- int ∗ **po**
- int **pce**
- int **pne**
- int **poe**
- int **goal** [2]
- int **start** [2]
- int ∗ **path**
- int **pathlen**
- int **pathbuflen**

该类的文档由以下文件生成:

- navfn/include/navfn/navwin.h

5.50 costmap_2d::Observation类 参考

Stores an observation in terms of a point cloud and the origin of the source

```
#include <observation.h>
```

Public 成员函数

- Observation ()

  Creates an empty observation
- Observation (geometry_msgs::Point &origin, const sensor_msgs::PointCloud2 &cloud, double obstacle_range, double raytrace_range)

  Creates an observation from an origin point and a point cloud
- Observation (const Observation &obs)

  Copy constructor
- Observation (const sensor_msgs::PointCloud2 &cloud, double obstacle_range)

  Creates an observation from a point cloud

成员变量

- geometry_msgs::Point **origin_**
- sensor_msgs::PointCloud2 ∗ **cloud_**
- double **obstacle_range_**
- double **raytrace_range_**

5.50.1 详细描述

Stores an observation in terms of a point cloud and the origin of the source

注解

> Tried to make members and constructor arguments const but the compiler would not accept the default assignment operator for vector insertion!

5.50.2 构造及析构函数说明

5.50.2.1 Observation() [1/3]

```
costmap_2d::Observation::Observation (
            geometry_msgs::Point & origin,
            const sensor_msgs::PointCloud2 & cloud,
            double obstacle_range,
            double raytrace_range )  [inline]
```

Creates an observation from an origin point and a point cloud

参数

origin	The origin point of the observation
cloud	The point cloud of the observation
obstacle_range	The range out to which an observation should be able to insert obstacles
raytrace_range	The range out to which an observation should be able to clear via raytracing

5.50.2.2 Observation() [2/3]

```
costmap_2d::Observation::Observation (
            const Observation & obs )  [inline]
```

Copy constructor

参数

obs	The observation to copy

5.50.2.3 Observation() [3/3]

```
costmap_2d::Observation::Observation (
            const sensor_msgs::PointCloud2 & cloud,
            double obstacle_range )  [inline]
```

Creates an observation from a point cloud

参数

cloud	The point cloud of the observation
obstacle_range	The range out to which an observation should be able to insert obstacles

该类的文档由以下文件生成:

- costmap_2d/include/costmap_2d/observation.h

5.51 costmap_2d::ObservationBuffer类 参考

Takes in point clouds from sensors, transforms them to the desired frame, and stores them

```
#include <observation_buffer.h>
```

Public 成员函数

- ObservationBuffer (std::string topic_name, double observation_keep_time, double expected_update_rate, double min_obstacle_height, double max_obstacle_height, double obstacle_range, double raytrace_range, tf2_←ros::Buffer &tf2_buffer, std::string global_frame, std::string sensor_frame, double tf_tolerance)

  Constructs an observation buffer
- ∼ObservationBuffer ()

  Destructor... cleans up
- bool setGlobalFrame (const std::string new_global_frame)

  Sets the global frame of an observation buffer. This will transform all the currently cached observations to the new global frame
- void bufferCloud (const sensor_msgs::PointCloud2 &cloud)

  Transforms a PointCloud to the global frame and buffers it ***Note: The burden is on the user to make sure the transform is available... ie they should use a MessageNotifier***
- void getObservations (std::vector< Observation > &observations)

  Pushes copies of all current observations onto the end of the vector passed in
- bool isCurrent () const

  Check if the observation buffer is being update at its expected rate
- void lock ()

  Lock the observation buffer
- void unlock ()

  Lock the observation buffer
- void resetLastUpdated ()

  Reset last updated timestamp

5.51.1 详细描述

Takes in point clouds from sensors, transforms them to the desired frame, and stores them

5.51.2 构造及析构函数说明

5.51.2.1 ObservationBuffer()

```
costmap_2d::ObservationBuffer::ObservationBuffer (
            std::string topic_name,
            double observation_keep_time,
            double expected_update_rate,
            double min_obstacle_height,
            double max_obstacle_height,
            double obstacle_range,
            double raytrace_range,
            tf2_ros::Buffer & tf2_buffer,
            std::string global_frame,
            std::string sensor_frame,
            double tf_tolerance )
```

Constructs an observation buffer

参数

topic_name	The topic of the observations, used as an identifier for error and warning messages
observation_keep_time	Defines the persistence of observations in seconds, 0 means only keep the latest
expected_update_rate	How often this buffer is expected to be updated, 0 means there is no limit
min_obstacle_height	The minimum height of a hitpoint to be considered legal
max_obstacle_height	The minimum height of a hitpoint to be considered legal
obstacle_range	The range to which the sensor should be trusted for inserting obstacles
raytrace_range	The range to which the sensor should be trusted for raytracing to clear out space
tf2_buffer	A reference to a tf2 Buffer
global_frame	The frame to transform PointClouds into
sensor_frame	The frame of the origin of the sensor, can be left blank to be read from the messages
tf_tolerance	The amount of time to wait for a transform to be available when setting a new global frame

5.51.3 成员函数说明

5.51.3.1 bufferCloud()

```
void costmap_2d::ObservationBuffer::bufferCloud (
            const sensor_msgs::PointCloud2 & cloud )
```

Transforms a PointCloud to the global frame and buffers it **Note: The burden is on the user to make sure the transform is available... ie they should use a MessageNotifier**

参数

cloud	The cloud to be buffered

5.51.3.2 getObservations()

```
void costmap_2d::ObservationBuffer::getObservations (
            std::vector< Observation > & observations )
```

Pushes copies of all current observations onto the end of the vector passed in

参数

observations	The vector to be filled

5.51.3.3 isCurrent()

```
bool costmap_2d::ObservationBuffer::isCurrent ( ) const
```

Check if the observation buffer is being update at its expected rate

返回

True if it is being updated at the expected rate, false otherwise

5.51.3.4 setGlobalFrame()

```
bool costmap_2d::ObservationBuffer::setGlobalFrame (
            const std::string new_global_frame )
```

Sets the global frame of an observation buffer. This will transform all the currently cached observations to the new global frame

参数

new_global_frame	The name of the new global frame.

返回

True if the operation succeeds, false otherwise

该类的文档由以下文件生成:

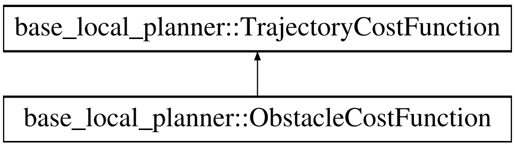
- costmap_2d/include/costmap_2d/observation_buffer.h

5.52 base_local_planner::ObstacleCostFunction类 参考

Uses costmap 2d to assign negative costs if robot footprint is in obstacle on any point of the trajectory.

```
#include <obstacle_cost_function.h>
```

类 base_local_planner::ObstacleCostFunction 继承关系图:


base_local_planner::TrajectoryCostFunction
base_local_planner::ObstacleCostFunction

Public 成员函数

- **ObstacleCostFunction** (costmap_2d::Costmap2D *costmap)
- bool prepare ()
- double scoreTrajectory (Trajectory &traj)
- void **setSumScores** (bool score_sums)
- void **setParams** (double max_trans_vel, double max_scaling_factor, double scaling_speed)
- void **setFootprint** (std::vector< geometry_msgs::Point > footprint_spec)

静态 Public 成员函数

- static double **getScalingFactor** (Trajectory &traj, double scaling_speed, double max_trans_vel, double max_scaling_factor)
- static double **footprintCost** (const double &x, const double &y, const double &th, double scale, std::vector< geometry_msgs::Point > footprint_spec, costmap_2d::Costmap2D *costmap, base_local_planner::WorldModel *world_model)

额外继承的成员函数

5.52.1 详细描述

Uses costmap 2d to assign negative costs if robot footprint is in obstacle on any point of the trajectory.

class ObstacleCostFunction

5.52.2 成员函数说明

5.52.2.1 prepare()

```
bool base_local_planner::ObstacleCostFunction::prepare ( )  [virtual]
```

General updating of context values if required. Subclasses may overwrite. Return false in case there is any error.

实现了 base_local_planner::TrajectoryCostFunction.

5.52.2.2 scoreTrajectory()

```
double base_local_planner::ObstacleCostFunction::scoreTrajectory (
            Trajectory & traj )  [virtual]
```

return a score for trajectory traj

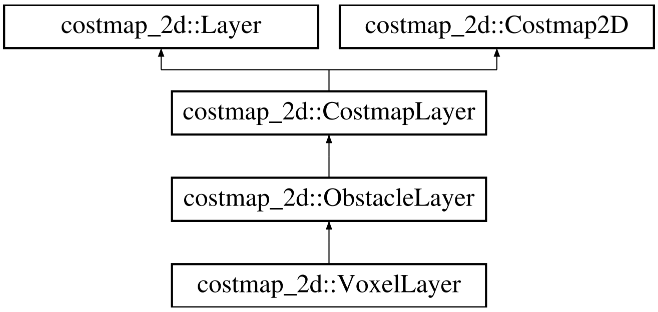
实现了 base_local_planner::TrajectoryCostFunction.

该类的文档由以下文件生成:

- base_local_planner/include/base_local_planner/obstacle_cost_function.h

5.53 costmap_2d::ObstacleLayer类 参考

类 costmap_2d::ObstacleLayer 继承关系图:



Public 成员函数

- virtual void onInitialize ()

  This is called at the end of initialize(). Override to implement subclass-specific initialization.
- virtual void updateBounds (double robot_x, double robot_y, double robot_yaw, double ∗min_x, double ∗min_y, double ∗max_x, double ∗max_y)

  This is called by the LayeredCostmap to poll this plugin as to how much of the costmap it needs to update. Each layer can increase the size of this bounds.
- virtual void updateCosts (costmap_2d::Costmap2D &master_grid, int min_i, int min_j, int max_i, int max_j)

  Actually update the underlying costmap, only within the bounds calculated during UpdateBounds().
- virtual void activate ()

  Restart publishers if they've been stopped.

- virtual void deactivate ()

  Stop publishers.
- virtual void **reset** ()
- void laserScanCallback (const sensor_msgs::LaserScanConstPtr &message, const boost::shared_ptr< costmap_2d::ObservationBuffer > &buffer)

  A callback to handle buffering LaserScan messages
- void laserScanValidInfCallback (const sensor_msgs::LaserScanConstPtr &message, const boost::shared_↩ ptr< ObservationBuffer > &buffer)

  A callback to handle buffering LaserScan messages which need filtering to turn Inf values into range_max.
- void pointCloudCallback (const sensor_msgs::PointCloudConstPtr &message, const boost::shared_ptr< costmap_2d::ObservationBuffer > &buffer)

  A callback to handle buffering PointCloud messages
- void pointCloud2Callback (const sensor_msgs::PointCloud2ConstPtr &message, const boost::shared_ptr< costmap_2d::ObservationBuffer > &buffer)

  A callback to handle buffering PointCloud2 messages
- void **addStaticObservation** (costmap_2d::Observation &obs, bool marking, bool clearing)
- void **clearStaticObservations** (bool marking, bool clearing)

Protected 成员函数

- virtual void **setupDynamicReconfigure** (ros::NodeHandle &nh)
- bool getMarkingObservations (std::vector< costmap_2d::Observation > &marking_observations) const

  Get the observations used to mark space
- bool getClearingObservations (std::vector< costmap_2d::Observation > &clearing_observations) const

  Get the observations used to clear space
- virtual void raytraceFreespace (const costmap_2d::Observation &clearing_observation, double ∗min_x, double ∗min_y, double ∗max_x, double ∗max_y)

  Clear freespace based on one observation
- void **updateRaytraceBounds** (double ox, double oy, double wx, double wy, double range, double ∗min_x, double ∗min_y, double ∗max_x, double ∗max_y)
- void **updateFootprint** (double robot_x, double robot_y, double robot_yaw, double ∗min_x, double ∗min_y, double ∗max_x, double ∗max_y)

Protected 属性

- std::vector< geometry_msgs::Point > **transformed_footprint_**
- bool **footprint_clearing_enabled_**
- std::string global_frame_

  The global frame for the costmap
- double max_obstacle_height_

  Max Obstacle Height
- laser_geometry::LaserProjection projector_

  Used to project laser scans into point clouds
- std::vector< boost::shared_ptr< message_filters::SubscriberBase > > observation_subscribers_

  Used for the observation message filters
- std::vector< boost::shared_ptr< tf2_ros::MessageFilterBase > > observation_notifiers_

  Used to make sure that transforms are available for each sensor
- std::vector< boost::shared_ptr< costmap_2d::ObservationBuffer > > observation_buffers_

  Used to store observations from various sensors
- std::vector< boost::shared_ptr< costmap_2d::ObservationBuffer > > marking_buffers_

  Used to store observation buffers used for marking obstacles

- std::vector< boost::shared_ptr< costmap_2d::ObservationBuffer > > clearing_buffers_

  Used to store observation buffers used for clearing obstacles
- std::vector< costmap_2d::Observation > **static_clearing_observations_**
- std::vector< costmap_2d::Observation > **static_marking_observations_**
- bool **rolling_window_**
- dynamic_reconfigure::Server< costmap_2d::ObstaclePluginConfig > ∗ **dsrv_**
- int **combination_method_**

额外继承的成员函数

5.53.1 成员函数说明

5.53.1.1 getClearingObservations()

```
bool costmap_2d::ObstacleLayer::getClearingObservations (
            std::vector< costmap_2d::Observation > & clearing_observations ) const  [protected]
```

Get the observations used to clear space

参数

clearing_observations	A reference to a vector that will be populated with the observations

返回

True if all the observation buffers are current, false otherwise

5.53.1.2 getMarkingObservations()

```
bool costmap_2d::ObstacleLayer::getMarkingObservations (
            std::vector< costmap_2d::Observation > & marking_observations ) const  [protected]
```

Get the observations used to mark space

参数

marking_observations	A reference to a vector that will be populated with the observations

返回

True if all the observation buffers are current, false otherwise

5.53.1.3 laserScanCallback()

```
void costmap_2d::ObstacleLayer::laserScanCallback (
             const sensor_msgs::LaserScanConstPtr & message,
             const boost::shared_ptr< costmap_2d::ObservationBuffer > & buffer )
```

A callback to handle buffering LaserScan messages

参数

message	The message returned from a message notifier
buffer	A pointer to the observation buffer to update

5.53.1.4 laserScanValidInfCallback()

```
void costmap_2d::ObstacleLayer::laserScanValidInfCallback (
             const sensor_msgs::LaserScanConstPtr & message,
             const boost::shared_ptr< ObservationBuffer > & buffer )
```

A callback to handle buffering LaserScan messages which need filtering to turn Inf values into range_max.

参数

message	The message returned from a message notifier
buffer	A pointer to the observation buffer to update

5.53.1.5 onInitialize()

```
virtual void costmap_2d::ObstacleLayer::onInitialize ( ) [virtual]
```

This is called at the end of initialize(). Override to implement subclass-specific initialization.

tf_, name_, and layered_costmap_ will all be set already when this is called.

重载 costmap_2d::Layer .

被 costmap_2d::VoxelLayer 重载.

5.53.1.6 pointCloud2Callback()

```
void costmap_2d::ObstacleLayer::pointCloud2Callback (
             const sensor_msgs::PointCloud2ConstPtr & message,
             const boost::shared_ptr< costmap_2d::ObservationBuffer > & buffer )
```

A callback to handle buffering PointCloud2 messages

参数

message	The message returned from a message notifier
buffer	A pointer to the observation buffer to update

5.53.1.7 pointCloudCallback()

```
void costmap_2d::ObstacleLayer::pointCloudCallback (
             const sensor_msgs::PointCloudConstPtr & message,
             const boost::shared_ptr< costmap_2d::ObservationBuffer > & buffer )
```

A callback to handle buffering PointCloud messages

参数

message	The message returned from a message notifier
buffer	A pointer to the observation buffer to update

5.53.1.8 raytraceFreespace()

```
virtual void costmap_2d::ObstacleLayer::raytraceFreespace (
             const costmap_2d::Observation & clearing_observation,
             double * min_x,
             double * min_y,
             double * max_x,
             double * max_y )  [protected], [virtual]
```

Clear freespace based on one observation

参数

clearing_observation	The observation used to raytrace
min_x	
min_y	
max_x	
max_y	

5.53.1.9 updateBounds()

```
virtual void costmap_2d::ObstacleLayer::updateBounds (
             double robot_x,
             double robot_y,
             double robot_yaw,
             double * min_x,
             double * min_y,
             double * max_x,
             double * max_y )  [virtual]
```

This is called by the LayeredCostmap to poll this plugin as to how much of the costmap it needs to update. Each layer can increase the size of this bounds.

For more details, see ”Layered Costmaps for Context-Sensitive Navigation”, by Lu et. Al, IROS 2014.

重载 costmap_2d::Layer .

被 costmap_2d::VoxelLayer 重载.

该类的文档由以下文件生成:

- costmap_2d/include/costmap_2d/obstacle_layer.h

5.54 base_local_planner::OdometryHelperRos类 参考

Public 成员函数

- OdometryHelperRos (std::string odom_topic="")

  Constructor.
- void odomCallback (const nav_msgs::Odometry::ConstPtr &msg)

  Callback for receiving odometry data
- void **getOdom** (nav_msgs::Odometry &base_odom)
- void **getRobotVel** (geometry_msgs::PoseStamped &robot_vel)
- void setOdomTopic (std::string odom_topic)

  Set the odometry topic. This overrides what was set in the constructor, if anything.
- std::string getOdomTopic () const

  Return the current odometry topic.

5.54.1 构造及析构函数说明

5.54.1.1 OdometryHelperRos()

```
base_local_planner::OdometryHelperRos::OdometryHelperRos (
          std::string odom_topic = "" )
```

Constructor.

参数

odom_topic	The topic on which to subscribe to Odometry messages. If the empty string is given (the default), no subscription is done.

5.54.2 成员函数说明

5.54.2.1 odomCallback()

```
void base_local_planner::OdometryHelperRos::odomCallback (
            const nav_msgs::Odometry::ConstPtr & msg )
```

Callback for receiving odometry data

参数

msg	An Odometry message

5.54.2.2 setOdomTopic()

```
void base_local_planner::OdometryHelperRos::setOdomTopic (
            std::string odom_topic )
```

Set the odometry topic. This overrides what was set in the constructor, if anything.

This unsubscribes from the old topic (if any) and subscribes to the new one (if any).

If odom_topic is the empty string, this just unsubscribes from the previous topic.

该类的文档由以下文件生成:

- base_local_planner/include/base_local_planner/odometry_helper_ros.h

5.55 global_planner::OrientationFilter类 参考

Public 成员函数

- virtual void **processPath** (const geometry_msgs::PoseStamped &start, std::vector< geometry_msgs::Pose↩ Stamped > &path)
- void **setAngleBasedOnPositionDerivative** (std::vector< geometry_msgs::PoseStamped > &path, int index)
- void **interpolate** (std::vector< geometry_msgs::PoseStamped > &path, int start_index, int end_index)
- void **setMode** (OrientationMode new_mode)
- void **setMode** (int new_mode)
- void **setWindowSize** (size_t window_size)

Protected 属性

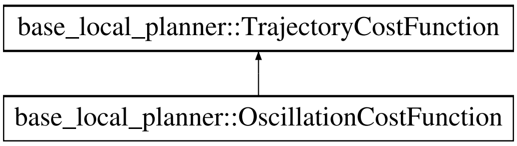
- OrientationMode **omode_**
- int **window_size_**

该类的文档由以下文件生成:

- global_planner/include/global_planner/orientation_filter.h

5.56 base_local_planner::OscillationCostFunction类 参考

类 base_local_planner::OscillationCostFunction 继承关系图:


base_local_planner::TrajectoryCostFunction
base_local_planner::OscillationCostFunction

Public 成员函数

- double scoreTrajectory (Trajectory &traj)
- bool prepare ()
- void resetOscillationFlags ()

  Reset the oscillation flags for the local planner
- void **updateOscillationFlags** (Eigen::Vector3f pos, base_local_planner::Trajectory *traj, double min_vel_↩ trans)
- void **setOscillationResetDist** (double dist, double angle)

额外继承的成员函数

5.56.1 成员函数说明

5.56.1.1 prepare()

```
bool base_local_planner::OscillationCostFunction::prepare ( )  [inline], [virtual]
```

General updating of context values if required. Subclasses may overwrite. Return false in case there is any error.

实现了 base_local_planner::TrajectoryCostFunction.

5.56.1.2 scoreTrajectory()

```
double base_local_planner::OscillationCostFunction::scoreTrajectory (
            Trajectory & traj )  [virtual]
```

return a score for trajectory traj

实现了 base_local_planner::TrajectoryCostFunction.

该类的文档由以下文件生成:

- base_local_planner/include/base_local_planner/oscillation_cost_function.h

5.57 pf_cluster_t结构体 参考

成员变量

- int **count**
- double **weight**
- pf_vector_t **mean**
- pf_matrix_t **cov**
- double **m** [4]
- double **c** [2][2]

该结构体的文档由以下文件生成:

- amcl/include/amcl/pf/pf.h

5.58 pf_kdtree_node结构体 参考

成员变量

- int **leaf**
- int **depth**
- int **pivot_dim**
- double **pivot_value**
- int **key** [3]
- double **value**
- int **cluster**
- struct pf_kdtree_node ∗ **children** [2]

该结构体的文档由以下文件生成:

- amcl/include/amcl/pf/pf_kdtree.h

5.59 pf_kdtree_t结构体 参考

成员变量

- double **size** [3]
- pf_kdtree_node_t ∗ **root**
- int **node_count**
- int **node_max_count**
- pf_kdtree_node_t ∗ **nodes**
- int **leaf_count**

该结构体的文档由以下文件生成:

- amcl/include/amcl/pf/pf_kdtree.h

5.60 pf_matrix_t结构体 参考

成员变量

- double **m** [3][3]

该结构体的文档由以下文件生成:

- amcl/include/amcl/pf/pf_vector.h

5.61 pf_pdf_gaussian_t结构体 参考

成员变量

- pf_vector_t **x**
- pf_matrix_t **cx**
- double **cxdet**
- pf_matrix_t **cr**
- pf_vector_t **cd**

该结构体的文档由以下文件生成:

- amcl/include/amcl/pf/pf_pdf.h

5.62 pf_sample_t结构体 参考

成员变量

- pf_vector_t **pose**
- double **weight**

该结构体的文档由以下文件生成:

- amcl/include/amcl/pf/pf.h

5.63 pf_vector_t结构体 参考

成员变量

- double **v** [3]

该结构体的文档由以下文件生成:

- amcl/include/amcl/pf/pf_vector.h

5.64 base_local_planner::PlanarLaserScan类 参考

Stores a scan from a planar laser that can be used to clear freespace

```
#include <planar_laser_scan.h>
```

成员变量

- geometry_msgs::Point32 **origin**
- sensor_msgs::PointCloud **cloud**
- double **angle_min**
- double **angle_max**
- double **angle_increment**

5.64.1 详细描述

Stores a scan from a planar laser that can be used to clear freespace

该类的文档由以下文件生成:

- base_local_planner/include/base_local_planner/planar_laser_scan.h

5.65 PlannerCore类 参考

Provides a ROS wrapper for the global_planner planner which runs a fast, interpolated navigation function on a costmap.

```
#include <planner_core.h>
```

5.65.1 详细描述

Provides a ROS wrapper for the global_planner planner which runs a fast, interpolated navigation function on a costmap.

该类的文档由以下文件生成:

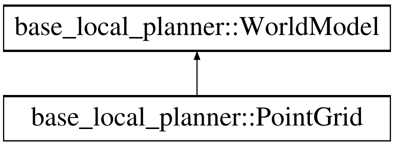
- global_planner/include/global_planner/planner_core.h

5.66 base_local_planner::PointGrid类 参考

A class that implements the WorldModel interface to provide free-space collision checks for the trajectory controller. This class stores points binned into a grid and performs point-in-polygon checks when necessary to determine the legality of a footprint at a given position/orientation.

```
#include <point_grid.h>
```

类 base_local_planner::PointGrid 继承关系图:



Public 成员函数

- PointGrid (double width, double height, double resolution, geometry_msgs::Point origin, double max_z, double obstacle_range, double min_separation)

  Constuctor for a grid that stores points in the plane
- virtual ∼PointGrid ()

  Destructor for a point grid
- void getPointsInRange (const geometry_msgs::Point &lower_left, const geometry_msgs::Point &upper_right, std::vector< std::list< geometry_msgs::Point32 > ∗ > &points)

  Returns the points that lie within the cells contained in the specified range. Some of these points may be outside the range itself.
- virtual double footprintCost (const geometry_msgs::Point &position, const std::vector< geometry_msgs::Point > &footprint, double inscribed_radius, double circumscribed_radius)

  Checks if any points in the grid lie inside a convex footprint
- void updateWorld (const std::vector< geometry_msgs::Point > &footprint, const std::vector< costmap_2d::Observation > &observations, const std::vector< PlanarLaserScan > &laser_scans)

  Inserts observations from sensors into the point grid
- bool gridCoords (geometry_msgs::Point pt, unsigned int &gx, unsigned int &gy) const

  Convert from world coordinates to grid coordinates
- void getCellBounds (unsigned int gx, unsigned int gy, geometry_msgs::Point &lower_left, geometry_msgs::←Point &upper_right) const

  Get the bounds in world coordinates of a cell in the point grid, assumes a legal cell when called
- double sq_distance (const geometry_msgs::Point32 &pt1, const geometry_msgs::Point32 &pt2)

  Compute the squared distance between two points
- bool gridCoords (const geometry_msgs::Point32 &pt, unsigned int &gx, unsigned int &gy) const

  Convert from world coordinates to grid coordinates
- unsigned int gridIndex (unsigned int gx, unsigned int gy) const

  Converts cell coordinates to an index value that can be used to look up the correct grid cell
- double orient (const geometry_msgs::Point &a, const geometry_msgs::Point &b, const geometry_msgs::←Point32 &c)

  Check the orientation of a pt c with respect to the vector a->b
- template<typename T >
  double orient (const T &a, const T &b, const T &c)

  Check the orientation of a pt c with respect to the vector a->b
- bool segIntersect (const geometry_msgs::Point32 &v1, const geometry_msgs::Point32 &v2, const geometry←_msgs::Point32 &u1, const geometry_msgs::Point32 &u2)

Check if two line segmenst intersect

- void intersectionPoint (const geometry_msgs::Point &v1, const geometry_msgs::Point &v2, const geometry_msgs::Point &u1, const geometry_msgs::Point &u2, geometry_msgs::Point &result)

  Find the intersection point of two lines
- bool ptInPolygon (const geometry_msgs::Point32 &pt, const std::vector< geometry_msgs::Point > &poly)

  Check if a point is in a polygon
- void insert (const geometry_msgs::Point32 &pt)

  Insert a point into the point grid
- double nearestNeighborDistance (const geometry_msgs::Point32 &pt)

  Find the distance between a point and its nearest neighbor in the grid
- double getNearestInCell (const geometry_msgs::Point32 &pt, unsigned int gx, unsigned int gy)

  Find the distance between a point and its nearest neighbor in a cell
- void removePointsInPolygon (const std::vector< geometry_msgs::Point > poly)

  Removes points from the grid that lie within the polygon
- void removePointsInScanBoundry (const PlanarLaserScan &laser_scan)

  Removes points from the grid that lie within a laser scan
- bool ptInScan (const geometry_msgs::Point32 &pt, const PlanarLaserScan &laser_scan)

  Checks to see if a point is within a laser scan specification
- void getPoints (sensor_msgs::PointCloud2 &cloud)

  Get the points in the point grid
- virtual double footprintCost (const geometry_msgs::Point &position, const std::vector< geometry_msgs::Point > &footprint, double inscribed_radius, double circumscribed_radius)=0

  Subclass will implement this method to check a footprint at a given position and orientation for legality in the world
- double **footprintCost** (double x, double y, double theta, const std::vector< geometry_msgs::Point > &footprint_spec, double inscribed_radius=0.0, double circumscribed_radius=0.0)
- double footprintCost (const geometry_msgs::Point &position, const std::vector< geometry_msgs::Point > &footprint, double inscribed_radius, double circumscribed_radius, double extra)

  Checks if any obstacles in the costmap lie inside a convex footprint that is rasterized into the grid

5.66.1 详细描述

A class that implements the WorldModel interface to provide free-space collision checks for the trajectory controller. This class stores points binned into a grid and performs point-in-polygon checks when necessary to determine the legality of a footprint at a given position/orientation.

5.66.2 构造及析构函数说明

5.66.2.1 PointGrid()

```
base_local_planner::PointGrid::PointGrid (
            double width,
            double height,
            double resolution,
            geometry_msgs::Point origin,
            double max_z,
            double obstacle_range,
            double min_separation )
```

Constuctor for a grid that stores points in the plane

参数

width	The width in meters of the grid
height	The height in meters of the gird
resolution	The resolution of the grid in meters/cell
origin	The origin of the bottom left corner of the grid
max_z	The maximum height for an obstacle to be added to the grid
obstacle_range	The maximum distance for obstacles to be added to the grid
min_separation	The minimum distance between points in the grid

5.66.3 成员函数说明

5.66.3.1 footprintCost() [1/3]

```
virtual double base_local_planner::PointGrid::footprintCost (
            const geometry_msgs::Point & position,
            const std::vector< geometry_msgs::Point > & footprint,
            double inscribed_radius,
            double circumscribed_radius )  [virtual]
```

Checks if any points in the grid lie inside a convex footprint

参数

position	The position of the robot in world coordinates
footprint	The specification of the footprint of the robot in world coordinates
inscribed_radius	The radius of the inscribed circle of the robot
circumscribed_radius	The radius of the circumscribed circle of the robot

返回

Positive if all the points lie outside the footprint, negative otherwise

实现了 base_local_planner::WorldModel.

5.66.3.2 footprintCost() [2/3]

```
virtual double base_local_planner::WorldModel::footprintCost
```

Subclass will implement this method to check a footprint at a given position and orientation for legality in the world

参数

position	The position of the robot in world coordinates
footprint	The specification of the footprint of the robot in world coordinates
inscribed_radius	The radius of the inscribed circle of the robot
circumscribed_radius	The radius of the circumscribed circle of the robot

返回

> Positive if all the points lie outside the footprint, negative otherwise: -1 if footprint covers at least a lethal obstacle cell, or -2 if footprint covers at least a no-information cell, or -3 if footprint is partially or totally outside of the map

5.66.3.3 footprintCost() [3/3]

```
double base_local_planner::WorldModel::footprintCost  [inline]
```

Checks if any obstacles in the costmap lie inside a convex footprint that is rasterized into the grid

参数

position	The position of the robot in world coordinates
footprint	The specification of the footprint of the robot in world coordinates
inscribed_radius	The radius of the inscribed circle of the robot
circumscribed_radius	The radius of the circumscribed circle of the robot

返回

> Positive if all the points lie outside the footprint, negative otherwise

5.66.3.4 getCellBounds()

```
void base_local_planner::PointGrid::getCellBounds (
            unsigned int gx,
            unsigned int gy,
            geometry_msgs::Point & lower_left,
            geometry_msgs::Point & upper_right ) const  [inline]
```

Get the bounds in world coordinates of a cell in the point grid, assumes a legal cell when called

参数

gx	The x coordinate of the grid cell
gy	The y coordinate of the grid cell
lower_left	The lower left bounds of the cell in world coordinates to be filled in
upper_right	The upper right bounds of the cell in world coordinates to be filled in

5.66.3.5 getNearestInCell()

```
double base_local_planner::PointGrid::getNearestInCell (
             const geometry_msgs::Point32 & pt,
             unsigned int gx,
             unsigned int gy )
```

Find the distance between a point and its nearest neighbor in a cell

参数

pt	The point used for comparison
gx	The x coordinate of the cell
gy	The y coordinate of the cell

返回

The distance between the point passed in and its nearest neighbor in the cell

5.66.3.6 getPoints()

```
void base_local_planner::PointGrid::getPoints (
             sensor_msgs::PointCloud2 & cloud )
```

Get the points in the point grid

参数

cloud	The point cloud to insert the points into

5.66.3.7 getPointsInRange()

```
void base_local_planner::PointGrid::getPointsInRange (
             const geometry_msgs::Point & lower_left,
             const geometry_msgs::Point & upper_right,
             std::vector< std::list< geometry_msgs::Point32 > * > & points )
```

Returns the points that lie within the cells contained in the specified range. Some of these points may be outside the range itself.

参数

lower_left	The lower left corner of the range search
upper_right	The upper right corner of the range search
points	A vector of pointers to lists of the relevant points

5.66.3.8 gridCoords() [1/2]

```
bool base_local_planner::PointGrid::gridCoords (
            const geometry_msgs::Point32 & pt,
            unsigned int & gx,
            unsigned int & gy ) const  [inline]
```

Convert from world coordinates to grid coordinates

参数

pt	A point in world space
gx	The x coordinate of the corresponding grid cell to be set by the function
gy	The y coordinate of the corresponding grid cell to be set by the function

返回

True if the conversion was successful, false otherwise

5.66.3.9 gridCoords() [2/2]

```
bool base_local_planner::PointGrid::gridCoords (
            geometry_msgs::Point pt,
            unsigned int & gx,
            unsigned int & gy ) const  [inline]
```

Convert from world coordinates to grid coordinates

参数

pt	A point in world space
gx	The x coordinate of the corresponding grid cell to be set by the function
gy	The y coordinate of the corresponding grid cell to be set by the function

返回

True if the conversion was successful, false otherwise

5.66.3.10 gridIndex()

```
unsigned int base_local_planner::PointGrid::gridIndex (
            unsigned int gx,
            unsigned int gy ) const [inline]
```

Converts cell coordinates to an index value that can be used to look up the correct grid cell

参数

gx	The x coordinate of the cell
gy	The y coordinate of the cell

返回

The index of the cell in the stored cell list

5.66.3.11 insert()

```
void base_local_planner::PointGrid::insert (
            const geometry_msgs::Point32 & pt )
```

Insert a point into the point grid

参数

pt	The point to be inserted

5.66.3.12 intersectionPoint()

```
void base_local_planner::PointGrid::intersectionPoint (
            const geometry_msgs::Point & v1,
            const geometry_msgs::Point & v2,
            const geometry_msgs::Point & u1,
            const geometry_msgs::Point & u2,
            geometry_msgs::Point & result )
```

Find the intersection point of two lines

参数

v1	The first point of the first segment

参数

v2	The second point of the first segment
u1	The first point of the second segment
u2	The second point of the second segment
result	The point to be filled in

5.66.3.13 nearestNeighborDistance()

```
double base_local_planner::PointGrid::nearestNeighborDistance (
            const geometry_msgs::Point32 & pt )
```

Find the distance between a point and its nearest neighbor in the grid

参数

pt	The point used for comparison

返回

The distance between the point passed in and its nearest neighbor in the point grid

5.66.3.14 orient() [1/2]

```
double base_local_planner::PointGrid::orient (
            const geometry_msgs::Point & a,
            const geometry_msgs::Point & b,
            const geometry_msgs::Point32 & c ) [inline]
```

Check the orientation of a pt c with respect to the vector a->b

参数

a	The start point of the vector
b	The end point of the vector
c	The point to compute orientation for

返回

orient(a, b, c) < 0 —-> Right, orient(a, b, c) > 0 —-> Left

5.66.3.15 orient() [2/2]

```
template<typename T >
double base_local_planner::PointGrid::orient (
             const T & a,
             const T & b,
             const T & c )  [inline]
```

Check the orientation of a pt c with respect to the vector a->b

参数

a	The start point of the vector
b	The end point of the vector
c	The point to compute orientation for

返回

orient(a, b, c) < 0 -—> Right, orient(a, b, c) > 0 -—> Left

5.66.3.16 ptInPolygon()

```
bool base_local_planner::PointGrid::ptInPolygon (
             const geometry_msgs::Point32 & pt,
             const std::vector< geometry_msgs::Point > & poly )
```

Check if a point is in a polygon

参数

pt	The point to be checked
poly	The polygon to check against

返回

True if the point is in the polygon, false otherwise

5.66.3.17 ptInScan()

```
bool base_local_planner::PointGrid::ptInScan (
             const geometry_msgs::Point32 & pt,
             const PlanarLaserScan & laser_scan )
```

Checks to see if a point is within a laser scan specification

参数

pt	The point to check
laser_scan	The specification of the scan to check against

返回

True if the point is contained within the scan, false otherwise

5.66.3.18 removePointsInPolygon()

```
void base_local_planner::PointGrid::removePointsInPolygon (
            const std::vector< geometry_msgs::Point > poly )
```

Removes points from the grid that lie within the polygon

参数

poly	A specification of the polygon to clear from the grid

5.66.3.19 removePointsInScanBoundry()

```
void base_local_planner::PointGrid::removePointsInScanBoundry (
            const PlanarLaserScan & laser_scan )
```

Removes points from the grid that lie within a laser scan

参数

laser_scan	A specification of the laser scan to use for clearing

5.66.3.20 segIntersect()

```
bool base_local_planner::PointGrid::segIntersect (
            const geometry_msgs::Point32 & v1,
            const geometry_msgs::Point32 & v2,
            const geometry_msgs::Point32 & u1,
            const geometry_msgs::Point32 & u2 )  [inline]
```

Check if two line segmenst intersect

参数

v1	The first point of the first segment
v2	The second point of the first segment
u1	The first point of the second segment
u2	The second point of the second segment

返回

True if the segments intersect, false otherwise

5.66.3.21 sq_distance()

```
double base_local_planner::PointGrid::sq_distance (
             const geometry_msgs::Point32 & pt1,
             const geometry_msgs::Point32 & pt2 )  [inline]
```

Compute the squared distance between two points

参数

pt1	The first point
pt2	The second point

返回

The squared distance between the two points

5.66.3.22 updateWorld()

```
void base_local_planner::PointGrid::updateWorld (
             const std::vector< geometry_msgs::Point > & footprint,
             const std::vector< costmap_2d::Observation > & observations,
             const std::vector< PlanarLaserScan > & laser_scans )
```

Inserts observations from sensors into the point grid

参数

footprint	The footprint of the robot in its current location
observations	The observations from various sensors
laser_scans	The laser scans used to clear freespace (the point grid only uses the first scan which is assumed to be the base laser)

该类的文档由以下文件生成:

- base_local_planner/include/base_local_planner/point_grid.h

5.67 costmap_2d::Costmap2D::PolygonOutlineCells类 参考

Public 成员函数

- **PolygonOutlineCells** (const Costmap2D &costmap, const unsigned char ∗char_map, std::vector< MapLocation > &cells)
- void **operator()** (unsigned int offset)

该类的文档由以下文件生成:

- costmap_2d/include/costmap_2d/costmap_2d.h

5.68 navfn::PotarrPoint结构体 参考

成员变量

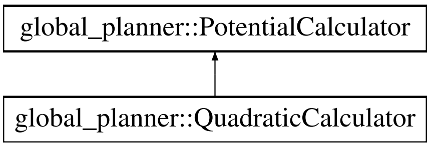
- float **x**
- float **y**
- float **z**
- float **pot_value**

该结构体的文档由以下文件生成:

- navfn/include/navfn/potarr_point.h

5.69 global_planner::PotentialCalculator类 参考

类 global_planner::PotentialCalculator 继承关系图:


global_planner::PotentialCalculator
global_planner::QuadraticCalculator

Public 成员函数

- **PotentialCalculator** (int nx, int ny)
- virtual float **calculatePotential** (float ∗potential, unsigned char cost, int n, float prev_potential=-1)
- virtual void setSize (int nx, int ny)

  Sets or resets the size of the map

Protected 成员函数

- int **toIndex** (int x, int y)

Protected 属性

- int **nx_**
- int **ny_**
- int ns_

5.69.1 成员函数说明

5.69.1.1 setSize()

```
virtual void global_planner::PotentialCalculator::setSize (
            int nx,
            int ny )  [inline], [virtual]
```

Sets or resets the size of the map

参数

nx	The x size of the map
ny	The y size of the map sets or resets the size of the map

5.69.2 结构体成员变量说明

5.69.2.1 ns_

```
int global_planner::PotentialCalculator::ns_  [protected]
```

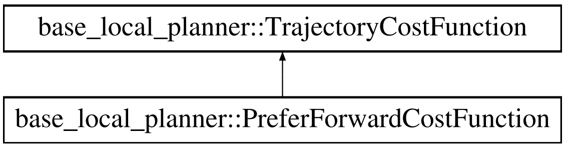
size of grid, in pixels

该类的文档由以下文件生成:

- global_planner/include/global_planner/potential_calculator.h

5.70 base_local_planner::PreferForwardCostFunction类 参考

类 base_local_planner::PreferForwardCostFunction 继承关系图:


base_local_planner::TrajectoryCostFunction
base_local_planner::PreferForwardCostFunction

Public 成员函数

- **PreferForwardCostFunction** (double penalty)
- double scoreTrajectory (Trajectory &traj)
- bool prepare ()
- void **setPenalty** (double penalty)

额外继承的成员函数

5.70.1 成员函数说明

5.70.1.1 prepare()

```
bool base_local_planner::PreferForwardCostFunction::prepare ( )  [inline], [virtual]
```

General updating of context values if required. Subclasses may overwrite. Return false in case there is any error.

实现了 base_local_planner::TrajectoryCostFunction.

5.70.1.2 scoreTrajectory()

```
double base_local_planner::PreferForwardCostFunction::scoreTrajectory (
            Trajectory & traj )  [virtual]
```

return a score for trajectory traj

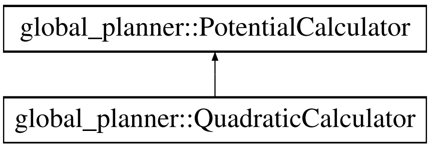
实现了 base_local_planner::TrajectoryCostFunction.

该类的文档由以下文件生成:

- base_local_planner/include/base_local_planner/prefer_forward_cost_function.h

5.71 global_planner::QuadraticCalculator类 参考

类 global_planner::QuadraticCalculator 继承关系图:


global_planner::PotentialCalculator
global_planner::QuadraticCalculator

Public 成员函数

- **QuadraticCalculator** (int nx, int ny)
- float **calculatePotential** (float ∗potential, unsigned char cost, int n, float prev_potential)

额外继承的成员函数

该类的文档由以下文件生成:

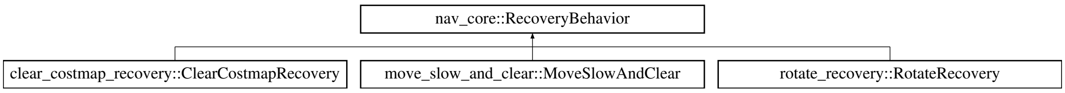
- global_planner/include/global_planner/quadratic_calculator.h

5.72 nav_core::RecoveryBehavior类 参考

Provides an interface for recovery behaviors used in navigation. All recovery behaviors written as plugins for the navigation stack must adhere to this interface.

```
#include <recovery_behavior.h>
```

类 nav_core::RecoveryBehavior 继承关系图:


nav_core::RecoveryBehavior
clear_costmap_recovery::ClearCostmapRecovery
move_slow_and_clear::MoveSlowAndClear
rotate_recovery::RotateRecovery

Public 成员函数

- virtual void initialize (std::string name, tf2_ros::Buffer ∗tf, costmap_2d::Costmap2DROS ∗global_costmap, costmap_2d::Costmap2DROS ∗local_costmap)=0

  Initialization function for the RecoveryBehavior
- virtual void runBehavior ()=0

  Runs the RecoveryBehavior
- virtual ∼RecoveryBehavior ()

  Virtual destructor for the interface

5.72.1 详细描述

Provides an interface for recovery behaviors used in navigation. All recovery behaviors written as plugins for the navigation stack must adhere to this interface.

5.72.2 成员函数说明

5.72.2.1 initialize()

```
virtual void nav_core::RecoveryBehavior::initialize (
             std::string name,
             tf2_ros::Buffer * tf,
             costmap_2d::Costmap2DROS * global_costmap,
             costmap_2d::Costmap2DROS * local_costmap )  [pure virtual]
```

Initialization function for the RecoveryBehavior

参数

tf	A pointer to a transform listener
global_costmap	A pointer to the global_costmap used by the navigation stack
local_costmap	A pointer to the local_costmap used by the navigation stack

在 clear_costmap_recovery::ClearCostmapRecovery, rotate_recovery::RotateRecovery , 以及 move_slow_and_clear::MoveSlowAndCle
内被实现.

该类的文档由以下文件生成:

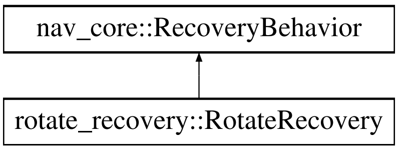
- nav_core/include/nav_core/recovery_behavior.h

5.73 rotate_recovery::RotateRecovery类 参考

A recovery behavior that rotates the robot in-place to attempt to clear out space

```
#include <rotate_recovery.h>
```

类 rotate_recovery::RotateRecovery 继承关系图:



Public 成员函数

- RotateRecovery ()

  Constructor, make sure to call initialize in addition to actually initialize the object
- void initialize (std::string name, tf2_ros::Buffer ∗, costmap_2d::Costmap2DROS ∗, costmap_2d::Costmap2DROS ∗local_costmap)

  Initialization function for the RotateRecovery recovery behavior
- void runBehavior ()

  Run the RotateRecovery recovery behavior.
- ∼RotateRecovery ()

  Destructor for the rotate recovery behavior

5.73.1 详细描述

A recovery behavior that rotates the robot in-place to attempt to clear out space

5.73.2 成员函数说明

5.73.2.1 initialize()

```
void rotate_recovery::RotateRecovery::initialize (
            std::string name,
            tf2_ros::Buffer * ,
            costmap_2d::Costmap2DROS * ,
            costmap_2d::Costmap2DROS * local_costmap )  [virtual]
```

Initialization function for the RotateRecovery recovery behavior

参数

name	Namespace used in initialization
tf	(unused)
global_costmap	(unused)
local_costmap	A pointer to the local_costmap used by the navigation stack

实现了 nav_core::RecoveryBehavior.

该类的文档由以下文件生成:

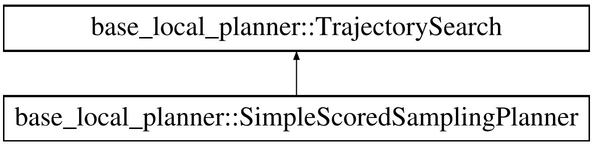
- rotate_recovery/include/rotate_recovery/rotate_recovery.h

5.74 base_local_planner::SimpleScoredSamplingPlanner类 参考

Generates a local plan using the given generator and cost functions. Assumes less cost are best, and negative costs indicate infinite costs

```
#include <simple_scored_sampling_planner.h>
```

类 base_local_planner::SimpleScoredSamplingPlanner 继承关系图:


base_local_planner::TrajectorySearch
base_local_planner::SimpleScoredSamplingPlanner

Public 成员函数

- SimpleScoredSamplingPlanner (std::vector< TrajectorySampleGenerator ∗ > gen_list, std::vector< TrajectoryCostFunction ∗ > &critics, int max_samples=-1)
- double scoreTrajectory (Trajectory &traj, double best_traj_cost)
- bool findBestTrajectory (Trajectory &traj, std::vector< Trajectory > ∗all_explored=0)

5.74.1 详细描述

Generates a local plan using the given generator and cost functions. Assumes less cost are best, and negative costs indicate infinite costs

This is supposed to be a simple and robust implementation of the TrajectorySearch interface. More efficient search may well be possible using search heuristics, parallel search, etc.

5.74.2 构造及析构函数说明

5.74.2.1 SimpleScoredSamplingPlanner()

```
base_local_planner::SimpleScoredSamplingPlanner::SimpleScoredSamplingPlanner (
                std::vector< TrajectorySampleGenerator * > gen_list,
                std::vector< TrajectoryCostFunction * > & critics,
                int max_samples = -1 )
```

Takes a list of generators and critics. Critics return costs > 0, or negative costs for invalid trajectories. Generators other than the first are fallback generators, meaning they only get to generate if the previous generator did not find a valid trajectory. Will use every generator until it stops returning trajectories or count reaches max_samples. Then resets count and tries for the next in the list. passing max_samples = -1 (default): Each Sampling planner will continue to call generator until generator runs out of samples (or forever if that never happens)

5.74.3 成员函数说明

5.74.3.1 findBestTrajectory()

```
bool base_local_planner::SimpleScoredSamplingPlanner::findBestTrajectory (
                Trajectory & traj,
                std::vector< Trajectory > * all_explored = 0 )  [virtual]
```

Calls generator until generator has no more samples or max_samples is reached. For each generated traj, calls critics in turn. If any critic returns negative value, that value is assumed as costs, else the costs are the sum of all critics result. Returns true and sets the traj parameter to the first trajectory with minimal non-negative costs if sampling yields trajectories with non-negative costs, else returns false.

参数

traj	The container to write the result to
all_explored	pass NULL or a container to collect all trajectories for debugging (has a penalty)

实现了 base_local_planner::TrajectorySearch.

5.74.3.2 scoreTrajectory()

```
double base_local_planner::SimpleScoredSamplingPlanner::scoreTrajectory (
            Trajectory & traj,
            double best_traj_cost )
```

runs all scoring functions over the trajectory creating a weigthed sum of positive costs, aborting as soon as a negative cost are found or costs greater than positive best_traj_cost accumulated

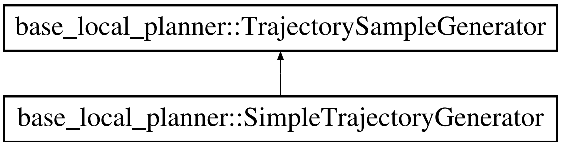
该类的文档由以下文件生成:

- base_local_planner/include/base_local_planner/simple_scored_sampling_planner.h

5.75 base_local_planner::SimpleTrajectoryGenerator类 参考

```
#include <simple_trajectory_generator.h>
```

类 base_local_planner::SimpleTrajectoryGenerator 继承关系图:


base_local_planner::TrajectorySampleGenerator
base_local_planner::SimpleTrajectoryGenerator

Public 成员函数

- void initialise (const Eigen::Vector3f &pos, const Eigen::Vector3f &vel, const Eigen::Vector3f &goal, base_local_planner::LocalPlannerLimits ∗limits, const Eigen::Vector3f &vsamples, std::vector< Eigen::↩ Vector3f > additional_samples, bool discretize_by_time=false)
- void initialise (const Eigen::Vector3f &pos, const Eigen::Vector3f &vel, const Eigen::Vector3f &goal, base_local_planner::LocalPlannerLimits ∗limits, const Eigen::Vector3f &vsamples, bool discretize_by_↩ time=false)
- void setParameters (double sim_time, double sim_granularity, double angular_sim_granularity, bool use_↩ dwa=false, double sim_period=0.0)
- bool hasMoreTrajectories ()
- bool nextTrajectory (Trajectory &traj)
- bool **generateTrajectory** (Eigen::Vector3f pos, Eigen::Vector3f vel, Eigen::Vector3f sample_target_vel, base_local_planner::Trajectory &traj)

静态 Public 成员函数

- static Eigen::Vector3f **computeNewPositions** (const Eigen::Vector3f &pos, const Eigen::Vector3f &vel, double dt)
- static Eigen::Vector3f **computeNewVelocities** (const Eigen::Vector3f &sample_target_vel, const Eigen::↩ Vector3f &vel, Eigen::Vector3f acclimits, double dt)

Protected 属性

- unsigned int **next_sample_index_**
- std::vector< Eigen::Vector3f > **sample_params_**
- base_local_planner::LocalPlannerLimits ∗ **limits_**
- Eigen::Vector3f **pos_**
- Eigen::Vector3f **vel_**
- bool **continued_acceleration_**
- bool **discretize_by_time_**
- double **sim_time_**
- double **sim_granularity_**
- double **angular_sim_granularity_**
- bool **use_dwa_**
- double **sim_period_**

5.75.1 详细描述

generates trajectories based on equi-distant discretisation of the degrees of freedom. This is supposed to be a simple and robust implementation of the TrajectorySampleGenerator interface, more efficient implementations are thinkable.

This can be used for both dwa and trajectory rollout approaches. As an example, assuming these values: sim_time = 1s, sim_period=200ms, dt = 200ms, vsamples_x=5, acc_limit_x = 1m/s^2, vel_x=0 (robot at rest, values just for easy calculations) dwa_planner will sample max-x-velocities from 0m/s to 0.2m/s. trajectory rollout approach will sample max-x-velocities 0m/s up to 1m/s trajectory rollout approach does so respecting the acceleration limit, so it gradually increases velocity

5.75.2 成员函数说明

5.75.2.1 hasMoreTrajectories()

```
bool base_local_planner::SimpleTrajectoryGenerator::hasMoreTrajectories ( )  [virtual]
```

Whether this generator can create more trajectories

实现了 base_local_planner::TrajectorySampleGenerator.

5.75.2.2 initialise() [1/2]

```
void base_local_planner::SimpleTrajectoryGenerator::initialise (
            const Eigen::Vector3f & pos,
            const Eigen::Vector3f & vel,
            const Eigen::Vector3f & goal,
            base_local_planner::LocalPlannerLimits * limits,
            const Eigen::Vector3f & vsamples,
            bool discretize_by_time = false )
```

参数

pos	current robot position
vel	current robot velocity
limits	Current velocity limits
vsamples	in how many samples to divide the given dimension
use_acceleration_limits	if true use physical model, else idealized robot model
discretize_by_time	if true, the trajectory is split according in chunks of the same duration, else of same length

5.75.2.3 initialise() [2/2]

```
void base_local_planner::SimpleTrajectoryGenerator::initialise (
             const Eigen::Vector3f & pos,
             const Eigen::Vector3f & vel,
             const Eigen::Vector3f & goal,
             base_local_planner::LocalPlannerLimits * limits,
             const Eigen::Vector3f & vsamples,
             std::vector< Eigen::Vector3f > additional_samples,
             bool discretize_by_time = false )
```

参数

pos	current robot position
vel	current robot velocity
limits	Current velocity limits
vsamples	in how many samples to divide the given dimension
use_acceleration_limits	if true use physical model, else idealized robot model
additional_samples	(deprecated): Additional velocity samples to generate individual trajectories from.
discretize_by_time	if true, the trajectory is split according in chunks of the same duration, else of same length

5.75.2.4 nextTrajectory()

```
bool base_local_planner::SimpleTrajectoryGenerator::nextTrajectory (
             Trajectory & traj )  [virtual]
```

Whether this generator can create more trajectories

实现了 base_local_planner::TrajectorySampleGenerator.

5.75.2.5 setParameters()

```
void base_local_planner::SimpleTrajectoryGenerator::setParameters (
             double sim_time,
             double sim_granularity,
             double angular_sim_granularity,
             bool use_dwa = false,
             double sim_period = 0.0 )
```

This function is to be called only when parameters change

参数

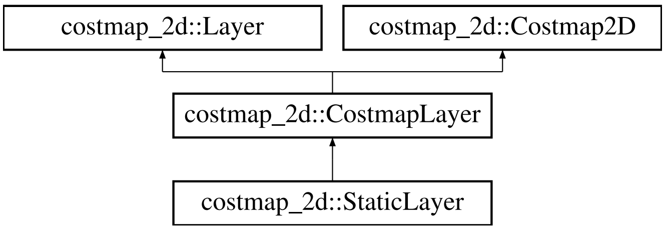
sim_granularity	granularity of collision detection
angular_sim_granularity	angular granularity of collision detection
use_dwa	whether to use DWA or trajectory rollout
sim_period	distance between points in one trajectory

该类的文档由以下文件生成:

- base_local_planner/include/base_local_planner/simple_trajectory_generator.h

5.76 costmap_2d::StaticLayer类 参考

类 costmap_2d::StaticLayer 继承关系图:



Public 成员函数

- virtual void onInitialize ()

  This is called at the end of initialize(). Override to implement subclass-specific initialization.
- virtual void activate ()

  Restart publishers if they've been stopped.
- virtual void deactivate ()

  Stop publishers.
- virtual void **reset** ()
- virtual void updateBounds (double robot_x, double robot_y, double robot_yaw, double ∗min_x, double ∗min_y, double ∗max_x, double ∗max_y)

  This is called by the LayeredCostmap to poll this plugin as to how much of the costmap it needs to update. Each layer can increase the size of this bounds.
- virtual void updateCosts (costmap_2d::Costmap2D &master_grid, int min_i, int min_j, int max_i, int max_j)

  Actually update the underlying costmap, only within the bounds calculated during UpdateBounds().
- virtual void matchSize ()

  Implement this to make this layer match the size of the parent costmap.

额外继承的成员函数

5.76.1 成员函数说明

5.76.1.1 onInitialize()

```
virtual void costmap_2d::StaticLayer::onInitialize ( )  [virtual]
```

This is called at the end of initialize(). Override to implement subclass-specific initialization.

tf_, name_, and layered_costmap_ will all be set already when this is called.

重载 costmap_2d::Layer .

5.76.1.2 updateBounds()

```
virtual void costmap_2d::StaticLayer::updateBounds (
               double robot_x,
               double robot_y,
               double robot_yaw,
               double * min_x,
               double * min_y,
               double * max_x,
               double * max_y )  [virtual]
```

This is called by the LayeredCostmap to poll this plugin as to how much of the costmap it needs to update. Each layer can increase the size of this bounds.

For more details, see ”Layered Costmaps for Context-Sensitive Navigation”, by Lu et. Al, IROS 2014.

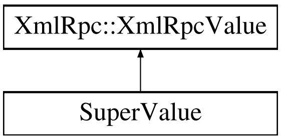
重载 costmap_2d::Layer .

该类的文档由以下文件生成:

- costmap_2d/include/costmap_2d/static_layer.h

5.77 SuperValue类 参考

类 SuperValue 继承关系图:



Public 成员函数

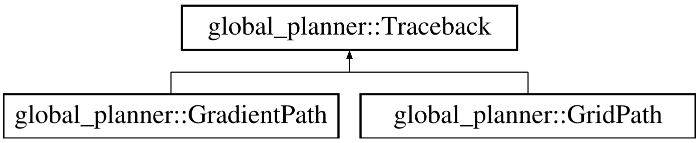
- void **setStruct** (XmlRpc::XmlRpcValue::ValueStruct ∗a)
- void **setArray** (XmlRpc::XmlRpcValue::ValueArray ∗a)

该类的文档由以下文件生成:

- costmap_2d/include/costmap_2d/costmap_2d_ros.h

5.78 global_planner::Traceback类 参考

类 global_planner::Traceback 继承关系图:



Public 成员函数

- **Traceback** (PotentialCalculator ∗p_calc)
- virtual bool **getPath** (float ∗potential, double start_x, double start_y, double end_x, double end_y, std::vector< std::pair< float, float > > &path)=0
- virtual void **setSize** (int xs, int ys)
- int **getIndex** (int x, int y)
- void **setLethalCost** (unsigned char lethal_cost)

Protected 属性

- int **xs_**
- int **ys_**
- unsigned char **lethal_cost_**
- PotentialCalculator ∗ **p_calc_**

该类的文档由以下文件生成:

- global_planner/include/global_planner/traceback.h

5.79 base_local_planner::Trajectory类 参考

Holds a trajectory generated by considering an x, y, and theta velocity

```
#include <trajectory.h>
```

Public 成员函数

- Trajectory ()

  Default constructor
- Trajectory (double xv, double yv, double thetav, double time_delta, unsigned int num_pts)

  Constructs a trajectory
- void getPoint (unsigned int index, double &x, double &y, double &th) const

  Get a point within the trajectory
- void setPoint (unsigned int index, double x, double y, double th)

  Set a point within the trajectory
- void addPoint (double x, double y, double th)

  Add a point to the end of a trajectory
- void getEndpoint (double &x, double &y, double &th) const

  Get the last point of the trajectory
- void resetPoints ()

  Clear the trajectory's points
- unsigned int getPointsSize () const

  Return the number of points in the trajectory

成员变量

- double **xv_**
- double **yv_**
- double thetav_

  The x, y, and theta velocities of the trajectory
- double cost_

  The cost/score of the trajectory
- double time_delta_

  The time gap between points

5.79.1 详细描述

Holds a trajectory generated by considering an x, y, and theta velocity

5.79.2 构造及析构函数说明

5.79.2.1 Trajectory()

```
base_local_planner::Trajectory::Trajectory (
            double xv,
            double yv,
            double thetav,
            double time_delta,
            unsigned int num_pts )
```

Constructs a trajectory

参数

xv	The x velocity used to seed the trajectory
yv	The y velocity used to seed the trajectory
thetav	The theta velocity used to seed the trajectory
num_pts	The expected number of points for a trajectory

5.79.3 成员函数说明

5.79.3.1 addPoint()

```
void base_local_planner::Trajectory::addPoint (
             double x,
             double y,
             double th )
```

Add a point to the end of a trajectory

参数

x	The x position
y	The y position
th	The theta position

5.79.3.2 getEndpoint()

```
void base_local_planner::Trajectory::getEndpoint (
             double & x,
             double & y,
             double & th ) const
```

Get the last point of the trajectory

参数

x	Will be set to the x position of the point
y	Will be set to the y position of the point
th	Will be set to the theta position of the point

5.79.3.3 getPoint()

```
void base_local_planner::Trajectory::getPoint (
            unsigned int index,
            double & x,
            double & y,
            double & th ) const
```

Get a point within the trajectory

参数

index	The index of the point to get
x	Will be set to the x position of the point
y	Will be set to the y position of the point
th	Will be set to the theta position of the point

5.79.3.4 getPointsSize()

```
unsigned int base_local_planner::Trajectory::getPointsSize ( ) const
```

Return the number of points in the trajectory

返回

The number of points in the trajectory

5.79.3.5 setPoint()

```
void base_local_planner::Trajectory::setPoint (
            unsigned int index,
            double x,
            double y,
            double th )
```

Set a point within the trajectory

参数

index	The index of the point to set
x	The x position
y	The y position
th	The theta position

该类的文档由以下文件生成:

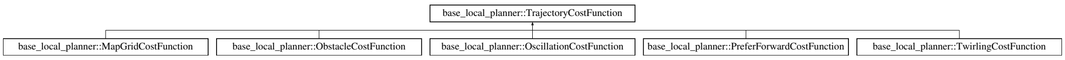
- base_local_planner/include/base_local_planner/trajectory.h

5.80 base_local_planner::TrajectoryCostFunction类 参考

Provides an interface for critics of trajectories During each sampling run, a batch of many trajectories will be scored using such a cost function. The prepare method is called before each batch run, and then for each trajectory of the sampling set, score_trajectory may be called.

```
#include <trajectory_cost_function.h>
```

类 base_local_planner::TrajectoryCostFunction 继承关系图:


base_local_planner::TrajectoryCostFunction
base_local_planner::MapGridCostFunction
base_local_planner::ObstacleCostFunction
base_local_planner::OscillationCostFunction
base_local_planner::PreferForwardCostFunction
base_local_planner::TwirlingCostFunction

Public 成员函数

- virtual bool prepare ()=0
- virtual double scoreTrajectory (Trajectory &traj)=0
- double **getScale** ()
- void **setScale** (double scale)

Protected 成员函数

- **TrajectoryCostFunction** (double scale=1.0)

5.80.1 详细描述

Provides an interface for critics of trajectories During each sampling run, a batch of many trajectories will be scored using such a cost function. The prepare method is called before each batch run, and then for each trajectory of the sampling set, score_trajectory may be called.

5.80.2 成员函数说明

5.80.2.1 prepare()

```
virtual bool base_local_planner::TrajectoryCostFunction::prepare ( )  [pure virtual]
```

General updating of context values if required. Subclasses may overwrite. Return false in case there is any error.

在 base_local_planner::TwirlingCostFunction, base_local_planner::PreferForwardCostFunction, base_local_planner::OscillationCostFur base_local_planner::ObstacleCostFunction , 以及 base_local_planner::MapGridCostFunction 内被实现.

5.80.2.2 scoreTrajectory()

```
virtual double base_local_planner::TrajectoryCostFunction::scoreTrajectory (
            Trajectory & traj )  [pure virtual]
```

return a score for trajectory traj

在 base_local_planner::TwirlingCostFunction, base_local_planner::PreferForwardCostFunction, base_local_planner::OscillationCostFur base_local_planner::ObstacleCostFunction , 以及 base_local_planner::MapGridCostFunction 内被实现.

该类的文档由以下文件生成:

- base_local_planner/include/base_local_planner/trajectory_cost_function.h

5.81 base_local_planner::TrajectoryPlanner类 参考

Computes control velocities for a robot given a costmap, a plan, and the robot's position in the world.

`#include <trajectory_planner.h>`

Public 成员函数

- TrajectoryPlanner (WorldModel &world_model, const costmap_2d::Costmap2D &costmap, std::vector< geometry_msgs::Point > footprint_spec, double acc_lim_x=1.0, double acc_lim_y=1.0, double acc_lim_theta=1.0, double sim_time=1.0, double sim_granularity=0.025, int vx_samples=20, int vtheta_samples=20, double path_distance_bias=0.6, double goal_distance_bias=0.8, double occdist_scale=0.2, double heading_lookahead=0.325, double oscillation_reset_dist=0.05, double escape_reset_dist=0.10, double escape_reset_theta=M_PI_2, bool holonomic_robot=true, double max_vel_x=0.5, double min_vel_x=0.1, double max_vel_th=1.0, double min_vel_th=-1.0, double min_in_place_vel_th=0.4, double backup_vel=-0.1, bool dwa=false, bool heading_scoring=false, double heading_scoring_timestep=0.1, bool meter_scoring=true, bool simple_attractor=false, std::vector< double > y_vels=std::vector< double >(0), double stop_time_buffer=0.2, double sim_period=0.1, double angular_sim_granularity=0.025)

  Constructs a trajectory controller
- ∼TrajectoryPlanner ()

  Destructs a trajectory controller
- void reconfigure (BaseLocalPlannerConfig &cfg)

  Reconfigures the trajectory planner
- Trajectory findBestPath (const geometry_msgs::PoseStamped &global_pose, geometry_msgs::PoseStamped &global_vel, geometry_msgs::PoseStamped &drive_velocities)

  Given the current position, orientation, and velocity of the robot, return a trajectory to follow
- void updatePlan (const std::vector< geometry_msgs::PoseStamped > &new_plan, bool compute_dists=false)

  Update the plan that the controller is following
- void getLocalGoal (double &x, double &y)

  Accessor for the goal the robot is currently pursuing in world corrdinates
- bool checkTrajectory (double x, double y, double theta, double vx, double vy, double vtheta, double vx_samp, double vy_samp, double vtheta_samp)

  Generate and score a single trajectory
- double scoreTrajectory (double x, double y, double theta, double vx, double vy, double vtheta, double vx_samp, double vy_samp, double vtheta_samp)

  Generate and score a single trajectory
- bool getCellCosts (int cx, int cy, float &path_cost, float &goal_cost, float &occ_cost, float &total_cost)

  Compute the components and total cost for a map grid cell
- void setFootprint (std::vector< geometry_msgs::Point > footprint)

  Set the footprint specification of the robot.
- geometry_msgs::Polygon getFootprintPolygon () const

  Return the footprint specification of the robot.
- std::vector< geometry_msgs::Point > **getFootprint** () const

友元

- class **TrajectoryPlannerTest**

5.81.1 详细描述

Computes control velocities for a robot given a costmap, a plan, and the robot's position in the world.

5.81.2 构造及析构函数说明

5.81.2.1 TrajectoryPlanner()

```
base_local_planner::TrajectoryPlanner::TrajectoryPlanner (
             WorldModel & world_model,
             const costmap_2d::Costmap2D & costmap,
             std::vector< geometry_msgs::Point > footprint_spec,
             double acc_lim_x = 1.0,
             double acc_lim_y = 1.0,
             double acc_lim_theta = 1.0,
             double sim_time = 1.0,
             double sim_granularity = 0.025,
             int vx_samples = 20,
             int vtheta_samples = 20,
             double path_distance_bias = 0.6,
             double goal_distance_bias = 0.8,
             double occdist_scale = 0.2,
             double heading_lookahead = 0.325,
             double oscillation_reset_dist = 0.05,
             double escape_reset_dist = 0.10,
             double escape_reset_theta = M_PI_2,
             bool holonomic_robot = true,
             double max_vel_x = 0.5,
             double min_vel_x = 0.1,
             double max_vel_th = 1.0,
             double min_vel_th = -1.0,
             double min_in_place_vel_th = 0.4,
             double backup_vel = -0.1,
             bool dwa = false,
             bool heading_scoring = false,
             double heading_scoring_timestep = 0.1,
             bool meter_scoring = true,
             bool simple_attractor = false,
             std::vector< double > y_vels = std::vector< double >(0),
             double stop_time_buffer = 0.2,
             double sim_period = 0.1,
             double angular_sim_granularity = 0.025 )
```

Constructs a trajectory controller

参数

world_model	The WorldModel the trajectory controller uses to check for collisions
costmap	A reference to the Costmap the controller should use
footprint_spec	A polygon representing the footprint of the robot. (Must be convex)
inscribed_radius	The radius of the inscribed circle of the robot
circumscribed_radius	The radius of the circumscribed circle of the robot
acc_lim_x	The acceleration limit of the robot in the x direction
acc_lim_y	The acceleration limit of the robot in the y direction
acc_lim_theta	The acceleration limit of the robot in the theta direction
sim_time	The number of seconds to ”roll-out” each trajectory
sim_granularity	The distance between simulation points should be small enough that the robot doesn't hit things
vx_samples	The number of trajectories to sample in the x dimension
vtheta_samples	The number of trajectories to sample in the theta dimension
path_distance_bias	A scaling factor for how close the robot should stay to the path
goal_distance_bias	A scaling factor for how aggresively the robot should pursue a local goal
occdist_scale	A scaling factor for how much the robot should prefer to stay away from obstacles
heading_lookahead	How far the robot should look ahead of itself when differentiating between different rotational velocities
oscillation_reset_dist	The distance the robot must travel before it can explore rotational velocities that were unsuccessful in the past
escape_reset_dist	The distance the robot must travel before it can exit escape mode
escape_reset_theta	The distance the robot must rotate before it can exit escape mode
holonomic_robot	Set this to true if the robot being controlled can take y velocities and false otherwise
max_vel_x	The maximum x velocity the controller will explore
min_vel_x	The minimum x velocity the controller will explore
max_vel_th	The maximum rotational velocity the controller will explore
min_vel_th	The minimum rotational velocity the controller will explore
min_in_place_vel_th	The absolute value of the minimum in-place rotational velocity the controller will explore
backup_vel	The velocity to use while backing up
dwa	Set this to true to use the Dynamic Window Approach, false to use acceleration limits
heading_scoring	Set this to true to score trajectories based on the robot's heading after 1 timestep
heading_scoring_timestep	How far to look ahead in time when we score heading based trajectories
meter_scoring	adapt parameters to costmap resolution
simple_attractor	Set this to true to allow simple attraction to a goal point instead of intelligent cost propagation
y_vels	A vector of the y velocities the controller will explore
angular_sim_granularity	The distance between simulation points for angular velocity should be small enough that the robot doesn't hit things

5.81.3 成员函数说明

5.81.3.1 checkTrajectory()

```
bool base_local_planner::TrajectoryPlanner::checkTrajectory (
             double x,
             double y,
             double theta,
             double vx,
             double vy,
             double vtheta,
             double vx_samp,
             double vy_samp,
             double vtheta_samp )
```

Generate and score a single trajectory

参数

x	The x position of the robot
y	The y position of the robot
theta	The orientation of the robot
vx	The x velocity of the robot
vy	The y velocity of the robot
vtheta	The theta velocity of the robot
vx_samp	The x velocity used to seed the trajectory
vy_samp	The y velocity used to seed the trajectory
vtheta_samp	The theta velocity used to seed the trajectory

返回

True if the trajectory is legal, false otherwise

5.81.3.2 findBestPath()

```
Trajectory base_local_planner::TrajectoryPlanner::findBestPath (
             const geometry_msgs::PoseStamped & global_pose,
             geometry_msgs::PoseStamped & global_vel,
             geometry_msgs::PoseStamped & drive_velocities )
```

Given the current position, orientation, and velocity of the robot, return a trajectory to follow

参数

global_pose	The current pose of the robot in world space
global_vel	The current velocity of the robot in world space
drive_velocities	Will be set to velocities to send to the robot base

返回

The selected path or trajectory

5.81.3.3 getCellCosts()

```
bool base_local_planner::TrajectoryPlanner::getCellCosts (
            int cx,
            int cy,
            float & path_cost,
            float & goal_cost,
            float & occ_cost,
            float & total_cost )
```

Compute the components and total cost for a map grid cell

参数

cx	The x coordinate of the cell in the map grid
cy	The y coordinate of the cell in the map grid
path_cost	Will be set to the path distance component of the cost function
goal_cost	Will be set to the goal distance component of the cost function
occ_cost	Will be set to the costmap value of the cell
total_cost	Will be set to the value of the overall cost function, taking into account the scaling parameters

返回

True if the cell is traversible and therefore a legal location for the robot to move to

5.81.3.4 getLocalGoal()

```
void base_local_planner::TrajectoryPlanner::getLocalGoal (
            double & x,
            double & y )
```

Accessor for the goal the robot is currently pursuing in world corrdinates

参数

x	Will be set to the x position of the local goal
y	Will be set to the y position of the local goal

5.81.3.5 scoreTrajectory()

```
double base_local_planner::TrajectoryPlanner::scoreTrajectory (
             double x,
             double y,
             double theta,
             double vx,
             double vy,
             double vtheta,
             double vx_samp,
             double vy_samp,
             double vtheta_samp )
```

Generate and score a single trajectory

参数

x	The x position of the robot
y	The y position of the robot
theta	The orientation of the robot
vx	The x velocity of the robot
vy	The y velocity of the robot
vtheta	The theta velocity of the robot
vx_samp	The x velocity used to seed the trajectory
vy_samp	The y velocity used to seed the trajectory
vtheta_samp	The theta velocity used to seed the trajectory

返回

The score (as double)

5.81.3.6 updatePlan()

```
void base_local_planner::TrajectoryPlanner::updatePlan (
             const std::vector< geometry_msgs::PoseStamped > & new_plan,
             bool compute_dists = false )
```

Update the plan that the controller is following

参数

new_plan	A new plan for the controller to follow
compute_dists	Wheter or not to compute path/goal distances when a plan is updated

该类的文档由以下文件生成:

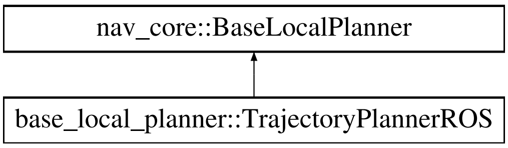
- base_local_planner/include/base_local_planner/trajectory_planner.h

5.82 base_local_planner::TrajectoryPlannerROS类 参考

A ROS wrapper for the trajectory controller that queries the param server to construct a controller

`#include <trajectory_planner_ros.h>`

类 base_local_planner::TrajectoryPlannerROS 继承关系图:


nav_core::BaseLocalPlanner
base_local_planner::TrajectoryPlannerROS

Public 成员函数

- TrajectoryPlannerROS ()

  Default constructor for the ros wrapper
- TrajectoryPlannerROS (std::string name, tf2_ros::Buffer *tf, costmap_2d::Costmap2DROS *costmap_ros)

  Constructs the ros wrapper
- void initialize (std::string name, tf2_ros::Buffer *tf, costmap_2d::Costmap2DROS *costmap_ros)

  Constructs the ros wrapper
- ∼TrajectoryPlannerROS ()

  Destructor for the wrapper
- bool computeVelocityCommands (geometry_msgs::Twist &cmd_vel)

  Given the current position, orientation, and velocity of the robot, compute velocity commands to send to the base
- bool setPlan (const std::vector< geometry_msgs::PoseStamped > &orig_global_plan)

  Set the plan that the controller is following
- bool isGoalReached ()

  Check if the goal pose has been achieved
- bool checkTrajectory (double vx_samp, double vy_samp, double vtheta_samp, bool update_map=true)

  Generate and score a single trajectory
- double scoreTrajectory (double vx_samp, double vy_samp, double vtheta_samp, bool update_map=true)

  Generate and score a single trajectory
- bool **isInitialized** ()
- TrajectoryPlanner * getPlanner () const

  Return the inner TrajectoryPlanner object. Only valid after initialize().

5.82.1 详细描述

A ROS wrapper for the trajectory controller that queries the param server to construct a controller

5.82.2 构造及析构函数说明

5.82.2.1 TrajectoryPlannerROS()

```
base_local_planner::TrajectoryPlannerROS::TrajectoryPlannerROS (
            std::string name,
            tf2_ros::Buffer * tf,
            costmap_2d::Costmap2DROS * costmap_ros )
```

Constructs the ros wrapper

参数

name	The name to give this instance of the trajectory planner
tf	A pointer to a transform listener
costmap	The cost map to use for assigning costs to trajectories

5.82.3 成员函数说明

5.82.3.1 checkTrajectory()

```
bool base_local_planner::TrajectoryPlannerROS::checkTrajectory (
            double vx_samp,
            double vy_samp,
            double vtheta_samp,
            bool update_map = true )
```

Generate and score a single trajectory

参数

vx_samp	The x velocity used to seed the trajectory
vy_samp	The y velocity used to seed the trajectory
vtheta_samp	The theta velocity used to seed the trajectory
update_map	Whether or not to update the map for the planner when computing the legality of the trajectory, this is useful to set to false if you're going to be doing a lot of trajectory checking over a short period of time

返回

True if the trajectory is legal, false otherwise

5.82.3.2 computeVelocityCommands()

```
bool base_local_planner::TrajectoryPlannerROS::computeVelocityCommands (
            geometry_msgs::Twist & cmd_vel )  [virtual]
```

Given the current position, orientation, and velocity of the robot, compute velocity commands to send to the base

参数

cmd_vel	Will be filled with the velocity command to be passed to the robot base

返回

True if a valid trajectory was found, false otherwise

实现了 nav_core::BaseLocalPlanner.

5.82.3.3 initialize()

```
void base_local_planner::TrajectoryPlannerROS::initialize (
            std::string name,
            tf2_ros::Buffer * tf,
            costmap_2d::Costmap2DROS * costmap_ros ) [virtual]
```

Constructs the ros wrapper

参数

name	The name to give this instance of the trajectory planner
tf	A pointer to a transform listener
costmap	The cost map to use for assigning costs to trajectories

实现了 nav_core::BaseLocalPlanner.

5.82.3.4 isGoalReached()

```
bool base_local_planner::TrajectoryPlannerROS::isGoalReached ( ) [virtual]
```

Check if the goal pose has been achieved

返回

True if achieved, false otherwise

实现了 nav_core::BaseLocalPlanner.

5.82.3.5 scoreTrajectory()

```
double base_local_planner::TrajectoryPlannerROS::scoreTrajectory (
            double vx_samp,
            double vy_samp,
            double vtheta_samp,
            bool update_map = true )
```

Generate and score a single trajectory

参数

vx_samp	The x velocity used to seed the trajectory
vy_samp	The y velocity used to seed the trajectory
vtheta_samp	The theta velocity used to seed the trajectory
update_map	Whether or not to update the map for the planner when computing the legality of the trajectory, this is useful to set to false if you're going to be doing a lot of trajectory checking over a short period of time

返回

score of trajectory (double)

5.82.3.6 setPlan()

```
bool base_local_planner::TrajectoryPlannerROS::setPlan (
            const std::vector< geometry_msgs::PoseStamped > & orig_global_plan )  [virtual]
```

Set the plan that the controller is following

参数

orig_global_plan	The plan to pass to the controller

返回

True if the plan was updated successfully, false otherwise

实现了 nav_core::BaseLocalPlanner.

该类的文档由以下文件生成:

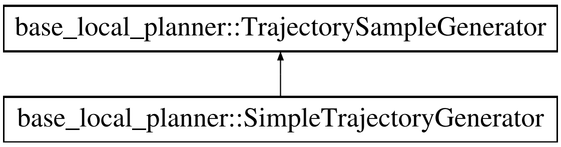
- base_local_planner/include/base_local_planner/trajectory_planner_ros.h

5.83 base_local_planner::TrajectorySampleGenerator类 参考

Provides an interface for navigation trajectory generators

```
#include <trajectory_sample_generator.h>
```

类 base_local_planner::TrajectorySampleGenerator 继承关系图:


base_local_planner::TrajectorySampleGenerator
base_local_planner::SimpleTrajectoryGenerator

Public 成员函数

- virtual bool hasMoreTrajectories ()=0
- virtual bool nextTrajectory (Trajectory &traj)=0
- virtual ~TrajectorySampleGenerator ()

  Virtual destructor for the interface

5.83.1 详细描述

Provides an interface for navigation trajectory generators

5.83.2 成员函数说明

5.83.2.1 hasMoreTrajectories()

```
virtual bool base_local_planner::TrajectorySampleGenerator::hasMoreTrajectories ( )  [pure
virtual]
```

Whether this generator can create more trajectories

在 base_local_planner::SimpleTrajectoryGenerator 内被实现.

5.83.2.2 nextTrajectory()

```
virtual bool base_local_planner::TrajectorySampleGenerator::nextTrajectory (
            Trajectory & traj )  [pure virtual]
```

Whether this generator can create more trajectories

在 base_local_planner::SimpleTrajectoryGenerator 内被实现.

该类的文档由以下文件生成:

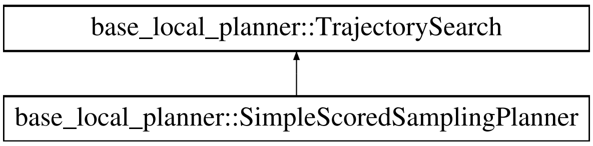
- base_local_planner/include/base_local_planner/trajectory_sample_generator.h

5.84 base_local_planner::TrajectorySearch类 参考

Interface for modules finding a trajectory to use for navigation commands next

```
#include <trajectory_search.h>
```

类 base_local_planner::TrajectorySearch 继承关系图:


base_local_planner::TrajectorySearch
base_local_planner::SimpleScoredSamplingPlanner

Public 成员函数

- virtual bool findBestTrajectory (Trajectory &traj, std::vector< Trajectory > ∗all_explored)=0

5.84.1 详细描述

Interface for modules finding a trajectory to use for navigation commands next

5.84.2 成员函数说明

5.84.2.1 findBestTrajectory()

```
virtual bool base_local_planner::TrajectorySearch::findBestTrajectory (
            Trajectory & traj,
            std::vector< Trajectory > * all_explored )  [pure virtual]
```

searches the space of allowed trajectory and returns one considered the optimal given the constraints of the particular search.

参数

traj	The container to write the result to
all_explored	pass NULL or a container to collect all trajectories for debugging (has a penalty)

在 base_local_planner::SimpleScoredSamplingPlanner 内被实现.

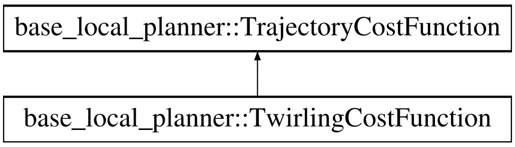
该类的文档由以下文件生成:

- base_local_planner/include/base_local_planner/trajectory_search.h

5.85 base_local_planner::TwirlingCostFunction类 参考

```
#include <twirling_cost_function.h>
```

类 base_local_planner::TwirlingCostFunction 继承关系图:


base_local_planner::TrajectoryCostFunction
base_local_planner::TwirlingCostFunction

Public 成员函数

- double scoreTrajectory (Trajectory &traj)
- bool prepare ()

额外继承的成员函数

5.85.1 详细描述

This class provides a cost based on how much a robot ”twirls” on its way to the goal. With differential-drive robots, there isn't a choice, but with holonomic or near-holonomic robots, sometimes a robot spins more than you'd like on its way to a goal. This class provides a way to assign a penalty purely to rotational velocities.

5.85.2 成员函数说明

5.85.2.1 prepare()

```
bool base_local_planner::TwirlingCostFunction::prepare ( )  [inline], [virtual]
```

General updating of context values if required. Subclasses may overwrite. Return false in case there is any error.

实现了 base_local_planner::TrajectoryCostFunction.

5.85.2.2 scoreTrajectory()

```
double base_local_planner::TwirlingCostFunction::scoreTrajectory (
            Trajectory & traj )  [virtual]
```

return a score for trajectory traj

实现了 base_local_planner::TrajectoryCostFunction.

该类的文档由以下文件生成:

- base_local_planner/include/base_local_planner/twirling_cost_function.h

5.86 base_local_planner::VelocityIterator类 参考

```
#include <velocity_iterator.h>
```

Public 成员函数

- **VelocityIterator** (double min, double max, int num_samples)
- double **getVelocity** ()
- VelocityIterator & **operator++** (int)
- void **reset** ()
- bool **isFinished** ()

5.86.1 详细描述

We use the class to get even sized samples between min and max, inluding zero if it is not included (and range goes from negative to positive

该类的文档由以下文件生成:

- base_local_planner/include/base_local_planner/velocity_iterator.h

5.87 voxel_grid::VoxelGrid类 参考

Public 成员函数

- VoxelGrid (unsigned int size_x, unsigned int size_y, unsigned int size_z)

  Constructor for a voxel grid
- void resize (unsigned int size_x, unsigned int size_y, unsigned int size_z)

  Resizes a voxel grid to the desired size
- void **reset** ()
- uint32_t ∗ **getData** ()
- void **markVoxel** (unsigned int x, unsigned int y, unsigned int z)
- bool **markVoxelInMap** (unsigned int x, unsigned int y, unsigned int z, unsigned int marked_threshold)
- void **clearVoxel** (unsigned int x, unsigned int y, unsigned int z)
- void **clearVoxelColumn** (unsigned int index)
- void **clearVoxelInMap** (unsigned int x, unsigned int y, unsigned int z)
- bool **bitsBelowThreshold** (unsigned int n, unsigned int bit_threshold)
- void **markVoxelLine** (double x0, double y0, double z0, double x1, double y1, double z1, unsigned int max↩ _length=UINT_MAX)
- void **clearVoxelLine** (double x0, double y0, double z0, double x1, double y1, double z1, unsigned int max↩ _length=UINT_MAX)
- void **clearVoxelLineInMap** (double x0, double y0, double z0, double x1, double y1, double z1, unsigned char ∗map_2d, unsigned int unknown_threshold, unsigned int mark_threshold, unsigned char free_cost=0, unsigned char unknown_cost=255, unsigned int max_length=UINT_MAX)
- VoxelStatus **getVoxel** (unsigned int x, unsigned int y, unsigned int z)
- VoxelStatus **getVoxelColumn** (unsigned int x, unsigned int y, unsigned int unknown_threshold=0, unsigned int marked_threshold=0)
- void **printVoxelGrid** ()
- void **printColumnGrid** ()
- unsigned int **sizeX** ()
- unsigned int **sizeY** ()
- unsigned int **sizeZ** ()
- template<class ActionType >

  void **raytraceLine** (ActionType at, double x0, double y0, double z0, double x1, double y1, double z1, unsigned int max_length=UINT_MAX)

静态 Public 成员函数

- static unsigned int **numBits** (unsigned int n)
- static VoxelStatus **getVoxel** (unsigned int x, unsigned int y, unsigned int z, unsigned int size_x, unsigned int size_y, unsigned int size_z, const uint32_t ∗data)

5.87.1 构造及析构函数说明

5.87.1.1 VoxelGrid()

```
voxel_grid::VoxelGrid::VoxelGrid (
            unsigned int size_x,
            unsigned int size_y,
            unsigned int size_z )
```

Constructor for a voxel grid

参数

size_x	The x size of the grid
size_y	The y size of the grid
size_z	The z size of the grid, only sizes <= 16 are supported

5.87.2 成员函数说明

5.87.2.1 resize()

```
void voxel_grid::VoxelGrid::resize (
            unsigned int size_x,
            unsigned int size_y,
            unsigned int size_z )
```

Resizes a voxel grid to the desired size

参数

size_x	The x size of the grid
size_y	The y size of the grid
size_z	The z size of the grid, only sizes <= 16 are supported

该类的文档由以下文件生成:

- voxel_grid/include/voxel_grid/voxel_grid.h

5.88 VoxelGrid类 参考

A 3D grid structure that stores points as an integer array. X and Y index the array and Z selects which bit of the integer is used giving a limit of 16 vertical cells.

```
#include <voxel_grid.h>
```

5.88.1 详细描述

A 3D grid structure that stores points as an integer array. X and Y index the array and Z selects which bit of the integer is used giving a limit of 16 vertical cells.

该类的文档由以下文件生成:

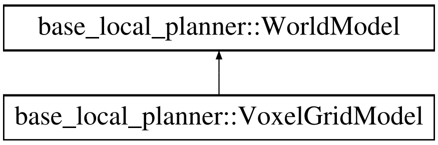
- voxel_grid/include/voxel_grid/voxel_grid.h

5.89 base_local_planner::VoxelGridModel类 参考

A class that implements the WorldModel interface to provide grid based collision checks for the trajectory controller using a 3D voxel grid.

```
#include <voxel_grid_model.h>
```

类 base_local_planner::VoxelGridModel 继承关系图:


base_local_planner::WorldModel
base_local_planner::VoxelGridModel

Public 成员函数

- VoxelGridModel (double size_x, double size_y, double size_z, double xy_resolution, double z_resolution, double origin_x, double origin_y, double origin_z, double max_z, double obstacle_range)

    Constructor for the VoxelGridModel
- virtual ∼VoxelGridModel ()

    Destructor for the world model
- virtual double footprintCost (const geometry_msgs::Point &position, const std::vector< geometry_msgs::Point > &footprint, double inscribed_radius, double circumscribed_radius)

    Checks if any obstacles in the voxel grid lie inside a convex footprint that is rasterized into the grid
- void updateWorld (const std::vector< geometry_msgs::Point > &footprint, const std::vector< costmap_2d::Observation > &observations, const std::vector< PlanarLaserScan > &laser_scans)

    The costmap already keeps track of world observations, so for this world model this method does nothing
- void getPoints (sensor_msgs::PointCloud2 &cloud)

    Function copying the Voxel points into a point cloud
- virtual double footprintCost (const geometry_msgs::Point &position, const std::vector< geometry_msgs::Point > &footprint, double inscribed_radius, double circumscribed_radius)=0

    Subclass will implement this method to check a footprint at a given position and orientation for legality in the world
- double **footprintCost** (double x, double y, double theta, const std::vector< geometry_msgs::Point > &footprint_spec, double inscribed_radius=0.0, double circumscribed_radius=0.0)
- double footprintCost (const geometry_msgs::Point &position, const std::vector< geometry_msgs::Point > &footprint, double inscribed_radius, double circumscribed_radius, double extra)

    Checks if any obstacles in the costmap lie inside a convex footprint that is rasterized into the grid

5.89.1 详细描述

A class that implements the WorldModel interface to provide grid based collision checks for the trajectory controller using a 3D voxel grid.

5.89.2 构造及析构函数说明

5.89.2.1 VoxelGridModel()

```
base_local_planner::VoxelGridModel::VoxelGridModel (
                double size_x,
                double size_y,
                double size_z,
                double xy_resolution,
                double z_resolution,
                double origin_x,
                double origin_y,
                double origin_z,
                double max_z,
                double obstacle_range )
```

Constructor for the VoxelGridModel

参数

size_x	The x size of the map
size_y	The y size of the map
size_z	The z size of the map... up to 32 cells
xy_resolution	The horizontal resolution of the map in meters/cell
z_resolution	The vertical resolution of the map in meters/cell
origin_x	The x value of the origin of the map
origin_y	The y value of the origin of the map
origin_z	The z value of the origin of the map
max_z	The maximum height for an obstacle to be added to the grid
obstacle_range	The maximum distance for obstacles to be added to the grid

5.89.3 成员函数说明

5.89.3.1 footprintCost() [1/3]

```
virtual double base_local_planner::VoxelGridModel::footprintCost (
            const geometry_msgs::Point & position,
            const std::vector< geometry_msgs::Point > & footprint,
            double inscribed_radius,
            double circumscribed_radius ) [virtual]
```

Checks if any obstacles in the voxel grid lie inside a convex footprint that is rasterized into the grid

参数

position	The position of the robot in world coordinates
footprint	The specification of the footprint of the robot in world coordinates
inscribed_radius	The radius of the inscribed circle of the robot
circumscribed_radius	The radius of the circumscribed circle of the robot

返回

Positive if all the points lie outside the footprint, negative otherwise

实现了 base_local_planner::WorldModel.

5.89.3.2 footprintCost() [2/3]

```
virtual double base_local_planner::WorldModel::footprintCost
```

Subclass will implement this method to check a footprint at a given position and orientation for legality in the world

参数

position	The position of the robot in world coordinates
footprint	The specification of the footprint of the robot in world coordinates
inscribed_radius	The radius of the inscribed circle of the robot
circumscribed_radius	The radius of the circumscribed circle of the robot

返回

> Positive if all the points lie outside the footprint, negative otherwise: -1 if footprint covers at least a lethal obstacle cell, or -2 if footprint covers at least a no-information cell, or -3 if footprint is partially or totally outside of the map

5.89.3.3 footprintCost() [3/3]

```
double base_local_planner::WorldModel::footprintCost  [inline]
```

Checks if any obstacles in the costmap lie inside a convex footprint that is rasterized into the grid

参数

position	The position of the robot in world coordinates
footprint	The specification of the footprint of the robot in world coordinates
inscribed_radius	The radius of the inscribed circle of the robot
circumscribed_radius	The radius of the circumscribed circle of the robot

返回

> Positive if all the points lie outside the footprint, negative otherwise

5.89.3.4 getPoints()

```
void base_local_planner::VoxelGridModel::getPoints (
             sensor_msgs::PointCloud2 & cloud )
```

Function copying the Voxel points into a point cloud

参数

cloud	the point cloud to copy data to. It has the usual x,y,z channels

5.89.3.5 updateWorld()

```
void base_local_planner::VoxelGridModel::updateWorld (
            const std::vector< geometry_msgs::Point > & footprint,
            const std::vector< costmap_2d::Observation > & observations,
            const std::vector< PlanarLaserScan > & laser_scans )
```

The costmap already keeps track of world observations, so for this world model this method does nothing

参数

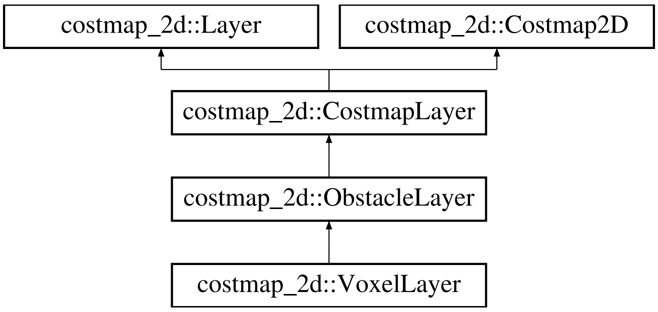
footprint	The footprint of the robot in its current location
observations	The observations from various sensors
laser_scan	The scans used to clear freespace

该类的文档由以下文件生成:

- base_local_planner/include/base_local_planner/voxel_grid_model.h

5.90 costmap_2d::VoxelLayer类 参考

类 costmap_2d::VoxelLayer 继承关系图:



Public 成员函数

- virtual void onInitialize ()

  This is called at the end of initialize(). Override to implement subclass-specific initialization.
- virtual void updateBounds (double robot_x, double robot_y, double robot_yaw, double ∗min_x, double ∗min_y, double ∗max_x, double ∗max_y)

  This is called by the LayeredCostmap to poll this plugin as to how much of the costmap it needs to update. Each layer can increase the size of this bounds.
- void updateOrigin (double new_origin_x, double new_origin_y)

  Move the origin of the costmap to a new location.... keeping data when it can
- bool **isDiscretized** ()
- virtual void matchSize ()

  Implement this to make this layer match the size of the parent costmap.
- virtual void **reset** ()

Protected 成员函数

- virtual void **setupDynamicReconfigure** (ros::NodeHandle &nh)
- virtual void resetMaps ()

  Resets the costmap and static_map to be unknown space

额外继承的成员函数

5.90.1 成员函数说明

5.90.1.1 onInitialize()

```
virtual void costmap_2d::VoxelLayer::onInitialize ( )  [virtual]
```

This is called at the end of initialize(). Override to implement subclass-specific initialization.

tf_, name_, and layered_costmap_ will all be set already when this is called.

重载 costmap_2d::ObstacleLayer .

5.90.1.2 updateBounds()

```
virtual void costmap_2d::VoxelLayer::updateBounds (
            double robot_x,
            double robot_y,
            double robot_yaw,
            double * min_x,
            double * min_y,
            double * max_x,
            double * max_y )  [virtual]
```

This is called by the LayeredCostmap to poll this plugin as to how much of the costmap it needs to update. Each layer can increase the size of this bounds.

For more details, see "Layered Costmaps for Context-Sensitive Navigation", by Lu et. Al, IROS 2014.

重载 costmap_2d::ObstacleLayer .

5.90.1.3 updateOrigin()

```
void costmap_2d::VoxelLayer::updateOrigin (
            double new_origin_x,
            double new_origin_y )  [virtual]
```

Move the origin of the costmap to a new location.... keeping data when it can

参数

new_↩ origin_x	The x coordinate of the new origin
new_↩ origin_y	The y coordinate of the new origin

重载 costmap_2d::Costmap2D .

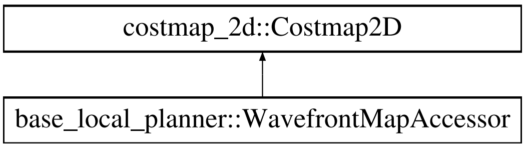
该类的文档由以下文件生成:

- costmap_2d/include/costmap_2d/voxel_layer.h

5.91 base_local_planner::WavefrontMapAccessor类 参考

```
#include <wavefront_map_accessor.h>
```

类 base_local_planner::WavefrontMapAccessor 继承关系图:



Public 成员函数

- **WavefrontMapAccessor** (MapGrid ∗map, double outer_radius)
- void **synchronize** ()

额外继承的成员函数

5.91.1 详细描述

Map_grids rely on costmaps to identify obstacles. We need a costmap that we can easily manipulate for unit tests. This class has a grid map where we can set grid cell state, and a synchronize method to make the costmap match.

该类的文档由以下文件生成:

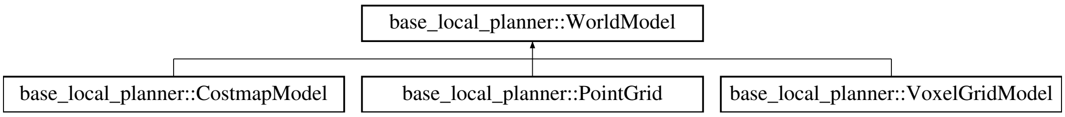
- base_local_planner/test/wavefront_map_accessor.h

5.92 base_local_planner::WorldModel类 参考

An interface the trajectory controller uses to interact with the world regardless of the underlying world model.

```
#include <world_model.h>
```

类 base_local_planner::WorldModel 继承关系图:


base_local_planner::WorldModel
base_local_planner::CostmapModel
base_local_planner::PointGrid
base_local_planner::VoxelGridModel

Public 成员函数

- virtual double footprintCost (const geometry_msgs::Point &position, const std::vector< geometry_msgs::Point > &footprint, double inscribed_radius, double circumscribed_radius)=0

  Subclass will implement this method to check a footprint at a given position and orientation for legality in the world
- double **footprintCost** (double x, double y, double theta, const std::vector< geometry_msgs::Point > &footprint_spec, double inscribed_radius=0.0, double circumscribed_radius=0.0)
- double footprintCost (const geometry_msgs::Point &position, const std::vector< geometry_msgs::Point > &footprint, double inscribed_radius, double circumscribed_radius, double extra)

  Checks if any obstacles in the costmap lie inside a convex footprint that is rasterized into the grid
- virtual ∼WorldModel ()

  Subclass will implement a destructor

5.92.1 详细描述

An interface the trajectory controller uses to interact with the world regardless of the underlying world model.

5.92.2 成员函数说明

5.92.2.1 footprintCost() [1/2]

```
virtual double base_local_planner::WorldModel::footprintCost (
            const geometry_msgs::Point & position,
            const std::vector< geometry_msgs::Point > & footprint,
            double inscribed_radius,
            double circumscribed_radius )  [pure virtual]
```

Subclass will implement this method to check a footprint at a given position and orientation for legality in the world

参数

position	The position of the robot in world coordinates
footprint	The specification of the footprint of the robot in world coordinates
inscribed_radius	The radius of the inscribed circle of the robot
circumscribed_radius	The radius of the circumscribed circle of the robot

返回

Positive if all the points lie outside the footprint, negative otherwise: -1 if footprint covers at least a lethal obstacle cell, or -2 if footprint covers at least a no-information cell, or -3 if footprint is partially or totally outside of the map

在 base_local_planner::VoxelGridModel, base_local_planner::PointGrid , 以及 base_local_planner::CostmapModel 内被实现.

5.92.2.2 footprintCost() [2/2]

```
double base_local_planner::WorldModel::footprintCost (
            const geometry_msgs::Point & position,
            const std::vector< geometry_msgs::Point > & footprint,
            double inscribed_radius,
            double circumscribed_radius,
            double extra ) [inline]
```

Checks if any obstacles in the costmap lie inside a convex footprint that is rasterized into the grid

参数

position	The position of the robot in world coordinates
footprint	The specification of the footprint of the robot in world coordinates
inscribed_radius	The radius of the inscribed circle of the robot
circumscribed_radius	The radius of the circumscribed circle of the robot

返回

Positive if all the points lie outside the footprint, negative otherwise

该类的文档由以下文件生成:

- base_local_planner/include/base_local_planner/world_model.h

Index