INF552

Generated by Doxygen 1.8.13

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Class Documentation

3.1 line3d Class Reference

A class to represent lines in the 3d space.

`#include <line3d.hpp>`

Public Member Functions

- **line3d** ()

  A constructor.
- **line3d** (Vec3d point, Vec3d v, bool second_Vec3d_isvector)

  A constructor.
- Vec3d **getVector** ()

  A method to get the vector.
- Vec3d **getPoint** ()

  A method to get the point.
- double **distance** (Vec3d p)

  *Finds the distance between the **line3d** (p. 5) and a point.*
- double **distance** (**line3d** l)

  *Finds the distance between the **line3d** (p. 5) and another **line3d** (p. 5).*
- bool **isDegenerated** ()

  Tels if the line is a correct line.
- double **cosAngle** (Vec3d v)

  Gives the cosine between the line and a vector.

Friends

- ostream & **operator**<< (ostream &os, const **line3d** &l)

  Overloads ofstream for printing purposes.

3.1.1 Detailed Description

A class to represent lines in the 3d space.

We describe a **line3d** (p. 5) by one of its point and a direction vector. We are using Vec3d objects because we don't need the color argument to represent this explicit object.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 line3d() [1/2]

```
line3d::line3d ( )
```

A constructor.

This default constructor sets this->point and this->vector to zero.

3.1.2.2 line3d() [2/2]

```
line3d::line3d (
            Vec3d point,
            Vec3d v,
            bool second_Vec3d_isvector )
```

A constructor.

This constructor allows us to define a **line3d** (p. 5) with either two points or a point and a vector.

Parameters

point	A point in the **line3d** (p. 5).
vector	A direction vector or a second point of the **line3d** (p. 5).
second_Vec3d_isvector	A boolean allowing us to call the constructor properly.

3.1.3 Member Function Documentation

3.1.3.1 cosAngle()

```
double line3d::cosAngle (
            Vec3d v )
```

Gives the cosine between the line and a vector.

Parameters

v	The considered vector.

Returns

The cosine between the line and the vector.

3.1.3.2 distance() [1/2]

```
double line3d::distance (
            Vec3d p )
```

Finds the distance between the **line3d** (p. 5) and a point.

Parameters

p	The considered point.

Returns

The distance between the **line3d** (p. 5) and the considered point.

3.1.3.3 distance() [2/2]

```
double line3d::distance (
              line3d l )
```

Finds the distance between the **line3d** (p. 5) and another **line3d** (p. 5).

Parameters

l	The considered **line3d** (p. 5).

Returns

The distance between the two line3ds.

3.1.3.4 getPoint()

```
Vec3d line3d::getPoint ( )
```

A method to get the point.

Returns

The point.

3.1.3.5 getVector()

```
Vec3d line3d::getVector ( )
```

A method to get the vector.

Returns

The vector.

3.1.3.6 isDegenerated()

```
bool line3d::isDegenerated ( )
```

Tels if the line is a correct line.

A line is degenerated if is direction vector is equal to zero.

Returns

A boolean telling if the line is a correct line.

3.1.4 Friends And Related Function Documentation

3.1.4.1 operator<<

```
ostream& operator<< (
            ostream & os,
            const  line3d & l )  [friend]
```

Overloads ofstream for printing purposes.

Parameters

os	Considered stream.
l	Considered line.

Returns

The ofstream to be printed.

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

- C:/Users/romai/Documents/Mes_documents/X/3A/INF552/Projet-Inf552/project/include/ **line3d.hpp**
- C:/Users/romai/Documents/Mes_documents/X/3A/INF552/Projet-Inf552/project/src/ **line3d.cpp**

3.2 line3dCloud Class Reference

A class to represent clouds of lines in the 3d space.

`#include <line3dCloud.hpp>`

Public Member Functions

- **line3dCloud** ()
  A constructor.
- void **push_back** (**line3d** line, int npoints=0)
  Adds a line in the linecloud.
- **line3d operator[ ]** (int i)
  Overloads [].
- int **size** ()
  Gives the size of the linecloud.
- int **getMinNpointsIndex** ()
  Finds the index of the line with the least neighbours of the linecloud.
- int **getMinNpoints** ()
  Finds the number of neighbours of the line with the least neighbours of the linecloud.
- void **set** (int i, **line3d** line, int npoints)
  Replace a line for another in the linecloud.
- double **minDistance** (**line3d** line, **plane** p)
  Finds distance of a line from it's closest in the pointcloud.
- double **minDistance** (Vec3d p)
  Finds distance of a point from it's closest line in the pointcloud.

3.2.1 Detailed Description

A class to represent clouds of lines in the 3d space.

We describe a cloud by a vector of **line3d** (p. 5) and a vector of int. The vector of int allows us to store the number of points next to the line from a pointcloud.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 line3dCloud()

```
line3dCloud::line3dCloud ( )
```

A constructor.

This default constructor initializes this->cloud and this->npoints.

3.2.3 Member Function Documentation

3.2.3.1 getMinNpoints()

```
int line3dCloud::getMinNpoints ( )
```

Finds the number of neighbours of the line with the least neighbours of the linecloud.

Returns

The desired number of neighbours.

3.2.3.2 getMinNpointsIndex()

```
int line3dCloud::getMinNpointsIndex ( )
```

Finds the index of the line with the least neighbours of the linecloud.

Returns

The desired index.

3.2.3.3 minDistance() [1/2]

```
double line3dCloud::minDistance (
            line3d line,
            plane p )
```

Finds distance of a line from it's closest in the pointcloud.

The distances are calculated at the intersection with a plane.

Parameters

line	The considered line.
p	The considered plane.

Returns

The distance.

3.2.3.4 minDistance() [2/2]

```
double line3dCloud::minDistance (
            Vec3d p )
```

Finds distance of a point from it's closest line in the pointcloud.

Parameters

p	The considered point.

Returns

The distance.

3.2.3.5 operator[]()

```
line3d line3dCloud::operator[] (
            int i )
```

Overloads [].

Parameters

i	The index to get.

Returns

The **line3d** (p. 5).

3.2.3.6 push_back()

```
void line3dCloud::push_back (
              line3d line,
             int npoints = 0 )
```

Adds a line in the linecloud.

Parameters

line	The **line3d** (p. 5) to add.
npoints	The number of points next to the considered line.

3.2.3.7 set()

```
void line3dCloud::set (
             int i,
              line3d line,
             int npoints )
```

Replace a line for another in the linecloud.

Parameters

i	The index to change.
line	The **line3d** (p. 5).
npoints	The number of neighbours of this **line3d** (p. 5).

3.2.3.8 size()

```
int line3dCloud::size ( )
```

Gives the size of the linecloud.

Returns

The size of the linecloud.

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

- C:/Users/romai/Documents/Mes_documents/X/3A/INF552/Projet-Inf552/project/include/ **line3dCloud.hpp**
- C:/Users/romai/Documents/Mes_documents/X/3A/INF552/Projet-Inf552/project/src/ **line3dCloud.cpp**

3.3 plane Class Reference

A class to represent plane in the 3d space.

`#include <plane.hpp>`

Public Member Functions

- **plane** ()
  A constructor.
- **plane** (double a, double b, double c, double d)
  A constructor.
- **plane** (Vec3d p1, Vec3d p2, Vec3d p3)
  A constructor.
- double **distance** (Vec3d p)
  Finds the distance between the plane and a point.
- void **regression** (**point3dCloud** pointcloud)
  Finds the closest plane to the given pointcloud.
- bool **isDegenerated** ()
  Tels if the plane is a correct plane.
- Vec3d **getDirection** ()
  Gives a direction vector of the plane.
- Vec3d **intersection** (**line3d** l)

Friends

- ostream & **operator**<< (ostream &os, const **plane** &p)
  Overloads ofstream for printing purposes.

3.3.1 Detailed Description

A class to represent plane in the 3d space.

We describe a plane by the four variables of its equation : a∗x + b∗y + c∗z + d = 0.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 plane() [1/3]

```
plane::plane ( )
```

A constructor.

This default constructor sets all the parameters to zero.

3.3.2.2 plane() [2/3]

```
plane::plane (
            double a,
            double b,
            double c,
            double d )
```

A constructor.

Parameters

a	1st parameter of a∗x + b∗y + c∗z + d = 0
b	2nd parameter of a∗x + b∗y + c∗z + d = 0
c	3rd parameter of a∗x + b∗y + c∗z + d = 0
d	4th parameter of a∗x + b∗y + c∗z + d = 0

3.3.2.3 plane() [3/3]

```
plane::plane (
            Vec3d p1,
            Vec3d p2,
            Vec3d p3 )
```

A constructor.

Parameters

p1	1st point of the plane
p2	2nd point of the plane
p3	3rd point of the plane

3.3.3 Member Function Documentation

3.3.3.1 distance()

```
double plane::distance (
            Vec3d p )
```

Finds the distance between the plane and a point.

Parameters

p	The considered point.

Returns

The distance between the plane and the considered point.

3.3.3.2 getDirection()

```
Vec3d plane::getDirection ( )
```

Gives a direction vector of the plane.

Returns

A direction vector of the plane.

3.3.3.3 intersection()

```
Vec3d plane::intersection (
              line3d l )
```

3.3.3.4 isDegenerated()

```
bool plane::isDegenerated ( )
```

Tels if the plane is a correct plane.

Returns

A boolean telling if the plane is a correct plane.

3.3.3.5 regression()

```
void plane::regression (
              point3dCloud pointcloud )
```

Finds the closest plane to the given pointcloud.

This method modify the plane inplace.

Parameters

pointcloud	The considered pointcloud.

3.3.4 Friends And Related Function Documentation

3.3.4.1 operator<<

```
ostream& operator<< (
            ostream & os,
            const  plane & p )  [friend]
```

Overloads ofstream for printing purposes.

Parameters

os	Considered stream.
p	Considered plane.

Returns

The ofstream to be printed.

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

- C:/Users/romai/Documents/Mes_documents/X/3A/INF552/Projet-Inf552/project/include/ **plane.hpp**
- C:/Users/romai/Documents/Mes_documents/X/3A/INF552/Projet-Inf552/project/src/ **plane.cpp**

3.4 point3d Class Reference

A class to represent points in the 3d space.

```
#include <point3d.hpp>
```

Public Member Functions

- **point3d** (Vec3d position, Vec3b color, pair< int, int > pixel)
  A constructor.
- Vec3d **getPosition** ()
  A method to get the position of the point.
- Vec3b **getColor** ()
  A method to get the color of the point.
- pair< int, int > **getPixelCoordinates** ()
  A method to get the pixel coordinates of the point.
- double **distance** (**point3d** p)
  Finds the distance between the point and another point.

3.4.1 Detailed Description

A class to represent points in the 3d space.

We describe a **point3d** (p. 16) by its position, its color and its correponding pixel in the origin image.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 point3d()

```
point3d::point3d (
            Vec3d position,
            Vec3b color,
            pair< int, int > pixel )
```

A constructor.

Parameters

position	The position of the 3dPoint.
color	The color of the 3dPoint.
pixel	The corresponding pixel in the image.

3.4.3 Member Function Documentation

3.4.3.1 distance()

```
double point3d::distance (
              point3d p )
```

Finds the distance between the point and another point.

Parameters

p	The considered point.

Returns

The distance between the two points.

3.4.3.2 getColor()

```
Vec3b point3d::getColor ( )
```

A method to get the color of the point.

Returns

The color of the point.

3.4.3.3 getPixelCoordinates()

```
pair< int, int > point3d::getPixelCoordinates ( )
```

A method to get the pixel coordinates of the point.

Returns

The pixel coordinates of the point.

3.4.3.4 getPosition()

```
Vec3d point3d::getPosition ( )
```

A method to get the position of the point.

Returns

The position of the point.

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

- C:/Users/romai/Documents/Mes_documents/X/3A/INF552/Projet-Inf552/project/include/ **point3d.hpp**
- C:/Users/romai/Documents/Mes_documents/X/3A/INF552/Projet-Inf552/project/src/ **point3d.cpp**

3.5 point3dCloud Class Reference

A class to represent clouds of points in the 3d space.

```
#include <point3dCloud.hpp>
```

Public Member Functions

- **point3dCloud** ()
  A constructor.
- void **push_back** (**point3d** point)
  Adds a point in the pointcloud.
- **point3d operator[ ]** (int i)
  Overloads [].
- int **size** ()
  Gives the size of the pointcloud.
- double **meanNeighboursDistance** ()
  Gives the mean of the distances between neighbours.
- void **showOnImage** (Mat &image)
  Shows the result on an image.
- void **pointCloud2ply** (string target)
  Generates .ply file from point cloud values.

3.5.1 Detailed Description

A class to represent clouds of points in the 3d space.

We describe a cloud by a vector of **point3d** (p. 16).

3.5.2 Constructor & Destructor Documentation

3.5.2.1 point3dCloud()

```
point3dCloud::point3dCloud ( )
```

A constructor.

This default constructor initializes this->cloud.

3.5.3 Member Function Documentation

3.5.3.1 meanNeighboursDistance()

```
double point3dCloud::meanNeighboursDistance ( )
```

Gives the mean of the distances between neighbours.

Returns

The mean.

3.5.3.2 operator[]()

```
point3d point3dCloud::operator[] (
            int i )
```

Overloads [].

Parameters

i	The index to get.

Returns

The **point3d** (p. 16).

3.5.3.3 pointCloud2ply()

```
void point3dCloud::pointCloud2ply (
            string target )
```

Generates .ply file from point cloud values.

Parameters

poincloud	The corresponding point cloud.

3.5.3.4 push_back()

```
void point3dCloud::push_back (
             point3d point )
```

Adds a point in the pointcloud.

Parameters

point	The **point3d** (p. 16) to add.

3.5.3.5 showOnImage()

```
void point3dCloud::showOnImage (
            Mat & image )
```

Shows the result on an image.

This method allows us to see directly the result without visualizing a 3d pointcloud.

3.5.3.6 size()

```
int point3dCloud::size ( )
```

Gives the size of the pointcloud.

Returns

The size of the pointcloud.

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

- C:/Users/romai/Documents/Mes_documents/X/3A/INF552/Projet-Inf552/project/include/ **point3dCloud.hpp**
- C:/Users/romai/Documents/Mes_documents/X/3A/INF552/Projet-Inf552/project/src/ **point3dCloud.cpp**

3.6 product Class Reference

A class to encapsulate the calculation of scalar and vectorial product.

`#include <product.hpp>`

Public Member Functions

- **product** (Vec3d v1, Vec3d v2)
  A constructor.
- double **getScalar** ()
  Computes the scalar product between the two points.
- Vec3d **getVectorial** ()
  Computes the vectorial product between the two points.

3.6.1 Detailed Description

A class to encapsulate the calculation of scalar and vectorial product.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 product()

```
product::product (
            Vec3d v1,
            Vec3d v2 )
```

A constructor.

Parameters

v1	The first vector.
v2	The second vector.

3.6.3 Member Function Documentation

3.6.3.1 getScalar()

```
double product::getScalar ( )
```

Computes the scalar product between the two points.

Returns

The scalar product.

3.6.3.2 getVectorial()

```
Vec3d product::getVectorial ( )
```

Computes the vectorial product between the two points.

Returns

The vectorial product.

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

- C:/Users/romai/Documents/Mes_documents/X/3A/INF552/Projet-Inf552/project/include/ **product.hpp**
- C:/Users/romai/Documents/Mes_documents/X/3A/INF552/Projet-Inf552/project/src/ **product.cpp**

3.7 projectData Class Reference

A class to represent the **projectData** (p. 22) for a situation.

```
#include <projectData.hpp>
```

Public Member Functions

- **projectData** (string filename, int gaussianBlur=3)
  A constructor.
- Matx33d **getCameraMatrix** ()
  A method to get the camera matrix.
- Mat **getLeftImage** ()
  A method to get the left image.
- Mat **getDisparity** ()
  A method to get the disparity.
- **point3dCloud pointCloudFromData** ()
  Generates a 3d point cloud from left image + disparity + transformation matrix.

3.7.1 Detailed Description

A class to represent the **projectData** (p. 22) for a situation.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 projectData()

```
projectData::projectData (
            string filename,
            int gaussianBlur = 3 )
```

A constructor.

Parameters

filename	The path to the file. It should be like : "something/aachen_000029_000019" and the constructor will manage to open the files with the good name and extentions.
n	Range of the Gaussian Blur. Equals to 2 by default.

3.7.3 Member Function Documentation

3.7.3.1 getCameraMatrix()

```
Matx33d projectData::getCameraMatrix ( )
```

A method to get the camera matrix.

Returns

The camera matrix.

3.7.3.2 getDisparity()

```
Mat projectData::getDisparity ( )
```

A method to get the disparity.

Returns

The disparity.

3.7.3.3 getLeftImage()

```
Mat projectData::getLeftImage ( )
```

A method to get the left image.

Returns

The left image.

3.7.3.4 pointCloudFromData()

```
point3dCloud projectData::pointCloudFromData ( )
```

Generates a 3d point cloud from left image + disparity + transformation matrix.

Exports the result as .ply file.

Parameters

left_image	The left image.
disparity	The disparity.
N	The matrix of correspondence : it can transform the disparity into 3d point.

Returns

The point cloud as vector<pair<Vec3d, Vec3b>>.

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

- C:/Users/romai/Documents/Mes_documents/X/3A/INF552/Projet-Inf552/project/include/ **projectData.hpp**
- C:/Users/romai/Documents/Mes_documents/X/3A/INF552/Projet-Inf552/project/src/ **projectData.cpp**

3.8 ransac Class Reference

A class to calculate the ransac from pointcloud to find correlated planes or lines.

```
#include <ransac.hpp>
```

Public Member Functions

- **ransac** (int n_iterations, double epsilon)

  A constructor.
- **point3dCloud fit3dPlane** (**point3dCloud** pointCloud, bool uniformColor=false, Vec3b color=Vec3b(0, 0, 0))

  Extract the most correlated points (plane model).
- **point3dCloud fit3dLine** (**point3dCloud** pointCloud, **plane** p, bool uniformColor=false, Vec3b color=Vec3b(0, 0, 0), int nlines=1, double minDistBetweenLines=0)

  Extract the most correlated points (line model).

3.8.1 Detailed Description

A class to calculate the ransac from pointcloud to find correlated planes or lines.

3.8.2 Constructor & Destructor Documentation

3.8.2.1 ransac()

```
ransac::ransac (
            int n_iterations,
            double epsilon )
```

A constructor.

Parameters

n_iterations	The number of iterations for the algorithm.
epsilon	Threshold.

3.8.3 Member Function Documentation

3.8.3.1 fit3dLine()

```
point3dCloud ransac::fit3dLine (
             point3dCloud pointCloud,
             plane p,
            bool uniformColor = false,
            Vec3b color = Vec3b(0, 0, 0),
            int nlines = 1,
            double minDistBetweenLines = 0 )
```

Extract the most correlated points (line model).

Parameters

pointCloud	The considered point cloud.
p	The plane to be $\sim$ orthogonal of.
uniformColor	A boolean to tell if we want to have a uniform color or not.
color	The color.
nlines	The number of lines desired.
minDistBetweenLines	The minimum distance between two lines (to avoid getting all the lines in the same space).

Returns

List of points that correlate the most (line model) as **point3dCloud** (p. 18).

3.8.3.2 fit3dPlane()

```
point3dCloud ransac::fit3dPlane (
             point3dCloud pointCloud,
            bool uniformColor = false,
            Vec3b color = Vec3b(0, 0, 0) )
```

Extract the most correlated points (plane model).

Parameters

pointCloud	The considered point cloud.
uniformColor	A boolean to tell if we want to have a uniform color or not.
color	The color.

Returns

List of points that correlate the most (plane model) as **point3dCloud** (p. 18).

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

- C:/Users/romai/Documents/Mes_documents/X/3A/INF552/Projet-Inf552/project/include/ **ransac.hpp**
- C:/Users/romai/Documents/Mes_documents/X/3A/INF552/Projet-Inf552/project/src/ **ransac.cpp**

Chapter 4

File Documentation

4.1 C:/Users/romai/Documents/Mes_documents/X/3A/INF552/Projet-Inf552/project/include/line3d.hpp File Reference

```
#include <iostream>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/features2d/features2d.hpp>
#include <opencv2/calib3d/calib3d.hpp>
#include <opencv2/imgproc.hpp>
#include "product.hpp"
```

Classes

- class **line3d**

  A class to represent lines in the 3d space.

4.2 C:/Users/romai/Documents/Mes_documents/X/3A/INF552/Projet-Inf552/project/include/line3dCloud.hpp File Reference

```
#include <iostream>
#include <windows.h>
#include <fstream>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/features2d/features2d.hpp>
#include <opencv2/calib3d/calib3d.hpp>
#include <opencv2/imgproc.hpp>
#include "line3d.hpp"
#include "plane.hpp"
```

Classes

- class **line3dCloud**

  A class to represent clouds of lines in the 3d space.

4.3 C:/Users/romai/Documents/Mes_documents/X/3A/INF552/Projet-Inf552/project/include/parameters.h File Reference

Variables

- const double **MIN_DISPARITY** = 10
- const double **MIN_COSINE** = 0.8

4.3.1 Variable Documentation

4.3.1.1 MIN_COSINE

```
const double MIN_COSINE = 0.8
```

4.3.1.2 MIN_DISPARITY

```
const double MIN_DISPARITY = 10
```

4.4 C:/Users/romai/Documents/Mes_documents/X/3A/INF552/Projet-Inf552/project/include/plane.hpp File Reference

```
#include <iostream>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/features2d/features2d.hpp>
#include <opencv2/calib3d/calib3d.hpp>
#include <opencv2/imgproc.hpp>
#include "point3dCloud.hpp"
#include "line3d.hpp"
```

Classes

- class **plane**

  A class to represent plane in the 3d space.

4.5 C:/Users/romai/Documents/Mes_documents/X/3A/INF552/Projet-Inf552/project/include/point3d.hpp File Reference

```
#include <iostream>
#include <windows.h>
#include <fstream>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/features2d/features2d.hpp>
#include <opencv2/calib3d/calib3d.hpp>
#include <opencv2/imgproc.hpp>
```

Classes

- class **point3d**

  A class to represent points in the 3d space.

4.6 C:/Users/romai/Documents/Mes_documents/X/3A/INF552/Projet-Inf552/project/include/point3dCloud.hpp File Reference

```
#include <iostream>
#include <windows.h>
#include <fstream>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/features2d/features2d.hpp>
#include <opencv2/calib3d/calib3d.hpp>
#include <opencv2/imgproc.hpp>
#include "point3d.hpp"
```

Classes

- class **point3dCloud**

  A class to represent clouds of points in the 3d space.

4.7 C:/Users/romai/Documents/Mes_documents/X/3A/INF552/Projet-Inf552/project/include/product.hpp File Reference

```
#include <iostream>
#include <windows.h>
#include <fstream>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/features2d/features2d.hpp>
#include <opencv2/calib3d/calib3d.hpp>
#include <opencv2/imgproc.hpp>
```

Classes

- class **product**

  A class to encapsulate the calculation of scalar and vectorial product.

4.8 C:/Users/romai/Documents/Mes_documents/X/3A/INF552/Projet-Inf552/project/include/projectData.hpp File Reference

```
#include <iostream>
#include <windows.h>
#include <fstream>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/features2d/features2d.hpp>
#include <opencv2/calib3d/calib3d.hpp>
#include <opencv2/imgproc.hpp>
#include "lib\json.hpp"
#include "point3dCloud.hpp"
#include "parameters.hpp"
```

Classes

- class **projectData**

  *A class to represent the **projectData** (p. 22) for a situation.*

Typedefs

- using **json** = nlohmann::json

4.8.1 Typedef Documentation

4.8.1.1 json

```
using json = nlohmann::json
```

4.9 C:/Users/romai/Documents/Mes_documents/X/3A/INF552/Projet-Inf552/project/include/ransac.hpp File Reference

```
#include <iostream>
#include <windows.h>
#include <fstream>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/features2d/features2d.hpp>
#include <opencv2/calib3d/calib3d.hpp>
#include <opencv2/imgproc.hpp>
#include "plane.hpp"
#include "line3dCloud.hpp"
#include "parameters.hpp"
```

Classes

- class **ransac**

  A class to calculate the ransac from pointcloud to find correlated planes or lines.

4.10 C:/Users/romai/Documents/Mes_documents/X/3A/INF552/Projet-Inf552/project/src/line3d.cpp File Reference

```
#include "line3d.hpp"
```

Functions

- ostream & **operator**<< (ostream &os, const **line3d** &l)

4.10.1 Function Documentation

4.10.1.1 operator<<()

```
ostream& operator<< (
            ostream & os,
            const  line3d & l )
```

Parameters

os	Considered stream.
l	Considered line.

Returns

The ofstream to be printed.

4.11 C:/Users/romai/Documents/Mes_documents/X/3A/INF552/Projet-Inf552/project/src/line3dCloud.cpp File Reference

```
#include "line3dCloud.hpp"
```

4.12 C:/Users/romai/Documents/Mes_documents/X/3A/INF552/Projet-Inf552/project/src/plane.cpp File Reference

```
#include "plane.hpp"
```

Functions

- ostream & **operator**<< (ostream &os, const **plane** &p)

4.12.1 Function Documentation

4.12.1.1 operator<<()

```
ostream& operator<< (
            ostream & os,
            const  plane & p )
```

Parameters

os	Considered stream.
p	Considered plane.

Returns

The ofstream to be printed.

4.13 C:/Users/romai/Documents/Mes_documents/X/3A/INF552/Projet-Inf552/project/src/point3d.cpp File Reference

```
#include "point3d.hpp"
```

4.14 C:/Users/romai/Documents/Mes_documents/X/3A/INF552/Projet-Inf552/project/src/point3dCloud.cpp File Reference

```
#include "point3dCloud.hpp"
```

4.15 C:/Users/romai/Documents/Mes_documents/X/3A/INF552/Projet-Inf552/project/src/product.cpp File Reference

```
#include "product.hpp"
```

4.16 C:/Users/romai/Documents/Mes_documents/X/3A/INF552/Projet-Inf552/project/src/projectData.cpp File Reference

```
#include "projectData.hpp"
```

Functions

- void **setcursor** (bool visible, DWORD size)

  Function for hiding/showing cursor : hiding with setcursror(0, 0); reinitialisation with setcursor(1, 10).
- bool **hasToBeTreated** (int i, int j, double d, const Mat &left_image)

  Determines if the pixel should be treated.

4.16.1 Function Documentation

4.16.1.1 hasToBeTreated()

```
bool hasToBeTreated (
            int i,
            int j,
            double d,
            const Mat & left_image )  [inline]
```

Determines if the pixel should be treated.

Parameters

d	The disparity between left/right images.

Returns

Whether the pixel should be considered.

4.16.1.2 setcursor()

```
void setcursor (
            bool visible,
            DWORD size )
```

Function for hiding/showing cursor : hiding with setcursror(0, 0); reinitialisation with setcursor(1, 10).

Parameters

visible	Whether the cursor should be visible.
size	The size of the cursor.

4.17 C:/Users/romai/Documents/Mes_documents/X/3A/INF552/Projet-Inf552/project/src/ransac.cpp File Reference

```
#include "ransac.hpp"
```

Index