Assignment1_Group19

Generated by Doxygen 1.8.17

Chapter 1

IR-Project

Intelligent Robotics project - UniPD - 2023/2024

For start the assignment 1 follow these instructions:

- Open a terminal and go to `catkin_ws` folder
- Run the command `catkin build`
- Run the command `roslaunch tiago_iaslab_simulation start_simulation.launch world_name:=robotics_library` on the first terminal
- Open another terminal and run the command `roslaunch tiago_iaslab_simulation navigation.launch`
- Open the third terminal go to `catkin_ws` folder and run `rosrun assignment_1 goal_receiver`
- Open the fourth terminal go again to `catkin_ws` folder and run `rosrun assignment_1 goal_sender`
- Now you can add the coordinate in the following way `12 0` or `12 -3` and so on where the first number is the x coordinate and the second number is the y coordinate
- To stop the program enter `q` end press `ENTER` on the terminal where you run `rosrun assignment_1 goal_sender`

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

Class Documentation

3.1 CartesianPoint Class Reference

CartesianPoint class.

`#include <cartesian_point.h>`

Public Member Functions

- CartesianPoint (double x=0, double y=0)
  Constructor for the CartesianPoint class.
- double getX () const
  Get the x coordinate.
- double getY () const
  Get the y coordinate.
- void setCartesian (double x, double y)
  Set the x and y coordinates.
- void setX (double x)
  Set the x coordinate.
- void setY (double y)
  Set the y coordinate.
- CartesianPoint & shift (CartesianPoint to_add)
  Shift the point by a given point.
- CartesianPoint & rotate (double angle_radians)
  Rotate the point by a given angle.
- PolarPoint to_polar ()
  Convert the point to polar coordinates.

Static Public Member Functions

- static CartesianPoint fromPolar (PolarPoint p)
  Convert a polar point to a cartesian point.
- static CartesianPoint middlePoint (CartesianPoint a, CartesianPoint b)
  Get the middle point between two points.
- static double distance (CartesianPoint a, CartesianPoint b)
  Get the distance between two points.

Friends

- std::ostream & operator<< (std::ostream &os, const CartesianPoint &point)

  Overload of the << operator for CartesianPoint objects.
- std::ostream & operator<< (std::ostream &os, const std::vector< CartesianPoint > &points)

  Overload of the << operator for std::vector<CartesianPoint> objects.

3.1.1 Detailed Description

CartesianPoint class.

This class is used to represent a point in the cartesian space. It stores the x and y coordinates of the point.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 CartesianPoint()

```
CartesianPoint::CartesianPoint (
            double x = 0,
            double y = 0 )
```

Constructor for the CartesianPoint class.

Parameters

x	x coordinate
y	y coordinate

3.1.3 Member Function Documentation

3.1.3.1 distance()

```
double CartesianPoint::distance (
            CartesianPoint a,
            CartesianPoint b )  [static]
```

Get the distance between two points.

Parameters

a	first point
b	second point

Returns

distance between the two points

3.1.3.2 fromPolar()

```
CartesianPoint CartesianPoint::fromPolar (
            PolarPoint p )  [static]
```

Convert a polar point to a cartesian point.

Get the polar coordinates of the point.

Parameters

p	PolarPoint object

Returns

CartesianPoint object

PolarPoint object

3.1.3.3 getX()

```
double CartesianPoint::getX ( ) const
```

Get the x coordinate.

Returns

x coordinate

3.1.3.4 getY()

```
double CartesianPoint::getY ( ) const
```

Get the y coordinate.

Returns

y coordinate

3.1.3.5 middlePoint()

```
CartesianPoint CartesianPoint::middlePoint (
            CartesianPoint a,
            CartesianPoint b )  [static]
```

Get the middle point between two points.

Parameters

a	first point
b	second point

Returns

middle point

3.1.3.6 rotate()

```
CartesianPoint & CartesianPoint::rotate (
            double angle_radians )
```

Rotate the point by a given angle.

Parameters

angle_radians	angle in radians

Returns

reference to the rotated point

3.1.3.7 setCartesian()

```
void CartesianPoint::setCartesian (
            double x,
            double y )
```

Set the x and y coordinates.

Parameters

x	x coordinate
y	y coordinate

3.1.3.8 setX()

```
void CartesianPoint::setX (
            double x )
```

Set the x coordinate.

Parameters

x	x coordinate

3.1.3.9 setY()

```
void CartesianPoint::setY (
            double y )
```

Set the y coordinate.

Parameters

y	y coordinate

3.1.3.10 shift()

```
CartesianPoint & CartesianPoint::shift (
            CartesianPoint to_add )
```

Shift the point by a given point.

Parameters

to_add	point to add

Returns

reference to the shifted point

3.1.3.11 to_polar()

```
PolarPoint CartesianPoint::to_polar ( )
```

Convert the point to polar coordinates.

Get the polar coordinates of the point.

Returns

PolarPoint object

3.1.4 Friends And Related Function Documentation

3.1.4.1 operator<< [1/2]

```
std::ostream& operator<< (
            std::ostream & os,
            const CartesianPoint & point )  [friend]
```

Overload of the << operator for CartesianPoint objects.

Parameters

os	output stream
point	CartesianPoint object

Returns

output stream

3.1.4.2 operator<< [2/2]

```
std::ostream& operator<< (
            std::ostream & os,
            const std::vector< CartesianPoint > & points )  [friend]
```

Overload of the << operator for std::vector<CartesianPoint> objects.

Parameters

os	output stream
points	vector of CartesianPoint objects

Returns

output stream

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

- cartesian_point.h
- cartesian_point.cpp

3.2 DetectionAction Class Reference

detection action server class

`#include <detection_action.h>`

Public Member Functions

- DetectionAction (std::string name, ros::NodeHandle nh)

  Constructor for the detection action server.
- void executeCB (const assignment_1::DetectionGoalConstPtr &goal)

  Callback function for the detection action server.

3.2.1 Detailed Description

detection action server class

This class implements the detection action server. It is used to send the robot to a given position and detect the cylinders in the environment. The action server sends feedback messages to the client with the current position of the robot and the detected cylinders. It also sends a result message when the robot reaches the goal position.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 DetectionAction()

```
DetectionAction::DetectionAction (
            std::string name,
            ros::NodeHandle nh )
```

Constructor for the detection action server.

Parameters

name	Name of the action server
nh	ROS node handle

3.2.3 Member Function Documentation

3.2.3.1 executeCB()

```
void DetectionAction::executeCB (
            const assignment_1::DetectionGoalConstPtr & goal )
```

Callback function for the detection action server.

This function is called every time a new goal is received by the action server. It sends the robot to the goal position and detects the cylinders in the environment. It sends feedback messages to the client with the current position of the robot and the detected cylinders. It also sends a result message when the robot reaches the goal position.

Parameters

goal	Goal message

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

- detection_action.h
- detection_action.cpp

3.3 Obstacle Class Reference

Obstacle class.

`#include <obstacle.h>`

Public Types

- enum **Shape** { **Unknown**, **Cylinder** }

Public Member Functions

- Obstacle (std::vector< PolarPoint > profile=std::vector< PolarPoint >(), Position position=Position())
  Constructor for the Obstacle class.
- Shape getShape () const
  Get the Shape object.
- std::vector< PolarPoint > getProfile () const
  Get the Profile object.
- CartesianPoint getCenter () const
  Get the Center of the obstacle.
- double getRadius () const
  Get the radius of the obstacle.

Static Public Member Functions

- static std::vector< std::vector< PolarPoint > > getObstacleProfiles (const sensor_msgs::LaserScan &m, double threshold)
  Get the Obstacle Profiles object.

3.3.1 Detailed Description

Obstacle class.

This class is used to represent an obstacle in the environment. It stores the obstacle profile, the center and the radius of the obstacle. It also stores the position of the obstacle in the map.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Obstacle()

```
Obstacle::Obstacle (
            std::vector< PolarPoint > profile = std::vector<PolarPoint>(),
            Position position = Position() )
```

Constructor for the Obstacle class.

This constructor is used to create an obstacle with a given profile and position. The shape of the obstacle is identified using a chain of rules on the obstacle profile.

Parameters

profile	Obstacle profile
position	Position of the obstacle in the map

3.3.3 Member Function Documentation

3.3.3.1 getCenter()

```
CartesianPoint Obstacle::getCenter ( ) const
```

Get the Center of the obstacle.

Returns

Center of the obstacle (cartesian point)

3.3.3.2 getObstacleProfiles()

```
std::vector< std::vector< PolarPoint > > Obstacle::getObstacleProfiles (
            const sensor_msgs::LaserScan & m,
            double threshold ) [static]
```

Get the Obstacle Profiles object.

This function is used to extract the obstacle profiles from the laser scan message. Given a threshold, it identifies the obstacle profiles in the laser scan message. If the signal make a jump of more than the threshold, it is considered a new obstacle profile.

Parameters

m	Laser scan message
threshold	Threshold to identify the obstacle profiles

Returns

Vector of obstacle profiles

3.3.3.3 getProfile()

```
std::vector< PolarPoint > Obstacle::getProfile ( ) const
```

Get the Profile object.

Returns

Profile of the obstacle (vector of polar points)

3.3.3.4 getRadius()

```
double Obstacle::getRadius ( ) const
```

Get the radius of the obstacle.

Returns

Radius of the obstacle in meters

3.3.3.5 getShape()

```
Obstacle::Shape Obstacle::getShape ( ) const
```

Get the Shape object.

Returns

Shape enum class

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

- obstacle.h
- obstacle.cpp

3.4 PolarPoint Class Reference

This class represents a point in polar coordinates. It provides methods to convert to and from cartesian coordinates. and to calculate the average and median of a vector of points.

`#include <polar_point.h>`

Public Member Functions

- PolarPoint (double ang_rad=0, double dist=0)
  Constructor for a point in polar coordinates.
- double getAngleRadians () const
  Getter for the angle in radians.
- double getDistance () const
  Getter for the angle in degrees.
- void setPolar (double ang_rad, double dist)
  Setter for the polar coordinates.
- void setCartesian (double x, double y)
  Setter for the cartesian coordinates.
- void convertToCartesian (double &x, double &y) const
  Converts the point to cartesian coordinates.
- CartesianPoint **toCartesian** () const

Static Public Member Functions

- static PolarPoint getAveragePoint (const std::vector< PolarPoint > &points)
  Calculates the average point of a vector of points.
- static PolarPoint getMedianPoint (const std::vector< PolarPoint > &points)
  Calculates the median point of a vector of points.
- static double **distanceBetweenPoints** (const PolarPoint &point1, const PolarPoint &point2)
- static std::pair< PolarPoint, PolarPoint > **getClosestPoints** (std::vector< PolarPoint > &points)
- static std::pair< double, double > **getMiddlePoint** (const PolarPoint &point1, const PolarPoint &point2)

Friends

- std::ostream & operator<< (std::ostream &os, const PolarPoint &point)
  Output stream operator for a PolarPoint.
- std::ostream & operator<< (std::ostream &os, const std::vector< PolarPoint > &points)
  Output stream operator for a vector of PolarPoints.

3.4.1 Detailed Description

This class represents a point in polar coordinates. It provides methods to convert to and from cartesian coordinates. and to calculate the average and median of a vector of points.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 PolarPoint()

```
PolarPoint::PolarPoint (
            double ang_rad = 0,
            double dist = 0 )
```

Constructor for a point in polar coordinates.

Parameters

ang_rad	Angle in radians.
dist	Distance from the origin.

Returns

A PolarPoint object.

This constructor initializes the angle and distance of the point. The angle is given in radians and the distance is given in meters.

3.4.3 Member Function Documentation

3.4.3.1 convertToCartesian()

```
void PolarPoint::convertToCartesian (
            double & x,
            double & y ) const
```

Converts the point to cartesian coordinates.

Parameters

x	X coordinate.
y	Y coordinate.

This method converts the point to cartesian coordinates. The coordinates are given in meters.

3.4.3.2 getAngleRadians()

```
double PolarPoint::getAngleRadians ( ) const
```

Getter for the angle in radians.

Returns

The angle in radians.

3.4.3.3 getAveragePoint()

```
PolarPoint PolarPoint::getAveragePoint (
            const std::vector< PolarPoint > & points )  [static]
```

Calculates the average point of a vector of points.

Parameters

points	Vector of points.

Returns

The average point.

This method calculates the average point of a vector of points. The average point is calculated as the average of the angles and the average of the distances of the points in the vector.

3.4.3.4 getDistance()

```
double PolarPoint::getDistance ( ) const
```

Getter for the angle in degrees.

Returns

The angle in degrees.

3.4.3.5 getMedianPoint()

```
PolarPoint PolarPoint::getMedianPoint (
            const std::vector< PolarPoint > & points )  [static]
```

Calculates the median point of a vector of points.

Parameters

points	Vector of points.

Returns

The median point.

This method calculates the median point of a vector of points. The median point is calculated as the median of the angles and the median of the distances of the points in the vector.

3.4.3.6 setCartesian()

```
void PolarPoint::setCartesian (
            double x,
            double y )
```

Setter for the cartesian coordinates.

Parameters

x	X coordinate.
y	Y coordinate.

This method sets the x and y coordinates of the point. The coordinates are given in meters.

3.4.3.7 setPolar()

```
void PolarPoint::setPolar (
            double ang_rad,
            double dist )
```

Setter for the polar coordinates.

Parameters

ang_rad	Angle in radians.
dist	Distance from the origin.

This method sets the angle and distance of the point. The angle is given in radians and the distance is given in meters.

3.4.4 Friends And Related Function Documentation

3.4.4.1 operator<< [1/2]

```
std::ostream& operator<< (
            std::ostream & os,
            const PolarPoint & point )  [friend]
```

Output stream operator for a PolarPoint.

Parameters

os	Output stream.
point	Point to be printed.

Returns

The output stream.

This method overloads the output stream operator for a PolarPoint. It prints the point in the format [angle,distance].

3.4.4.2 operator<< [2/2]

```
std::ostream& operator<< (
            std::ostream & os,
            const std::vector< PolarPoint > & points )  [friend]
```

Output stream operator for a vector of PolarPoints.

Parameters

os	Output stream.
points	Vector of points to be printed.

Returns

The output stream.

This method overloads the output stream operator for a vector of PolarPoints. It prints the vector in the format {point1,point2,...}.

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

- polar_point.h
- polar_point.cpp

3.5 Position Class Reference

Position class.

```
#include <position.h>
```

Public Member Functions

- Position (CartesianPoint p=CartesianPoint(), double o=0)

  Constructor for the Position class.
- CartesianPoint getPoint () const

  Get the Point object.
- double getOrientation () const

  Get orientation in radiants.
- void setPoint (CartesianPoint p)

  Set the Point object.
- void setPoint (double x, double y)

  Set Point object using x and y coordinates.
- void setOrientation (double o=0)

  Set orientation in radiants.
- void setPosition (CartesianPoint p, double o=0)

  Set position in the cartesian space and orientation in radiants.
- void setPosition (double x, double y, double o=0)

  Set position in the cartesian space and orientation in radiants using x and y coordinates.

Static Public Member Functions

- static double sinCosToRad (double sin, double cos)

  get radian angle from sin and cos

Friends

- std::ostream & operator<< (std::ostream &os, const Position &pos)

  Shift the point by a given point.

3.5.1 Detailed Description

Position class.

This class is used to represent the position of the robot in the map. It stores the position of the robot in the cartesian space and its orientation.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 Position()

```
Position::Position (
            CartesianPoint p = CartesianPoint(),
            double o = 0 )
```

Constructor for the Position class.

Parameters

p	Position in the cartesian space
o	Orientation in radiants

3.5.3 Member Function Documentation

3.5.3.1 getOrientation()

```
double Position::getOrientation ( ) const
```

Get orientation in radiants.

Returns

Orientation in radiants

3.5.3.2 getPoint()

```
CartesianPoint Position::getPoint ( ) const
```

Get the Point object.

Returns

CartesianPoint

3.5.3.3 setOrientation()

```
void Position::setOrientation (
            double o = 0 )
```

Set orientation in radiants.

Set orientation in degrees.

Parameters

o	Orientation in radiants
o	Orientation in degrees

3.5.3.4 setPoint() [1/2]

```
void Position::setPoint (
            CartesianPoint p )
```

Set the Point object.

Parameters

p	Point using CartesianPoint

3.5.3.5 setPoint() [2/2]

```
void Position::setPoint (
            double x,
            double y )
```

Set Point object using x and y coordinates.

Parameters

x	x coordinate
y	y coordinate

3.5.3.6 setPosition() [1/2]

```
void Position::setPosition (
            CartesianPoint p,
            double o = 0 )
```

Set position in the cartesian space and orientation in radiants.

Parameters

p	Position in the cartesian space
o	Orientation in radiants

3.5.3.7 setPosition() [2/2]

```
void Position::setPosition (
            double x,
            double y,
            double o = 0 )
```

Set position in the cartesian space and orientation in radiants using x and y coordinates.

Parameters

x	x coordinate
y	y coordinate
o	Orientation in radiants

3.5.3.8 sinCosToRad()

```
double Position::sinCosToRad (
            double sin,
            double cos )  [static]
```

get radian angle from sin and cos

Parameters

sin	sin of the angle
cos	cos of the angle

Returns

double angle in radians

3.5.4 Friends And Related Function Documentation

3.5.4.1 operator<<

```
std::ostream& operator<< (
            std::ostream & os,
            const Position & pos )  [friend]
```

Shift the point by a given point.

Parameters

p	Point to shift by

Returns

Shifted point

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

- position.h
- position.cpp