Robot planning

Generated by Doxygen 1.8.20

Chapter 1

Laboratory of Applied Robotics Student Interface

Package used by student to complete the assignment of the course.

1.1 Necessary libraries

- OMPL (The Open Motion Planning Library)

Install the OMPL lib for ROS with the following command on ubuntu as specified here:

```
sudo apt-get install ros-‘rosversion -d‘-ompl
```

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

Class Documentation

4.1 student::Dubins Class Reference

Public Member Functions

- std::vector< Pose > solveDubinsProblem (const RobotPosition &start, const RobotPosition &end, double kmax, float &minLength)

4.1.1 Member Function Documentation

4.1.1.1 solveDubinsProblem()

```
vector< Pose > student::Dubins::solveDubinsProblem (
             const RobotPosition & start,
             const RobotPosition & end,
             double kmax,
             float & minLength )
```

Solve a Dubins problem

Parameters

start	The starting position of the robot
end	The final position of the robot
kmax	The maximum curvature of the robot
minLength	The resulting minimum length of the path

Returns

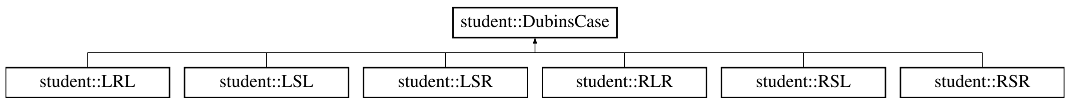
A path composed by a vector of Poses

The documentation for this class was generated from the following files:

- src/planning/dubins.hpp
- src/planning/dubins.cpp

4.2 student::DubinsCase Class Reference

Inheritance diagram for student::DubinsCase:



Public Member Functions

- **DubinsCase** (int _k1Sign, int _k2Sign, int _k3Sign)
- DubinsResult **solve** (const DubinsParams &params, float kmax)
- virtual DubinsResult **compute** (const DubinsParams &params)=0

Public Attributes

- float **k1**
- float **k2**
- float **k3**

The documentation for this class was generated from the following file:

- src/planning/dubins.hpp

4.3 student::DubinsMultipoint Class Reference

Public Member Functions

- **DubinsMultipoint** (int k, float startTheta, float kmax)
- void getShortestPath (const std::vector< Point > &path, Path &resultPath)

4.3.1 Member Function Documentation

4.3.1.1 getShortestPath()

```
void student::DubinsMultipoint::getShortestPath (
             const std::vector< Point > & path,
             Path & resultPath )
```

Using dynamic programming it calculates the best combination of Poses to follow the given path

Parameters

path	The path to follow
resultPath	The resulting vector of Poses

The documentation for this class was generated from the following files:

- src/planning/dubins_multipoint.hpp
- src/planning/dubins_multipoint.cpp

4.4 student::DubinsParams Struct Reference

Public Attributes

- float **theta_0**
- float **theta_f**
- float **k_max**
- float **lambda**

The documentation for this struct was generated from the following file:

- src/planning/dubins.hpp

4.5 student::DubinsResult Struct Reference

Public Member Functions

- float **getSum** ()
- void **scaleFromStandard** (const DubinsParams &params)

Public Attributes

- float **s1**
- float **s2**
- float **s3**

The documentation for this struct was generated from the following file:

- src/planning/dubins.hpp

4.6 boost::polygon::geometry_concept< student::VorPoint > Struct Reference

Public Types

- typedef point_concept **type**

The documentation for this struct was generated from the following file:

- src/planning/voronoi.hpp

4.7 boost::polygon::geometry_concept< student::VorSegment > Struct Reference

Public Types

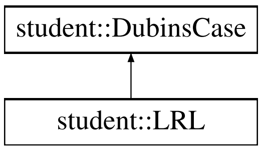
- typedef segment_concept **type**

The documentation for this struct was generated from the following file:

- src/planning/voronoi.hpp

4.8 student::LRL Class Reference

Inheritance diagram for student::LRL:



Public Member Functions

- DubinsResult compute (const DubinsParams &params) override

Additional Inherited Members

4.8.1 Member Function Documentation

4.8.1.1 compute()

```
DubinsResult student::LRL::compute (
            const DubinsParams & p )  [override], [virtual]
```

Solve the Dubins problem with a LRL movement

Parameters

p	The Dubins parameters

Returns

The solution of the Dubins problem

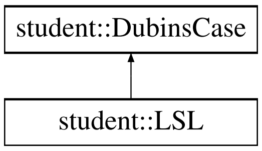
Implements student::DubinsCase.

The documentation for this class was generated from the following files:

- src/planning/dubins.hpp
- src/planning/dubins.cpp

4.9 student::LSL Class Reference

Inheritance diagram for student::LSL:



Public Member Functions

- DubinsResult compute (const DubinsParams &params) override

Additional Inherited Members

4.9.1 Member Function Documentation

4.9.1.1 compute()

```
DubinsResult student::LSL::compute (
             const DubinsParams & p )  [override], [virtual]
```

Solve the Dubins problem with a LSL movement

Parameters

p	The Dubins parameters

Returns

The solution of the Dubins problem

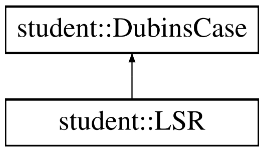
Implements student::DubinsCase.

The documentation for this class was generated from the following files:

- src/planning/dubins.hpp
- src/planning/dubins.cpp

4.10 student::LSR Class Reference

Inheritance diagram for student::LSR:



Public Member Functions

- DubinsResult compute (const DubinsParams &params) override

Additional Inherited Members

4.10.1 Member Function Documentation

4.10.1.1 compute()

```
DubinsResult student::LSR::compute (
              const DubinsParams & p )  [override], [virtual]
```

Solve the Dubins problem with a LSR movement

Parameters

p	The Dubins parameters

Returns

The solution of the Dubins problem

Implements student::DubinsCase.

The documentation for this class was generated from the following files:

- src/planning/dubins.hpp
- src/planning/dubins.cpp

4.11 boost::polygon::point_traits< student::VorPoint > Struct Reference

Public Types

- typedef int **coordinate_type**

Static Public Member Functions

- static coordinate_type **get** (const student::VorPoint &point, orientation_2d orient)

The documentation for this struct was generated from the following file:

- src/planning/voronoi.hpp

4.12 boost::polygon::point_traits< student::VorSegment > Struct Reference

Public Types

- typedef int **coordinate_type**
- typedef student::VorPoint **point_type**

Static Public Member Functions

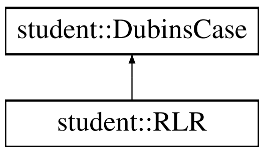
- static point_type **get** (const student::VorSegment &segment, direction_1d dir)

The documentation for this struct was generated from the following file:

- src/planning/voronoi.hpp

4.13 student::RLR Class Reference

Inheritance diagram for student::RLR:



Public Member Functions

- DubinsResult compute (const DubinsParams &params) override

Additional Inherited Members

4.13.1 Member Function Documentation

4.13.1.1 compute()

```
DubinsResult student::RLR::compute (
             const DubinsParams & p )  [override], [virtual]
```

Solve the Dubins problem with a RLR movement

Parameters

p	The Dubins parameters

Returns

The solution of the Dubins problem

Implements student::DubinsCase.

The documentation for this class was generated from the following files:

- src/planning/dubins.hpp
- src/planning/dubins.cpp

4.14 student::RobotPosition Struct Reference

Public Member Functions

- **RobotPosition** (float x, float y, float theta)
- **RobotPosition** (Point p, float theta)

Public Attributes

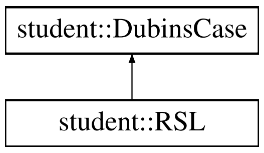
- float **x**
- float **y**
- float **theta**

The documentation for this struct was generated from the following file:

- src/planning/dubins.hpp

4.15 student::RSL Class Reference

Inheritance diagram for student::RSL:



Public Member Functions

- DubinsResult compute (const DubinsParams &params) override

Additional Inherited Members

4.15.1 Member Function Documentation

4.15.1.1 compute()

```
DubinsResult student::RSL::compute (
             const DubinsParams & p )  [override], [virtual]
```

Solve the Dubins problem with a RSL movement

Parameters

p	The Dubins parameters

Returns

The solution of the Dubins problem

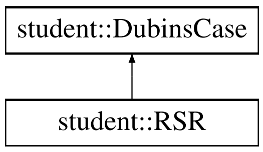
Implements student::DubinsCase.

The documentation for this class was generated from the following files:

- src/planning/dubins.hpp
- src/planning/dubins.cpp

4.16 student::RSR Class Reference

Inheritance diagram for student::RSR:



Public Member Functions

- DubinsResult compute (const DubinsParams &params) override

Additional Inherited Members

4.16.1 Member Function Documentation

4.16.1.1 compute()

```
DubinsResult student::RSR::compute (
             const DubinsParams & p )  [override], [virtual]
```

Solve the Dubins problem with a RSR movement

Parameters

p	The Dubins parameters

Returns

The solution of the Dubins problem

Implements student::DubinsCase.

The documentation for this class was generated from the following files:

- src/planning/dubins.hpp
- src/planning/dubins.cpp

4.17 Settings Class Reference

Public Types

- enum **Pattern** { **NOT_EXISTING**, **CHESSBOARD**, **CIRCLES_GRID**, **ASYMMETRIC_CIRCLES_GRID** }
- enum **InputType** { **INVALID**, **CAMERA**, **VIDEO_FILE**, **IMAGE_LIST** }

Public Member Functions

- void **write** (FileStorage &fs) const
- void **read** (const FileNode &node)
- void **validate** ()
- Mat **nextImage** ()

Static Public Member Functions

- static bool **readStringList** (const string &filename, vector$<$ string $>$ &l)
- static bool **isListOfImages** (const string &filename)

Public Attributes

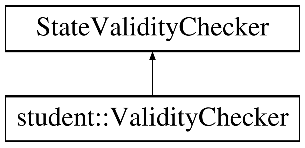
- Size **boardSize**
- Pattern **calibrationPattern**
- float **squareSize**
- int **nrFrames**
- float **aspectRatio**
- int **delay**
- bool **writePoints**
- bool **writeExtrinsics**
- bool **calibZeroTangentDist**
- bool **calibFixPrincipalPoint**
- bool **flipVertical**
- string **outputFileName**
- bool **showUndistorsed**
- string **input**
- bool **useFisheye**
- bool **fixK1**
- bool **fixK2**
- bool **fixK3**
- bool **fixK4**
- bool **fixK5**
- int **cameraID**
- vector$<$ string $>$ **imageList**
- size_t **atImageList**
- VideoCapture **inputCapture**
- InputType **inputType**
- bool **goodInput**
- int **flag**

The documentation for this class was generated from the following file:

- src/camera_calibration.cpp

4.18 student::ValidityChecker Class Reference

Inheritance diagram for student::ValidityChecker:



Public Member Functions

- **ValidityChecker** (const ob::SpaceInformationPtr &si, const std::vector< Polygon > &toAvoid, const Polygon &arenaBorders, const Point &targetPosition)
- bool isValid (const ob::State ∗state) const

Public Attributes

- std::vector< Polygon > **obstacles**
- Polygon **borders**
- Polygon **targetPolygon**

4.18.1 Detailed Description

Container class for the `isValid` method needed by the RRT∗ planning

4.18.2 Member Function Documentation

4.18.2.1 isValid()

```
bool student::ValidityChecker::isValid (
            const ob::State * state ) const  [inline]
```

Verify that the given point is outside of any obstacle polygon.

The documentation for this class was generated from the following file:

- src/planning/planning.cpp

4.19 student::VorPoint Struct Reference

Public Member Functions

- **VorPoint** (int a, int b)
- **VorPoint** (Point p)

Public Attributes

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

The documentation for this struct was generated from the following file:

- src/planning/voronoi.hpp

4.20 student::VorSegment Struct Reference

Public Types

- typedef int **coordinate_type**
- typedef student::VorPoint **point_type**

Public Member Functions

- **VorSegment** (int x1, int y1, int x2, int y2)
- VorPoint **get** (boost::polygon::direction_1d &dir) const

Public Attributes

- VorPoint **p0**
- VorPoint **p1**

The documentation for this struct was generated from the following file:

- src/planning/voronoi.hpp

Index