pmvs-triangulation

1.1

Generated by Doxygen 1.6.3

Tue Aug 20 10:28:08 2013

Contents

Chapter 1

pmvs-triangulation

There are essentially two main programs : delaunay and triangclean.

They are based on the CGAL library (Computational Geometry Algorithms Library, http://www.cgal.org)

1.1 delaunay

it builds the delaunay triangulation and does the ray tracing. Depends on: delaunay.h triangdefs.h delaunay.cpp config.cpp delaunay_io.cpp addcells.cpp intersect.cpp gviewer.cpp

1.2 triangclean

It analyzes ray tracing information and extracts surface facets. Depends on : delaunay.h triangdefs.h triangclean.cpp config.cpp extract.cpp smooth.cpp gviewer.cpp delaunay_io.cpp

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 Intersect Struct Reference

information about an intersectection between a ray and a facet

```
#include "triangdefs.h"
```

Public Attributes

- InterType **int_type**
- Vertex_handle v1

  type of intersection (inside facet, on edge ...)

- Vertex_handle v2

  vertex for vertex intersection, 1st edge vertex for edge intersection

- int intersected_facet

  2nd edge vertex for edge intersection

4.1.1 Detailed Description

information about an intersectection between a ray and a facet

Definition at line 89 of file triangdefs.h.

4.1.2 Member Data Documentation

4.1.2.1 int Intersect::intersected_facet

2nd edge vertex for edge intersection

Definition at line 93 of file triangdefs.h.

4.1.2.2 Vertex_handle Intersect::v1

type of intersection (inside facet, on edge ...)

Definition at line 91 of file triangdefs.h.

4.1.2.3 Vertex_handle Intersect::v2

vertex for vertex intersection, 1st edge vertex for edge intersection

Definition at line 92 of file triangdefs.h.

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

- triangdefs.h

4.2 TrParams Class Reference

params for facets extraction

```
#include "delaunay.h"
```

Public Attributes

- int **do_clean**
- double lgr_threshold

    OR of various cleaning methods.

- double surf_threshold

    inf limit for edge square lgr

- double average_volume

    inf limit for facet surf

- float cosinus_thresh

    volume of a regular tetra with edges = average edge

- bool moy_normal

    cosine max value for normal angles

- float smooth_coef

    use average vertices normal instead of individual normals

- int smooth_nb_iter

    coef for smothness

- int nb_intersect

    nb of iteration in smoothing process

- ExtractType extract_type

    minimum nb of intersection

- int extract_mode

    extraction type : maxflow, threshold, weigthed threshold

4.2.1 Detailed Description

params for facets extraction

Definition at line 44 of file delaunay.h.

4.2.2 Member Data Documentation

4.2.2.1 double TrParams::average_volume

inf limit for facet surf

Definition at line 56 of file delaunay.h.

4.2.2.2 float TrParams::cosinus_thresh

volume of a regular tetra with edges = average edge

Definition at line 57 of file delaunay.h.

4.2.2.3 int TrParams::extract_mode

extraction type : maxflow, threshold, weigthed threshold

Definition at line 63 of file delaunay.h.

4.2.2.4 ExtractType TrParams::extract_type

minimum nb of intersection

Definition at line 62 of file delaunay.h.

4.2.2.5 double TrParams::lgr_threshold

OR of various cleaning methods.

Definition at line 54 of file delaunay.h.

4.2.2.6 bool TrParams::moy_normal

cosine max value for normal angles

Definition at line 58 of file delaunay.h.

4.2.2.7 int TrParams::nb_intersect

nb of iteration in smoothing process

Definition at line 61 of file delaunay.h.

4.2.2.8 float TrParams::smooth_coef

use average vertices normal instead of individual normals

Definition at line 59 of file delaunay.h.

4.2.2.9 int TrParams::smooth_nb_iter

coef for smothness

Definition at line 60 of file delaunay.h.

4.2.2.10 double TrParams::surf_threshold

inf limit for edge square lgr

Definition at line 55 of file delaunay.h.

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

- delaunay.h

4.3 TrStats Class Reference

Store statistics.

```
#include "delaunay.h"
```

Public Member Functions

- void sum (TrStats ∗stats2)

  Sum of 2 statistics objects (to gather statistics from multiple threads).

Public Attributes

- unsigned long **nb_tested_facets**
- unsigned long nb_tot_intersect

  nb of tested facets

- unsigned long nb_intv

  nb of intersections found

- unsigned long nb_int_edge

  nb of intersections on vertex

- unsigned long nb_test_facets

  nb of intersections on edge

- int nb_tetra0

  nb of facets candidate for ray intersection

- unsigned long nb_tested_lv0

  nb of tetraedrons with a camera vertex

- unsigned long nb_rayons

  nb of intersected facets of tetrahedrons with a camera as vertex

4.3.1 Detailed Description

Store statistics.

Definition at line 27 of file delaunay.h.

4.3.2 Member Function Documentation

4.3.2.1 void TrStats::sum (TrStats ∗ *stats2*)

Sum of 2 statistics objects (to gather statistics from multiple threads).

Definition at line 95 of file config.cpp.

4.3.3 Member Data Documentation

4.3.3.1 unsigned long TrStats::nb_int_edge

nb of intersections on vertex

Definition at line 36 of file delaunay.h.

4.3.3.2 unsigned long TrStats::nb_intv

nb of intersections found

Definition at line 35 of file delaunay.h.

4.3.3.3 unsigned long TrStats::nb_rayons

nb of intersected facets of tetrahedrons with a camera as vertex

Definition at line 40 of file delaunay.h.

4.3.3.4 unsigned long TrStats::nb_test_facets

nb of intersections on edge

Definition at line 37 of file delaunay.h.

4.3.3.5 unsigned long TrStats::nb_tested_lv0

nb of tetraedrons with a camera vertex

Definition at line 39 of file delaunay.h.

4.3.3.6 int TrStats::nb_tetra0

nb of facets candidate for ray intersection

Definition at line 38 of file delaunay.h.

4.3.3.7 unsigned long TrStats::nb_tot_intersect

nb of tested facets

Definition at line 34 of file delaunay.h.

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

- delaunay.h
- config.cpp

4.4 Viewer Class Reference

Public Member Functions

- **Viewer** (char ∗name1, char ∗name2, bool in_bbox, float ∗xybox, TPoint &points1, PointColor &pcolors1, std::vector< Face > &faces1, TPoint &points2, PointColor &pcolors2, std::vector< Face > &faces2, CGAL::Bbox_3 bbox)
- void **draw** ()
- void **switch_data** ()
- void **init** ()
- virtual void **keyPressEvent** (QKeyEvent ∗e)
- virtual QString **helpString** () const
- void **postSelection** (const QPoint &point)
- **Viewer** (int nbcams, int ∗cam_index, TPoint &points, PointColor &pcolors, std::vector< Face > &faces, std::map< int, VisiblePatches ∗ > &image_patches, CGAL::Bbox_3 bbox, bool in_bbox, CGAL::Bbox_3 limit_bbox)
- void **draw** ()
- void **init** ()
- virtual void **keyPressEvent** (QKeyEvent ∗e)
- virtual QString **helpString** () const
- void **draw_cam** ()
- void **postSelection** (const QPoint &point)
- void **initlight** ()

Public Attributes

- int **m_point_size**
- bool **m_data1**

4.4.1 Detailed Description

Definition at line 27 of file cmpcgal.h.

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

- cmpcgal.h
- qviewer.h
- cmpcgal.cpp
- qviewer.cpp

Chapter 5

File Documentation

5.1 addcells.cpp File Reference

Add the barycenter of "large" tetrahedrons to the triangulation.

```
#include "delaunay.h"
#include <CGAL/basic.h>
#include <CGAL/Object.h>
#include <CGAL/Exact_predicates_inexact_constructions_kernel.h>
#include <CGAL/Triangulation_3.h>
#include <CGAL/Delaunay_triangulation_3.h>
#include <CGAL/Triangulation_vertex_base_with_info_3.h>
#include <CGAL/Triangulation_cell_base_with_info_3.h>
#include <CGAL/intersections.h>
#include <CGAL/circulator.h>
#include <CGAL/IO/Triangulation_geomview_ostream_3.h>
#include <iostream>
#include <fstream>
#include <iterator>
#include <unistd.h>
#include <time.h>
```

Functions

- float mean_edge (Delaunay &T)

  Compute the average edge length in a delaunay triangulation.

- int add_cells (Delaunay &T, float edge_min, float edge_max, TPoint &points, PointColor &pcolors, std::vector< Point > &normals)

  Add the barycenter of "large" tetrahedrons to the set of points and update the triangulation.

5.1.1 Detailed Description

Add the barycenter of "large" tetrahedrons to the triangulation.

Definition in file addcells.cpp.

5.1.2 Function Documentation

5.1.2.1 int add_cells (Delaunay & *T*, float *edge_min*, float *edge_max*, TPoint & *points*, PointColor & *pcolors*, std::vector< Point > & *normals*)

Add the barycenter of "large" tetrahedrons to the set of points and update the triangulation.

Parameters

$\leftrightarrow$ ***T*** : the delaunay data

edge_min : consider tetrahedrons that have at least one edge longer than **edge_min**

edge_max : but ignore tetrahedrons that have at least one edge longer than **edge_max**

$\leftrightarrow$ ***points*** : the vector of points. New points will be appended.

$\leftrightarrow$ ***pcolors*** : the vector of colors. Colors of new points will be estimated and appended.

$\leftrightarrow$ ***normals*** : the vector of normals. Normals of new points will be estimated and appended.

Definition at line 46 of file addcells.cpp.

5.1.2.2 float mean_edge (Delaunay & *T*)

Compute the average edge length in a delaunay triangulation.

Definition at line 24 of file addcells.cpp.

5.2 config.cpp File Reference

Configuration parameters.

```
#include "delaunay.h"
```

Functions

- float **delta_t** (time_t ∗t1, time_t ∗t2)
- void **prompt** (std::string s)
- bool mygetopt (const char ∗opt, OP_TYPE op_type, int i, int argc, char ∗∗argv, void ∗res, int nbargs)

  Find presence and arguments of a program option.

Variables

- char **file_version** [ ] = "GG05"
- int debug_stop = 0

  used to tag .cgal files

- int **debug_vis** = 0
- bool **gv_on** = false
- int **facet_vertex_order** [ ] = {2,1,3,2,2,3,0,2,0,3,1,0,0,1,2,0}

5.2.1 Detailed Description

Configuration parameters.

Definition in file config.cpp.

5.2.2 Function Documentation

5.2.2.1 bool mygetopt (const char ∗ *opt*, OP_TYPE *op_type*, int *i*, int *argc*, char ∗∗ *argv*, void ∗ *res*, int *nbargs*)

Find presence and arguments of a program option.

Parameters

opt	The option string (for example -i)
op_type	The argument(s) type : OPT_INT,OPT_FLOAT,OPT_STRING,OPT_NOARGS
i	Index of argv entry to compare with opt
argc	Nb of elements in argv
argv	string list of program arguments
res	Pointer to store the value or values associated to option
nbargs	Number of args of option (default 1, ignored for OPT_NOARGS)

Returns

bool Whether option was found or not.

Definition at line 57 of file config.cpp.

5.2.3 Variable Documentation

5.2.3.1 int debug_stop = 0

used to tag .cgal files

Definition at line 19 of file config.cpp.

5.3 delaunay.cpp File Reference

Main program for delaunay triangulation and ray tracing.

```
#include "delaunay.h"
#include <time.h>
```

Functions

- void usage (char ∗prog)
- std::vector< Facet >::iterator check_point (Delaunay &Tr, std::vector< Cell_handle > &marked_-cells, TPoint &points, std::vector< Cell_handle > &cells, std::vector< Facet > &ifacets, std::vector< Facet >::iterator itf0, int icam, Vertex_handle vcam, int ipt, TrStats ∗stats, CGAL::Geomview_stream &gv) throw (const char ∗)
- void **dump_cell** (Cell_handle &c)
- void check_cam (Delaunay &Tr, TPoint &points, int icam, VisiblePatches ∗vpatches, TrStats ∗stats, CGAL::Geomview_stream &gv) throw (const char ∗)
- void dump_data (char ∗file, Delaunay &T, TPoint &cam_points, TPoint &points, PointColor &pcol-ors, std::vector< Point > &normals, CGAL::Bbox_3 &bb, int totpts, int nbpts0, std::vector< int > &cameras, std::vector< int > &bad_cameras, std::map< int, VisiblePatches ∗ > &image_patches, float edge_mean, float ∗tetra_coefs)
- int **main** (int argc, char ∗∗argv)

5.3.1 Detailed Description

Main program for delaunay triangulation and ray tracing.

Definition in file delaunay.cpp.

5.3.2 Function Documentation

5.3.2.1 void check_cam (Delaunay & *Tr*, TPoint & *points*, int *icam*, VisiblePatches ∗ *vpatches*, TrStats ∗ *stats*, CGAL::Geomview_stream & *gv*) throw (const char ∗)

Find cells intersected by the rays joining the camera number icam to all its visible points. The number of intersections is added to field ’info’ of cells

Parameters

points,:	vector of points
icam	: camera number
vpatches	: vector of indices of the points that are visible from the camera
$\rightarrow$ ***stats***	: statistics

Definition at line 144 of file delaunay.cpp.

5.3.2.2 std::vector<Facet>::iterator check_point (Delaunay & *Tr*, std::vector< Cell_handle > & *marked_cells*, TPoint & *points*, std::vector< Cell_handle > & *cells*, std::vector< Facet > & *ifacets*, std::vector< Facet >::iterator *itf0*, int *icam*, Vertex_handle *vcam*, int *ipt*, TrStats ∗ *stats*, CGAL::Geomview_stream & *gv*) throw (const char ∗)

Find cells intersected by the ray joining camera num icam to point num ipt

Parameters

points : vector of points

cells : incident cells at camera point

ifacets : vector of facets of incident cells to be tested.

itf0 : where to start in ifacets

icam : cam index

vcam : vertex of cam

ipt : target point index

→ ***marked_cells*** : intersected cells are added to this vector

→ ***stats*** : statistics

Returns

iterator where to start in the camera incident facet vector (try to optimize initial facets search).

Definition at line 59 of file delaunay.cpp.

5.3.2.3 void dump_data (char ∗ *file*, Delaunay & *T*, TPoint & *cam_points*, TPoint & *points*, PointColor & *pcolors*, std::vector< Point > & *normals*, CGAL::Bbox_3 & *bb*, int *totpts*, int *nbpts0*, std::vector< int > & *cameras*, std::vector< int > & *bad_cameras*, std::map< int, VisiblePatches ∗ > & *image_patches*, float *edge_mean*, float ∗ *tetra_coefs*)

Dump to a binary file all data needed to extract surface facets extraction :

Parameters

file : name of file

T : the delaunay structure

cam_points : vector of cameras coords and index in cam_points.

pcolors : vector of initial points colors

normals : vector of intitial points normals

bb : bounding box of PMVS points

totpts : total number of points (PMVS points + camera points)

nbpts0 : nb of points without cameras (= index of 1st camera)

cameras : vector of valid cameras giving their index in cam_points

bad_cameras : vector of bad cameras giving their index in cam_points

image_patches : map of visible points per camera

edge_mean : average edge length (PMVS2 points only)

tetra_coefs : the coefficients of option -a

The structure of the file is :

- 4 bytes version (CGxx)
- delaunay data
- 6 ∗ float : bounding box (without cams)
- 3 ∗ float : average edge length (in pmvs points), 0 0 or coefs min and max used for points addition in large tetras
- int : nbcams = nb of "good" cameras
- npts ∗ 3 uchars : colors of points
- bool : 'with_normals'
- npts ∗ 3-floats : normals of finite vertices
- nbcams ∗ 2 ∗ int : nbcams camera indexes (in finite vertices list), nbcams original cameras nums
- nbcell ∗ int : finites cells info
- nbcams ∗ (points visible by each camera) :
  - int : cam num
  - int : nb of visible points
  - nbv ∗ int : indexes (in finite vertices list) of visible points

int : nbbadcams

- nbbadcams ∗ 3 float : bad cameras coords

Definition at line 233 of file delaunay.cpp.

5.3.2.4 void usage (char ∗ *prog*)

print the unsage message

Parameters

prog : program name

Definition at line 31 of file delaunay.cpp.

5.4 delaunay.h File Reference

```
#include "triangdefs.h"
#include <time.h>
```

Classes

- class TrStats

  Store statistics.

- class TrParams

  params for facets extraction

Defines

- #define **CLEAN_LGR** 1
- #define **CLEAN_SURF** 2
- #define **CLEAN_TETRA** 4
- #define **CG_PATCHES** 1
- #define **CG_BADCAMS** 2

Functions

- bool mygetopt (const char ∗opt, OP_TYPE op_type, int i, int argc, char ∗∗argv, void ∗res, int nbargs=1)

  Find presence and arguments of a program option.

- void **draw_all** (CGAL::Geomview_stream &gv, Delaunay &Tr)
- void **draw_seg** (CGAL::Geomview_stream &gv, Segment &seg)
- void **prompt** (std::string s)
- void **draw_tetra** (Delaunay &Tr, std::vector< Cell_handle > &cells, CGAL::Geomview_stream &gv)
- void **draw_line_tetra** (Delaunay &Tr, std::vector< Cell_handle > &cells, CGAL::Geomview_-stream &gv)
- void **draw_facets** (Delaunay &Tr, std::vector< Facet > facets, PointColor pcolors, CGAL::Geomview_stream &gv)
- void **draw_facets** (Delaunay &Tr, std::vector< Facet > &facets, CGAL::Geomview_stream &gv)
- void **draw_cells_edges** (Delaunay &Tr, std::vector< Cell_handle > &cells, CGAL::Geomview_-stream &gv)
- void **draw_segs** (CGAL::Geomview_stream &gv, std::vector< Segment > &segs)
- void delaunay_extract (const char ∗outply, Delaunay &T, std::vector< Point > &normals, Point-Color &pcolors, int nbcams, int ∗cams_index, TPoint &bad_cameras, TrParams &params, CGAL::Geomview_stream &gv, char ∗comment=NULL)

  Extract surface facets from the delaunay triangulation.

- CGAL::Bbox_3 read_ply (const char ∗filename, TPoint &points, std::vector< Point > &normals, PointColor &colors) throw (const char ∗)

  read a ply file containing only points, optionnaly with colors and normals

- CGAL::Bbox_3 read_ply (const char ∗filename, TPoint &points, std::vector< Point > &normals, PointColor &colors, std::vector< Face > &faces, char ∗∗coment=NULL) throw (const char ∗)

  read a ply file containing points and facets.

- void **read_patches** (const char ∗filename, int firstpoint, int nbcams, TPoint &points, std::map< int, VisiblePatches ∗ > &image_patches, bool read_points=false) throw (const char ∗)
- void **read_cgal_data** (char ∗file, Delaunay &T, PointColor &pcolors, std::vector< Point > &normals, CGAL::Bbox_3 &bb, int ∗nbcams, int ∗∗cams_index, std::map< int, VisiblePatches ∗ > &image_patches, TPoint &bad_cameras, int data_mode, float ∗edge_mean, float ∗tetra_coefs) throw (const char ∗)
- void **read_cgal_xdata** (char ∗file, int ∗nbcams, int ∗∗cams_index, CGAL::Bbox_3 &bb, std::map< int, VisiblePatches ∗ > &image_patches, TPoint &bad_cameras, int data_mode, float ∗edge_mean, float ∗tetra_coefs) throw (const char ∗)
- std::vector< Facet > ∗ intersect (int ∗err, int ∗nb_tested, Segment &seg, Intersect &in_inter, Intersect &out_inter, std::vector< Facet > ∗facets, Delaunay &Tr, std::vector< Cell_handle > &marked_cells, TrStats ∗stats, CGAL::Geomview_stream &gv) throw (const char ∗)

  Find which facet is intersected and return next facets to check.

- Vector ∗ smooth (Delaunay &T, std::vector< Point > &normals, std::vector< Facet > &facets, float lambda, int nbiter, int nbcams, int ∗cams_index)

  Replace vertices coordinates by a function of their neighbours.

- float **delta_t** (time_t ∗t1, time_t ∗t2)
- float mean_edge (Delaunay &T)

  Compute the average edge length in a delaunay triangulation.

- int add_cells (Delaunay &T, float edge_min, float edge_max, TPoint &points, PointColor &pcolors, std::vector< Point > &normals)

  Add the barycenter of "large" tetrahedrons to the set of points and update the triangulation.

Variables

- bool **gv_on**
- int debug_stop

  used to tag .cgal files

- int **debug_vis**
- int **facet_vertex_order** [ ]
- char **file_version** [ ]

5.4.1 Detailed Description

Definition in file delaunay.h.

5.4.2 Function Documentation

5.4.2.1 int add_cells (Delaunay & *T*, float *edge_min*, float *edge_max*, TPoint & *points*, PointColor & *pcolors*, std::vector< Point > & *normals*)

Add the barycenter of "large" tetrahedrons to the set of points and update the triangulation.

Parameters

↔ ***T***	: the delaunay data
edge_min	: consider tetrahedrons that have at least one edge longer than **edge_min**
edge_max	: but ignore tetrahedrons that have at least one edge longer than **edge_max**
↔ ***points***	: the vector of points. New points will be appended.
↔ ***pcolors***	: the vector of colors. Colors of new points will be estimated and appended.
↔ ***normals***	: the vector of normals. Normals of new points will be estimated and appended.

Definition at line 46 of file addcells.cpp.

5.4.2.2 void delaunay_extract (const char ∗ *outply*, Delaunay & *T*, std::vector< Point > & *normals*, PointColor & *pcolors*, int *nbcams*, int ∗ *cams_index*, TPoint & *bad_cameras*, TrParams & *params*, CGAL::Geomview_stream & *gv*, char ∗ *comment*)

Extract surface facets from the delaunay triangulation.

Parameters

outply	: name of output ply file.
T	: delaunay data
normals	: vector of PMVS normals
pcolors	: vector of PMVS point colors
nbcams	: number of cameras
cams_index	: indices of camera coords in points vector
bad_cameras	: vector of the vertices of bad cameras.
params.extract_type	: call prepare_extract_mxf or prepare_extract_std
params.smooth_coef	: if >0., run the smooth program on vertices.
comment	: comment to put in the header (the arguments of the command line)

Definition at line 579 of file extract.cpp.

5.4.2.3 std::vector<Facet>∗ intersect (int ∗ *err*, int ∗ *nb_tested*, Segment & *seg*, Intersect & *in_inter*, Intersect & *out_inter*, std::vector< Facet > ∗ *facets*, Delaunay & *Tr*, std::vector< Cell_handle > & *marked_cells*, TrStats ∗ *stats*, CGAL::Geomview_stream & *gv*) throw (const char ∗)

Find which facet is intersected and return next facets to check.

Parameters

$\rightarrow$ ***err*** : -1 if no intersection and not at end of segment, 0 otherwise
$\rightarrow$ ***nb_tested*** : number of facets effectively tested (for statistics).
seg : segment from a camera to a visible poins.
in_inter : kind of previous intersection (entry in cell).
$\rightarrow$ ***out_inter*** : kind of intersection.
facets : vector of facets to check.
T : delaunay data.
$\leftrightarrow$ ***marked_cells*** : intersected cell will be added to this vector.
$\leftrightarrow$ ***stats*** : statistics counters.

Returns

: pointer to vector of facets to check at next step.

Definition at line 181 of file intersect.cpp.

5.4.2.4 float mean_edge (Delaunay & *T*)

Compute the average edge length in a delaunay triangulation.

Definition at line 24 of file addcells.cpp.

5.4.2.5 bool mygetopt (const char ∗ *opt*, OP_TYPE *op_type*, int *i*, int *argc*, char ∗∗ *argv*, void ∗ *res*, int *nbargs*)

Find presence and arguments of a program option.

Parameters

opt The option string (for example -i)
op_type The argument(s) type : OPT_INT,OPT_FLOAT,OPT_STRING,OPT_NOARGS
i Index of argv entry to compare with opt
argc Nb of elements in argv
argv string list of program arguments
res Pointer to store the value or values associated to option
nbargs Number of args of option (default 1, ignored for OPT_NOARGS)

Returns

bool Whether option was found or not.

Definition at line 57 of file config.cpp.

5.4.2.6 CGAL::Bbox_3 read_ply (const char ∗*filename*, TPoint &*points*, std::vector< Point > & *normals*, PointColor & *colors*, std::vector< Face > &*faces*, char ∗∗ *coment* = `NULL`) throw (const char ∗)

read a ply file containing points and facets.

Definition at line 177 of file delaunay_io.cpp.

5.4.2.7 CGAL::Bbox_3 read_ply (const char $*$ *filename*, TPoint & *points*, std::vector$<$ Point $>$ & *normals*, PointColor & *colors*) throw (const char $*$)

read a ply file containing only points, optionnaly with colors and normals

Definition at line 170 of file delaunay_io.cpp.

5.4.2.8 Vector$*$ smooth (Delaunay & *T*, std::vector$<$ Point $>$ & *normals*, std::vector$<$ Facet $>$ & *facets*, float *lambda*, int *nbiter*, int *nbcams*, int $*$ *cams_index*)

Replace vertices coordinates by a function of their neighbours.

$p = \lambda p + (1 - \lambda) \sum_{q \in N_p} w_q q$

N_p is the set of neighbours of p, λ is a positive scalar < 1, weights w_q sums to 1 and are a function of the unit normals $n_p n_q$.

For example $w_q = \mu[n_p.n_q]_+^k$, where $x_+ = x$ if $x \geq 0$ and 0 otherwise, and μ is choosen so that $\sum_a \in N_p w_q = 1$.

Parameters

- ***T*** : delaunay data
- ***normals*** : vector of vertices PMVS normals.
- ***facets*** : vector of facets on the surface. The neighbourhood is restricted to cells containing te facets.
- ***lambda*** : the λ coefficient.
- ***nbiter*** : number of iterations. If negative,
- ***nbcams*** : number of cameras
- ***cams_index*** : indexes of cameras coords in PMVS points table.

Definition at line 131 of file smooth.cpp.

5.4.3 Variable Documentation

5.4.3.1 int debug_stop

used to tag .cgal files

Definition at line 19 of file config.cpp.

5.5 delaunay_io.cpp File Reference

functions to read data from ply or cgal files

`#include "delaunay.h"`

`#include <stdlib.h>`

Functions

- CGAL::Bbox_3 ply_binary_data (std::ifstream &ifstr, TPoint &points, std::vector< Point > &normals, PointColor &colors, std::vector< Face > &faces, int nbpts, int nbfaces, int nbflt, int nbint) throw (const char ∗)
- CGAL::Bbox_3 **ply_ascii_data** (std::ifstream &ifstr, TPoint &points, std::vector< Point > &normals, PointColor &colors, std::vector< Face > &faces, int nbpts, int nbfaces, int nbflt, int nbint) throw (const char ∗)
- CGAL::Bbox_3 **read_all_ply** (const char ∗filename, TPoint &points, std::vector< Point > &normals, PointColor &colors, std::vector< Face > &faces, char ∗∗comment, bool with_faces) throw (const char ∗)
- CGAL::Bbox_3 read_ply (const char ∗filename, TPoint &points, std::vector< Point > &normals, PointColor &colors) throw (const char ∗)

  read a ply file containing only points, optionnaly with colors and normals
- CGAL::Bbox_3 read_ply (const char ∗filename, TPoint &points, std::vector< Point > &normals, PointColor &colors, std::vector< Face > &faces, char ∗∗comment) throw (const char ∗)

  read a ply file containing points and facets.
- void **read_patches** (const char ∗filename, int firstpoint, int nbcams, TPoint &points, std::map< int, VisiblePatches ∗ > &image_patches, bool read_points) throw (const char ∗)
- void **get_patches** (std::ifstream &iFileT, int nbcams, std::map< int, VisiblePatches ∗ > &image_patches, TPoint &bad_cameras, int data_mode) throw (const char ∗)
- void **read_cgal_data** (char ∗file, Delaunay &T, PointColor &pcolors, std::vector< Point > &normals, CGAL::Bbox_3 &bb, int ∗nbcams, int ∗∗cams_index, std::map< int, VisiblePatches ∗ > &image_patches, TPoint &bad_cameras, int data_mode, float ∗edge_mean, float ∗tetra_coefs) throw (const char ∗)
- void **read_cgal_xdata** (char ∗file, int ∗nbcams, int ∗∗cams_index, CGAL::Bbox_3 &bb, std::map< int, VisiblePatches ∗ > &image_patches, TPoint &bad_cameras, int data_mode, float ∗edge_mean, float ∗tetra_coefs) throw (const char ∗)

5.5.1 Detailed Description

functions to read data from ply or cgal files

Definition in file delaunay_io.cpp.

5.5.2 Function Documentation

5.5.2.1 CGAL::Bbox_3 ply_binary_data (std::ifstream & *ifstr*, TPoint & *points*, std::vector< Point > & *normals*, PointColor & *colors*, std::vector< Face > & *faces*, int *nbpts*, int *nbfaces*, int *nbflt*, int *nbint*) throw (const char ∗)

Read the data part of a binary ply file

Definition at line 23 of file delaunay_io.cpp.

5.5.2.2 CGAL::Bbox_3 read_ply (const char ∗ *filename*, TPoint & *points*, std::vector< Point > & *normals*, PointColor & *colors*, std::vector< Face > & *faces*, char ∗∗ *comment*) throw (const char ∗)

read a ply file containing points and facets.

Definition at line 177 of file delaunay_io.cpp.

5.5.2.3 CGAL::Bbox_3 read_ply (const char ∗ *filename*, TPoint & *points*, std::vector< Point > & *normals*, PointColor & *colors*) throw (const char ∗)

read a ply file containing only points, optionnaly with colors and normals

Definition at line 170 of file delaunay_io.cpp.

5.6 extract.cpp File Reference

Functions for surface facets extraction.

```
#include "delaunay.h"
#include <map>
#include <time.h>
#include "graph.h"
```

Typedefs

- typedef Graph< int, int, int > **GraphType**

Functions

- static int **nb_rm_normals** (0)
- void prepare_extract_mxf (Delaunay &T, TrParams &params, CGAL::Geomview_stream &gv)

  Find tetrahedrons to keep and remove with a cost function minimization.

- void prepare_extract_std (Delaunay &T, TrParams &params, CGAL::Geomview_stream &gv)

  Find which tetrahedrons to remove or keep, based on nb of ray intersections.

- int clean1 (Delaunay &T)

  try to remove rough elements, that is tetrahedrons with only one face adjacent to a non infinite tetrahedron.

- bool clean2 (Delaunay &T, Facet &f, int ∗nb_rmlg, int ∗nb_rmsurf, TrParams &params)

  Check if a facet is too wide and/or has a too long edge.

- void prepare_facets (Delaunay &T, std::vector< Facet > &facets, std::vector< Point > &normals, int nbcams, int ∗cams_index, TrParams &params, CGAL::Geomview_stream &gv)

  build the list of surface facets

- void save_ply (const char ∗file, Delaunay &T, std::vector< Facet > &facets, Vector ∗points, std::vector< Point > &normals, PointColor &pcolors, int nbcams, int ∗cams_index, TPoint &bad_-cameras, TrParams &params, char ∗comment) throw (const char ∗)

  Save results (points and facets) in an ascii ply file. For test only (data is partial).

- void save_ply_binary (const char ∗file, Delaunay &T, std::vector< Facet > &facets, Vector ∗points, std::vector< Point > &normals, PointColor &pcolors, int nbcams, int ∗cams_index, TPoint &bad_-cameras, TrParams &params, char ∗comment) throw (const char ∗)

  Save vertices and extracted faces in a binary ply file.

- void delaunay_extract (const char ∗outply, Delaunay &T, std::vector< Point > &normals, Point-Color &pcolors, int nbcams, int ∗cams_index, TPoint &bad_cameras, TrParams &params, CGAL::Geomview_stream &gv, char ∗comment)

  Extract surface facets from the delaunay triangulation.

5.6.1 Detailed Description

Functions for surface facets extraction.

Definition in file extract.cpp.

5.6.2 Function Documentation

5.6.2.1 int clean1 (Delaunay & *T*)

try to remove rough elements, that is tetrahedrons with only one face adjacent to a non infinite tetrahedron.

Definition at line 156 of file extract.cpp.

5.6.2.2 bool clean2 (Delaunay & *T*, Facet & *f*, int ∗ *nb_rmlg*, int ∗ *nb_rmsurf*, TrParams & *params*)

Check if a facet is too wide and/or has a too long edge.

Parameters

T	: the delaunay data.
f	: the facet to test.
↔ ***nb_rmlg***	: nb of faces removed by edge length check; incremented .
↔ ***nb_rmsurf***	: nb of faces removed by surface check; incremented .
params.do_clean	: define what checking to do
params.lgr_threshold	: threshold for length check.
params.surf_threshold	: threshold for surface check.

Definition at line 185 of file extract.cpp.

5.6.2.3 void delaunay_extract (const char ∗ *outply*, Delaunay & *T*, std::vector< Point > & *normals*, PointColor & *pcolors*, int *nbcams*, int ∗ *cams_index*, TPoint & *bad_cameras*, TrParams & *params*, CGAL::Geomview_stream & *gv*, char ∗ *comment*)

Extract surface facets from the delaunay triangulation.

Parameters

outply	: name of output ply file.
T	: delaunay data
normals	: vector of PMVS normals
pcolors	: vector of PMVS point colors
nbcams	: number of cameras
cams_index	: indices of camera coords in points vector
bad_cameras	: vector of the vertices of bad cameras.
params.extract_type	: call prepare_extract_mxf or prepare_extract_std
params.smooth_coef	: if >0., run the smooth program on vertices.

comment : comment to put in the header (the arguments of the command line)

Definition at line 579 of file extract.cpp.

5.6.2.4 void prepare_extract_mxf (Delaunay & *T*, TrParams & *params*, CGAL::Geomview_stream & *gv*)

Find tetrahedrons to keep and remove with a cost function minimization.

Parameters

$\leftrightarrow$ ***T*** : delaunay triangulation. Removed and valid cells are flagged in field info

params.nb_intersect : number of intersection threshold

Definition at line 31 of file extract.cpp.

5.6.2.5 void prepare_extract_std (Delaunay & *T*, TrParams & *params*, CGAL::Geomview_stream & *gv*)

Find which tetrahedrons to remove or keep, based on nb of ray intersections.

Parameters

$\leftrightarrow$ ***T*** : delaunay triangulation. Removed and valid cells are flagged in field info

params.nb_intersect : number of intersection threshold

params.average_volume : increase the threshold for cells having a larger volume.

Definition at line 106 of file extract.cpp.

5.6.2.6 void prepare_facets (Delaunay & *T*, std::vector< Facet > & *facets*, std::vector< Point > & *normals*, int *nbcams*, int $*$ *cams_index*, TrParams & *params*, CGAL::Geomview_stream & *gv*)

build the list of surface facets

Parameters

normals,: the vector of PMVS normals

nbcams : number of cameras

cams_index : indexes of cameras coords in PMVS points table

params.extract_mode : 0/1 = use removed/valid cells to find facets +2 = retrieve all facets of corresponding tetrahedrons (for test purpose)

params.do_clean : type of desired cleaning

Definition at line 262 of file extract.cpp.

5.6.2.7 void save_ply (const char ∗ *file*, Delaunay & *T*, std::vector< Facet > & *facets*, Vector ∗ *points*, std::vector< Point > & *normals*, PointColor & *pcolors*, int *nbcams*, int ∗ *cams_index*, TPoint & *bad_cameras*, TrParams & *params*, char ∗ *comment*) throw (const char ∗)

Save results (points and facets) in an ascii ply file. For test only (data is partial).

Definition at line 374 of file extract.cpp.

5.6.2.8 void save_ply_binary (const char ∗ *file*, Delaunay & *T*, std::vector< Facet > & *facets*, Vector ∗ *points*, std::vector< Point > & *normals*, PointColor & *pcolors*, int *nbcams*, int ∗ *cams_index*, TPoint & *bad_cameras*, TrParams & *params*, char ∗ *comment*) throw (const char ∗)

Save vertices and extracted faces in a binary ply file.

Parameters

file	: destination file
T	: delaunay data
points	: optional vector of points. If non nul use it for points coords instead of delaunay vertices.
normals	: vector of PMVS normals
pcolors	: vector of PMVS point colors
nbcams	: number of cameras
cams_index	: indices of camera coords in points vector
bad_cameras	: vector of the vertices of bad cameras.
params.extract_mode	: use to print facet vertice in the correct order
comment	: comment to put in the header (the arguments of the command line)

Definition at line 464 of file extract.cpp.

5.7 intersect.cpp File Reference

Compute imtersection of tetrahedrons with rays issued from cameras.

```
#include "delaunay.h"
#include <map>
```

Functions

- int is_cell_vertex (Cell_handle c, Point &pt)

  *Return the vertex index of a cell **c** corresponding to point **pt**.*

- int check_edges (Point &a, Point &b, Point &c, Point &p, Point &q, int ∗coplanar) throw (const char ∗)

  Check if a segement is coplanar with edges of a triangle.

- int my_intersect (Triangle &t, Segment &s, int ∗coplanar)

  *Check intersection between a facet **f** and a segment **seg**. Derived from **CGAL::do_intersect** in include/CGAL/Triangle_3_Segment_3_do_intersect.h.*

- std::vector< Facet > ∗ intersect (int ∗err, int ∗nb_tested, Segment &seg, Intersect &in_inter, Intersect &out_inter, std::vector< Facet > ∗facets, Delaunay &Tr, std::vector< Cell_handle > &marked_cells, TrStats ∗stats, CGAL::Geomview_stream &gv) throw (const char ∗)

  Find which facet is intersected and return next facets to check.

5.7.1 Detailed Description

Compute imtersection of tetrahedrons with rays issued from cameras.

Definition in file intersect.cpp.

5.7.2 Function Documentation

5.7.2.1 int check_edges (Point & *a*, Point & *b*, Point & *c*, Point & *p*, Point & *q*, int ∗ *coplanar*) throw (const char ∗)

Check if a segement is coplanar with edges of a triangle.

Parameters

a,b,c : coords of the triangle vertices

p,q : segment with end points **p** and **q**

↔ ***coplanar*** : pointer to the 3 int array of indices of the edges coplanr with **pq**.

Returns

Nb of entries in **coplanar**.

Definition at line 39 of file intersect.cpp.

5.7.2.2 std::vector<Facet>∗ intersect (int ∗ *err*, int ∗ *nb_tested*, Segment & *seg*, Intersect & *in_inter*, Intersect & *out_inter*, std::vector< Facet > ∗ *facets*, Delaunay & *Tr*, std::vector< Cell_handle > & *marked_cells*, TrStats ∗ *stats*, CGAL::Geomview_stream & *gv*) throw (const char ∗)

Find which facet is intersected and return next facets to check.

Parameters

$\rightarrow$ ***err*** : -1 if no intersection and not at end of segment, 0 otherwise

$\rightarrow$ ***nb_tested*** : number of facets effectively tested (for statistics).

seg : segment from a camera to a visible poins.

in_inter : kind of previous intersection (entry in cell).

$\rightarrow$ ***out_inter*** : kind of intersection.

facets : vector of facets to check.

T : delaunay data.

$\leftrightarrow$ ***marked_cells*** : intersected cell will be added to this vector.

$\leftrightarrow$ ***stats*** : statistics counters.

Returns

: pointer to vector of facets to check at next step.

Definition at line 181 of file intersect.cpp.

5.7.2.3 int is_cell_vertex (Cell_handle *c*, Point & *pt*)

Return the vertex index of a cell **c** corresponding to point **pt**.

Returns

the vertex index (0-3), -1 if the point is not a cell vertex.

Definition at line 26 of file intersect.cpp.

5.7.2.4 int my_intersect (Triangle & *t*, Segment & *s*, int ∗ *coplanar*)

Check intersection between a facet **f** and a segment **seg**. Derived from **CGAL::do_intersect** in include/CGAL/Triangle_3_Segment_3_do_intersect.h.

Parameters

$\leftrightarrow$ ***coplanar*** : pointer to the 3 int array of indices of the edges coplanr with **pq**.

Returns

-1 if there is no intersection, the number of edges coplanar with the segment otherwise (0 = intersection inside the triangle).

Definition at line 68 of file intersect.cpp.

5.8 smooth.cpp File Reference

"Smoothing" of verices coordinates.

```
#include "delaunay.h"
#include <map>
#include <time.h>
```

Functions

- void smooth1 (Delaunay &T, std::vector< Point > &normals, std::map< Facet, bool > keep_facets, Vector ∗in_pts, Vector ∗out_pts, float lambda, std::map< int, bool > &cams_map, bool same_-weight)

  Does one iteration of smoothing.

- Vector ∗ smooth (Delaunay &T, std::vector< Point > &normals, std::vector< Facet > &facets, float lambda, int nbiter, int nbcams, int ∗cams_index)

  Replace vertices coordinates by a function of their neighbours.

5.8.1 Detailed Description

"Smoothing" of verices coordinates.

Definition in file smooth.cpp.

5.8.2 Function Documentation

5.8.2.1 Vector∗ smooth (Delaunay & *T*, std::vector< Point > & *normals*, std::vector< Facet > & *facets*, float *lambda*, int *nbiter*, int *nbcams*, int ∗ *cams_index*)

Replace vertices coordinates by a function of their neighbours.

$p = \lambda p + (1 - \lambda) \sum_{q \in N_p} w_q q$

N_p is the set of neighbours of p, λ is a positive scalar < 1, weights w_q sums to 1 and are a function of the unit normals $n_p n_q$.

For example $w_q = \mu[n_p.n_q]_+^k$, where $x_+ = x$ if $x \geq 0$ and 0 otherwise, and μ is choosen so that $\sum_a \in N_p w_q = 1$.

Parameters

T : delaunay data

normals : vector of vertices PMVS normals.

facets : vector of facets on the surface. The neighbourhood is restricted to cells containing te facets.

lambda : the λ coefficient.

nbiter : number of iterations. If negative,

nbcams : number of cameras

cams_index : indexes of cameras coords in PMVS points table.

Definition at line 131 of file smooth.cpp.

5.8.2.2 void smooth1 (Delaunay & *T*, std::vector< Point > & *normals*, std::map< Facet, bool > *keep_facets*, Vector $*$ *in_pts*, Vector $*$ *out_pts*, float *lambda*, std::map< int, bool > & *cams_map*, bool *same_weight*)

Does one iteration of smoothing.

Definition at line 25 of file smooth.cpp.

5.9 triangdefs.h File Reference

Definitions of types.

```
#include <CGAL/basic.h>
#include <CGAL/Object.h>
#include <CGAL/Exact_predicates_inexact_constructions_kernel.h>
#include <CGAL/Triangulation_3.h>
#include <CGAL/Delaunay_triangulation_3.h>
#include <CGAL/Triangulation_vertex_base_with_info_3.h>
#include <CGAL/Triangulation_cell_base_with_info_3.h>
#include <CGAL/intersections.h>
#include <CGAL/circulator.h>
#include <CGAL/IO/Triangulation_geomview_ostream_3.h>
#include <iostream>
#include <fstream>
#include <iterator>
#include <unistd.h>
```

Classes

- struct Intersect

  information about an intersectection between a ray and a facet

Defines

- #define **DEF_LGR_COEF** 0.14
- #define **DEF_SURF_COEF** 0.01
- #define **MAX_INTERSECT** 2 ∗ 100000

Typedefs

- typedef CGAL::Exact_predicates_inexact_constructions_kernel **K**
- typedef CGAL::Triangulation_vertex_base_with_info_3< int, K > **Vb**
- typedef K::Vector_3 **Vector**
- typedef CGAL::Triangulation_cell_base_with_info_3< int, K > **Cb**
- typedef CGAL::Triangulation_data_structure_3< Vb, Cb > **Tds**
- typedef CGAL::Delaunay_triangulation_3< K, Tds > **Delaunay**
- typedef Delaunay::Point **Point**
- typedef Delaunay::Edge **Edge**
- typedef Delaunay::Cell_handle **Cell_handle**
- typedef Delaunay::Vertex_handle **Vertex_handle**
- typedef Delaunay::Locate_type **Locate_type**

- typedef Delaunay::Segment **Segment**
- typedef Delaunay::Triangle **Triangle**
- typedef Delaunay::Tetrahedron **Tetrahedron**
- typedef std::pair< Cell_handle, int > **Facet**
- typedef CGAL::Triple< int, int, int > **Face**
- typedef std::vector< Facet >::iterator **I**
- typedef CGAL::Circulator_from_iterator< I > **Circulator**
- typedef std::vector< int > **VisiblePatches**
- typedef std::vector< CGAL::Color > **PointColor**
- typedef std::vector< std::pair< Point, int > > **TPoint**

Enumerations

- enum **OP_TYPE** { **OPT_INT**, **OPT_FLOAT**, **OPT_STRING**, **OPT_NOARGS** }
- enum **ExtractType** { **XTR_DEFAULT**, **XTR_STD**, **XTR_STD_VOL**, **XTR_MAXFLOW** }
- enum **InterType** { **INT_FACET**, **INT_EDGE**, **INT_VERTEX** }

5.9.1 Detailed Description

Definitions of types.

Definition in file triangdefs.h.

Index