project

..

Generated by Doxygen 1.8.11

Contents

Chapter 1

Empty doxygen CMake subproject

Screenshots

Markdown main page

Doxgen generated API documentation

Use in your project

```
1 cd yourproject
2 git clone EmptyDoxygenCMake doc
3 rm -rf doc/.git
```

Edit `doc/doxygen/Doxyfile.in` *INPUT* and *STRIP_FROM_PATH* to fit your project. Also you can change where the documentation will be installed in `doc/CMakeLists.txt` in the first ∗#TODO∗ section. If you do not want your documentation to be part of installation, just leave the section commented.

To your main CMakeLists.txt you want to add `add_subdirectory(doc)` in order to include documentation build.

Standalone doc builder

Sometimes you just need to build docs from some pile of sources, and here can this repository help you. You may not even have a CMake Project setup yet, in this case you need to uncomment code after second ∗#TODO∗ section in `doc/CMakeLists.txt` and documentation will act as a standalone CMake project.

Prerequisites

- CMake 3.2 and newer
- C++14 capable compiler
- Doxygen for docs (*Graphviz for more graphs in docs, PlantUML for more UML diagrams*, PlantUML needs java)

Linux

- Arch Linux: `sudo pacman -S cmake g++ graphviz git`
  - download plantuml.jar and have it somewhere where *PATH* points to
- Ubuntu 16.04: `sudo apt-get install cmake g++ graphviz plantuml git`

Windows

- Install msys2 and install these packages:
  - `mingw32/mingw-w64-i686-gcc mingw32/mingw-w64-i686-cmake git mingw32/mingw-w64-i6 msys/make msys/doxygen msys/make`
- For graphs in documentation install Graphviz (to `c:\Program Files\Graphviz`, so scripts can find it) and add its `bin` subdirectory to *PATH*, install java (have it on *PATH*), download PlantUML jar file and have it on *PATH*.

Building doc

You can use *doc* target to build and *install* target to install the documentation (if installation part of CMakeLists.txt is uncommented).

Documentation will be in your `[build_directory]/doc/index.html` or `[build_directory]/doc/doc/index.htm` depending of whether used as standalone or as a CMake subdirectory.

Linux

```
1 mkdir build
2 cmake ../path/to/CMake/directory/
3 make doc
```

Windows

```
1 mkdir build
2 cmake ../path/to/CMake/directory/ -G "MSYS Makefiles"
3 make doc
```

Chapter 2

docs

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

File Documentation

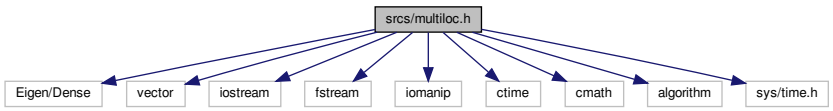
4.1 docs/README.md File Reference

4.2 readme.md File Reference

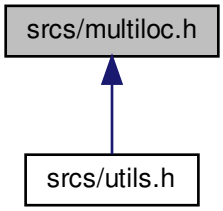
4.3 srcs/multiloc.h File Reference

```
#include "Eigen/Dense"
#include <vector>
#include <iostream>
#include <fstream>
#include <iomanip>
#include <ctime>
#include <cmath>
#include <algorithm>
#include <sys/time.h>
```

Include dependency graph for multiloc.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef vector< double > one_Tuple
- typedef vector< one_Tuple > two_Tuple
- typedef vector< two_Tuple > three_Tuple

Functions

- void sortarry (two_Tuple &list, int temp)

  对一个二维的*vector*<*vector*<*double*>>的*list*进行按照每一个*vector*<*double*>的第 2 个元素进行升序
- bool finditems (double value, two_Tuple &list)

  判断*value*元素是否属于*2*维*vector*中的第一个元素,如果*value*属于*list*中返回*true,*否则返回*false.*
- bool finditems (double value, one_Tuple &list)

  判断*value*元素是否属于*1*维*vector*的*list,*如果*value*属于*list*中返回*true,*否则返回*false.*
- int classification (const one_Tuple &list_one, three_Tuple &list)

  坐标点聚类算法,即判断一维的*vector*<*double*>类型是否属于交点坐标聚类集合*list*中的一类,如果属于其中的一类, 则将该*1*维的*vector*归入到该类中,否则单独归为*list*中新的一类.
- bool isequal (const two_Tuple &list, const two_Tuple &list1)

  判断两个二维*vector*即*list*和*list1*是否相等,如果两个相等,返回*true.*否则返回*false.*
- void productiondot (one_Tuple &m_data, two_Tuple &angle, three_Tuple &list_three, int &numberruncount)

  测向线交点坐标聚类算法，即第一步读取靶点和测向线的数据信息，将两两测向线进行相交的到交点坐标． 第*2*步即将交点坐标送入坐标聚类算法中，进行坐标点的聚类．其中*m_data*为靶点的坐标．*angle*为每个靶点对应的测向 线的角度信息．*list_three*为输出的坐标点聚类集合．*numberruncount*为测向线数量用于剔除独立的测向线聚类集合．
- void clearreplace (three_Tuple &means_result, three_Tuple &f_result, int &numberruncount)

  将坐标点聚类集合转化为测向线聚类集合．其中*means_result*为输入的坐标点聚类集合． *f_result*为输出的测向线聚类集合．*numberruncount*为测向线数量用于剔除独立的测向线聚类集合．
- void Demo4Mult_Targetlocalization (three_Tuple &f_result, two_Tuple &list_two, string &str)

  将测向线聚类集合送入目标定位算法中，进行目标点定位．其中*f_result*为输入的测向线聚类集合．*list_two*为所有目标点的坐标．*str*为目标点的坐标信息以文件形式输出的文件路径．
- void Demo1Single_Target (one_Tuple &angle, one_Tuple &data, MatrixXd &result)

  单目标定位算法根据输入的测向线角度和靶点坐标求解目标点的具体位置信息． *data*为所有靶点的坐标．*angle*为每个靶点对应测向线的角度信息．定位出来的目标点坐标将保存在*result*.
- void Demo2 (one_Tuple &m_data, two_Tuple &angle, three_Tuple &list_three, three_Tuple &list_three2)

  对靶点数据和测向线数据通过聚类算法得到测向线聚类坐标集合．其中*m_data*为所有靶点的坐标．*angle*为每个靶点对应测向线的角度信息．交点坐标聚类集合为*list_three*，测向线组聚类集合为*list_*↩*threetwo*.

- void Demo3count (two_Tuple &A, two_Tuple &B)

  判断测向线组中任意两类是否具有重复的测向线信息．将第一类中非重复的部分归入第*2*类中． *A*为一类的测向线聚类组．*B*为另一类的测向线聚类组．
- void Demo3intersect (two_Tuple &A, two_Tuple &B, ofstream &out_stream, int numberruncount)

  判断测向线组中任意两类是否具有重复的测向线信息．将重复的多余的部分进行删除． *A*为一类的测向线聚类组． *B*为另一类的测向线聚类组．*numberruncount*为测向线数量用于剔除独立的测向线聚类集合．
- void Demo3 (three_Tuple &f_result, three_Tuple &list_three, string &str, int &numberruncount)

  判断一类测向线聚类组与整个测向线聚类集合中每一类是否具有重复的测向线信息．将重复多余的测向线信息进行删除，同时更新测向线聚类集合． *A*为一类的测向线聚类组．*B*为所有测向线聚类集合．*numberruncount*为测向线数量用于剔除独立的测向线聚类集合．
- void Demo3remove (two_Tuple &A, three_Tuple &B, ofstream &out_stream, int &numberruncount)

  将测向线聚类集合进行测向线野值检测. 将重复多余的测向线信息进行删除,同时更新测向线聚类集合.将删除的测向线信息保存到文件中. *f_result*为一类的测向线聚类组.*list_three*为所有测向线聚类集合.*str*为删除的测向线信息以文件形式输出的路径.*numberruncount*为测向线 数量用于剔除独立的测向线聚类集合．
- bool Demo5inputdata (two_Tuple &angles, one_Tuple &m_coordinate)

  测试数据数据通过命令行进行在线输入,对输入的靶点数据坐标和测向线角度信息进行保存,剔除重复输入的信息. 输入的测向线角度信息保存在*angles*,输入的靶点坐标信息保存在*m_coordinate*.当输入出现数据格式出现错误时,返回*false*.正确返回*true*.

Variables

- int numberruncount

4.3.1 Typedef Documentation

4.3.1.1 typedef vector<double> one_Tuple

使用one_Tuple代替vector<double>

4.3.1.2 typedef vector<two_Tuple> three_Tuple

使用three_Tuple代替vector<vector<vector<double>>>

4.3.1.3 typedef vector<one_Tuple> two_Tuple

使用two_Tuple代替vector<vector<double>>

4.3.2 Function Documentation

4.3.2.1 int classification (const one_Tuple & *list_one,* three_Tuple & *list*)

坐标点聚类算法,即判断一维的vector<double>类型是否属于交点坐标聚类集合list中的一类,如果属于其中的一类, 则将该1维的vector归入到该类中,否则单独归为list中新的一类.

Parameters

*list_one*是*vector<double>*类型	
*list*为*vector<vector<vector<double>>>*类型	

Returns

int类型

4.3.2.2 void clearreplace (three_Tuple & *means_result,* three_Tuple & *f_result,* int & *numberruncount*)

将坐标点聚类集合转化为测向线聚类集合．其中means_result为输入的坐标点聚类集合． f_result为输出的测向线聚类集合．numberruncount为测向线数量用于剔除独立的测向线聚类集合．

Parameters

*means_result*为*vector*<*vector*<*double*>>类型	
*f_result*为*vector*<*vector*<*double*>>类型	
*numberruncount*为*int*类型	

Returns

无.

4.3.2.3 void Demo1Single_Target (one_Tuple & *angle,* one_Tuple & *data,* MatrixXd & *result*)

单目标定位算法根据输入的测向线角度和靶点坐标求解目标点的具体位置信息． data为所有靶点的坐标．angle为每个靶点对应测向线的角度信息．定位出来的目标点坐标将保存在result．

Parameters

*angle*为*vector*<*double*>类型	
*data*为*vector*<*double*>类型	
*result*为*MatrixXd*类型	

Returns

无.

4.3.2.4 void Demo2 (one_Tuple & *m_data,* two_Tuple & *angle,* three_Tuple & *list_three,* three_Tuple & *list_three2*)

对靶点数据和测向线数据通过聚类算法得到测向线聚类坐标集合．其中m_data为所有靶点的坐标．angle为每个靶点对应测向线的角度信息．交点坐标聚类集合为list_three，测向线组聚类集合为list_↩threetwo．

Parameters

*angle*为*vector*<*double*>类型	
*data*为*vector*<*double*>类型	
*result*为*MatrixXd*类型	

Returns

无

4.3.2.5 void Demo3 (three_Tuple & *f_result,* three_Tuple & *list_three,* string & *str,* int & *numberruncount*)

判断一类测向线聚类组与整个测向线聚类集合中每一类是否具有重复的测向线信息．将重复多余的测向线信息进行删除，同时更新测向线聚类集合． A为一类的测向线聚类组．B为所有测向线聚类集合．numberruncount为测向线数量用于剔除独立的测向线聚类集合．

Parameters

*A*为*vector*<*vector*<*double*>>类型	
*B*为*vector*<*vector*<*vector*<*double*>>>类型	
*numberruncount*为*int*类型	

Returns

无

4.3.2.6 void Demo3count (two_Tuple & *A,* two_Tuple & *B*)

判断测向线组中任意两类是否具有重复的测向线信息．将第一类中非重复的部分归入第2类中． A为一类的测向线聚类组．B为另一类的测向线聚类组．

Parameters

*A*为*vector*<*vector*<*double*>>类型	
*B*为*vector*<*vector*<*double*>>类型	

Returns

无

4.3.2.7 void Demo3intersect (two_Tuple & *A,* two_Tuple & *B,* ofstream & *out_stream,* int *numberruncount*)

判断测向线组中任意两类是否具有重复的测向线信息．将重复的多余的部分进行删除． A为一类的测向线聚类组． B为另一类的测向线聚类组．numberruncount为测向线数量用于剔除独立的测向线聚类集合．

Parameters

*A*为*vector*<*vector*<*double*>>类型	
*B*为*vector*<*vector*<*double*>>类型	
*numberruncount*为*int*类型	

Returns

无

4.3.2.8 void Demo3remove (two_Tuple & *A,* three_Tuple & *B,* ofstream & *out_stream,* int & *numberruncount*)

将测向线聚类集合进行测向线野值检测. 将重复多余的测向线信息进行删除,同时更新测向线聚类集合.将删除的测向线信息保存到文件中. f_result为一类的测向线聚类组.list_three为所有测向线聚类集合.str为删除的测向线信息以文件形式输出的路径.numberruncount为测向线 数量用于剔除独立的测向线聚类集合.

Parameters

A为*vector*<*vector*<*double*>>类型	
B为*vector*<*vector*<*vector*<*double*>>>类型	
*str*为*string*类型	
*numberruncount*为*int*类型	

Returns

无

4.3.2.9 void Demo4Mult_Targetlocalization (three_Tuple & *f_result,* two_Tuple & *list_two,* string & *str*)

将测向线聚类集合送入目标定位算法中，进行目标点定位．其中f_result为输入的测向线聚类集合．list_two为所有目标点的坐标．str为目标点的坐标信息以文件形式输出的文件路径．

Parameters

*f_result*为*vector*<*vector*<*double*>>类型	
*list_two*为*vector*<*vector*<*double*>>类型	
*str*为*string*类型	

Returns

无.

4.3.2.10 bool Demo5inputdata (two_Tuple & *angles,* one_Tuple & *m_coordinate*)

测试数据数据通过命令行进行在线输入,对输入的靶点数据坐标和测向线角度信息进行保存,剔除重复输入的信息. 输入的测向线角度信息保存在angles,输入的靶点坐标信息保存在m_coordinate.当输入出现数据格式出现错误时,返回false.正确返回true.

Parameters

*angles*为*vector*<*vector*<*double*>>类型	
*m_coordinate*为<*vector*<*double*>类型	

Returns

bool类型

4.3.2.11 bool finditems (double *value,* two_Tuple & *list*)

判断value元素是否属于2维vector中的第一个元素,如果value属于list中返回true,否则返回false.

Parameters

*value*为*double*类型	
*list*是*vector*<*vector*<*double*>>类型	

Returns

bool类型

4.3.2.12 bool finditems (double *value,* one_Tuple & *list*)

判断value元素是否属于1维vector的list,如果value属于list中返回true,否则返回false.

Parameters

*list*是*vector*<*double*>类型	
*value*为*double*类型	

Returns

bool类型

4.3.2.13 bool isequal (const two_Tuple & *list,* const two_Tuple & *list1*)

判断两个二维vector即list和list1是否相等,如果两个相等,返回true.否则返回false.

Parameters

*list*为*vector*<*vector*<*double*>>类型	
*list1*为*vector*<*vector*<*double*>>类型	

Returns

bool类型

4.3.2.14 void productiondot (one_Tuple & *m_data,* two_Tuple & *angle,* three_Tuple & *list_three,* int & *numberruncount*)

测向线交点坐标聚类算法，即第一步读取靶点和测向线的数据信息，将两两测向线进行相交的到交点坐标． 第2步即将交点坐标送入坐标聚类算法中，进行坐标点的聚类．其中m_data为靶点的坐标．angle为每个靶点对应的测向 线的角度信息．list_three为输出的坐标点聚类集合．numberruncount为测向线数量用于剔除独立的测向线聚类集合．

Parameters

*m_data*为*vector*<*double*>类型	
*angle*为*vector*<*vector*<*double*>>类型	
*list_three*为*vector*<*vector*<*double*>>类型	
*numberruncount*为*int*类型	

Returns

无

4.3.2.15 void sortarry (two_Tuple & *list,* int *temp*)

对一个二维的vector<vector<double>>的list进行按照每一个vector<double>的第 2 个元素进行升序

Parameters

*list*是*vector*<*vector*<*double*>>类型	
*temp*为*int*类型	

Returns

无返回

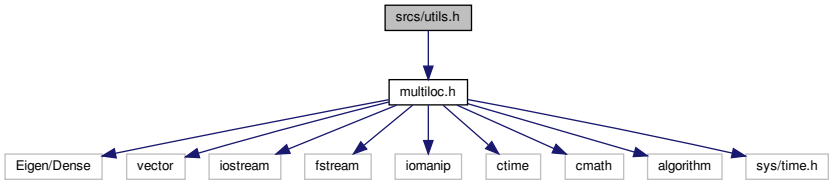
4.3.3 Variable Documentation

4.3.3.1 int numberruncount

doxygen对c++代码注释参考:https://blog.csdn.net/czyt1988/article/details/8901191

4.4 srcs/utils.h File Reference

`#include <multiloc.h>`
Include dependency graph for utils.h:



Functions

- void print (one_Tuple &list)

  输出*1*维*vector*<*double*>的数据到命令行
- void print (two_Tuple &list)

  输出*2*维*vector*<*vector*<*double*>>的数据到命令行
- void print (three_Tuple &list)

  输出*3*维*vector*<*vector*<*vector*<*double*>>>的数据到命令行
- void print (MatrixXd &list, int &total)

  输出*2*维*MatrixXd*前*total*个的数据到命令行
- bool read_data (const char *filename, one_Tuple &tuples)

  : 从文件中读取数据内容，将输出的结果保存在一维的*vector*<*double*>中． *filename*为需要读取数据的文件名路径，*tuples*保存读取的数据．读取数据成功，返回*true*．否则，返回*false*．
- bool read_file (string &Docement, one_Tuple &m_data, two_Tuple &angle)

  从文件中读取用于多目标定位的靶点和测向线信息，其中*filename*为需要读取数据的文件路径， 将读取的靶点坐标保存在*m_data*中，将读取的测向线信息保存在*angle*中．
- bool read_file (string &Docement, one_Tuple &m_data, one_Tuple &angle)

  从文件中读取用于单目标定位的靶点和测向线信息，其中*filename*为需要读取数据的文件路径， 将读取的靶点坐标保存在*m_data*中，将读取的测向线信息保存在*angle*中．

4.4.1 Function Documentation

4.4.1.1 void print (one_Tuple & *list*)

输出1维vector<double>的数据到命令行

Parameters

*list*为*vector*<*double*>类型	

Returns

无

4.4.1.2 void print (two_Tuple & *list*)

输出2维vector<vector<double>>的数据到命令行

Parameters

*list*为*vector*<*vector*<*double*>>类型	

Returns

无

4.4.1.3 void print (three_Tuple & *list*)

输出3维vector<vector<vector<double>>>的数据到命令行

Parameters

*list*为*vector*<*vector*<*vector*<*double*>>>类型	

Returns

无

4.4.1.4 void print (MatrixXd & *list,* int & *total*)

输出2维MatrixXd前total个的数据到命令行

Parameters

*list*为*MatrixXd*类型	
*total*为*int*类型	

Returns

无

4.4.1.5 bool read_data (const char * *filename,* one_Tuple & *tuples*)

: 从文件中读取数据内容，将输出的结果保存在一维的vector<double>中． filename为需要读取数据的文件名路径，tuples保存读取的数据．读取数据成功，返回true．否则，返回false．

Parameters

4.4.1.6 bool read_file (string & *Docement,* one_Tuple & *m_data,* two_Tuple & *angle*)

从文件中读取用于多目标定位的靶点和测向线信息，其中filename为需要读取数据的文件路径， 将读取的靶点坐标保存在m_data中，将读取的测向线信息保存在angle中.

Parameters

*Docement*为*string*类型	
*m_data*为*vector*<*double*>类型	
*angle*为*vector*<*vector*<*double*>>类型	

Returns

bool类型，读取数据成功，返回true．否则，返回false.

4.4.1.7 bool read_file (string & *Docement,* one_Tuple & *m_data,* one_Tuple & *angle*)

从文件中读取用于单目标定位的靶点和测向线信息，其中filename为需要读取数据的文件路径， 将读取的靶点坐标保存在m_data中，将读取的测向线信息保存在angle中.

Parameters

*Docement*为*string*类型	
m_data	为vector<double>类型
angle	为vector<double>类型

Returns

bool类型，读取数据成功，返回true；否则，返回 false。

Index