project_name

ver:1.0

制作者 Doxygen 1.9.1

Chapter 1

继承关系索引

1.1 类继承关系

此继承关系列表按字典顺序粗略的排序:

Chapter 2

结构体索引

2.1 结构体

这里列出了所有结构体，并附带简要说明:

Chapter 3

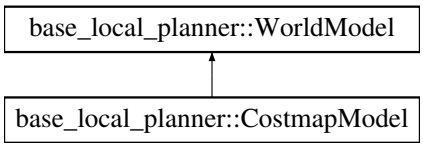
结构体说明

3.1 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:



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

3.1.1 详细描述

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

3.1.2 构造及析构函数说明

3.1.2.1 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

3.1.3 成员函数说明

3.1.3.1 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.

3.1.3.2 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

3.1.3.3 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

3.1.3.4 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

3.1.3.5 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:

- include/base_local_planner/costmap_model.h

3.2 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

3.2.1 成员函数说明

3.2.1.1 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

3.2.1.2 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

3.2.1.3 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:

- include/base_local_planner/footprint_helper.h

3.3 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)

3.3.1 成员函数说明

3.3.1.1 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

3.3.1.2 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:

- include/base_local_planner/latched_stop_rotate_controller.h

3.4 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

3.4.1 详细描述

An iterator implementing Bresenham Ray-Tracing.

该类的文档由以下文件生成:

- include/base_local_planner/line_iterator.h

3.5 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**

3.5.1 成员函数说明

3.5.1.1 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:

- include/base_local_planner/local_planner_limits.h

3.6 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** ()

3.6.1 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:

- include/base_local_planner/local_planner_util.h

3.7 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

3.7.1 详细描述

Stores path distance and goal distance information used for scoring trajectories

3.7.2 构造及析构函数说明

3.7.2.1 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:

- include/base_local_planner/map_cell.h

3.8 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

3.8.1 详细描述

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

3.8.2 构造及析构函数说明

3.8.2.1 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

3.8.2.2 MapGrid() [2/2]

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

Copy constructor for a MapGrid

参数

mg	The MapGrid to copy

3.8.3 成员函数说明

3.8.3.1 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

3.8.3.2 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

3.8.3.3 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

3.8.3.4 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

3.8.3.5 obstacleCosts()

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

return a value that indicates cell is in obstacle

3.8.3.6 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

3.8.3.7 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

3.8.3.8 operator=()

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

Assignment operator for a MapGrid

参数

mg	The MapGrid to assign from

3.8.3.9 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

3.8.3.10 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)

3.8.3.11 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

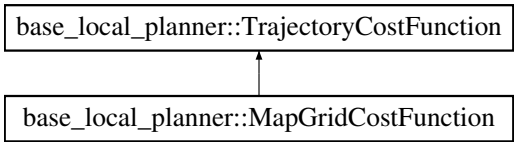
该类的文档由以下文件生成:

- include/base_local_planner/map_grid.h

3.9 base_local_planner::MapGridCostFunction类 参考

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

类 base_local_planner::MapGridCostFunction 继承关系图:



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

3.9.1 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

3.9.2 成员函数说明

3.9.2.1 obstacleCosts()

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

return a value that indicates cell is in obstacle

3.9.2.2 prepare()

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

propagate distances

实现了 base_local_planner::TrajectoryCostFunction.

3.9.2.3 scoreTrajectory()

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

return a score for trajectory traj

实现了 base_local_planner::TrajectoryCostFunction.

3.9.2.4 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.

3.9.2.5 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

3.9.2.6 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:

- include/base_local_planner/map_grid_cost_function.h

3.10 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.

3.10.1 成员函数说明

3.10.1.1 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:

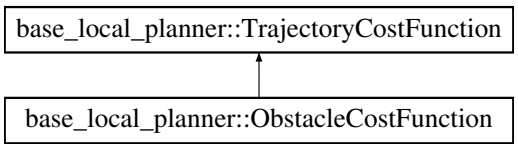
- include/base_local_planner/map_grid_visualizer.h

3.11 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:



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

3.11.1 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

3.11.2 成员函数说明

3.11.2.1 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.

3.11.2.2 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:

- include/base_local_planner/obstacle_cost_function.h

3.12 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.

3.12.1 构造及析构函数说明

3.12.1.1 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.

3.12.2 成员函数说明

3.12.2.1 odomCallback()

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

Callback for receiving odometry data

参数

msg	An Odometry message

3.12.2.2 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).

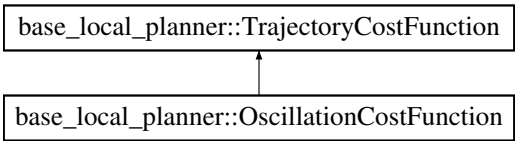
If odom_topic is the empty string, this just unsubscribes from the previous topic.

该类的文档由以下文件生成:

- include/base_local_planner/odometry_helper_ros.h

3.13 base_local_planner::OscillationCostFunction类 参考

类 base_local_planner::OscillationCostFunction 继承关系图:



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

3.13.1 成员函数说明

3.13.1.1 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.

3.13.1.2 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:

- include/base_local_planner/oscillation_cost_function.h

3.14 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**

3.14.1 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:

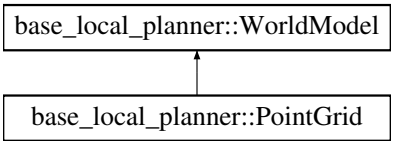
- include/base_local_planner/planar_laser_scan.h

3.15 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

3.15.1 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.

3.15.2 构造及析构函数说明

3.15.2.1 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

3.15.3 成员函数说明

3.15.3.1 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.

3.15.3.2 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

3.15.3.3 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

3.15.3.4 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

3.15.3.5 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

3.15.3.6 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

3.15.3.7 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

3.15.3.8 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

3.15.3.9 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

3.15.3.10 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

3.15.3.11 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

3.15.3.12 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
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

3.15.3.13 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

3.15.3.14 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

3.15.3.15 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

3.15.3.16 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

3.15.3.17 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

3.15.3.18 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

3.15.3.19 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

3.15.3.20 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

3.15.3.21 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

3.15.3.22 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

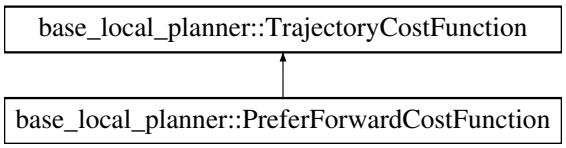
footprint	The footprint of the robot in its current location
observations	The observations from various sensors
laser_scans	The laser scans used to clear freespace (the point grid only uses the first scan which is assumed to be the base laser)

该类的文档由以下文件生成:

- include/base_local_planner/point_grid.h

3.16 base_local_planner::PreferForwardCostFunction类 参考

类 base_local_planner::PreferForwardCostFunction 继承关系图:



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

3.16.1 成员函数说明

3.16.1.1 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.

3.16.1.2 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:

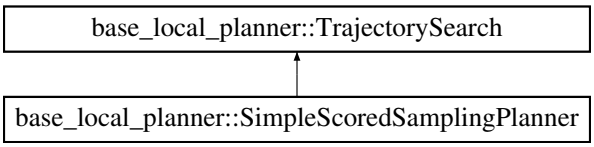
- include/base_local_planner/prefer_forward_cost_function.h

3.17 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:



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)

3.17.1 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.

3.17.2 构造及析构函数说明

3.17.2.1 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)

3.17.3 成员函数说明

3.17.3.1 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.

3.17.3.2 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

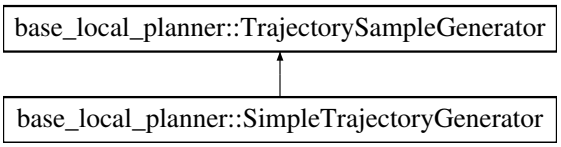
该类的文档由以下文件生成:

- include/base_local_planner/simple_scored_sampling_planner.h

3.18 base_local_planner::SimpleTrajectoryGenerator类 参考

`#include <simple_trajectory_generator.h>`

类 base_local_planner::SimpleTrajectoryGenerator 继承关系图:



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_**

3.18.1 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

3.18.2 成员函数说明

3.18.2.1 hasMoreTrajectories()

```
bool base_local_planner::SimpleTrajectoryGenerator::hasMoreTrajectories ( )  [virtual]
```

Whether this generator can create more trajectories

实现了 base_local_planner::TrajectorySampleGenerator.

3.18.2.2 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

3.18.2.3 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

3.18.2.4 nextTrajectory()

```
bool base_local_planner::SimpleTrajectoryGenerator::nextTrajectory (
            Trajectory & traj )  [virtual]
```

Whether this generator can create more trajectories

实现了 base_local_planner::TrajectorySampleGenerator.

3.18.2.5 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:

- include/base_local_planner/simple_trajectory_generator.h

3.19 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

3.19.1 详细描述

Holds a trajectory generated by considering an x, y, and theta velocity

3.19.2 构造及析构函数说明

3.19.2.1 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

3.19.3 成员函数说明

3.19.3.1 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

3.19.3.2 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

3.19.3.3 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

3.19.3.4 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

3.19.3.5 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:

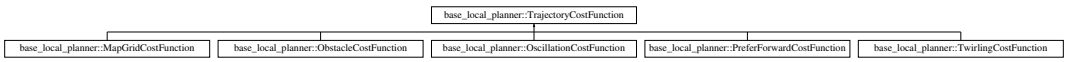
- include/base_local_planner/trajectory.h

3.20 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:



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)

3.20.1 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.

3.20.2 成员函数说明

3.20.2.1 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::OscillationCostFun
base_local_planner::ObstacleCostFunction , 以及 base_local_planner::MapGridCostFunction 内被实现.

3.20.2.2 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::OscillationCostFun
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:

- include/base_local_planner/trajectory_cost_function.h

3.21 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**

3.21.1 详细描述

Computes control velocities for a robot given a costmap, a plan, and the robot's position in the world.

3.21.2 构造及析构函数说明

3.21.2.1 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

参数

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

3.21.3 成员函数说明

3.21.3.1 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

3.21.3.2 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

3.21.3.3 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

3.21.3.4 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

3.21.3.5 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)

3.21.3.6 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:

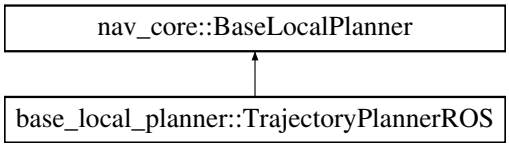
- include/base_local_planner/trajectory_planner.h

3.22 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:



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().

3.22.1 详细描述

A ROS wrapper for the trajectory controller that queries the param server to construct a controller

3.22.2 构造及析构函数说明

3.22.2.1 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

3.22.3 成员函数说明

3.22.3.1 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

3.22.3.2 computeVelocityCommands()

```
bool base_local_planner::TrajectoryPlannerROS::computeVelocityCommands (
            geometry_msgs::Twist & cmd_vel )
```

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

3.22.3.3 initialize()

```
void base_local_planner::TrajectoryPlannerROS::initialize (
            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

3.22.3.4 isGoalReached()

```
bool base_local_planner::TrajectoryPlannerROS::isGoalReached ( )
```

Check if the goal pose has been achieved

返回

True if achieved, false otherwise

3.22.3.5 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)

3.22.3.6 setPlan()

```
bool base_local_planner::TrajectoryPlannerROS::setPlan (
            const std::vector< geometry_msgs::PoseStamped > & orig_global_plan )
```

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

该类的文档由以下文件生成:

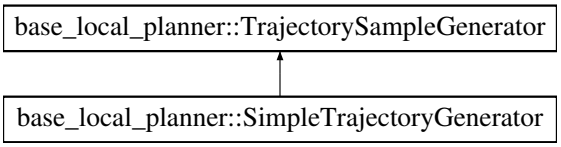
- include/base_local_planner/trajectory_planner_ros.h

3.23 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:



Public 成员函数

- virtual bool hasMoreTrajectories ()=0
- virtual bool nextTrajectory (Trajectory &traj)=0
- virtual ~TrajectorySampleGenerator ()

  Virtual destructor for the interface

3.23.1 详细描述

Provides an interface for navigation trajectory generators

3.23.2 成员函数说明

3.23.2.1 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.

3.23.2.2 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:

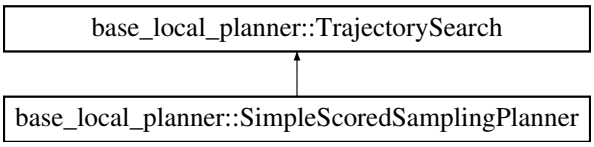
- include/base_local_planner/trajectory_sample_generator.h

3.24 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:



Public 成员函数

- virtual bool findBestTrajectory (Trajectory &traj, std::vector< Trajectory > *all_explored)=0

3.24.1 详细描述

Interface for modules finding a trajectory to use for navigation commands next

3.24.2 成员函数说明

3.24.2.1 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.

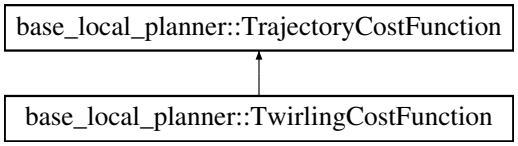
该类的文档由以下文件生成:

- include/base_local_planner/trajectory_search.h

3.25 base_local_planner::TwirlingCostFunction类 参考

`#include <twirling_cost_function.h>`

类 base_local_planner::TwirlingCostFunction 继承关系图:



Public 成员函数

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

额外继承的成员函数

3.25.1 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.

3.25.2 成员函数说明

3.25.2.1 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.

3.25.2.2 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:

- include/base_local_planner/twirling_cost_function.h

3.26 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** ()

3.26.1 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:

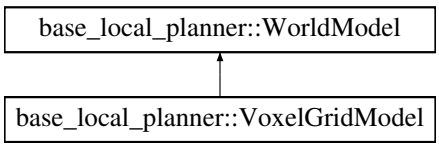
- include/base_local_planner/velocity_iterator.h

3.27 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:



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

3.27.1 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.

3.27.2 构造及析构函数说明

3.27.2.1 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

3.27.3 成员函数说明

3.27.3.1 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.

3.27.3.2 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

3.27.3.3 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

3.27.3.4 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

3.27.3.5 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

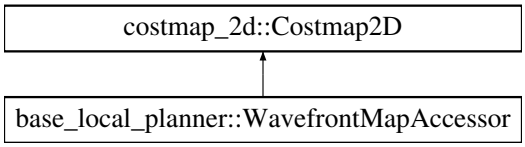
该类的文档由以下文件生成:

- include/base_local_planner/voxel_grid_model.h

3.28 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** ()

3.28.1 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:

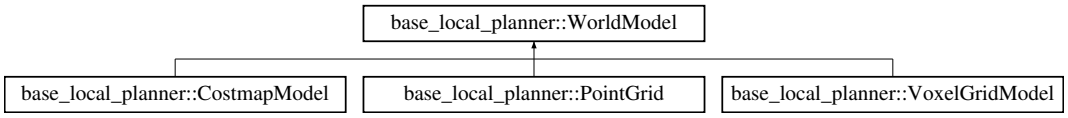
- test/wavefront_map_accessor.h

3.29 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:



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

3.29.1 详细描述

An interface the trajectory controller uses to interact with the world regardless of the underlying world model.

3.29.2 成员函数说明

3.29.2.1 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.

3.29.2.2 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:

- include/base_local_planner/world_model.h

Index