

## Robot planning

Generated by Doxygen 1.8.20



<b>1 Laboratory of Applied Robotics Student Interface</b>	<b>1</b>
1.1 Necessary libraries	1
<b>2 Hierarchical Index</b>	<b>3</b>
2.1 Class Hierarchy	3
<b>3 Class Index</b>	<b>5</b>
3.1 Class List	5
<b>4 Class Documentation</b>	<b>7</b>
4.1 student::Dubins Class Reference	7
4.1.1 Member Function Documentation	7
4.1.1.1 solveDubinsProblem()	7
4.2 student::DubinsCase Class Reference	8
4.3 student::DubinsMultipoint Class Reference	8
4.3.1 Member Function Documentation	8
4.3.1.1 getShortestPath()	8
4.4 student::DubinsParams Struct Reference	9
4.5 student::DubinsResult Struct Reference	9
4.6 boost::polygon::geometry_concept< student::VorPoint > Struct Reference	10
4.7 boost::polygon::geometry_concept< student::VorSegment > Struct Reference	10
4.8 student::LRL Class Reference	10
4.8.1 Member Function Documentation	10
4.8.1.1 compute()	10
4.9 student::LSL Class Reference	11
4.9.1 Member Function Documentation	11
4.9.1.1 compute()	11
4.10 student::LSR Class Reference	12
4.10.1 Member Function Documentation	12
4.10.1.1 compute()	12
4.11 boost::polygon::point_traits< student::VorPoint > Struct Reference	13
4.12 boost::polygon::point_traits< student::VorSegment > Struct Reference	13
4.13 student::RLR Class Reference	13
4.13.1 Member Function Documentation	14
4.13.1.1 compute()	14
4.14 student::RobotPosition Struct Reference	14
4.15 student::RSL Class Reference	15
4.15.1 Member Function Documentation	15
4.15.1.1 compute()	15
4.16 student::RSR Class Reference	15
4.16.1 Member Function Documentation	16
4.16.1.1 compute()	16
4.17 Settings Class Reference	16

4.18 student::ValidityChecker Class Reference . . . . .	18
4.18.1 Detailed Description . . . . .	18
4.18.2 Member Function Documentation . . . . .	18
4.18.2.1 isValid() . . . . .	18
4.19 student::VorPoint Struct Reference . . . . .	18
4.20 student::VorSegment Struct Reference . . . . .	19
<b>Index</b>	<b>21</b>

## 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:

student::Dubins	7
student::DubinsCase	8
student::LRL	10
student::LSL	11
student::LSR	12
student::RLR	13
student::RSL	15
student::RSR	15
student::DubinsMultipoint	8
student::DubinsParams	9
student::DubinsResult	9
boost::polygon::geometry_concept< student::VorPoint >	10
boost::polygon::geometry_concept< student::VorSegment >	10
boost::polygon::point_traits< student::VorPoint >	13
boost::polygon::point_traits< student::VorSegment >	13
student::RobotPosition	14
Settings	16
StateValidityChecker	
student::ValidityChecker	18
student::VorPoint	18
student::VorSegment	19





## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">student::Dubins</a>	7
<a href="#">student::DubinsCase</a>	8
<a href="#">student::DubinsMultipoint</a>	8
<a href="#">student::DubinsParams</a>	9
<a href="#">student::DubinsResult</a>	9
<a href="#">boost::polygon::geometry_concept&lt; student::VorPoint &gt;</a>	10
<a href="#">boost::polygon::geometry_concept&lt; student::VorSegment &gt;</a>	10
<a href="#">student::LRL</a>	10
<a href="#">student::LSL</a>	11
<a href="#">student::LSR</a>	12
<a href="#">boost::polygon::point_traits&lt; student::VorPoint &gt;</a>	13
<a href="#">boost::polygon::point_traits&lt; student::VorSegment &gt;</a>	13
<a href="#">student::RLR</a>	13
<a href="#">student::RobotPosition</a>	14
<a href="#">student::RSL</a>	15
<a href="#">student::RSR</a>	15
<a href="#">Settings</a>	16
<a href="#">student::ValidityChecker</a>	18
<a href="#">student::VorPoint</a>	18
<a href="#">student::VorSegment</a>	19



## 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

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

#### Returns

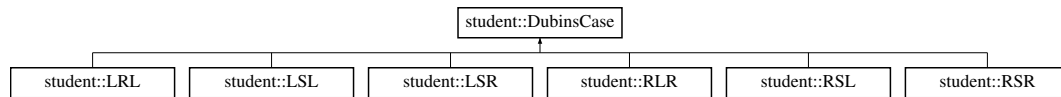
A path composed by a vector of Poses

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

<i>path</i>	The path to follow
<i>resultPath</i>	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

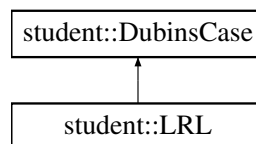
- `typedef segment_concept type`

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 <a href="#">Dubins</a> parameters
-----	---------------------------------------

## Returns

The solution of the [Dubins](#) problem

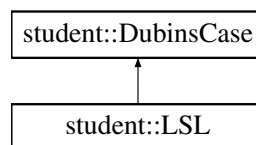
Implements [student::DubinsCase](#).

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 <a href="#">Dubins</a> parameters
-----	---------------------------------------

**Returns**

The solution of the [Dubins](#) problem

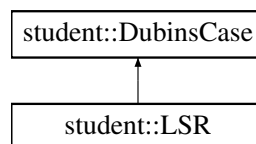
Implements [student::DubinsCase](#).

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**

<i>p</i>	The <a href="#">Dubins</a> 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

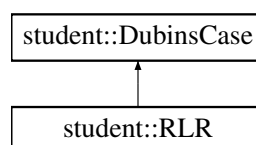
- static [point\\_type](#) **get** (const [student::VorSegment](#) &segment, direction\_1d dir)

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 <a href="#">Dubins</a> 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

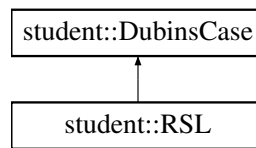
- float **x**
- float **y**
- float **theta**

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 <a href="#">Dubins</a> parameters
-----	---------------------------------------

#### Returns

The solution of the [Dubins](#) problem

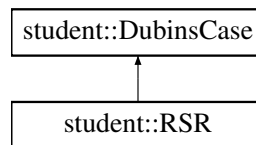
Implements [student::DubinsCase](#).

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 <a href="#">Dubins</a> 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

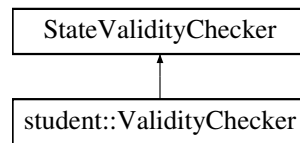
- Size **boardSize**
- Pattern **calibrationPattern**
- float **squareSize**
- int **nrFrames**
- float **aspectRatio**
- int **delay**
- bool **writePoints**
- bool **writeExtrinsics**
- bool **calibZeroTangentDist**
- bool **calibFixPrincipalPoint**
- bool **flipVertical**
- string **outputFileName**
- bool **showUndistorted**
- 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

boost::polygon::geometry\_concept< student::VorPoint  
    >, [10](#)  
boost::polygon::geometry\_concept< student::VorSegment  
    >, [10](#)  
boost::polygon::point\_traits< student::VorPoint >, [13](#)  
boost::polygon::point\_traits< student::VorSegment >,  
    [13](#)

compute  
    student::LRL, [10](#)  
    student::LSL, [11](#)  
    student::LSR, [12](#)  
    student::RLR, [14](#)  
    student::RSL, [15](#)  
    student::RSR, [16](#)

getShortestPath  
    student::DubinsMultipoint, [8](#)

isValid  
    student::ValidityChecker, [18](#)

Settings, [16](#)  
solveDubinsProblem  
    student::Dubins, [7](#)  
student::Dubins, [7](#)  
    solveDubinsProblem, [7](#)  
student::DubinsCase, [8](#)  
student::DubinsMultipoint, [8](#)  
    getShortestPath, [8](#)  
student::DubinsParams, [9](#)  
student::DubinsResult, [9](#)  
student::LRL, [10](#)  
    compute, [10](#)  
student::LSL, [11](#)  
    compute, [11](#)  
student::LSR, [12](#)  
    compute, [12](#)  
student::RLR, [13](#)  
    compute, [14](#)  
student::RobotPosition, [14](#)  
student::RSL, [15](#)  
    compute, [15](#)  
student::RSR, [15](#)  
    compute, [16](#)  
student::ValidityChecker, [18](#)  
    isValid, [18](#)  
student::VorPoint, [18](#)  
student::VorSegment, [19](#)