Velodyne VLP16 Lidar Interface and Obstacle Finder

Generated by Doxygen 1.9.1

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

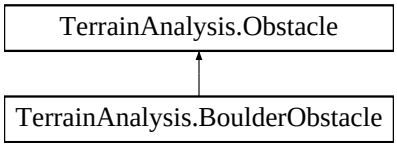
Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Class Documentation

3.1 TerrainAnalysis.BoulderObstacle Class Reference

Inheritance diagram for TerrainAnalysis.BoulderObstacle:



Public Member Functions

- **BoulderObstacle** (double[ ] coords, double groundRefIn)
- **BoulderObstacle** (int azimuth, int elevation, double distance, double groundRefIn)
- void updateCenterCoords ()
- double getArea ()
- void updateBounds (double[ ] coords)
- void updateBounds (int azimuth, int elevation, double distance)
- void combineObstacles (Obstacle o)
- obstacleType getType ()
- double[ ] getLeftMostPoint ()
- double[ ] getRightMostPoint ()
- double[ ] getFardestPoint ()
- double[ ] getClosestPoint ()
- boolean isSameAs (Obstacle o)
- String toString ()

Additional Inherited Members

3.1.1 Member Function Documentation

3.1.1.1 combineObstacles()

```
void TerrainAnalysis.BoulderObstacle.combineObstacles (
            Obstacle o )
```

Call this function to combine self with another Obstacle. This function expands the boundary box of self and destroys the other Object.

Parameters

o	Obstacle to combine with

Reimplemented from TerrainAnalysis.Obstacle.

3.1.1.2 getArea()

```
double TerrainAnalysis.BoulderObstacle.getArea ( )
```

Call function to get the area of the boundary box surrounding the Obstacle

Returns

Area of boundary box surrounding the Obtacle.

Reimplemented from TerrainAnalysis.Obstacle.

3.1.1.3 getClosestPoint()

```
double [] TerrainAnalysis.BoulderObstacle.getClosestPoint ( )
```

Get the closest (to rover) point of the boundary box surrounding the obstacle

Returns

double array containing the X, Y, and Z coordinates of the closest (to rover) boundary point

Reimplemented from TerrainAnalysis.Obstacle.

3.1.1.4 getFardestPoint()

```
double [] TerrainAnalysis.BoulderObstacle.getFardestPoint ( )
```

Get the farthest (from rover) point of the boundary box surrounding the obstacle

Returns

double array containing the X, Y, and Z coordinates of the farthest (from rover) boundary point

Reimplemented from TerrainAnalysis.Obstacle.

3.1.1.5 getLeftMostPoint()

```
double [] TerrainAnalysis.BoulderObstacle.getLeftMostPoint ( )
```

Get the left most point of the boundary box surrounding the obstacle

Returns

double array containing the X, Y, and Z coordinates of the left most boundary point

Reimplemented from TerrainAnalysis.Obstacle.

3.1.1.6 getRightMostPoint()

```
double [] TerrainAnalysis.BoulderObstacle.getRightMostPoint ( )
```

Get the right most point of the boundary box surrounding the obstacle

Returns

double array containing the X, Y, and Z coordinates of the right most boundary point

Reimplemented from TerrainAnalysis.Obstacle.

3.1.1.7 getType()

```
obstacleType TerrainAnalysis.BoulderObstacle.getType ( )
```

Get the type of Obstacle

Returns

obstacleType enumerator

Reimplemented from TerrainAnalysis.Obstacle.

3.1.1.8 isSameAs()

```
boolean TerrainAnalysis.BoulderObstacle.isSameAs (
            Obstacle o )
```

Call function on an Obstacle to compare it to another Obstacle. Obstacles are compared based on their type and boundary boxes.

Parameters

o	Obstacle to compare Self to

Returns

Boolean value indicating if the two Obstacles match

Reimplemented from TerrainAnalysis.Obstacle.

3.1.1.9 toString()

```
String TerrainAnalysis.BoulderObstacle.toString ( )
```

Function to retreive a string detailing the Obstacle. Type: Center X: Center Y: Center Z:

Returns

String detailing Obstacle

Reimplemented from TerrainAnalysis.Obstacle.

3.1.1.10 updateBounds() [1/2]

```
void TerrainAnalysis.BoulderObstacle.updateBounds (
            double[] coords )
```

Feed in a point represented as a double array which is determine to be part of the Obstacle. The function then uses this new found point to update boundary box.

Parameters

coords	point as a double array.

Reimplemented from TerrainAnalysis.Obstacle.

3.1.1.11 updateBounds() [2/2]

```
void TerrainAnalysis.BoulderObstacle.updateBounds (
            int azimuth,
            int elevation,
            double distance )
```

Update the boundary box of an Obstacle by using polar coordinates. The fed in values will then be used to calculate the pertaining point in cartician.

Parameters

azimuth	azimuth angle (degrees) as an integer and multiplied by 100 (ex: 10.20 -> 1020)
elevation	elevation angle (degrees) as an integer and multiplied by 100 (ex: 10.20 -> 1020)
distance	distance in meters as a double

Reimplemented from TerrainAnalysis.Obstacle.

3.1.1.12 updateCenterCoords()

```
void TerrainAnalysis.BoulderObstacle.updateCenterCoords ( )
```

Call function to update center coordinates based on the current bounds

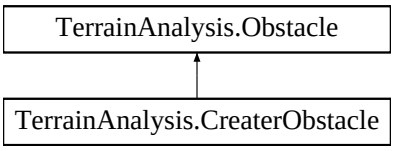
Reimplemented from TerrainAnalysis.Obstacle.

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

- src/TerrainAnalysis/BoulderObstacle.java

3.2 TerrainAnalysis.CreaterObstacle Class Reference

Inheritance diagram for TerrainAnalysis.CreaterObstacle:



Public Member Functions

- CreaterObstacle (double[ ] coords, double groundRefIn)
- CreaterObstacle (int azimuth, int elevation, double distance, double groundRefIn)
- void updateCenterCoords ()
- double getArea ()
- void updateBounds (double[ ] coords)
- void updateBounds (int azimuth, int elevation, double distance)
- void combineObstacles (Obstacle o)
- obstacleType getType ()
- double[ ] getLeftMostPoint ()
- double[ ] getRightMostPoint ()
- double[ ] getFardestPoint ()
- double[ ] getClosestPoint ()
- boolean isSameAs (Obstacle o)
- String toString ()

Additional Inherited Members

3.2.1 Detailed Description

CreaterObstacle class which extends the Obstacle class. It is meant to be used alongside BoulderObstacle. Both Obstacle types behave similarly except for how we draw the boundary box around it. For a creater, the boundary box is drawn arround the perimeter of the creater. For a bourder, we use it's height as the height box (meaning the box is vertical).

getterPoint() functions are called by other programs to outline boundary box surrounding obstacle

update() functions are called to update the bounds and height/depth of Obstacle depending on the new found point.

isSameAs() function is used to compare two obstacles and determe two Obstacles and determine if they should be combined or not

3.2.2 Constructor & Destructor Documentation

3.2.2.1 CreaterObstacle() [1/2]

```
TerrainAnalysis.CreaterObstacle.CreaterObstacle (
            double[] coords,
            double groundRefIn )
```

CreaterObstacle constructor. Assigns type as CREATER and updates boundary box to the initializer point. Use this function to initialize in cartician coordinates.

Parameters

coords	Initializer point. First point to be found as a possible Obstacle.
groundRefIn	Point to be used as the reference for ground.

3.2.2.2 CreaterObstacle() [2/2]

```
TerrainAnalysis.CreaterObstacle.CreaterObstacle (
            int azimuth,
            int elevation,
            double distance,
            double groundRefIn )
```

CreaterObstacle constructor. Assigns type as CREATER and updates boundary box to the initializer point. Use this function to initialize in polar cordinates which are then turned to cartician.

Parameters

azimuth	Azimuth angle, in degrees, at which the obstacle is first found.
elevation	Elevation angle, in degreees, at which the obstacle is first found.
distance	Distance, in meters, at which the obstacle is first found.
groundRefIn	Point to be used as the reference for ground.

3.2.3 Member Function Documentation

3.2.3.1 combineObstacles()

```
void TerrainAnalysis.CreaterObstacle.combineObstacles (
            Obstacle o )
```

If two obstacles are found to have intersecting boundary boxes, then combine both obstacles into a single one.

Parameters

o	Obstacle to be combined with.

Reimplemented from TerrainAnalysis.Obstacle.

3.2.3.2 getArea()

```
double TerrainAnalysis.CreaterObstacle.getArea ( )
```

Get the area of the boundary box surrounding the Obstacle.

Returns

Area of Obstacle as a double and in m^2

Reimplemented from TerrainAnalysis.Obstacle.

3.2.3.3 getClosestPoint()

```
double [] TerrainAnalysis.CreaterObstacle.getClosestPoint ( )
```

Get the closest (to rover) point of the boundary box surrounding the obstacle

Returns

double array containing the X, Y, and Z coordinates of the closest (to rover) boundary point

Reimplemented from TerrainAnalysis.Obstacle.

3.2.3.4 getFardestPoint()

```
double [] TerrainAnalysis.CreaterObstacle.getFardestPoint ( )
```

Get the farthest (from rover) point of the boundary box surrounding the obstacle

Returns

double array containing the X, Y, and Z coordinates of the farthest (from rover) boundary point

Reimplemented from TerrainAnalysis.Obstacle.

3.2.3.5 getLeftMostPoint()

```
double [] TerrainAnalysis.CreaterObstacle.getLeftMostPoint ( )
```

Get the left most point of the boundary box surrounding the obstacle

Returns

double array containing the X, Y, and Z coordinates of the left most boundary point

Reimplemented from TerrainAnalysis.Obstacle.

3.2.3.6 getRightMostPoint()

```
double [] TerrainAnalysis.CreaterObstacle.getRightMostPoint ( )
```

Get the right most point of the boundary box surrounding the obstacle

Returns

double array containing the X, Y, and Z coordinates of the right most boundary point

Reimplemented from TerrainAnalysis.Obstacle.

3.2.3.7 getType()

```
obstacleType TerrainAnalysis.CreaterObstacle.getType ( )
```

Get the type of Obstacle

Returns

Enumerator type inidicating a CreaterObstacle;

Reimplemented from TerrainAnalysis.Obstacle.

3.2.3.8 isSameAs()

```
boolean TerrainAnalysis.CreaterObstacle.isSameAs (
            Obstacle o )
```

Determine if two Obstacles are the same by seeing if their boundary boxes intersect.

Returns

Boolean value, True if the two obstacles are the same.

Reimplemented from TerrainAnalysis.Obstacle.

3.2.3.9 toString()

```
String TerrainAnalysis.CreaterObstacle.toString ( )
```

Turn Creater obstacle into a string for debugging:

Obstacle Type: X: Y: Z: Depth:

Returns

String detailing obstacle for debugging

Reimplemented from TerrainAnalysis.Obstacle.

3.2.3.10 updateBounds() [1/2]

```
void TerrainAnalysis.CreaterObstacle.updateBounds (
            double[] coords )
```

Use a new found point to update the boundary of the box. The box is extended to emcompass the new point.

Parameters

coords	New found point

Reimplemented from TerrainAnalysis.Obstacle.

3.2.3.11 updateBounds() [2/2]

```
void TerrainAnalysis.CreaterObstacle.updateBounds (
            int azimuth,
```

```
int elevation,
double distance )
```

Update the boundary box with a new found point but in polar cordinates. The coordinates are turned to cartician first.

Parameters

azimuth	Azimuth angle, in degrees, at which the obstacle is found
elevation	Elevation angle, in degrees, at which the obstacle is found
distance	Distance, in meters, at which the obstacle is found

Reimplemented from TerrainAnalysis.Obstacle.

3.2.3.12 updateCenterCoords()

```
void TerrainAnalysis.CreaterObstacle.updateCenterCoords ( )
```

Undate the center coordinates by taking the middle of the boundary box.

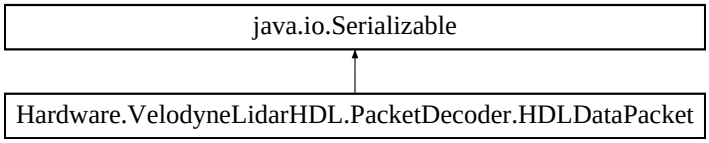
Reimplemented from TerrainAnalysis.Obstacle.

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

- src/TerrainAnalysis/CreaterObstacle.java

3.3 Hardware.VelodyneLidarHDL.PacketDecoder.HDLDataPacket Class Reference

Inheritance diagram for Hardware.VelodyneLidarHDL.PacketDecoder.HDLDataPacket:



Public Member Functions

- HDLDataPacket (byte[ ] rawData, int data_length)
- int getTimestamp ()
- void getFiringData (int blockID, HDLFiringData[ ] firingData)
- HDLFiringData getFiringData (int blockID)
- boolean addFiringData (HDLFiringData firingData)

3.3.1 Detailed Description

HDLDataPacket implements Serializable to ensure all data is continous. It is made up of HDLFiringData objects and serves to reprent a full packet broken down into its building blocks.

getTimeStamp() function returns the time at which the packet was created

getFiringData(blockId, firingData[out]) function returns the firing data for the indentifier provided.

getFirinfData(blockId) function returns the firind data, as a HDLFiringData object, for th indentifier provided.

addFiringData(HDLFiringData) function adds the already instantiated HDLFiring object to the ArrayList

3.3.2 Constructor & Destructor Documentation

3.3.2.1 HDLDataPacket()

```
Hardware.VelodyneLidarHDL.PacketDecoder.HDLDataPacket.HDLDataPacket (
            byte[] rawData,
            int data_length )
```

Construtor for creating an HDLDataPacket from a raw packet directly extracted from lidar socket

Parameters

rawData	byte array containing entire packet from socket
data_length	number of bytes contained in array

3.3.3 Member Function Documentation

3.3.3.1 addFiringData()

```
boolean Hardware.VelodyneLidarHDL.PacketDecoder.HDLDataPacket.addFiringData (
            HDLFiringData firingData )
```

Add a single firingData object to ArrayList if it contains less than HDL_FIRING_PER_PKT entries

Parameters

firingData	HDLFiringData to be added

Returns

Boolean value indicating if block was added or not

3.3.3.2 getFiringData() [1/2]

```
HDLFiringData Hardware.VelodyneLidarHDL.PacketDecoder.HDLDataPacket.getFiringData (
            int blockID )
```

Get firing data for an specific block (0 - 11). Returns as an actual HDLFiringData object which makes it slower

Parameters

blockID	Integer specifying which block within packet to get

Returns

HDLFiringData object requested

3.3.3.3 getFiringData() [2/2]

```
void Hardware.VelodyneLidarHDL.PacketDecoder.HDLDataPacket.getFiringData (
            int blockID,
            HDLFiringData[] firingData )
```

Get firing data for an specific block (0 - 11)

Parameters

blockID	integer specifying which block within packet to get
firingData	return value

3.3.3.4 getTimestamp()

```
int Hardware.VelodyneLidarHDL.PacketDecoder.HDLDataPacket.getTimestamp ( )
```

Get timestamp as a zero-padded integer

Returns

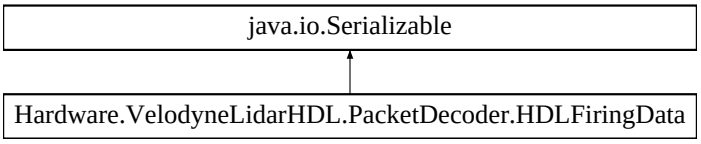
timestamp integer

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

- src/Hardware/VelodyneLidarHDL/PacketDecoder.java

3.4 Hardware.VelodyneLidarHDL.PacketDecoder.HDLFiringData Class Reference

Inheritance diagram for Hardware.VelodyneLidarHDL.PacketDecoder.HDLFiringData:



Public Member Functions

- HDLFiringData (byte[ ] rawData, int data_length)
- HDLFiringData (byte[ ] blockIdentifier, byte[ ] azimuthAngle)
- HDLFiringData (byte[ ] blockIdentifier, byte[ ] azimuthAngle, ArrayList< HDLLaserReturn > laserReturns)
- short getBlockIdentifier ()
- int getAzimuthAngle ()
- HDLLaserReturn getLaserReturn (int laserID)
- void addLaserReturn (HDLLaserReturn laserReturn)

3.4.1 Detailed Description

HDLFiringData class implements Serializable to assure data continuity. It is made up of HDLFiringReturns and serves as the building block for the HDLFirigPacket class.

getAzimuthAngle() function returns the azimuth angle at which the firing group was fired to produce the firing block

getLaserReturn(laserID) function returns the HDLLaserReturn object for the provided laser id

addLaserReturn(laserReturn) adds a single HDLLaserReturn object to the arrayList of laser returns

3.4.2 Constructor & Destructor Documentation

3.4.2.1 HDLFiringData() [1/3]

```
Hardware.VelodyneLidarHDL.PacketDecoder.HDLFiringData.HDLFiringData (
            byte[] rawData,
            int data_length )
```

Constructor for creating a HDLFiringData object from an array of bytes (usually a sub-array of packet)

Parameters

rawData	byte array containing all data needed to generate block
data_length	size of provided data array

3.4.2.2 HDLFiringData() [2/3]

```
Hardware.VelodyneLidarHDL.PacketDecoder.HDLFiringData.HDLFiringData (
            byte[] blockIdentifier,
            byte[] azimuthAngle )
```

Constructor for creaeting HDLFiringData object without any HDLLaserReturn objects within object. Function add↩ LaserReturn must then be used to populate array

Parameters

blockIdentifier	byte array containing the two bytes used to identify a new block
azimuthAngle	byte array containing the two bytes used to represent the azimuth angle (MSB first)

3.4.2.3 HDLFiringData() [3/3]

```
Hardware.VelodyneLidarHDL.PacketDecoder.HDLFiringData.HDLFiringData (
            byte[] blockIdentifier,
            byte[] azimuthAngle,
            ArrayList< HDLLaserReturn > laserReturns )
```

Constructor for creating a HDLFringData object with all of the HDLLaserReturn object already instantiated

Parameters

blockIdentifier	byte array containing the two bytes used to identify a new block
azimuthAngle	byte array containing the two bytes used to represent the azimuth angle (MSB first)
laserReturns	ArrayList containing all of the 32 laserReturn objects needed to make a full block

3.4.3 Member Function Documentation

3.4.3.1 addLaserReturn()

```
void Hardware.VelodyneLidarHDL.PacketDecoder.HDLFiringData.addLaserReturn (
            HDLLaserReturn laserReturn )
```

Add a single HDLLaseReturn to ArrayList if it contains less than HDL_LASER_PER_FIRING

Parameters

laserReturn	single HDLLaseReturn to be added

3.4.3.2 getAzimuthAngle()

```
int Hardware.VelodyneLidarHDL.PacketDecoder.HDLFiringData.getAzimuthAngle ( )
```

Function to get azimuth angle at which the firing group was fired to create block

Returns

Azimuth angle as an integer

3.4.3.3 getBlockIdentifier()

```
short Hardware.VelodyneLidarHDL.PacketDecoder.HDLFiringData.getBlockIdentifier ( )
```

Function to get block identifier as an unsigned char

Returns

block indentifier as an unsinged char

3.4.3.4 getLaserReturn()

```
HDLLaserReturn Hardware.VelodyneLidarHDL.PacketDecoder.HDLFiringData.getLaserReturn (
            int laserID )
```

Get a single laser return within the firing block

Parameters

laserID	indetifier for requested laser return

Returns

single laser return as an HDLLaserReturn object

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

- src/Hardware/VelodyneLidarHDL/PacketDecoder.java

3.5 Hardware.VelodyneLidarHDL.PacketDecoder.HDLFrame Class Reference

Public Member Functions

- HDLFrame ()
- void addPoint (double[ ] point, int laserID, int azimuth)
- void addDistance (double dist, int laserID, int azimuth)
- double[ ] getPoint (int laserID, int azimuth)
- double getDistance (int laserID, int azimuth)
- int getNumberOfAzimuthsInFrame ()
- void getRowForLaserID (int laserID, double[ ][ ] return_data, int data_length)
- void getDistanceRowForLaserID (int laserID, double[ ][ ] return_data, int data_length)
- void getSortedDistances (double[ ][ ][ ] distances, int number_of_azimuths)
- void getSortedPointCloud (double[ ][ ][ ] pointCloud, int number_of_azimuths)

3.5.1 Detailed Description

HDLFrame class intended to serve as an "snapshot" of what the lidar sees given the packets added to the frame. The entries are sorted in a table-like fashion with the laserID and azimuth angle serving as the vertical and horizontal axis respectivetly. For the point cloud data, it is a 3D table with the cartician cordinate as the third axis (order: X,Y,Z).

addpoint() function is called to add a single point cloud entry into the frame. The laser ID and azimuth angle are used to index point.

addDistance() function is called to add a single distance sample into the frame. The laser ID and azimuth anfle are used to index entry.

getPoint() function is used to retrieve a single point cloud point given a laser ID and azimuth angle.

getDistance() function is used to retrieve a single distance entry given a laser ID and azimuth angle.

getNumberOfAzimuthsInFrame() function is used to determine how many unique firing sequences (one per azimuth) were used to create frame.

getRowForLaserID() function is used to get all point cloud entries that correspond to a single laser ID.

getDistanceRowForLaserID() function is used to get all distance entries that correspond to a single laser ID.

getSortedPointCloud() function returns the entire point cloud 3D array which comes sorted based on the cartician values.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 HDLFrame()

```
Hardware.VelodyneLidarHDL.PacketDecoder.HDLFrame.HDLFrame ( )
```

Default constructor. Use the other functions to populate.

3.5.3 Member Function Documentation

3.5.3.1 addDistance()

```
void Hardware.VelodyneLidarHDL.PacketDecoder.HDLFrame.addDistance (
            double dist,
            int laserID,
            int azimuth )
```

Add a single distance entry to frame.

Parameters

dist	Double value indicating distance in meters
laserID	Integer representing the laser from which the distance was measured
azimuth	azimuth angle, as integer, for which the distance measurement was taken

3.5.3.2 addPoint()

```
void Hardware.VelodyneLidarHDL.PacketDecoder.HDLFrame.addPoint (
            double[] point,
            int laserID,
            int azimuth )
```

Add a single point cloud entrie into frame.

Parameters

point	point to be added. It should be a double array contaning X, Y, and Z values.
laserID	integer representing the laser which was used to create the point
azimuth	azimuth angle, as integer, for point

3.5.3.3 getDistance()

```
double Hardware.VelodyneLidarHDL.PacketDecoder.HDLFrame.getDistance (
            int laserID,
            int azimuth )
```

Get a single distance entry that corresponds to a single laser id and azimuth.

Parameters

laserID	laser ID number for which to get the point for
azimuth	Azimuth angle for which to get point for

Returns

Distance as double. Returns NULL if no entry found

3.5.3.4 getDistanceRowForLaserID()

```
void Hardware.VelodyneLidarHDL.PacketDecoder.HDLFrame.getDistanceRowForLaserID (
            int laserID,
            double return_data[][],
            int data_length )
```

Get distance entries within frame for a single laser ID. Returned data is sorted based on the azimuth angle.

Parameters

laserID	Laser for which user wishes to obtain all points within frame
return_data	data buffer which will contain all sorted points once function returns
data_length	Number of azimuths in the returned data buffer

3.5.3.5 getNumberOfAzimuthsInFrame()

```
int Hardware.VelodyneLidarHDL.PacketDecoder.HDLFrame.getNumberOfAzimuthsInFrame ( )
```

Get the number of azimuths that currently make up the frame

Returns

number of azimuths as an integer

3.5.3.6 getPoint()

```
double [] Hardware.VelodyneLidarHDL.PacketDecoder.HDLFrame.getPoint (
            int laserID,
            int azimuth )
```

Get a single point that corresponds to a single laser Id and azimuth.

Parameters

laserID	ID number for which to get the point for
azimuth	Azimuth angle for which to get point for

Returns

Point cloud entry if found, NULL if not found.

3.5.3.7 getRowForLaserID()

```
void Hardware.VelodyneLidarHDL.PacketDecoder.HDLFrame.getRowForLaserID (
            int laserID,
            double return_data[][],
            int data_length )
```

Get all points within frame for a single laser ID. Points return sorted in the X and Y directions.

Parameters

laserID	Laser for which user wishes to obtain all points within frame
return_data	data buffer which will contain all sorted points once function returns
data_length	Number of azimuths in the returned data buffer

3.5.3.8 getSortedDistances()

```
void Hardware.VelodyneLidarHDL.PacketDecoder.HDLFrame.getSortedDistances (
            double distances[][][],
            int number_of_azimuths )
```

Get all distances as a 2D array and sort them based on azimuth angle.

Parameters

distances	Array which will contain all returned data.
number_of_azimuths	Size of distances

3.5.3.9 getSortedPointCloud()

```
void Hardware.VelodyneLidarHDL.PacketDecoder.HDLFrame.getSortedPointCloud (
            double pointCloud[][][],
            int number_of_azimuths )
```

Get point cloud data as a 3D array and sorted on both X and Y directions.

Parameters

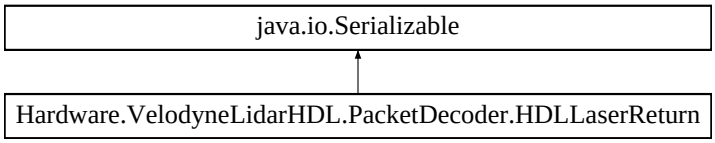
pointCloud	Return data
number_of_azimuths	Size of pointCloud

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

- src/Hardware/VelodyneLidarHDL/PacketDecoder.java

3.6 Hardware.VelodyneLidarHDL.PacketDecoder.HDLLaserReturn Class Reference

Inheritance diagram for Hardware.VelodyneLidarHDL.PacketDecoder.HDLLaserReturn:



Public Member Functions

- HDLLaserReturn (byte[ ] rawData)
- HDLLaserReturn (byte[ ] _distance, byte _intensity)
- int getDistance ()
- short getIntesity ()

3.6.1 Detailed Description

HDLLaserReturn class implements Serializable to assure continous data representation

The HDLLaserReturn class is meant to be the bulding blocks for the HDLFiringData class

getDistance() function returns the raw data distance as a zero-padded integer

getIntensity() function returns the raw data intensity as a zero-padded integer

3.6.2 Constructor & Destructor Documentation

3.6.2.1 HDLLaserReturn() [1/2]

```
Hardware.VelodyneLidarHDL.PacketDecoder.HDLLaserReturn.HDLLaserReturn (
            byte[] rawData )
```

Construtor for creating a HDLLaserReturn object from bytes directly out of packet

Parameters

rawData	Byte array containing the three bytes that make up a single laser return sample

3.6.2.2 HDLLaserReturn() [2/2]

```
Hardware.VelodyneLidarHDL.PacketDecoder.HDLLaserReturn.HDLLaserReturn (
            byte[] _distance,
            byte _intensity )
```

Constructor to create a HDLLaserReturn object from distance and intesity values

Parameters

_distance	Byte array containing MSB and LSB distance values
_intensity	Single byte to represent the intensity of the return

3.6.3 Member Function Documentation

3.6.3.1 getDistance()

```
int Hardware.VelodyneLidarHDL.PacketDecoder.HDLLaserReturn.getDistance ( )
```

Get distance as an integer made up of the two distance bytes

Returns

Integer value for distance (raw value, need to multiply by resolution)

3.6.3.2 getIntesity()

```
short Hardware.VelodyneLidarHDL.PacketDecoder.HDLLaserReturn.getIntesity ( )
```

Get intensity as a byte

Returns

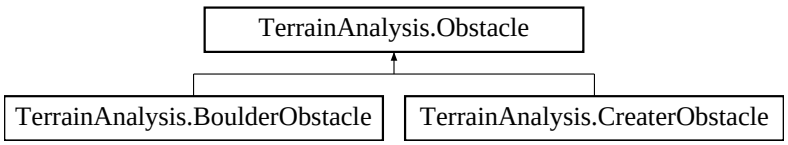
intesity represented as an unsigned char

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

- src/Hardware/VelodyneLidarHDL/PacketDecoder.java

3.7 TerrainAnalysis.Obstacle Class Reference

Inheritance diagram for TerrainAnalysis.Obstacle:



Classes

- enum obstacleType

Public Member Functions

- abstract void updateCenterCoords ()
- abstract double getArea ()
- abstract void updateBounds (double[ ] coords)
- abstract void updateBounds (int azimuth, int elevation, double distance)
- abstract obstacleType getType ()
- abstract double[ ] getLeftMostPoint ()
- abstract double[ ] getRightMostPoint ()
- abstract double[ ] getFardestPoint ()
- abstract double[ ] getClosestPoint ()
- abstract boolean isSameAs (Obstacle o)
- abstract void combineObstacles (Obstacle o)
- abstract String toString ()

Protected Member Functions

- double[ ] toCartesian (int azimuth, int laserID, double distance)

3.7.1 Detailed Description

Abstract class to outline how an Obstacle classs should behave and be seen by the rest of the mapping and pathing program. There are two types: Boulders or Creaters with a negative height indicating a Creater.

getterPoint() functions are called by other programs to outline boundary box surrounding obstacle

update() functions are called to update the bounds and height/depth of Obstacle depending on the new found point.

isSameAs() function is used to compare two obstacles and determe two Obstacles and determine if they should be combined or not

3.7.2 Member Function Documentation

3.7.2.1 combineObstacles()

```
abstract void TerrainAnalysis.Obstacle.combineObstacles (
            Obstacle o )  [abstract]
```

Call this function to combine self with another Obstacle. This function expands the boundary box of self and destroys the other Object.

Parameters

o	Obstacle to combine with

Reimplemented in TerrainAnalysis.CreaterObstacle, and TerrainAnalysis.BoulderObstacle.

3.7.2.2 getArea()

```
abstract double TerrainAnalysis.Obstacle.getArea ( )  [abstract]
```

Call function to get the area of the boundary box surrounding the Obstacle

Returns

Area of boundary box surrounding the Obtacle.

Reimplemented in TerrainAnalysis.CreaterObstacle, and TerrainAnalysis.BoulderObstacle.

3.7.2.3 getClosestPoint()

```
abstract double [] TerrainAnalysis.Obstacle.getClosestPoint ( )  [abstract]
```

Get the closest (to rover) point of the boundary box surrounding the obstacle

Returns

double array containing the X, Y, and Z coordinates of the closest (to rover) boundary point

Reimplemented in TerrainAnalysis.CreaterObstacle, and TerrainAnalysis.BoulderObstacle.

3.7.2.4 getFardestPoint()

```
abstract double [] TerrainAnalysis.Obstacle.getFardestPoint ( )  [abstract]
```

Get the farthest (from rover) point of the boundary box surrounding the obstacle

Returns

double array containing the X, Y, and Z coordinates of the farthest (from rover) boundary point

Reimplemented in TerrainAnalysis.CreaterObstacle, and TerrainAnalysis.BoulderObstacle.

3.7.2.5 getLeftMostPoint()

```
abstract double [] TerrainAnalysis.Obstacle.getLeftMostPoint ( )  [abstract]
```

Get the left most point of the boundary box surrounding the obstacle

Returns

double array containing the X, Y, and Z coordinates of the left most boundary point

Reimplemented in TerrainAnalysis.CreaterObstacle, and TerrainAnalysis.BoulderObstacle.

3.7.2.6 getRightMostPoint()

```
abstract double [] TerrainAnalysis.Obstacle.getRightMostPoint ( )  [abstract]
```

Get the right most point of the boundary box surrounding the obstacle

Returns

double array containing the X, Y, and Z coordinates of the right most boundary point

Reimplemented in TerrainAnalysis.CreaterObstacle, and TerrainAnalysis.BoulderObstacle.

3.7.2.7 getType()

```
abstract obstacleType TerrainAnalysis.Obstacle.getType ( )  [abstract]
```

Get the type of Obstacle

Returns

obstacleType enumerator

Reimplemented in TerrainAnalysis.CreaterObstacle, and TerrainAnalysis.BoulderObstacle.

3.7.2.8 isSameAs()

```
abstract boolean TerrainAnalysis.Obstacle.isSameAs (
            Obstacle o )  [abstract]
```

Call function on an Obstacle to compare it to another Obstacle. Obstacles are compared based on their type and boundary boxes.

Parameters

o	Obstacle to compare Self to

Returns

Boolean value indicating if the two Obstacles match

Reimplemented in TerrainAnalysis.CreaterObstacle, and TerrainAnalysis.BoulderObstacle.

3.7.2.9 toCartesian()

```
double [] TerrainAnalysis.Obstacle.toCartesian (
            int azimuth,
            int laserID,
            double distance )  [protected]
```

Turn given polar coordinates into cartician coordinates based on the laser Id, distance and azimuth values

Parameters

azimuth	
laserID	
distance	

Returns

Double array containing {X,Y,Z}

3.7.2.10 toString()

```
abstract String TerrainAnalysis.Obstacle.toString ( )  [abstract]
```

Function to retreive a string detailing the Obstacle. Type: Center X: Center Y: Center Z:

Returns

String detailing Obstacle

Reimplemented in TerrainAnalysis.CreaterObstacle, and TerrainAnalysis.BoulderObstacle.

3.7.2.11 updateBounds() [1/2]

```
abstract void TerrainAnalysis.Obstacle.updateBounds (
            double[] coords )  [abstract]
```

Feed in a point represented as a double array which is determine to be part of the Obstacle. The function then uses this new found point to update boundary box.

Parameters

coords	point as a double array.

Reimplemented in TerrainAnalysis.CreaterObstacle, and TerrainAnalysis.BoulderObstacle.

3.7.2.12 updateBounds() [2/2]

```
abstract void TerrainAnalysis.Obstacle.updateBounds (
            int azimuth,
            int elevation,
            double distance )  [abstract]
```

Update the boundary box of an Obstacle by using polar coordinates. The fed in values will then be used to calculate the pertaining point in cartician.

Parameters

azimuth	azimuth angle (degrees) as an integer and multiplied by 100 (ex: 10.20 -> 1020)
elevation	elevation angle (degrees) as an integer and multiplied by 100 (ex: 10.20 -> 1020)
distance	distance in meters as a double

Reimplemented in TerrainAnalysis.CreaterObstacle, and TerrainAnalysis.BoulderObstacle.

3.7.2.13 updateCenterCoords()

```
abstract void TerrainAnalysis.Obstacle.updateCenterCoords ( )  [abstract]
```

Call function to update center coordinates based on the current bounds

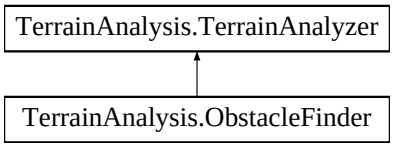
Reimplemented in TerrainAnalysis.CreaterObstacle, and TerrainAnalysis.BoulderObstacle.

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

- src/TerrainAnalysis/Obstacle.java

3.8 TerrainAnalysis.ObstacleFinder Class Reference

Inheritance diagram for TerrainAnalysis.ObstacleFinder:



Public Member Functions

- ObstacleFinder (double heightTolarence, double groundRef, int positiveHitsThreshold)
- void findObstaclesPolar (HDLFrame frame)
- void findObstaclesCartician (HDLFrame frame)
- void addObstacle (Obstacle o)
- int getNumberOfObticles ()
- void clearObsticlesSeen ()
- Obstacle getLatestObstacleFound ()

3.8.1 Detailed Description

ObstacleFinder implements TerrainAnalyzer to be used in tandum with PackedDecorder's output frames and the two types of Obstacles, Boulder and Creater.

findObstaclesPolar() takes in an HDLFrame and analyzes it in polar coordinates where the distance to a sampled point is compared to that of a flat plane.

findObstaclesCartician() takes in an HDLFrame and analyzes it in cartician coordinates where it looks for concistant changes in height.

Obstacle() related functions are then used to get number and retreive obstacles found if any.

3.8.2 Constructor & Destructor Documentation

3.8.2.1 ObstacleFinder()

```
TerrainAnalysis.ObstacleFinder.ObstacleFinder (
            double heightTolarence,
            double groundRef,
            int positiveHitsThreshold )
```

ObstacleFinder constructor.

Parameters

heightTolarence	Threshold, in meters, in either height (cartician) or distance (polar) to trigger a search
groundRef	Referance point, in meters, to be used as ground
positiveHitsThreshold	Number of possitive hits/points found to declare an Obstacle found

3.8.3 Member Function Documentation

3.8.3.1 addObstacle()

```
void TerrainAnalysis.ObstacleFinder.addObstacle (
            Obstacle o )
```

Add a single Obstacle to ArrayList

Parameters

o	Obstacle to be added to array

Implements TerrainAnalysis.TerrainAnalyzer.

3.8.3.2 clearObsticlesSeen()

```
void TerrainAnalysis.ObstacleFinder.clearObsticlesSeen ( )
```

Clear buffers

Implements TerrainAnalysis.TerrainAnalyzer.

3.8.3.3 findObstaclesCartician()

```
void TerrainAnalysis.ObstacleFinder.findObstaclesCartician (
            HDLFrame frame )
```

Look for Obstacles in cartician coordinates within the provided HDLFrame. Frame does not have to come from a calibrated decoder/lidar.

Parameters

frame	HDLFrame used to look for obstacles. Refer to PacketDecoder and VelodyneLidar class for more info with regards to HDLFrame

Implements TerrainAnalysis.TerrainAnalyzer.

3.8.3.4 findObstaclesPolar()

```
void TerrainAnalysis.ObstacleFinder.findObstaclesPolar (
            HDLFrame frame )
```

Look for obstacles in polar coordinates within the provided frame. Found Obstacles are added to ArrayList and retreived through getters.

Parameters

frame	HDLFrame used to look for obstacles. To work correctly, the frame has to have been produced by a calibrated decoder/lidar. Refer to PacketDecoder and VelodyneLidar class for more info with regards to HDLFrame

Implements TerrainAnalysis.TerrainAnalyzer.

3.8.3.5 getLatestObstacleFound()

```
Obstacle TerrainAnalysis.ObstacleFinder.getLatestObstacleFound ( )
```

Get the last Obstacle found and remove from buffer

Returns

Obstacle found from last analysis ran

Implements TerrainAnalysis.TerrainAnalyzer.

3.8.3.6 getNumberOfObticles()

```
int TerrainAnalysis.ObstacleFinder.getNumberOfObticles ( )
```

Get the number of Obstacles found

Returns

number of obstacles found

Implements TerrainAnalysis.TerrainAnalyzer.

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

- src/TerrainAnalysis/ObstacleFinder.java

3.9 TerrainAnalysis.Obstacle.obstacleType Enum Reference

Public Attributes

- NONE
- BOULDER
- **CREATER**

3.9.1 Detailed Description

Enumerator to indicate type of Obstacle object

3.9.2 Member Data Documentation

3.9.2.1 BOULDER

`TerrainAnalysis.Obstacle.obstacleType.BOULDER`

Obstacle with a height greater than zero

3.9.2.2 NONE

`TerrainAnalysis.Obstacle.obstacleType.NONE`

Default type

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

- src/TerrainAnalysis/Obstacle.java

3.10 Hardware.VelodyneLidarHDL.PacketDecoder Class Reference

Classes

- class HDLDataPacket
- class HDLFiringData
- class HDLFrame
- class HDLLaserReturn

Public Member Functions

- PacketDecoder (boolean generatePointCloud)
- void **finalize** ()
- void SetMaxNumberOffFrames (int max_num_frames)
- void DecodePacket (byte[ ] data, int[ ] data_length)
- void addToCalibrationFrame (byte[ ] data, int[ ] data_length)
- void SetCorrectionsFile (final String corrections_file)
- Deque< HDLFrame > GetFrames ()
- void ClearFrames ()
- HDLFrame GetLatestFrame (int numberOfAzimuthsInFrame)

Static Public Attributes

- static double[ ] Lidar_height_map = new double[Constants.HDL_NUM_ROT_ANGLES]
- static double[ ] Az_cos_lookup_table = new double[Constants.HDL_NUM_ROT_ANGLES]
- static double[ ] Az_sin_lookup_table = new double[Constants.HDL_NUM_ROT_ANGLES]
- static double[ ][ ] El_cos_lookup_table = new double[Constants.HDL_LASER_PER_FIRING][Constants.↩ HDL_NUM_ROT_ANGLES]
- static double[ ][ ] El_sin_lookup_table = new double[Constants.HDL_LASER_PER_FIRING][Constants.↩ HDL_NUM_ROT_ANGLES]
- static int[ ] **elAngle_lookup_table**
- static int[ ] laserIdMap = {15,13,11,9,7,5,3,1,14,12,10,8,6,4,2,0}

Protected Member Functions

- void ProcessesHDLPacket (byte[ ] data, int data_length)
- void PushFringData (int laserID, int azimuth, HDLLaserReturn laserReturn, boolean isCalibrationData)
- void UnloadData ()
- void InitTables ()
- void **LoadCorrectionsFile** (final String correctionsfile)
- void **SetCorrectionsCommon** ()
- void splitFrame ()

3.10.1 Detailed Description

PacketDecoder class

The PacketDecoder class is intended to be used alonside the PacketDriver inside the VelodyneLidar wrapper class

DecodePacket(Byte[ ] packet) function is intended to be used to convert packet into a frame (explained in HDLFrame class)

GetLatestFrame() function is intended to be used to retreive the lastest HDLFrame with all of the provided decoded packets

3.10.2 Constructor & Destructor Documentation

3.10.2.1 PacketDecoder()

```
Hardware.VelodyneLidarHDL.PacketDecoder.PacketDecoder (
            boolean generatePointCloud )
```

Constructor to PacketDecoder class. Input is used to indicate if the algorithm should take the packet and derive a point cloud 3D array or just store as distances (i.e. polar coordinates). Point cloud is only needed if no calibration data is avalible (flat surface was not sampled).

Parameters

generatePointCloud	Flag to allow point cloud calculations.

3.10.3 Member Function Documentation

3.10.3.1 addToCalibrationFrame()

```
void Hardware.VelodyneLidarHDL.PacketDecoder.addToCalibrationFrame (
            byte[] data,
            int[] data_length )
```

Function used to push a raw packet into the calibration frame. Ususally used right after a flat plane has been sampled.

Parameters

data	Raw data packet coming staright from lidar socket
data_length	Size of data buffer

3.10.3.2 ClearFrames()

```
void Hardware.VelodyneLidarHDL.PacketDecoder.ClearFrames ( )
```

Unload all frames.

3.10.3.3 DecodePacket()

```
void Hardware.VelodyneLidarHDL.PacketDecoder.DecodePacket (
            byte[] data,
            int[] data_length )
```

Decode a single packet and add to current frame.

Parameters

data	raw byte array containing a packet coming straight from lidar
data_length	number of bytes within array

3.10.3.4 GetFrames()

```
Deque<HDLFrame> Hardware.VelodyneLidarHDL.PacketDecoder.GetFrames ( )
```

Get all frames within Queue

Returns

Queue containing all frames currently in decoder.

3.10.3.5 GetLatestFrame()

```
HDLFrame Hardware.VelodyneLidarHDL.PacketDecoder.GetLatestFrame (
            int numberOfAzimuthsInFrame )
```

Get the lastest frame added to queue.

Parameters

numberOfAzimuthsInFrame	Minimum number of azimuths that frame must contain to be retrived.

Returns

HDLFrame requested

3.10.3.6 InitTables()

```
void Hardware.VelodyneLidarHDL.PacketDecoder.InitTables ( )  [protected]
```

Initialize all tables used in point cloud calculations. Lidar height is estimated by taking the point cloud Z values for the two outer most azimuth angles, and linearly interpolate each one with the center-most azimuth.

3.10.3.7 ProcessesHDLPacket()

```
void Hardware.VelodyneLidarHDL.PacketDecoder.ProcessesHDLPacket (
            byte[] data,
            int data_length )  [protected]
```

Protected function to decode a single packet and add to current frame. Function can only be called once it is determined packets is valid.

Parameters

data	Raw byte array containing a packet coming straight from lidar
data_length	Number of bytes within array

Uncomment this code out if you wish to split frames once the lidar loops back arround if(firingData.getAzimuth↩ Angle() < _last_azimuth){ splitFrame(); }

_last_azimuth = firingData.getAzimuthAngle();

3.10.3.8 PushFringData()

```
void Hardware.VelodyneLidarHDL.PacketDecoder.PushFringData (
            int laserID,
            int azimuth,
            HDLLaserReturn laserReturn,
            boolean isCalibrationData )  [protected]
```

Push firing data to current frame or calibration frame. Generates point clod data if decoder configured to do so.

Parameters

laserID	Laser identifier for sensor used to sample laser return
azimuth	Azimuth at which return was taken
laserReturn	HDLLaserReturn containing distance and intensity data
isCalibrationData	Indicates if the data is for a flat plane and should be added to clalibration frame

3.10.3.9 SetCorrectionsFile()

```
void Hardware.VelodyneLidarHDL.PacketDecoder.SetCorrectionsFile (
            final String corrections_file )
```

Function used to load factory specified corrections.

Parameters

corrections_file	String for name of correction file

3.10.3.10 SetMaxNumberOffFrames()

```
void Hardware.VelodyneLidarHDL.PacketDecoder.SetMaxNumberOffFrames (
            int max_num_frames )
```

Set the number of frames allowed to be stored at once.

Parameters

max_num_frames	Interger indicating number of frames to be stored at once in Queue.

3.10.3.11 splitFrame()

```
void Hardware.VelodyneLidarHDL.PacketDecoder.splitFrame ( )  [protected]
```

Function to create a new frame once lidar wraps arround.

3.10.3.12 UnloadData()

```
void Hardware.VelodyneLidarHDL.PacketDecoder.UnloadData ( )  [protected]
```

Clear all variables used to keep track of frames being decoded.

3.10.4 Member Data Documentation

3.10.4.1 Az_cos_lookup_table

```
double [] Hardware.VelodyneLidarHDL.PacketDecoder.Az_cos_lookup_table = new double[Constants.↩
HDL_NUM_ROT_ANGLES]  [static]
```

Azimuth angle cosine lookup table to speed up computation

3.10.4.2 Az_sin_lookup_table

```
double [] Hardware.VelodyneLidarHDL.PacketDecoder.Az_sin_lookup_table = new double[Constants.↩
HDL_NUM_ROT_ANGLES]  [static]
```

Azimuth angle sine lookup table to speed up coputation

3.10.4.3 El_cos_lookup_table

```
double [][] Hardware.VelodyneLidarHDL.PacketDecoder.El_cos_lookup_table = new double[Constants.↩
HDL_LASER_PER_FIRING][Constants.HDL_NUM_ROT_ANGLES]  [static]
```

Elevation angle cosine lookup table

3.10.4.4 El_sin_lookup_table

```
double [][] Hardware.VelodyneLidarHDL.PacketDecoder.El_sin_lookup_table = new double[Constants.↩
HDL_LASER_PER_FIRING][Constants.HDL_NUM_ROT_ANGLES]  [static]
```

Elevation angle sine lookup table

3.10.4.5 elAngle_lookup_table

```
int [] Hardware.VelodyneLidarHDL.PacketDecoder.elAngle_lookup_table  [static]
```

Initial value:

```
= {1500, -100, 1300, -300, 1100, -500, 900, -700,
                                                    700, -900, 500, -1100, 300, -1300, 100, -1500}
```

3.10.4.6 laserIdMap

```
int [] Hardware.VelodyneLidarHDL.PacketDecoder.laserIdMap = {15,13,11,9,7,5,3,1,14,12,10,8,6,4,2,0}
[static]
```

Table to map firing sequence to laserID (i.e laser id 15 gets fired first)

3.10.4.7 Lidar_height_map

```
double [] Hardware.VelodyneLidarHDL.PacketDecoder.Lidar_height_map = new double[Constants.↩
HDL_NUM_ROT_ANGLES]  [static]
```

Array to map an specific azimuth angle to the exact height of the sensors at that azimuth

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

- src/Hardware/VelodyneLidarHDL/PacketDecoder.java

3.11 Hardware.VelodyneLidarHDL.PacketDriver Class Reference

Public Member Functions

- PacketDriver (int port)
- void finalize ()
- void InitPacketDriver (int port)
- boolean GetPacket (byte[ ] data, int[ ] data_length)

3.11.1 Detailed Description

PacketDriver implements the dataGather interface for retriving packets from an RS32 connection

The PacketDriver is intended to be a subclass for a user implementing the VelodyneLidar Class

getPacket() function gets called by VelodyneLidar class to retreive the latest data packet sent by lidar.

3.11.2 Constructor & Destructor Documentation

3.11.2.1 PacketDriver()

```
Hardware.VelodyneLidarHDL.PacketDriver.PacketDriver (
            int port )
```

Constructor for PacketDriver.

Parameters

in	*port*	The port number to connect to.

3.11.3 Member Function Documentation

3.11.3.1 finalize()

```
void Hardware.VelodyneLidarHDL.PacketDriver.finalize ( )
```

Overwritten finalizer to ganrantee socket gets closed

3.11.3.2 GetPacket()

```
boolean Hardware.VelodyneLidarHDL.PacketDriver.GetPacket (
            byte[] data,
            int[] data_length )
```

Funtion for getting a single packet through binded socket

Parameters

out	*data*	Byte buffer to receive packet in
in	*data_length*	size for provided data buffer

3.11.3.3 InitPacketDriver()

```
void Hardware.VelodyneLidarHDL.PacketDriver.InitPacketDriver (
            int port )
```

Initialize PacketDriver if class constructed using default constructor

Parameters

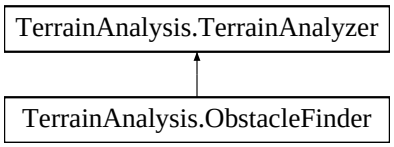
in	*port*	The port number to connect to.

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

- src/Hardware/VelodyneLidarHDL/PacketDriver.java

3.12 TerrainAnalysis.TerrainAnalyzer Interface Reference

Inheritance diagram for TerrainAnalysis.TerrainAnalyzer:



Public Member Functions

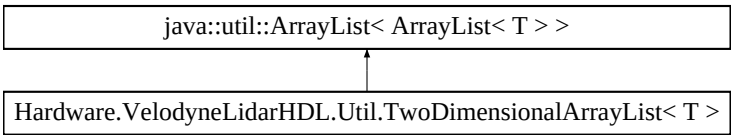
- void **findObstaclesCartician** (HDLFrame frame)
- void **findObstaclesPolar** (HDLFrame frame)
- int **getNumberOfObticles** ()
- Obstacle **getLatestObstacleFound** ()
- void **clearObsticlesSeen** ()
- void **addObstacle** (Obstacle o)

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

- src/TerrainAnalysis/TerrainAnalyzer.java

3.13 Hardware.VelodyneLidarHDL.Util.TwoDimensionalArrayList< T > Class Template Reference

Inheritance diagram for Hardware.VelodyneLidarHDL.Util.TwoDimensionalArrayList< T >:



Public Member Functions

- void **addToInnerArray** (int index, T element)
- void **addToInnerArray** (int index, int index2, T element)

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

- src/Hardware/VelodyneLidarHDL/Util/TwoDimensionalArrayList.java

3.14 Hardware.VelodyneLidarHDL.VelodyneLidar Class Reference

Public Member Functions

- VelodyneLidar (double heightTolarence, double groundRef, int positiveHitsThreshold, int numberOf↩ AzimuthsInFrame, boolean generatePointCloud)
- boolean calibrateLidar ()
- void changeRequiredAzimuths (int num)
- void scanFullFieldOfView ()
- void updateLatestFrame (int numberOfAzimuthsInFrame)
- void analyzeLatestFrame ()
- void clearAllDataBuffers ()
- Obstacle getClosestObstacle ()
- boolean anyObsticlesInFrame ()
- String toString ()

3.14.1 Detailed Description

VelodyneLidar class used to wrap PacketDriver, PacketDecoder, and ObstacleFinder classes. This class serves as the top-most abstraction layer for using the Velodyne VLP-16 for simple obtacle detection and avoidance.

calibrateLidar() function gets called if there is a file containing the raw packet data of a flat surface.

scanFullFieldOfView() function gets called to perform a full FOV scan and produce a single HDLFrame.

getClosetObstacle() fucntion is used to used the latest scan, look in it for possible obstacles, and return closest.

clearAllDataBuffers() function is used to reset lidar (calibration frame is kept).

3.14.2 Constructor & Destructor Documentation

3.14.2.1 VelodyneLidar()

```
Hardware.VelodyneLidarHDL.VelodyneLidar.VelodyneLidar (
            double heightTolarence,
            double groundRef,
            int positiveHitsThreshold,
            int numberOfAzimuthsInFrame,
            boolean generatePointCloud )
```

Main VelodyneLidar class intended to be used intandum with all the other classes in the package. It is the top-most abstraction layer and as such careful consideration must be taken when providing the initializion parameters.

Parameters

heightTolarence	Double to indicate at what height (in meters) to start checking for possible obstacles.
groundRef	Double to idicate (in meters) what the lidar should consider to be ground (i.e. if 0.01 or 0.00 should be ground)
positiveHitsThreshold	Number of laser returns indicating a possible obstacle needed to count as a Obstacle
numberOfAzimuthsInFrame	Number of azimuths required to be sampled befre creating a HDLFrame
generatePointCloud	Flag to idicate if point cloud calculations should be performed. If not, then lidar uses polar cordinates to look for obstacles

3.14.3 Member Function Documentation

3.14.3.1 analyzeLatestFrame()

```
void Hardware.VelodyneLidarHDL.VelodyneLidar.analyzeLatestFrame ( )
```

Analyze the most up-to-date frame and look for any obstacles inside of it.

3.14.3.2 anyObsticlesInFrame()

```
boolean Hardware.VelodyneLidarHDL.VelodyneLidar.anyObsticlesInFrame ( )
```

Returns true if there are any obtacles in the current frame.

Returns

True if atleast one obstacle in frame.

3.14.3.3 calibrateLidar()

```
boolean Hardware.VelodyneLidarHDL.VelodyneLidar.calibrateLidar ( )
```

Funtion used to load calibration file and feed it to _decoder to generate the calibraiton frame. Calibration is required if the user does not want to perform point-cloud calculations. Calibrations makes looking for obtacles more accurate in both Cartician and Polar cordinate searches.

Returns

Flag indicating if Lidar was successfully calibrated or not.

3.14.3.4 changeRequiredAzimuths()

```
void Hardware.VelodyneLidarHDL.VelodyneLidar.changeRequiredAzimuths (
            int num )
```

Change the number of azimuths required to create a frame. Number needs to be greater than 350.

Parameters

num	New number of azimuths required. Minimum number is 350.

3.14.3.5 clearAllDataBuffers()

```
void Hardware.VelodyneLidarHDL.VelodyneLidar.clearAllDataBuffers ( )
```

Clear all buffers within all objects used by lidar.

3.14.3.6 getClosestObstacle()

```
Obstacle Hardware.VelodyneLidarHDL.VelodyneLidar.getClosestObstacle ( )
```

If frame has already been analyzed, then remove and return the closest obsticle to lidar.

Returns

Obstacle object, NULL if no obstacle found

3.14.3.7 scanFullFieldOfView()

```
void Hardware.VelodyneLidarHDL.VelodyneLidar.scanFullFieldOfView ( )
```

Scan a full FOV by sampling at least _number_azimuths_in_frame number of times.

3.14.3.8 toString()

```
String Hardware.VelodyneLidarHDL.VelodyneLidar.toString ( )
```

Function meant for debugging. Returns all found obstacles as a string to be displayed during testing.

Returns

String with information detailing all Obstacles within frame

3.14.3.9 updateLatestFrame()

```
void Hardware.VelodyneLidarHDL.VelodyneLidar.updateLatestFrame (
            int numberOfAzimuthsInFrame )
```

Scan a full FOV with the specified number of azimuths in it.

Parameters

numberOfAzimuthsInFrame	Number of azimuths required to create a full Frame.

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

- src/Hardware/VelodyneLidarHDL/VelodyneLidar.java

Index