

Inshaping 0.1

Bug List

File anyarg.h

no bug found yet

Inshaping 0.1

Modules

Here is a list of all modules:

| | |
|----------------------|--|
| Utils | |
| Benchmark | |
| IO | |
| DataStructure | |
| Rigsteration | |

Generated by [doxygen](#) 1.8.14

Inshaping 0.1

[Namespaces](#) | [Classes](#) | [Functions](#)

Utils

Namespaces

Inshape::Convert

Inshape::ConvexHull

Classes

class **Inshape::Cutter**

struct **Inshape::tri**

class **Inshape::BoundingBox**

class **Inshape::Inshaper**

class **Inshape::NormalEstimationPlus**

class **Inshape::Plane**

class **Inshape::Poisson**

Functions

bool **Inshape::compareNorm** (tri i, tri j)

Detailed Description

Function Documentation

◆ compareNorm()

```
bool Inshape::compareNorm ( tri i,  
                             tri j  
                             )
```

Inshaping 0.1

Inshape

Convert

Functions

Inshape::Convert Namespace Reference

Utils

Functions

void **meshToCloud** (**MyMesh**
&head_mesh, pcl::PointCloud<
pcl::PointXYZ > &head_cloud)

pcl::PointCloud< pcl::PointXYZ >::Ptr **hyperToPointCloud**
(**CCLib::HyperCloud::Ptr**
&hyper)

CCLib::HyperCloud::Ptr **cloudToHyper** (pcl::PointCloud<
pcl::PointXYZ > &cloud)

MyMeshPtr **PolygonToMyMesh**
(pcl::PolygonMesh &mesh)

Function Documentation

◆ cloudToHyper()

CCLib::HyperCloud::Ptr

Inshape::Convert::cloudToHyper (pcl::PointCloud< pcl::PointXYZ > & clc

◆ hyperToPointCloud()

```
pcl::PointCloud<pcl::PointXYZ>::Ptr  
Inshape::Convert::hyperToPointCloud ( CCLib::HyperCloud::Ptr & hype
```

◆ meshToCloud()

```
void  
Inshape::Convert::meshToCloud ( MyMesh & he  
                                pcl::PointCloud< pcl::PointXYZ > & he  
                                )
```

◆ PolygonToMyMesh()

MyMeshPtr

Inshape::Convert::PolygonToMyMesh

(pcl::PolygonMesh & mesh)

Generated by [doxygen](https://doxygen.org) 1.8.14

Inshaping 0.1

Inshape

ConvexHull

Functions

Inshape::ConvexHull Namespace Reference

Utils

Functions

pcl::PolygonMesh::Ptr **getConvexHull** (pcl::PointCloud< pcl::PointXYZ >::Ptr &cloud)

pcl::PointCloud< pcl::PointXYZ >::Ptr **filterWithHull** (pcl::PointCloud< pcl::PointXYZ >::Ptr &bodyshad, pcl::PointCloud< pcl::PointXYZ >::Ptr &head, double threshold)

Function Documentation

◆ filterWithHull()

```
pcl::PointCloud<pcl::PointXYZ>::Ptr  
Inshape::ConvexHull::filterWithHull ( pcl::PointCloud< pcl::PointXYZ >::F  
    pcl::PointCloud< pcl::PointXYZ >::F  
    double  
    )
```

◆ getConvexHull()

```
pcl::PolygonMesh::Ptr  
Inshape::ConvexHull::getConvexHull ( pcl::PointCloud< pcl::PointXYZ >::
```

Inshaping 0.1

[Inshape](#) > [Cutter](#)

Inshape::Cutter Class Reference

[Utils](#)

[Public Member Functions](#) |
[Static Public Member Functions](#) |
[List of all members](#)

```
#include <AfterProcess.h>
```

Public Member Functions

Cutter (pcl::PointCloud< pcl::PointXYZ >::Ptr &_body, **MyMeshPtr** _head)

void **setBody** (pcl::PointCloud< pcl::PointXYZ >::Ptr &_body)

void **setHead** (**MyMeshPtr** _head)

pcl::PointCloud< pcl::PointXYZ >::Ptr **getCutResult** (double threshold, double percent)

Static Public Member Functions

```
static pcl::PointCloud< pcl::PointXYZ >::Ptr checkBodyWithHead  
(pcl::PointCloud<  
pcl::PointXYZ >::Ptr  
&bodyshead,  
pcl::PointCloud<  
pcl::PointXYZ >::Ptr &head,  
double threshold)
```

Constructor & Destructor Documentation

Member Function Documentation

◆ checkBodyWithHead()

```
static
pcl::PointCloud<pcl::PointXYZ>::Ptr
Inshape::Cutter::checkBodyWithHead ( pcl::PointCloud< pcl::PointXYZ >
                                     pcl::PointCloud< pcl::PointXYZ >
                                     double
                                     )
```

◆ getCutResult()

```
pcl::PointCloud<pcl::PointXYZ>::Ptr  
Inshape::Cutter::getCutResult      ( double threshold,  
                                     double percent  
                                     )
```

◆ setBody()

void

```
Inshape::Cutter::setBody ( pcl::PointCloud< pcl::PointXYZ >::Ptr & _body
```

◆ setHead()

```
void Inshape::Cutter::setHead ( MyMeshPtr _head )
```

inline

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

- include/**AfterProcess.h**

Generated by doxygen 1.8.14

Inshaping 0.1

Inshape > tri >

[Public Attributes](#) | [List of all members](#)

Inshape::tri Struct Reference

[Utils](#)

```
#include <BoundingBox.h>
```

Public Attributes

Eigen::Vector3d **edge**

int **from**

int **to**

Member Data Documentation

◆ edge

Eigen::Vector3d Inshape::tri::edge

◆ from

```
int Inshape::tri::from
```

◆ to

```
int Inshape::tri::to
```

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

- include/**BoundingBox.h**

Generated by **doxygen** 1.8.14

Inshaping 0.1

[Inshape](#)

[BoundingBox](#)

Inshape::BoundingBox Class Reference

[Utils](#)

[Public Member Functions](#) |

[Static Public Member Functions](#) |

[List of all members](#)

```
#include <BoundingBox.h>
```

Public Member Functions

BoundingBox (std::vector< Eigen::Vector3d > &_vertices)

BoundingBox (pcl::PointCloud< pcl::PointXYZ >::Ptr &cloud)

Eigen::Vector3d **getLength** ()

Eigen::Vector3d **getCloserPoint** ()

Eigen::Vector3d **getClosestPoint** (Eigen::Vector3d p)

Eigen::Vector3d **getFarerPoint** ()

void **computeNormals** (Eigen::Vector3d head_baryCenter)

void **saveOBB** (std::string filename)

Static Public Member Functions

```
static std::vector< Eigen::Vector3d > getOBB (pcl::PointCloud<  
pcl::PointXYZ >::Ptr &cloud)
```

Constructor & Destructor Documentation

◆ BoundingBox() [1/2]

```
Inshape::BoundingBox::BoundingBox ( std::vector< Eigen::Vector3d > &
```

◆ BoundingBox() [2/2]

```
Inshape::BoundingBox::BoundingBox ( pcl::PointCloud< pcl::PointXYZ >:
```

Member Function Documentation

◆ computeNormals()

void

Inshape::BoundingBox::computeNormals (Eigen::Vector3d head_baryC

◆ getCloserPoint()

```
Eigen::Vector3d Inshape::BoundingBox::getCloserPoint ( )
```

◆ getClosestPoint()

Eigen::Vector3d

Inshape::BoundingBox::getClosestPoint

(Eigen::Vector3d p)

◆ getFarerPoint()

```
Eigen::Vector3d Inshape::BoundingBox::getFarerPoint ( )
```

◆ getLength()

```
Eigen::Vector3d Inshape::BoundingBox::getLength ( )
```

◆ getOBB()

```
static  
std::vector<Eigen::Vector3d>  
Inshape::BoundingBox::getOBB ( pcl::PointCloud< pcl::PointXYZ >::Ptr &
```

◆ saveOBB()

```
void Inshape::BoundingBox::saveOBB ( std::string filename )
```

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

- include/**BoundingBox.h**

Generated by  1.8.14

Inshaping 0.1

[Inshape](#) > [Inshaper](#) >

[Public Member Functions](#) | [List of all members](#)

Inshape::Inshaper Class Reference

[Utils](#)

```
#include <Inshaper.h>
```

Public Member Functions

Inshaper ()

void **setMatch** (pcl::PointCloud< pcl::PointXYZ >::Ptr
&_subBody, pcl::PointCloud< pcl::PointXYZ >::Ptr
&_subHead)

void **unSetMatch** ()

void **setInputBody** (pcl::PointCloud< pcl::PointXYZ >::Ptr
_body)

void **setInputTemplate** (**MyMeshPtr** &_head, pcl::PointCloud<
pcl::PointXYZ >::Ptr _face)

void **setParam** (**Param** &_param)

void **setPreAligned** (bool _preAligned)

MyMeshPtr **getOutputMesh** ()

Constructor & Destructor Documentation

◆ Inshaper()

Inshape::Inshaper::Inshaper ()

inline

Member Function Documentation

◆ getOutputMesh()

```
MyMeshPtr Inshape::Inshaper::getOutputMesh ( )
```

◆ setInputBody()

void

```
Inshaper::Inshaper::setInputBody ( pcl::PointCloud< pcl::PointXYZ >::Ptr
```

◆ setInputTemplate()

```
void  
Inshape::Inshaper::setInputTemplate ( MyMeshPtr &  
                                       pcl::PointCloud< pcl::PointXYZ >::  
                                       )
```

◆ setMatch()

```
void  
Inshape::Inshaper::setMatch ( pcl::PointCloud< pcl::PointXYZ >::Ptr & _s  
                             pcl::PointCloud< pcl::PointXYZ >::Ptr & _s  
                             )
```

◆ setParam()

```
void Inshape::Inshaper::setParam ( Param & _param )
```

inline

◆ setPreAligned()

```
void Inshape::Inshaper::setPreAligned ( bool _preAligned )
```

inline

◆ unSetMatch()

```
void Inshape::Inshaper::unSetMatch ( )
```

inline

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

- [include/Inshaper.h](#)

Generated by [doxygen](#) 1.8.14

Inshaping 0.1

Inshape > NormalEstimationPlus >

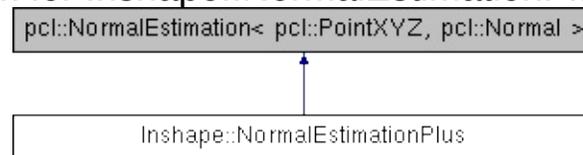
[Classes](#) | [Public Member Functions](#) |
[Static Public Member Functions](#) |
[List of all members](#)

Inshape::NormalEstimationPlus Class Reference

[Utils](#)

```
#include <Normal.h>
```

Inheritance diagram for Inshape::NormalEstimationPlus:



Public Member Functions

NormalEstimationPlus ()

void **setCoherence** (int
_coherence)

int **getCoherence** ()

pcl::PointCloud< pcl::PointNormal >::Ptr **getCoherencyPointNormals**
()

Static Public Member Functions

```
static void AddNeighboursToHeap (pcl::PointCloud< pcl::PointNormal  
>::Ptr cloud_with_normals, int index, int K,  
pcl::search::KdTree< pcl::PointNormal >::Ptr &tree,  
std::vector< WArc > &heap, std::shared_ptr< bool > &visited)
```

```
static void coherencyPass (pcl::PointCloud< pcl::PointNormal >::Ptr  
cloud_with_normals, int coherence=8)
```

Constructor & Destructor Documentation

◆ NormalEstimationPlus()

Inshape::NormalEstimationPlus::NormalEstimationPlus ()

inline

Member Function Documentation

◆ AddNeighboursToHeap()

```
static void  
Inshape::NormalEstimationPlus::AddNeighboursToHeap ( pcl::PointCloud  
                                                    int  
                                                    int  
                                                    pcl::search::Kd  
                                                    std::vector< W  
                                                    std::shared_pt  
                                                    )
```

◆ coherencyPass()

```
static void  
Inshape::NormalEstimationPlus::coherencyPass ( pcl::PointCloud< pcl::F  
int  
)
```

◆ getCoherence()

```
int Inshape::NormalEstimationPlus::getCoherence ( )
```

inline

◆ getCoherencyPointNormals()

```
pcl::PointCloud<pcl::PointNormal>::Ptr  
Inshape::NormalEstimationPlus::getCoherencyPointNormals ( )
```

◆ setCoherence()

```
void  
Inshape::NormalEstimationPlus::setCoherence ( int _coherence ) inline
```

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

- include/**Normal.h**

Generated by doxygen 1.8.14

Inshaping 0.1

[Inshape](#) > [Plane](#)

Inshape::Plane Class Reference

[Utils](#)

[Public Types](#) | [Public Member Functions](#) |
[Public Attributes](#) | [Friends](#) | [List of all members](#)

```
#include <Plane.h>
```

Public Types

```
enum Position { BELOW, ABOVE }
```

Public Member Functions

Plane ()

Plane (double a, double b, double c, double d)

Plane (Eigen::Vector3d normal, Eigen::Vector3d point)

Plane (std::vector< Eigen::Vector3d > &ptArr)

bool **belowPlane** (Eigen::Vector3d point, double offset)

void **segmentCloud** (pcl::PointCloud< pcl::PointXYZ > &cloud,
pcl::PointCloud< pcl::PointXYZ > &output, **Position** position,
double offset)

void **segmentMesh** (**MyMesh** &mesh, **MyMesh** &output, **Position**
position, double offset=0)

bool **pointNormForm** (Eigen::Vector3d normal, Eigen::Vector3d point)

bool **FromThreePt** (std::vector< Eigen::Vector3d > &ptArr)

Public Attributes

Eigen::Vector3d **Normal**

double **D**

Friends

std::ostream & **operator<<** (std::ostream &os, const **Plane** &plane)

Member Enumeration Documentation

◆ Position

```
enum Inshape::Plane::Position
```

Enumerator

BELOW

ABOVE

Constructor & Destructor Documentation

◆ Plane() [1/4]

Inshape::Plane::Plane ()

inline

◆ Plane() [2/4]

```
Inshape::Plane::Plane ( double a,  
                        double b,  
                        double c,  
                        double d  
                        )
```

inline

◆ Plane() [3/4]

```
Inshape::Plane::Plane ( Eigen::Vector3d normal,  
                        Eigen::Vector3d point  
                        )
```

inline

◆ Plane() [4/4]

```
Inshape::Plane::Plane ( std::vector< Eigen::Vector3d > & ptArr ) inline
```

Member Function Documentation

◆ belowPlane()

```
bool Inshape::Plane::belowPlane ( Eigen::Vector3d point,  
                                   double           offset  
                                   )
```

◆ FromThreePt()

bool

```
Inshape::Plane::FromThreePt ( std::vector< Eigen::Vector3d > & ptArr )
```

◆ pointNormForm()

```
bool Inshape::Plane::pointNormForm ( Eigen::Vector3d normal,  
                                      Eigen::Vector3d point  
                                      )
```

◆ segmentCloud()

```
void  
Inshape::Plane::segmentCloud ( pcl::PointCloud< pcl::PointXYZ > & clou  
                                pcl::PointCloud< pcl::PointXYZ > & outp  
                                Position                                pos  
                                double                                offs  
                                )
```

◆ segmentMesh()

```
void Inshape::Plane::segmentMesh ( MyMesh & mesh,  
                                   MyMesh & output,  
                                   Position position,  
                                   double offset = 0  
                                   )
```

Friends And Related Function Documentation

◆ operator<<

```
std::ostream& operator<< ( std::ostream & os,  
                          const Plane & plane  
                          )
```

friend

Member Data Documentation

◆ D

```
double Inshape::Plane::D
```

◆ Normal

Eigen::Vector3d Inshape::Plane::Normal

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

- include/**Plane.h**
-

Generated by  1.8.14

Inshaping 0.1

[Inshape](#)

[Poisson](#)

Inshape::Poisson Class Reference

[Utils](#)

[Public Member Functions](#) |

[Static Public Member Functions](#) |

[List of all members](#)

```
#include <Poisson.h>
```

Public Member Functions

Poisson ()

Poisson (pcl::PointCloud< pcl::PointXYZ >::Ptr &cloud)

void **setDepth** (int _depth)

void **setInputCloud** (pcl::PointCloud< pcl::PointXYZ >::Ptr
&cloud)

MyMeshPtr **getOutput** ()

Static Public Member Functions

```
static MyMeshPtr getOutput (pcl::PointCloud< pcl::PointNormal >::Ptr  
    &cloud, int depth)
```

Constructor & Destructor Documentation

◆ Poisson() [1/2]

Inshape::Poisson::Poisson ()

inline

◆ Poisson() [2/2]

```
Inshape::Poisson::Poisson ( pcl::PointCloud< pcl::PointXYZ >::Ptr & clou
```

Member Function Documentation

◆ `getOutput()` [1/2]

```
MyMeshPtr Inshape::Poisson::getOutput ( )
```


◆ setDepth()

```
void Inshape::Poisson::setDepth ( int _depth )
```

inline

◆ setInputCloud()

void

Inshape::Poisson::setInputCloud (pcl::PointCloud< pcl::PointXYZ >::Ptr &

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

- include/**Poisson.h**

Generated by doxygen 1.8.14

Inshaping 0.1

Benchmark

Functions

Functions

double **Inshape::avgE1** (TriSparse &triSprase)

double **Inshape::avgE3** (TriSparse &triSprase)

Detailed Description

Function Documentation

◆ avgE1()

```
double Inshape::avgE1 ( TriSparse & triSprase )
```

◆ avgE3()

```
double Inshape::avgE3 ( TriSparse & triSprase )
```

Generated by doxygen 1.8.14

Inshaping 0.1

Functions

IO

Functions

bool **Inshape::readPly** (const std::string &filename, pcl::PointCloud< pcl::PointXYZ > &cloud)

bool **Inshape::savePly** (const std::string &filename, pcl::PointCloud< pcl::PointXYZ > &cloud)

bool **Inshape::savePolygonMesh** (std::string filename, pcl::PolygonMesh &mesh)

Detailed Description

Function Documentation

◆ readPly()

```
bool Inshape::readPly ( const std::string & filename,  
                       pcl::PointCloud< pcl::PointXYZ > & cloud  
                       )
```

◆ savePly()

```
bool Inshape::savePly ( const std::string & filename,  
                       pcl::PointCloud< pcl::PointXYZ > & cloud  
                       )
```

◆ savePolygonMesh()

```
bool Inshape::savePolygonMesh ( std::string      filename,  
                                pcl::PolygonMesh & mesh  
                                )
```

Inshaping 0.1

[Classes](#) | [Macros](#)

DataStructure

Classes

class **CCLib::HyperCloud**
An Extended point cloud (with point duplication mechanism)
More...

struct **Inshape::ParamItem**

struct **Inshape::Param**

class **Inshape::Setting**

Macros

```
#define HEAD_FILENAME "HEAD_FILENAME"
```

```
#define BODY_FILENAME "BODY_FILENAME"
```

```
#define SUB_HEAD "SUB_HEAD"
```

```
#define SUB_BODY "SUB_BODY"
```

```
#define FACE_FILENAME "FACE_FILENAME"
```

```
#define FINAL_PARAM "FINAL_PARAM"
```

Detailed Description

Macro Definition Documentation

◆ BODY_FILENAME

```
#define BODY_FILENAME "BODY_FILENAME"
```

◆ FACE_FILENAME

```
#define FACE_FILENAME "FACE_FILENAME"
```

◆ FINAL_PARAM

```
#define FINAL_PARAM "FINAL_PARAM"
```

◆ HEAD_FILENAME

```
#define HEAD_FILENAME "HEAD_FILENAME"
```

◆ SUB_BODY

```
#define SUB_BODY "SUB_BODY"
```

◆ SUB_HEAD

```
#define SUB_HEAD "SUB_HEAD"
```

Inshaping 0.1

CCLib > HyperCloud >

CCLib::HyperCloud Class Reference

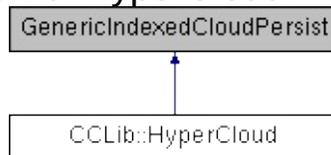
DataStructure

[Public Types](#) | [Public Member Functions](#) |
[Protected Types](#) | [Protected Attributes](#) |
[List of all members](#)

An Extended point cloud (with point duplication mechanism) [More...](#)

```
#include <HyperCloud.h>
```

Inheritance diagram for CCLib::HyperCloud:



Public Types

```
typedef std::shared_ptr< HyperCloud > Ptr
```

Public Member Functions

HyperCloud ()

The PowerfulCloud constructor.
[More...](#)

virtual **~HyperCloud ()**

The PowerfulCloud destructor.
[More...](#)

virtual unsigned **size ()** const

virtual void **forEach** (genericPointAction actio

virtual void **getBoundingBox** (CCVector3 &bbMin, CCVector3 &bbMax)

virtual std::vector< Eigen::Vector3d > **getOBB ()**

virtual void **placeIteratorAtBeginning ()**

virtual const CCVector3 * **getNextPoint ()**

virtual bool **enableScalarField ()**

virtual bool **isScalarFieldEnabled ()** const

virtual void **setPointScalarValue** (unsigned pointIndex, ScalarType value)

virtual ScalarType **getPointScalarValue** (unsigned pointIndex) const

virtual const CCVector3 * **getPoint** (unsigned index)

virtual const Eigen::Vector3d **getEigenVec** (unsigned index)

virtual const pcl::PointXYZ **getPointXYZ** (unsigned index)

Eigen::Vector3d **getBaryCenter** ()

void **addCloud** (CCLib::HyperCloud &cloud)

virtual void **getPoint** (unsigned index, CCVector3 &P) const

virtual const CCVector3 * **getPointPersistentPtr** (unsigned index)

void **clear** ()
Clears cloud. [More...](#)

virtual void **addPoint** (const CCVector3 &P)
Point insertion mechanism. [More..](#)

virtual void **addPoint** (const Eigen::Vector3d &P)

virtual void **addPoint** (const PointCoordinateType P[])
Point insertion mechanism. [More..](#)

virtual bool **reserve** (unsigned n)
Reserves some memory for hosting the points. [More...](#)

virtual bool **resize** (unsigned n)
Presets the size of the vector used to store the points. [More...](#)

virtual void **applyTransformation** (PointProjectionTools::Transformation &trans)

Applies a rigid transformation to the cloud. [More...](#)

ScalarField * **getScalarField ()**
Returns associated scalar field (if any) [More...](#)

const ScalarField * **getScalarField () const**
Returns associated scalar field (if any) (const version) [More...](#)

Protected Types

typedef GenericChunkedArray< 3, PointCoordinateType > **PointsConta**
Point contain
[More...](#)

Protected Attributes

PointsContainer * **m_points**
3D Points container [More...](#)

ScalarField * **m_scalarField**
The points distances. [More...](#)

unsigned **globalIterator**
Iterator on the points container. [More...](#)

bool **m_validBB**
Bounding-box validity. [More...](#)

Detailed Description

An Extended point cloud (with point duplication mechanism)

Implements the GenericIndexedCloud interface. A simple point cloud that stores its own point instances and distances in a vector.

Member Typedef Documentation

◆ PointsContainer

```
typedef GenericChunkedArray<3, PointCoordinateType>  
CCLib::HyperCloud::PointsContainer
```

protected

Point container.

◆ Ptr

```
typedef std::shared_ptr<HyperCloud> CCLib::HyperCloud::Ptr
```

Constructor & Destructor Documentation

◆ HyperCloud()

```
CCLib::HyperCloud::HyperCloud ( )
```

The PowerfulCloud constructor.

◆ ~HyperCloud()

virtual CCLib::HyperCloud::~~HyperCloud ()

virtual

The PowerfulCloud destructor.

Member Function Documentation

◆ addCloud()

```
void CCLib::HyperCloud::addCloud ( CCLib::HyperCloud & cloud )
```

◆ addPoint() [1/3]

```
virtual void  
CCLib::HyperCloud::addPoint      ( const CCVector3 & P ) virtual
```

Point insertion mechanism.

The point data will be duplicated in memory.

Parameters

P the point to insert

◆ addPoint() [2/3]

virtual void

CCLib::HyperCloud::addPoint (const Eigen::Vector3d & p) virtual

◆ addPoint() [3/3]

virtual void

CCLib::HyperCloud::addPoint (const PointCoordinateType P[]) virtual

Point insertion mechanism.

The point data will be duplicated in memory.

Parameters

P the point to insert (as a 3-size array)

◆ applyTransformation()

virtual void

CCLib::HyperCloud::applyTransformation (PointProjectionTools::Transfo

Applies a rigid transformation to the cloud.

WARNING: THIS METHOD IS NOT COMPATIBLE WITH PARALLEL ST

Parameters

trans transformation (scale * rotation matrix + translation vector)

◆ clear()

```
void CCLib::HyperCloud::clear ( )
```

Clears cloud.

◆ enableScalarField()

virtual bool CCLib::HyperCloud::enableScalarField ()

virtual

◆ forEach()

virtual void

CCLib::HyperCloud::forEach (genericPointAction *action*) virtual

◆ getBaryCenter()

```
Eigen::Vector3d CCLib::HyperCloud::getBaryCenter ( )
```

◆ getBoundingBox()

virtual void

```
CCLib::HyperCloud::getBoundingBox ( CCVector3 & bbMin,  
                                     CCVector3 & bbMax  
                                     )
```

virtual

◆ getEigenVec()

```
virtual const Eigen::Vector3d  
CCLib::HyperCloud::getEigenVec      ( unsigned index ) inline virtual
```

◆ getNextPoint()

```
virtual const CCVector3* CCLib::HyperCloud::getNextPoint ( )
```

virtual

◆ getOBB()

```
virtual std::vector<Eigen::Vector3d>  
CCLib::HyperCloud::getOBB
```

```
( ) virtual
```

◆ `getPoint()` [1/2]

virtual const CCVector3*

CCLib::HyperCloud::getPoint

(unsigned `index`)

inline

virtual

◆ `getPoint()` [2/2]

```
virtual void CCLib::HyperCloud::getPoint ( unsigned    index,  
                                             CCVector3 & P  
                                             ) const virtual
```

◆ getPointPersistentPtr()

virtual const CCVector3*

CCLib::HyperCloud::getPointPersistentPtr (unsigned index) virtual

◆ getPointScalarValue()

virtual ScalarType

CCLib::HyperCloud::getPointScalarValue (unsigned `pointIndex`) const

◆ getPointXYZ()

virtual const pcl::PointXYZ

CCLib::HyperCloud::PointXYZ

(unsigned **index**)

inline

virtual

◆ getScalarField() [1/2]

ScalarField* CCLib::HyperCloud::getScalarField ()

inline

Returns associated scalar field (if any)

◆ getScalarField() [2/2]

```
const ScalarField* CCLib::HyperCloud::getScalarField ( ) const
```

inline

Returns associated scalar field (if any) (const version)

◆ isScalarFieldEnabled()

```
virtual bool CCLib::HyperCloud::isScalarFieldEnabled ( ) const
```

virtual

◆ placeliteratorAtBeginning()

```
virtual void CCLib::HyperCloud::placeliteratorAtBeginning ( )
```

virtual

◆ reserve()

```
virtual bool CCLib::HyperCloud::reserve ( unsigned n )
```

virtual

Reserves some memory for hosting the points.

Parameters

n the number of points

◆ resize()

```
virtual bool CCLib::HyperCloud::resize ( unsigned n )
```

virtual

Presets the size of the vector used to store the points.

Parameters

n the number of points

◆ setPointScalarValue()

virtual void

```
CCLib::HyperCloud::setPointScalarValue ( unsigned   pointIndex,  
                                           ScalarType value  
                                           )
```

virtual

◆ size()

```
virtual unsigned CCLib::HyperCloud::size ( ) const
```

virtual

Member Data Documentation

◆ globalIterator

unsigned CCLib::HyperCloud::globalIterator

protected

Iterator on the points container.

◆ m_points

PointsContainer* CCLib::HyperCloud::m_points

protected

3D Points container

◆ m_scalarField

ScalarField* CCLib::HyperCloud::m_scalarField

protected

The points distances.

◆ m_validBB

```
bool CCLib::HyperCloud::m_validBB
```

protected

Bounding-box validity.

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

- include/**HyperCloud.h**

Generated by doxygen 1.8.14

Inshaping 0.1

[Inshape](#) > [ParamItem](#)

Inshape::ParamItem Struct Reference

[DataStructure](#)

[Public Member Functions](#) | [Public Attributes](#) |
[List of all members](#)

```
#include <Setting.h>
```

Public Member Functions

std::string **toString** (int i)

Public Attributes

double **theta**

double **lamda_min**

double **lamda_max**

double **eta**

double **threshold**

Member Function Documentation

◆ toString()

std::string Inshape::ParamItem::toString (int i)

inline

Member Data Documentation

◆ eta

```
double Inshape::ParamItem::eta
```

◆ lamda_max

```
double Inshape::ParamItem::lamda_max
```

◆ lamda_min

```
double Inshape::ParamItem::lamda_min
```

◆ theta

```
double Inshape::ParamItem::theta
```

◆ threshold

```
double Inshape::ParamItem::threshold
```

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

- include/**Setting.h**

Generated by  1.8.14

Inshaping 0.1

[Inshape](#) > [Param](#)

Inshape::Param Struct Reference

[DataStructure](#)

[Public Member Functions](#) | [Public Attributes](#) |
[List of all members](#)

```
#include <Setting.h>
```

Public Member Functions

```
void clear ()
```

Public Attributes

`std::vector< ParamItem > paramItems`

`ParamItem finalParam`

Member Function Documentation

◆ clear()

```
void Inshape::Param::clear ( )
```

inline

Member Data Documentation

◆ finalParam

ParamItem Inshape::Param::finalParam

◆ paramItems

```
std::vector<ParamItem> Inshape::Param::paramItems
```

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

- include/**Setting.h**

Generated by **doxygen** 1.8.14

Inshaping 0.1

Inshape > Setting >

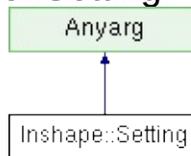
[Public Member Functions](#) | [Public Attributes](#) |
[List of all members](#)

Inshape::Setting Class Reference

DataStructure

```
#include <Setting.h>
```

Inheritance diagram for Inshape::Setting:



Public Member Functions

bool **readInputTemplate** (std::string filename)

bool **readMatch** (std::string filename)

bool **readParam** (std::string filename)

Setting ()

bool **parse_argv** (int argc, char **argv)

► Public Member Functions inherited from **Anyarg**

Anyarg ()

Construct a **Anyarg** object. [More...](#)

bool **parse_argv** (int argc, char **argv)

int **get_argc** () const

const char * **get_arg** (int i) const

const char * **auto_usage** ()

bool **add_flag** (const char *name, char letter, const char *desc)

bool **add_flag** (char letter, const char *desc)

bool **add_option_str** (const char *name, char letter, const char *v0, const char *desc)

bool **add_option_str** (char letter, const char *v0, const char *desc)

bool **add_option_int** (char letter, int v0, const char *desc)

bool **add_option_int** (const char *name, char letter, int v0, const char *desc)

bool **add_option_double** (char letter, double v0, const char *desc)

bool **add_option_double** (const char *name, char letter, double v0, const char *desc)

bool **is_true** (const char *name) const

bool **is_true** (char letter) const

const char * **get_value_str** (const char *name) const

const char * **get_value_str** (char letter) const

int **get_value_int** (const char *name) const

int **get_value_int** (char letter) const

double **get_value_double** (const char *name) const

double **get_value_double** (char letter) const

Public Attributes

std::string **headFilename**

std::string **faceFilename**

std::string **bodyFilename**

std::string **outputFilename**

std::string **subHead**

std::string **subBody**

Param **param**

bool **preAligned**

bool **outputIntermediate**

bool **useMatch**

Constructor & Destructor Documentation

◆ Setting()

```
Inshape::Setting::Setting ( )
```

Member Function Documentation

◆ parse_argv()

```
bool Inshape::Setting::parse_argv ( int   argc,  
                                     char ** argv  
                                     )
```

◆ readInputTemplate()

```
bool Inshape::Setting::readInputTemplate ( std::string filename )
```

◆ readMatch()

```
bool Inshape::Setting::readMatch ( std::string filename )
```

◆ readParam()

```
bool Inshape::Setting::readParam ( std::string filename )
```

Member Data Documentation

◆ bodyFilename

```
std::string Inshape::Setting::bodyFilename
```

◆ faceFilename

```
std::string Inshape::Setting::faceFilename
```

◆ headFilename

```
std::string Inshape::Setting::headFilename
```

◆ outputFilename

```
std::string Inshape::Setting::outputFilename
```

◆ outputIntermediate

```
bool Inshape::Setting::outputIntermediate
```

◆ param

Param Inshape::Setting::param

◆ preAligned

```
bool Inshape::Setting::preAligned
```

◆ subBody

```
std::string Inshape::Setting::subBody
```

◆ subHead

```
std::string Inshape::Setting::subHead
```

◆ useMatch

```
bool Inshape::Setting::useMatch
```

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

- include/**Setting.h**

Generated by doxygen 1.8.14

Inshaping 0.1

[Classes](#) | [Functions](#)

Rigsteration

Classes

class **Inshape::TriSparse**

Functions

Eigen::Matrix4f **Inshape::ICPRigsteration** (pcl::PointCloud< pcl::PointXYZ >::Ptr &data, pcl::PointCloud< pcl::PointXYZ >::Ptr &model)

Eigen::Matrix4f **Inshape::NICPRigsteration** (pcl::PointCloud< pcl::PointXYZ >::Ptr &data, pcl::PointCloud< pcl::PointXYZ >::Ptr &model)

Eigen::Matrix4f **Inshape::SVDRegistration** (pcl::PointCloud< pcl::PointXYZ >::Ptr &sub_data, pcl::PointCloud< pcl::PointXYZ >::Ptr &sub_model)

void **Inshape::AlignWithFace** (pcl::PointCloud< pcl::PointXYZ >::Ptr &body, pcl::PointCloud< pcl::PointXYZ >::Ptr &head_cloud, pcl::PointCloud< pcl::PointXYZ >::Ptr &face)

void **Inshape::rigidRigsterWithoutMatch** (pcl::PointCloud< pcl::PointXYZ >::Ptr &body, **MyMesh** &head_mesh, pcl::PointCloud< pcl::PointXYZ >::Ptr &face)

void **Inshape::rigidRigsterWithMatch** (pcl::PointCloud< pcl::PointXYZ >::Ptr &body, **MyMesh** &head_mesh, pcl::PointCloud< pcl::PointXYZ >::Ptr &face, pcl::PointCloud< pcl::PointXYZ >::Ptr &sub_head, pcl::PointCloud< pcl::PointXYZ >::Ptr &sub_body)

Detailed Description

Function Documentation

◆ AlignWithFace()

```
void  
Inshape::AlignWithFace ( pcl::PointCloud< pcl::PointXYZ >::Ptr & body,  
                        pcl::PointCloud< pcl::PointXYZ >::Ptr & head_  
                        pcl::PointCloud< pcl::PointXYZ >::Ptr & face  
                        )
```

◆ ICPRigsteration()

Eigen::Matrix4f

```
Inshape::ICPRigsteration ( pcl::PointCloud< pcl::PointXYZ >::Ptr & data,  
                           pcl::PointCloud< pcl::PointXYZ >::Ptr & mode  
                           )
```

◆ NICPRigsteration()

Eigen::Matrix4f

```
Inshape::NICPRigsteration ( pcl::PointCloud< pcl::PointXYZ >::Ptr & data  
                           pcl::PointCloud< pcl::PointXYZ >::Ptr & moc  
                           )
```

◆ rigidRigsterWithMatch()

```
void  
Inshape::rigidRigsterWithMatch ( pcl::PointCloud< pcl::PointXYZ >::Ptr &  
                                MyMesh &  
                                pcl::PointCloud< pcl::PointXYZ >::Ptr &  
                                pcl::PointCloud< pcl::PointXYZ >::Ptr &  
                                pcl::PointCloud< pcl::PointXYZ >::Ptr &  
                                )
```

◆ rigidRigsterWithoutMatch()

```
void  
Inshape::rigidRigsterWithoutMatch ( pcl::PointCloud< pcl::PointXYZ >::Pt  
                                   MyMesh &  
                                   pcl::PointCloud< pcl::PointXYZ >::Pt  
                                   )
```

◆ SVDRegistration()

Eigen::Matrix4f

```
Inshape::SVDRegistration ( pcl::PointCloud< pcl::PointXYZ >::Ptr & sub_  
                           pcl::PointCloud< pcl::PointXYZ >::Ptr & sub_  
                           )
```

Inshaping 0.1

[Inshape](#) > [TriSparse](#)

Inshape::TriSparse Class Reference

[Rigsteration](#)

[Public Member Functions](#) |
[Static Public Member Functions](#) |
[List of all members](#)

```
#include <TriSparse.h>
```

Public Member Functions

TriSparse

(pcl::PointCloud<
pcl::PointNormal >::Ptr
_body, **MyMeshPtr**
_head)

void **setParam** (const double
_THRESHOLD, const
double _THETA, const
double _LAMDA_MIN,
const double
_LAMDA_MAX, const
double _ETA, bool
_doubleCheck)

void **setParam** (const
ParamItem &_param,
bool _doubleCheck)

void **printParam** ()

void **setInput**
(pcl::PointCloud<
pcl::PointNormal >::Ptr
_body, **MyMeshPtr**
_head)

void **setA**
(Eigen::SparseMatrix<
double > &_A)

void **setB** (Eigen::VectorXd
&_B)

void **updateHead**
(pcl::PointCloud<

pcl::PointXYZ >::Ptr
&head_cloud)

double **getLAMDA** (unsigned
index)

MyMesh & **getHead** ()

pcl::PointCloud< pcl::PointNormal > & **getBody** ()

std::unordered_map< unsigned, unsigned > & **getMap** ()

bool **construct** ()

bool **solve** (Eigen::VectorXd
&x)

bool **solveToCloud**
(pcl::PointCloud<
pcl::PointXYZ >::Ptr
&cloud)

~TriSparse ()

Static Public Member Functions

static bool **solveTriSparse** (Eigen::SparseMatrix< double > &A, Eigen::VectorXd &B, Eigen::VectorXd &x)

static void **confirmNearestNeighbor** (pcl::PointCloud< pcl::PointNormal > &body, **MyMesh** &head_mesh, std::unordered_map< unsigned, unsigned > &mapping, double threshold)

static void **confirmNearestNeighbor** (pcl::PointCloud< pcl::PointXYZ > &from, pcl::PointCloud< pcl::PointNormal > &to, std::unordered_map< unsigned, unsigned > &mapping, double threshold)

static void **findNearestNeighbor** (pcl::PointCloud< pcl::PointNormal > &body, **MyMesh** &head, std::unordered_map< unsigned, unsigned > &mapping, double threshold, bool doubleCheck)

static void **findNearestNeighbor2** (pcl::PointCloud< pcl::PointXYZ > &body, pcl::PointCloud< pcl::PointXYZ > &head, std::unordered_map< unsigned, unsigned > &mapping, double threshold, bool doubleCheck)

static void **confirmNearestNeighbor2** (pcl::PointCloud< pcl::PointXYZ > &from, pcl::PointCloud< pcl::PointXYZ > &to, std::unordered_map< unsigned, unsigned > &mapping, double threshold)

static void **updateMesh** (**MyMesh** &mesh, pcl::PointCloud< pcl::PointXYZ >::Ptr &cloud)

Constructor & Destructor Documentation

◆ TriSparse()

```
Inshape::TriSparse::TriSparse ( pcl::PointCloud< pcl::PointNormal >::Ptr  
                               MyMeshPtr  
                               )
```

◆ ~TriSparse()

Inshape::TriSparse::~~TriSparse ()

inline

Member Function Documentation

◆ confirmNearestNeighbor() [1/2]

```
static void  
Inshape::TriSparse::confirmNearestNeighbor ( pcl::PointCloud< pcl::Poin  
MyMesh &  
std::unordered_map< uns  
double  
)
```

◆ confirmNearestNeighbor() [2/2]

```
static void  
Inshape::TriSparse::confirmNearestNeighbor ( pcl::PointCloud< pcl::Poin  
                                             pcl::PointCloud< pcl::Poin  
                                             std::unordered_map< uns  
                                             double  
                                             )
```

◆ confirmNearestNeighbor2()

```
static void  
Inshape::TriSparse::confirmNearestNeighbor2 ( pcl::PointCloud< pcl::Poi  
                                             pcl::PointCloud< pcl::Poi  
                                             std::unordered_map< un  
                                             double  
                                             )
```

◆ construct()

```
bool Inshape::TriSparse::construct ( )
```

◆ findNearestNeighbor()

```
static void  
Inshape::TriSparse::findNearestNeighbor ( pcl::PointCloud< pcl::PointNormal > &  
                                           MyMesh &  
                                           std::unordered_map< unsigned int, unsigned int > &  
                                           double  
                                           bool  
                                           )
```

◆ findNearestNeighbor2()

```
static void  
Inshape::TriSparse::findNearestNeighbor2 ( pcl::PointCloud< pcl::PointXYZ  
                                           pcl::PointCloud< pcl::PointXYZ  
                                           std::unordered_map< unsigned  
                                           double  
                                           bool  
                                           )
```

◆ `getBody()`

```
pcl::PointCloud<pcl::PointNormal>&  
Inshape::TriSparse::getBody
```

```
( ) inline
```

◆ getHead()

MyMesh& Inshape::TriSparse::getHead ()

inline

◆ getLAMDA()

```
double Inshape::TriSparse::getLAMDA ( unsigned index )
```

◆ getMap()

```
std::unordered_map<unsigned, unsigned>&  
Inshape::TriSparse::getMap
```

```
( ) inline
```

◆ printParam()

```
void Inshape::TriSparse::printParam ( )
```

◆ setA()

```
void Inshape::TriSparse::setA ( Eigen::SparseMatrix< double > & _A )
```

◆ setB()

```
void Inshape::TriSparse::setB ( Eigen::VectorXd & _B )
```

◆ setInput()

```
void  
Inshape::TriSparse::setInput ( pcl::PointCloud< pcl::PointNormal >::Ptr _  
                               MyMeshPtr _  
                               )
```

◆ setParam() [1/2]

```
void Inshape::TriSparse::setParam ( const double  _THRESHOLD,  
                                     const double  _THETA,  
                                     const double  _LAMDA_MIN,  
                                     const double  _LAMDA_MAX,  
                                     const double  _ETA,  
                                     bool          _doubleCheck  
                                     )
```

◆ setParam() [2/2]

```
void  
Inshape::TriSparse::setParam    ( const ParamItem & _param,  
                                bool                _doubleCheck  
                                )
```

◆ solve()

```
bool Inshape::TriSparse::solve ( Eigen::VectorXd & x )
```

◆ solveToCloud()

bool

```
Inshape::TriSparse::solveToCloud ( pcl::PointCloud< pcl::PointXYZ >::Ptr
```

◆ solveTriSparse()

```
static bool  
Inshape::TriSparse::solveTriSparse ( Eigen::SparseMatrix< double > & A  
                                     Eigen::VectorXd & B  
                                     Eigen::VectorXd & x  
                                     )
```

◆ updateHead()

void

Inshape::TriSparse::updateHead (pcl::PointCloud< pcl::PointXYZ >::Ptr d

◆ updateMesh()

```
static void  
Inshape::TriSparse::updateMesh ( MyMesh &  
                                pcl::PointCloud< pcl::PointXYZ >::Ptr &  
                                )
```

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

- include/**TriSparse.h**

Generated by doxygen 1.8.14

Inshaping 0.1

Namespace List

Here is a list of all namespaces with brief descriptions:

[detail level 1 2]

| | |
|---|--|
|  CCLib | |
|  Inshape | |
|  Convert | |
|  ConvexHull | |

Generated by  1.8.14

Inshaping 0.1

Classes

CCLib Namespace Reference

Classes

class **HyperCloud**

An Extended point cloud (with point duplication mechanism)

[More...](#)

Generated by  1.8.14

Inshaping 0.1

[Namespaces](#) | [Classes](#) | [Typedefs](#) | [Functions](#)

Inshape Namespace Reference

Namespaces

Convert

ConvexHull

Classes

class **BoundingBox**

class **Cutter**

class **Inshaper**

class **NormalEstimationPlus**

struct **Param**

struct **ParamItem**

class **Plane**

class **Poisson**

class **Setting**

struct **tri**

class **TriSparse**

Typedefs

```
typedef OpenMesh::PolyMesh_ArrayKernelT MyMesh
```

```
typedef std::shared_ptr< MyMesh > MyMeshPtr
```

```
typedef std::shared_ptr< CCLib::SimpleMesh > SimpleMeshPtr
```

Functions

| | | |
|---------------------------------------|--------|---|
| | double | avgE1 (TriSparse &triSprase) |
| | double | avgE3 (TriSparse &triSprase) |
| | bool | compareNorm (tri i, tri j) |
| pcl::PointCloud< pcl::PointXYZ >::Ptr | | hyperToPointCloud (CCLib::HyperCloud::Ptr &hyper) |
| CCLib::HyperCloud::Ptr | | cloudToHyper (pcl::PointCloud< pcl::PointXYZ > &cloud) |
| pcl::PolygonMesh::Ptr | | getConvexHull (pcl::PointCloud< pcl::PointXYZ >::Ptr &cloud) |
| pcl::PointCloud< pcl::PointXYZ >::Ptr | | filterWithHull (pcl::PointCloud< pcl::PointXYZ >::Ptr &bodyshead, pcl::PointCloud< pcl::PointXYZ >::Ptr &head, double threshold) |
| | bool | readPly (const std::string &filename, pcl::PointCloud< pcl::PointXYZ > &cloud) |
| | bool | savePly (const std::string &filename, pcl::PointCloud< pcl::PointXYZ > &cloud) |
| | bool | savePolygonMesh (std::string filename, pcl::PolygonMesh &mesh) |
| Eigen::Matrix4f | | ICPRigsteration |

(pcl::PointCloud< pcl::PointXYZ
>::Ptr &data, pcl::PointCloud<
pcl::PointXYZ >::Ptr &model)

Eigen::Matrix4f **NICPRigsteration**
(pcl::PointCloud< pcl::PointXYZ
>::Ptr &data, pcl::PointCloud<
pcl::PointXYZ >::Ptr &model)

Eigen::Matrix4f **SVDRegistration**
(pcl::PointCloud< pcl::PointXYZ
>::Ptr &sub_data,
pcl::PointCloud< pcl::PointXYZ
>::Ptr &sub_model)

void **AlignWithFace** (pcl::PointCloud<
pcl::PointXYZ >::Ptr &body,
pcl::PointCloud< pcl::PointXYZ
>::Ptr &head_cloud,
pcl::PointCloud< pcl::PointXYZ
>::Ptr &face)

void **rigidRigsterWithoutMatch**
(pcl::PointCloud< pcl::PointXYZ
>::Ptr &body, **MyMesh**
&head_mesh, pcl::PointCloud<
pcl::PointXYZ >::Ptr &face)

void **rigidRigsterWithMatch**
(pcl::PointCloud< pcl::PointXYZ
>::Ptr &body, **MyMesh**
&head_mesh, pcl::PointCloud<
pcl::PointXYZ >::Ptr &face,
pcl::PointCloud< pcl::PointXYZ
>::Ptr &sub_head,
pcl::PointCloud< pcl::PointXYZ
>::Ptr &sub_body)

Typedef Documentation

◆ MyMesh

```
typedef OpenMesh::PolyMesh_ArrayKernelT Inshape::MyMesh
```

◆ MyMeshPtr

```
typedef std::shared_ptr<MyMesh> Inshape::MyMeshPtr
```

◆ SimpleMeshPtr

```
typedef std::shared_ptr<CCLib::SimpleMesh>  
Inshape::SimpleMeshPtr
```

Function Documentation

◆ cloudToHyper()

CCLib::HyperCloud::Ptr

Inshape::cloudToHyper (pcl::PointCloud< pcl::PointXYZ > & cloud)

◆ filterWithHull()

```
pcl::PointCloud<pcl::PointXYZ>::Ptr  
Inshape::filterWithHull      ( pcl::PointCloud< pcl::PointXYZ >::F  
                               pcl::PointCloud< pcl::PointXYZ >::F  
                               double  
                               )
```

◆ getConvexHull()

```
pcl::PolygonMesh::Ptr  
Inshape::getConvexHull ( pcl::PointCloud< pcl::PointXYZ >::Ptr & cloud )
```

◆ hyperToPointCloud()

```
pcl::PointCloud<pcl::PointXYZ>::Ptr  
Inshape::hyperToPointCloud ( CCLib::HyperCloud::Ptr & hyper
```

Generated by [doxygen](#) 1.8.14

Inshaping 0.1

Here is a list of all namespace members with links to the namespace documentation for each member:

- [AlignWithFace\(\)](#) : [Inshape](#)
- [avgE1\(\)](#) : [Inshape](#)
- [avgE3\(\)](#) : [Inshape](#)
- [cloudToHyper\(\)](#) : [Inshape](#) , [Inshape::Convert](#)
- [compareNorm\(\)](#) : [Inshape](#)
- [filterWithHull\(\)](#) : [Inshape::ConvexHull](#) , [Inshape](#)
- [getConvexHull\(\)](#) : [Inshape::ConvexHull](#) , [Inshape](#)
- [hyperToPointCloud\(\)](#) : [Inshape::Convert](#) , [Inshape](#)
- [ICPRigsteration\(\)](#) : [Inshape](#)
- [meshToCloud\(\)](#) : [Inshape::Convert](#)
- [MyMesh](#) : [Inshape](#)
- [MyMeshPtr](#) : [Inshape](#)
- [NICPRigsteration\(\)](#) : [Inshape](#)
- [PolygonToMyMesh\(\)](#) : [Inshape::Convert](#)
- [readPly\(\)](#) : [Inshape](#)
- [rigidRigsterWithMatch\(\)](#) : [Inshape](#)
- [rigidRigsterWithoutMatch\(\)](#) : [Inshape](#)
- [savePly\(\)](#) : [Inshape](#)
- [savePolygonMesh\(\)](#) : [Inshape](#)
- [SimpleMeshPtr](#) : [Inshape](#)
- [SVDRegistration\(\)](#) : [Inshape](#)

Inshaping 0.1

- AlignWithFace() : **Inshape**
- avgE1() : **Inshape**
- avgE3() : **Inshape**
- cloudToHyper() : **Inshape** , **Inshape::Convert**
- compareNorm() : **Inshape**
- filterWithHull() : **Inshape::ConvexHull** , **Inshape**
- getConvexHull() : **Inshape::ConvexHull** , **Inshape**
- hyperToPointCloud() : **Inshape::Convert** , **Inshape**
- ICPRigsteration() : **Inshape**
- meshToCloud() : **Inshape::Convert**
- NICPRigsteration() : **Inshape**
- PolygonToMyMesh() : **Inshape::Convert**
- readPly() : **Inshape**
- rigidRigsterWithMatch() : **Inshape**
- rigidRigsterWithoutMatch() : **Inshape**
- savePly() : **Inshape**
- savePolygonMesh() : **Inshape**
- SVDRegistration() : **Inshape**

Inshaping 0.1

- MyMesh : **Inshape**
- MyMeshPtr : **Inshape**
- SimpleMeshPtr : **Inshape**

Generated by **doxygen** 1.8.14

Inshaping 0.1

Class List

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

[detail level 1 2]

| | |
|-----------------------------|---|
| ▼ CCLib | |
| HyperCloud | An Extended point cloud (with point duplication mechanism) |
| ▼ Inshape | |
| BoundingBox | |
| Cutter | |
| Inshaper | |
| NormalEstimationPlus | |
| Param | |
| ParamItem | |
| Plane | |
| Poisson | |
| Setting | |
| tri | |
| TriSparse | |
| Anyarg | Use this class to define program options and parse command line arguments |
| Option | Data structure of a program option |

Inshaping 0.1

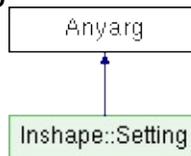
[Public Member Functions](#) | [List of all members](#)

Anyarg Class Reference

Use this class to define program options and parse command line arguments. [More...](#)

```
#include <anyarg.h>
```

Inheritance diagram for Anyarg:



Public Member Functions

Anyarg ()

Construct a **Anyarg** object. [More...](#)

bool **parse_argv** (int argc, char **argv)

int **get_argc** () const

const char * **get_arg** (int i) const

const char * **auto_usage** ()

bool **add_flag** (const char *name, char letter, const char *desc)

bool **add_flag** (char letter, const char *desc)

bool **add_option_str** (const char *name, char letter, const char *v0, const char *desc)

bool **add_option_str** (char letter, const char *v0, const char *desc)

bool **add_option_int** (char letter, int v0, const char *desc)

bool **add_option_int** (const char *name, char letter, int v0, const char *desc)

bool **add_option_double** (char letter, double v0, const char *desc)

bool **add_option_double** (const char *name, char letter, double v0, const char *desc)

bool **is_true** (const char *name) const

bool **is_true** (char letter) const

const char * **get_value_str** (const char *name) const

const char * **get_value_str** (char letter) const

int **get_value_int** (const char *name) const

int **get_value_int** (char letter) const

double **get_value_double** (const char *name) const

double **get_value_double** (char letter) const

Detailed Description

Use this class to define program options and parse command line arguments.

Constructor & Destructor Documentation

◆ Anyarg()

```
Anyarg::Anyarg ( )
```

Construct a **Anyarg** object.

Member Function Documentation

◆ add_flag() [1/2]

```
bool Anyarg::add_flag ( const char * name,  
                        char         letter,  
                        const char * desc  
                        )
```

Add a boolean flag to a program.

Parameters

- name** Long name of a flag. Using hyphen to connect multiple words, such as "buffer-size".
- letter** Single-letter label of a flag, such as 'i'. Set letter to 0 (not '0') if you do not need a short label.
- desc** Description of a flag, which will be used to generate usage. Set desc as "" if nothing to say.

Returns

true if success, false if the flag has been defined.

Note

The default value of flag is false.

◆ add_flag() [2/2]

```
bool Anyarg::add_flag ( char      letter,  
                        const char * desc  
                        )
```

◆ add_option_double() [1/2]

```
bool Anyarg::add_option_double ( char      letter,  
                                double     v0,  
                                const char * desc  
                                )
```

◆ add_option_double() [2/2]

```
bool Anyarg::add_option_double ( const char * name,  
                                char          letter,  
                                double        v0,  
                                const char * desc  
                                )
```

◆ add_option_int() [1/2]

```
bool Anyarg::add_option_int ( char      letter,  
                             int       v0,  
                             const char * desc  
                             )
```

◆ add_option_int() [2/2]

```
bool Anyarg::add_option_int ( const char * name,  
                             char        letter,  
                             int         v0,  
                             const char * desc  
                             )
```

◆ add_option_str() [1/2]

```
bool Anyarg::add_option_str ( const char * name,  
                             char         letter,  
                             const char * v0,  
                             const char * desc  
                             )
```

Add a taking-value option to a program.

Parameters

- name** Long name of a flag. Using hyphen to connect multiple words, such as "buffer-size".
- letter** Single-letter label of a flag, such as 'i'. Set letter to 0 if you do not need a short label.
- v0** The default value of this option.
- desc** Description of a flag, which will be used to generate usage. Set desc as "" if nothing to say. If description starts with an = (like: "=FILE Name of output file"), the word followed will be treated as a META word, which will be used in the generation of option usage, e.g.: "-f FILE, --filename=FILE".

Returns

true if success, false if the flag has been defined.

◆ add_option_str() [2/2]

```
bool Anyarg::add_option_str ( char      letter,  
                             const char * v0,  
                             const char * desc  
                             )
```

◆ auto_usage()

```
const char* Anyarg::auto_usage ( )
```

Generate well-formatted usage information for options defined for the program.

Returns

Usage information as a c-string.

◆ `get_arg()`

```
const char* Anyarg::get_arg ( int i ) const
```

Get a non-option argument by index

Parameters

- i** The index of a non-option arguments, ranging from 0 to `get_argc()` - 1. Non-option arguments are in the same order as they appeared in command line.

◆ get_argc()

```
int Anyarg::get_argc ( ) const
```

Get the number of non-option arguments.

Returns

The count of non-option arguments.

◆ `get_value_double()` [1/2]

```
double Anyarg::get_value_double ( const char * name ) const
```

◆ `get_value_double()` [2/2]

```
double Anyarg::get_value_double ( char letter ) const
```

◆ `get_value_int()` [1/2]

```
int Anyarg::get_value_int ( const char * name ) const
```

◆ `get_value_int()` [2/2]

```
int Anyarg::get_value_int ( char letter ) const
```

◆ `get_value_str()` [1/2]

```
const char* Anyarg::get_value_str ( const char * name ) const
```

Get option value.

Parameters

name The same name used in `add_option_xxx()`.

letter Single-letter label of a flag, such as 'i'.

Returns

The value of an option. If the value is not assigned in command line, the default value will be returned.

Note

`get_value` functions should match with `add_option` functions. Use `get_value_str()`, `get_value_int()`, `get_value_double()` to get the value of an option whose value is of type string, int and double, respectively.

◆ `get_value_str()` [2/2]

```
const char* Anyarg::get_value_str ( char letter ) const
```

◆ is_true() [1/2]

```
bool Anyarg::is_true ( const char * name ) const
```

Check whether a flag is set to true in the command line.

Parameters

name The same name used in **add_flag()**.

letter Single-letter label of a flag.

Returns

true if a flag is set in command line, false if not.

◆ is_true() [2/2]

```
bool Anyarg::is_true ( char letter ) const
```

◆ parse_argv()

```
bool Anyarg::parse_argv ( int    argc,  
                          char ** argv  
                          )
```

Parse command line arguments.

Precondition

Flags and options have to be defined by **add_flag()** and **add_option_xxx()** before the parsing of command line.

Parameters

argc The number of arguments in command line.

argv Array including all command line arguments.

Returns

true if success, false if fail.

Note

The first elements of argv MUST be the name of the program.

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

- include/**anyarg.h**

Inshaping 0.1

Option Struct Reference

[Public Member Functions](#) | [Public Attributes](#) |
[List of all members](#)

Data structure of a program option. [More...](#)

```
#include <anyarg.h>
```

Public Member Functions

Option ()

bool **set_value** (const char *opt_value)

bool **set_desc_meta** (const char *opt_desc)

Public Attributes

char **type**

char **letter**

string **name**

string **value_str**

union {
 bool **value_bool**

 int **value_int**

 double **value_double**

};

string **desc**

string **meta**

Detailed Description

Data structure of a program option.

Constructor & Destructor Documentation

◆ Option()

Option::Option ()

inline

Member Function Documentation

◆ set_desc_meta()

```
bool Option::set_desc_meta ( const char * opt_desc )
```

◆ set_value()

```
bool Option::set_value ( const char * opt_value )
```

Member Data Documentation

◆ @1

```
union { ... }
```

◆ desc

```
string Option::desc
```

◆ letter

char Option::letter

◆ meta

```
string Option::meta
```

◆ name

```
string Option::name
```

◆ type

char Option::type

◆ value_bool

```
bool Option::value_bool
```

◆ value_double

```
double Option::value_double
```

◆ value_int

```
int Option::value_int
```

◆ value_str

```
string Option::value_str
```

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

- include/**anyarg.h**

Generated by **doxygen** 1.8.14

Inshaping 0.1

Class Index

[a](#) | [b](#) | [c](#) | [h](#) | [i](#) | [n](#) | [o](#) | [p](#) | [s](#) | [t](#)

a

[Anyarg](#)

b

[BoundingBox \(Inshape\)](#)

c

[Cutter \(Inshape\)](#)

h

[HyperCloud \(CCLib\)](#)

i

[Inshaper \(Inshape\)](#)

n

[NormalEstimationPlus \(Inshape\)](#)

o

[Optior](#)

p

[Param](#)

[Param](#)

[a](#) | [b](#) | [c](#) | [h](#) | [i](#) | [n](#) | [o](#) | [p](#) | [s](#) | [t](#)

Inshaping 0.1

Class Hierarchy

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

[detail level 1 2]

| | |
|--|--|
| ▼  Anyarg | Use this class to define program options a parse command line argument |
|  Inshape::Setting | |
|  Inshape::BoundingBox | |
|  Inshape::Cutter | |
| ▼  GenericIndexedCloudPersist | |
|  CCLib::HyperCloud | An Extended point cloud (with pair duplication mechanism) |
|  Inshape::Inshaper | |
| ▼  NormalEstimation | |
|  Inshape::NormalEstimationPlus | |
|  Option | Data structure a program option |
|  Inshape::Param | |

 Inshape::ParamItem

 Inshape::Plane

 Inshape::NormalEstimationPlus::WArc::PointerAndIdx

 Inshape::Poisson

 Inshape::tri

 Inshape::TriSparse

Inshaping 0.1

[Inshape](#)[NormalEstimationPlus](#)[WArc](#)[PointerAndIdx](#)[Public Member Functions](#) | [Public Attributes](#) |[List of all members](#)

Inshape::NormalEstimationPlus::WArc::Pointer/ Struct Reference

```
#include <Normal.h>
```

Public Member Functions

Eigen::Vector3f **normalCopy** ()

void **invertNormal** ()

Public Attributes

pcl::PointNormal * **p**

int **index**

Member Function Documentation

◆ invertNormal()

void

```
Inshape::NormalEstimationPlus::WArc::PointerAndIdx::invertNormal ( )
```

◆ normalCopy()

Eigen::Vector3f

Inshape::NormalEstimationPlus::WArc::PointerAndIdx::normalCopy ()

Member Data Documentation

◆ index

```
int Inshape::NormalEstimationPlus::WArc::PointerAndIdx::index
```

◆ p

```
pcl::PointNormal*  
Inshape::NormalEstimationPlus::WArc::PointerAndIdx::p
```

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

- include/**Normal.h**

Generated by doxygen 1.8.14

Inshaping 0.1

Here is a list of all class members with links to the classes they belong to:

- a -

- ABOVE : [Inshape::Plane](#)
- add_flag() : [Anyarg](#)
- add_option_double() : [Anyarg](#)
- add_option_int() : [Anyarg](#)
- add_option_str() : [Anyarg](#)
- addCloud() : [CCLib::HyperCloud](#)
- AddNeighboursToHeap() : [Inshape::NormalEstimationPlus](#)
- addPoint() : [CCLib::HyperCloud](#)
- Anyarg() : [Anyarg](#)
- applyTransformation() : [CCLib::HyperCloud](#)
- auto_usage() : [Anyarg](#)

- b -

- BELOW : [Inshape::Plane](#)
- belowPlane() : [Inshape::Plane](#)
- bodyFilename : [Inshape::Setting](#)
- BoundingBox() : [Inshape::BoundingBox](#)

- c -

- checkBodyWithHead() : [Inshape::Cutter](#)
- clear() : [CCLib::HyperCloud](#) , [Inshape::Param](#)
- coherencyPass() : [Inshape::NormalEstimationPlus](#)
- computeNormals() : [Inshape::BoundingBox](#)
- confirmNearestNeighbor() : [Inshape::TriSparse](#)
- confirmNearestNeighbor2() : [Inshape::TriSparse](#)
- construct() : [Inshape::TriSparse](#)
- Cutter() : [Inshape::Cutter](#)

- d -

- D : **Inshape::Plane**
- desc : **Option**

- e -

- edge : **Inshape::tri**
- enableScalarField() : **CCLib::HyperCloud**
- eta : **Inshape::ParamItem**

- f -

- faceFilename : **Inshape::Setting**
- finalParam : **Inshape::Param**
- findNearestNeighbor() : **Inshape::TriSparse**
- findNearestNeighbor2() : **Inshape::TriSparse**
- forEach() : **CCLib::HyperCloud**
- from : **Inshape::tri**
- FromThreePt() : **Inshape::Plane**

- g -

- get_arg() : **Anyarg**
- get_argc() : **Anyarg**
- get_value_double() : **Anyarg**
- get_value_int() : **Anyarg**
- get_value_str() : **Anyarg**
- getBaryCenter() : **CCLib::HyperCloud**
- getBody() : **Inshape::TriSparse**
- getBoundingBox() : **CCLib::HyperCloud**
- getCloserPoint() : **Inshape::BoundingBox**
- getClosestPoint() : **Inshape::BoundingBox**
- getCoherence() : **Inshape::NormalEstimationPlus**
- getCoherencyPointNormals() : **Inshape::NormalEstimationPlus**
- getCutResult() : **Inshape::Cutter**
- getEigenVec() : **CCLib::HyperCloud**
- getFarerPoint() : **Inshape::BoundingBox**
- getHead() : **Inshape::TriSparse**
- getLAMDA() : **Inshape::TriSparse**
- getLength() : **Inshape::BoundingBox**
- getMap() : **Inshape::TriSparse**

- getNextPoint() : **CCLib::HyperCloud**
- getOBB() : **CCLib::HyperCloud** , **Inshape::BoundingBox**
- getOutput() : **Inshape::Poisson**
- getOutputMesh() : **Inshape::Inshaper**
- getPoint() : **CCLib::HyperCloud**
- getPointPersistentPtr() : **CCLib::HyperCloud**
- getPointScalarValue() : **CCLib::HyperCloud**
- getPointXYZ() : **CCLib::HyperCloud**
- getScalarField() : **CCLib::HyperCloud**
- globalIterator : **CCLib::HyperCloud**

- h -

- headFilename : **Inshape::Setting**
- HyperCloud() : **CCLib::HyperCloud**

- i -

- index : **Inshape::NormalEstimationPlus::WArc::PointerAndIdx**
- Inshaper() : **Inshape::Inshaper**
- invertNormal() : **Inshape::NormalEstimationPlus::WArc::PointerAndIdx**
- is_true() : **Anyarg**
- isScalarFieldEnabled() : **CCLib::HyperCloud**

- l -

- lamda_max : **Inshape::ParamItem**
- lamda_min : **Inshape::ParamItem**
- letter : **Option**

- m -

- m_points : **CCLib::HyperCloud**
- m_scalarField : **CCLib::HyperCloud**
- m_validBB : **CCLib::HyperCloud**
- meta : **Option**

- n -

- name : **Option**
- Normal : **Inshape::Plane**
- normalCopy() : **Inshape::NormalEstimationPlus::WArc::PointerAndIdx**
- NormalEstimationPlus() : **Inshape::NormalEstimationPlus**

- o -

- operator<< : **Inshape::Plane**
- Option() : **Option**
- outputFilename : **Inshape::Setting**
- outputIntermediate : **Inshape::Setting**

- p -

- p : **Inshape::NormalEstimationPlus::WArc::PointerAndIdx**
- param : **Inshape::Setting**
- paramItems : **Inshape::Param**
- parse_argv() : **Anyarg** , **Inshape::Setting**
- placeliteratorAtBeginning() : **CCLib::HyperCloud**
- Plane() : **Inshape::Plane**
- pointNormForm() : **Inshape::Plane**
- PointsContainer : **CCLib::HyperCloud**
- Poisson() : **Inshape::Poisson**
- Position : **Inshape::Plane**
- preAligned : **Inshape::Setting**
- printParam() : **Inshape::TriSparse**
- Ptr : **CCLib::HyperCloud**

- r -

- readInputTemplate() : **Inshape::Setting**
- readMatch() : **Inshape::Setting**
- readParam() : **Inshape::Setting**
- reserve() : **CCLib::HyperCloud**
- resize() : **CCLib::HyperCloud**

- s -

- saveOBB() : **Inshape::BoundingBox**
- segmentCloud() : **Inshape::Plane**
- segmentMesh() : **Inshape::Plane**
- set_desc_meta() : **Option**
- set_value() : **Option**
- setA() : **Inshape::TriSparse**
- setB() : **Inshape::TriSparse**
- setBody() : **Inshape::Cutter**
- setCoherence() : **Inshape::NormalEstimationPlus**
- setDepth() : **Inshape::Poisson**
- setHead() : **Inshape::Cutter**
- setInput() : **Inshape::TriSparse**
- setInputBody() : **Inshape::Inshaper**
- setInputCloud() : **Inshape::Poisson**
- setInputTemplate() : **Inshape::Inshaper**
- setMatch() : **Inshape::Inshaper**
- setParam() : **Inshape::Inshaper , Inshape::TriSparse**
- setPointScalarValue() : **CCLib::HyperCloud**
- setPreAligned() : **Inshape::Inshaper**
- Setting() : **Inshape::Setting**
- size() : **CCLib::HyperCloud**
- solve() : **Inshape::TriSparse**
- solveToCloud() : **Inshape::TriSparse**
- solveTriSparse() : **Inshape::TriSparse**
- subBody : **Inshape::Setting**
- subHead : **Inshape::Setting**

- t -

- theta : **Inshape::ParamItem**
- threshold : **Inshape::ParamItem**
- to : **Inshape::tri**
- toString() : **Inshape::ParamItem**
- TriSparse() : **Inshape::TriSparse**
- type : **Option**

- u -

- unSetMatch() : **Inshape::Inshaper**
- updateHead() : **Inshape::TriSparse**

- updateMesh() : **Inshape::TriSparse**
- useMatch : **Inshape::Setting**

- V -

- value_bool : **Option**
- value_double : **Option**
- value_int : **Option**
- value_str : **Option**

- ~ -

- ~HyperCloud() : **CCLib::HyperCloud**
- ~TriSparse() : **Inshape::TriSparse**

Inshaping 0.1

- a -

- add_flag() : **Anyarg**
- add_option_double() : **Anyarg**
- add_option_int() : **Anyarg**
- add_option_str() : **Anyarg**
- addCloud() : **CCLib::HyperCloud**
- AddNeighboursToHeap() : **Inshape::NormalEstimationPlus**
- addPoint() : **CCLib::HyperCloud**
- Anyarg() : **Anyarg**
- applyTransformation() : **CCLib::HyperCloud**
- auto_usage() : **Anyarg**

- b -

- belowPlane() : **Inshape::Plane**
- BoundingBox() : **Inshape::BoundingBox**

- c -

- checkBodyWithHead() : **Inshape::Cutter**
- clear() : **CCLib::HyperCloud** , **Inshape::Param**
- coherencyPass() : **Inshape::NormalEstimationPlus**
- computeNormals() : **Inshape::BoundingBox**
- confirmNearestNeighbor() : **Inshape::TriSparse**
- confirmNearestNeighbor2() : **Inshape::TriSparse**
- construct() : **Inshape::TriSparse**
- Cutter() : **Inshape::Cutter**

- e -

- enableScalarField() : **CCLib::HyperCloud**

- f -

- findNearestNeighbor() : **Inshape::TriSparse**
- findNearestNeighbor2() : **Inshape::TriSparse**
- forEach() : **CCLib::HyperCloud**
- FromThreePt() : **Inshape::Plane**

- g -

- get_arg() : **Anyarg**
- get_argc() : **Anyarg**
- get_value_double() : **Anyarg**
- get_value_int() : **Anyarg**
- get_value_str() : **Anyarg**
- getBaryCenter() : **CCLib::HyperCloud**
- getBody() : **Inshape::TriSparse**
- getBoundingBox() : **CCLib::HyperCloud**
- getCloserPoint() : **Inshape::BoundingBox**
- getClosestPoint() : **Inshape::BoundingBox**
- getCoherence() : **Inshape::NormalEstimationPlus**
- getCoherencyPointNormals() : **Inshape::NormalEstimationPlus**
- getCutResult() : **Inshape::Cutter**
- getEigenVec() : **CCLib::HyperCloud**
- getFarerPoint() : **Inshape::BoundingBox**
- getHead() : **Inshape::TriSparse**
- getLAMDA() : **Inshape::TriSparse**
- getLength() : **Inshape::BoundingBox**
- getMap() : **Inshape::TriSparse**
- getNextPoint() : **CCLib::HyperCloud**
- getOBB() : **CCLib::HyperCloud** , **Inshape::BoundingBox**
- getOutput() : **Inshape::Poisson**
- getOutputMesh() : **Inshape::Inshaper**
- getPoint() : **CCLib::HyperCloud**
- getPointPersistentPtr() : **CCLib::HyperCloud**
- getPointScalarValue() : **CCLib::HyperCloud**
- getPointXYZ() : **CCLib::HyperCloud**
- getScalarField() : **CCLib::HyperCloud**

- h -

- HyperCloud() : **CCLib::HyperCloud**

- i -

- Inshaper() : **Inshape::Inshaper**
- invertNormal() :
Inshape::NormalEstimationPlus::WArc::PointerAndIdx
- is_true() : **Anyarg**
- isScalarFieldEnabled() : **CCLib::HyperCloud**

- n -

- normalCopy() :
Inshape::NormalEstimationPlus::WArc::PointerAndIdx
- NormalEstimationPlus() : **Inshape::NormalEstimationPlus**

- o -

- Option() : **Option**

- p -

- parse_argv() : **Anyarg** , **Inshape::Setting**
- placeliteratorAtBeginning() : **CCLib::HyperCloud**
- Plane() : **Inshape::Plane**
- pointNormForm() : **Inshape::Plane**
- Poisson() : **Inshape::Poisson**
- printParam() : **Inshape::TriSparse**

- r -

- readInputTemplate() : **Inshape::Setting**
- readMatch() : **Inshape::Setting**
- readParam() : **Inshape::Setting**
- reserve() : **CCLib::HyperCloud**
- resize() : **CCLib::HyperCloud**

- s -

- saveOBB() : **Inshape::BoundingBox**
- segmentCloud() : **Inshape::Plane**

- segmentMesh() : **Inshape::Plane**
- set_desc_meta() : **Option**
- set_value() : **Option**
- setA() : **Inshape::TriSparse**
- setB() : **Inshape::TriSparse**
- setBody() : **Inshape::Cutter**
- setCoherence() : **Inshape::NormalEstimationPlus**
- setDepth() : **Inshape::Poisson**
- setHead() : **Inshape::Cutter**
- setInput() : **Inshape::TriSparse**
- setInputBody() : **Inshape::Inshaper**
- setInputCloud() : **Inshape::Poisson**
- setInputTemplate() : **Inshape::Inshaper**
- setMatch() : **Inshape::Inshaper**
- setParam() : **Inshape::Inshaper** , **Inshape::TriSparse**
- setPointScalarValue() : **CCLib::HyperCloud**
- setPreAligned() : **Inshape::Inshaper**
- Setting() : **Inshape::Setting**
- size() : **CCLib::HyperCloud**
- solve() : **Inshape::TriSparse**
- solveToCloud() : **Inshape::TriSparse**
- solveTriSparse() : **Inshape::TriSparse**

- t -

- toString() : **Inshape::ParamItem**
- TriSparse() : **Inshape::TriSparse**

- u -

- unSetMatch() : **Inshape::Inshaper**
- updateHead() : **Inshape::TriSparse**
- updateMesh() : **Inshape::TriSparse**

- ~ -

- ~HyperCloud() : **CCLib::HyperCloud**
 - ~TriSparse() : **Inshape::TriSparse**
-

Inshaping 0.1

- b -

- bodyFilename : **Inshape::Setting**

- d -

- D : **Inshape::Plane**
- desc : **Option**

- e -

- edge : **Inshape::tri**
- eta : **Inshape::ParamItem**

- f -

- faceFilename : **Inshape::Setting**
- finalParam : **Inshape::Param**
- from : **Inshape::tri**

- g -

- globalIterator : **CCLib::HyperCloud**

- h -

- headFilename : **Inshape::Setting**

- i -

- index : **Inshape::NormalEstimationPlus::WArc::PointerAndIdx**

- l -

- lamda_max : **Inshape::ParamItem**
- lamda_min : **Inshape::ParamItem**
- letter : **Option**

- m -

- m_points : **CCLib::HyperCloud**
- m_scalarField : **CCLib::HyperCloud**
- m_validBB : **CCLib::HyperCloud**
- meta : **Option**

- n -

- name : **Option**
- Normal : **Inshape::Plane**

- o -

- outputFilename : **Inshape::Setting**
- outputIntermediate : **Inshape::Setting**

- p -

- p : **Inshape::NormalEstimationPlus::WArc::PointerAndIdx**
- param : **Inshape::Setting**
- paramItems : **Inshape::Param**
- preAligned : **Inshape::Setting**

- s -

- subBody : **Inshape::Setting**
- subHead : **Inshape::Setting**

- t -

- theta : **Inshape::ParamItem**
- threshold : **Inshape::ParamItem**
- to : **Inshape::tri**
- type : **Option**

- u -

- useMatch : **Inshape::Setting**

- v -

- value_bool : **Option**
- value_double : **Option**
- value_int : **Option**
- value_str : **Option**

Inshaping 0.1

- PointsContainer : **CCLib::HyperCloud**
 - Ptr : **CCLib::HyperCloud**
-

Generated by doxygen 1.8.14

Inshaping 0.1

- Position : **Inshape::Plane**

Generated by **doxygen** 1.8.14

Inshaping 0.1

- ABOVE : **Inshape::Plane**
- BELOW : **Inshape::Plane**

Generated by **doxygen** 1.8.14

Inshaping 0.1

- operator<< : **Inshape::Plane**

Generated by doxygen 1.8.14

Inshaping 0.1

File List

Here is a list of all files with brief descriptions:

[detail level 1 2]

| | |
|---------------------|--------------------------------|
| ▼ include | |
| AfterProcess.h | |
| anyarg.h | A simple option parser for C++ |
| Benchmark.h | |
| BoundingBox.h | |
| Common.h | |
| Convert.h | |
| ConvexHull.cpp | |
| ConvexHull.h | |
| FileIO.h | |
| HyperCloud.h | |
| Inshaper.h | |
| Normal.h | |
| Plane.h | |
| Poisson.h | |
| RigidRigsteration.h | |
| Setting.h | |
| TriSparse.h | |

Inshaping 0.1

include

include Directory Reference

Files

file [AfterProcess.h](#) [code]

file [anyarg.h](#) [code]
A simple option parser for C++.

file [Benchmark.h](#) [code]

file [BoundingBox.h](#) [code]

file [Common.h](#) [code]

file [Convert.h](#) [code]

file [ConvexHull.cpp](#)

file [ConvexHull.h](#) [code]

file [FileIO.h](#) [code]

file [HyperCloud.h](#) [code]

file [Inshaper.h](#) [code]

file [Normal.h](#) [code]

file [Plane.h](#) [code]

file [Poisson.h](#) [code]

file [RigidRigsteration.h](#) [code]

file [Setting.h](#) [code]

file [TriSparse.h](#) [code]

Inshaping 0.1

include

[Classes](#) | [Namespaces](#)

AfterProcess.h File Reference

```
#include <FileIO.h> #include <Plane.h>
#include <BoundingBox.h>
#include <pcl/common/centroid.h>
#include <ConvexHull.h>
#include <Poisson.h>
```

[Go to the source code of this file.](#)

Classes

```
class Inshape::Cutter
```

Namespaces

Inshape

Inshaping 0.1

include

Classes

anyarg.h File Reference

A simple option parser for C++. [More...](#)

```
#include <string> #include <vector>
#include <cstdlib>
#include <cstring>
```

[Go to the source code of this file.](#)

Classes

struct **Option**

Data structure of a program option. [More...](#)

class **Anyarg**

Use this class to define program options and parse command line arguments. [More...](#)

Detailed Description

A simple option parser for C++.

Author

Li Qibin (liqb036@gmail.com)

Version

0.12

Bug:

no bug found yet

Date

2014/01/28

Anyarg supports both single letter-options (like: -i) and long options (like: -help).

The mixup of short and long options in command line is allowed (like: -v -help).

Options can be given as:

```
--help
--long=value
--long value
-h
-ab
-c value
```

where -help, -h, -a, -b are boolean flags, option -long and -c take an argument.

Although options can mix up with non-option arguments, it is a good practice to place options before non-option arguments.

All non-option arguments are collected to a new array in the same order as they are in command line.

Argument – stops the parsing of command line arguments. All the following arguments are treated as non-option arguments.

A single hyphen - is parsed as an ordinary non-option arguments. It is usually used to specify input from standard input or output to the standard output.

Assign multiple values to an option is not allowed.

Inshaping 0.1

include

[Namespaces](#) | [Functions](#)

Benchmark.h File Reference

```
#include <TriSparse.h>
```

[Go to the source code of this file.](#)

Namespaces

Inshape

Functions

double **Inshape::avgE1** (TriSparse &triSprase)

double **Inshape::avgE3** (TriSparse &triSprase)

Generated by **doxygen** 1.8.14

Inshaping 0.1

include

[Classes](#) | [Namespaces](#) | [Functions](#)

BoundingBox.h File Reference

```
#include <Eigen/Dense> #include <vector>
#include <iostream>
#include <pcl/point_cloud.h>
#include <pcl/point_types.h>
#include <FileIO.h>
```

[Go to the source code of this file.](#)

Classes

struct **Inshape::tri**

class **Inshape::BoundingBox**

Namespaces

Inshape

Functions

bool **Inshape::compareNorm** (tri i, tri j)

Generated by doxygen 1.8.14

Inshaping 0.1

include

[Namespaces](#) | [Typedefs](#)

Common.h File Reference

```
#include <Eigen/Dense> #include <Eigen/Sparse>
#include <unordered_map>
#include <memory>
#include <pcl/common/centroid.h>
#include <pcl/point_cloud.h>
#include <pcl/point_types.h>
#include <pcl/surface/gp3.h>
#include <pcl/surface/poisson.h>
#include <OpenMesh/Core/IO/MeshIO.hh>
#include <OpenMesh/Core/Mesh/PolyMesh_ArrayKernelT.hh>
#include <HyperCloud.h>
#include <SimpleMesh.h>
```

[Go to the source code of this file.](#)

Namespaces

Inshape

Typedefs

`typedef OpenMesh::PolyMesh_ArrayKernelT Inshape::MyMesh`

`typedef std::shared_ptr< MyMesh > Inshape::MyMeshPtr`

`typedef std::shared_ptr< CCLib::SimpleMesh > Inshape::SimpleMeshF`

Generated by [doxygen](http://doxygen.org) 1.8.14

Inshaping 0.1

include

[Namespaces](#) | [Functions](#)

Convert.h File Reference

```
#include <TriSparse.h>
```

[Go to the source code of this file.](#)

Namespaces

Inshape

Inshape::Convert

Functions

void **Inshape::Convert::meshToCloud**
(MyMesh &head_mesh,
pcl::PointCloud< pcl::PointXYZ >
&head_cloud)

pcl::PointCloud< pcl::PointXYZ >::Ptr **Inshape::Convert::hyperToPoint**
(**CCLib::HyperCloud::Ptr** &hyper

CCLib::HyperCloud::Ptr **Inshape::Convert::cloudToHype**
(pcl::PointCloud< pcl::PointXYZ >
&cloud)

MyMeshPtr **Inshape::Convert::PolygonToMy**
(pcl::PolygonMesh &mesh)

Inshaping 0.1

include

[Namespaces](#) | [Functions](#)

ConvexHull.cpp File Reference

```
#include <ConvexHull.h>
```

Namespaces

Inshape

Functions

`pcl::PointCloud< pcl::PointXYZ >::Ptr` **Inshape::hyperToPointCloud**
(**CCLib::HyperCloud::Ptr** &hyper)

CCLib::HyperCloud::Ptr **Inshape::cloudToHyper**
(`pcl::PointCloud< pcl::PointXYZ >` &cloud)

`pcl::PolygonMesh::Ptr` **Inshape::getConvexHull**
(`pcl::PointCloud< pcl::PointXYZ >::Ptr` &cloud)

`pcl::PointCloud< pcl::PointXYZ >::Ptr` **Inshape::filterWithHull**
(`pcl::PointCloud< pcl::PointXYZ >::Ptr` &bodyshad,
`pcl::PointCloud< pcl::PointXYZ >::Ptr` &head, double threshold)

Inshaping 0.1

include

[Namespaces](#) | [Functions](#)

ConvexHull.h File Reference

```
#include <Common.h> #include <DistanceComputationTools.h>
#include <pcl/surface/convex_hull.h>
#include <Convert.h>
```

[Go to the source code of this file.](#)

Namespaces

Inshape

Inshape::ConvexHull

Functions

pcl::PolygonMesh::Ptr **Inshape::ConvexHull::getConve**
(pcl::PointCloud< pcl::PointXYZ >:
&cloud)

pcl::PointCloud< pcl::PointXYZ >::Ptr **Inshape::ConvexHull::filterWith**
(pcl::PointCloud< pcl::PointXYZ >:
&bodyshead, pcl::PointCloud<
pcl::PointXYZ >::Ptr &head, double
threshold)

Inshaping 0.1

include >

[Namespaces](#) | [Functions](#)

FileIO.h File Reference

```
#include <fstream> #include <string>
#include <Common.h>
#include <Normal.h>
```

[Go to the source code of this file.](#)

Namespaces

Inshape

Functions

bool **Inshape::readPly** (const std::string &filename, pcl::PointCloud< pcl::PointXYZ > &cloud)

bool **Inshape::savePly** (const std::string &filename, pcl::PointCloud< pcl::PointXYZ > &cloud)

bool **Inshape::savePolygonMesh** (std::string filename, pcl::PolygonMesh &mesh)

Inshaping 0.1

include

[Classes](#) | [Namespaces](#)

HyperCloud.h File Reference

```
#include "GenericChunkedArray.h" #include
"GenericIndexedCloudPersist.h"
#include "PointProjectionTools.h"
#include <Eigen/Dense>
#include <pcl/point_types.h>
#include <memory>
```

[Go to the source code of this file.](#)

Classes

class **CCLib::HyperCloud**
An Extended point cloud (with point duplication mechanism)
[More...](#)

Namespaces

CCLib

Generated by doxygen 1.8.14

Inshaping 0.1

include

[Classes](#) | [Namespaces](#)

Inshaper.h File Reference

```
#include <RigidRigsteration.h> #include <FileIO.h>
#include <Setting.h>
#include <TriSparse.h>
#include <AfterProcess.h>
#include <iostream>
#include <Benchmark.h>
#include <Normal.h>
#include <Poisson.h>
#include <Common.h>
```

[Go to the source code of this file.](#)

Classes

class **Inshape::Inshaper**

Namespaces

Inshape

Generated by doxygen 1.8.14

Inshaping 0.1

include

[Classes](#) | [Namespaces](#)

Normal.h File Reference

```
#include <iostream> #include <pcl/visualization/pcl_visualizer.h>
#include <pcl/search/kdtree.h>
#include <pcl/features/normal_3d.h>
#include <pcl/kdtree/kdtree_flann.h>
```

[Go to the source code of this file.](#)

Classes

class **Inshape::NormalEstimationPlus**

struct **Inshape::NormalEstimationPlus::WArc::PointerAndIdx**

Namespaces

Inshape

Generated by doxygen 1.8.14

Inshaping 0.1

include

[Classes](#) | [Namespaces](#)

Plane.h File Reference

```
#include <Eigen/Dense> #include <iostream>
#include <vector>
#include <algorithm>
#include <BoundingBox.h>
#include <TriSparse.h>
```

[Go to the source code of this file.](#)

Classes

class **Inshape::Plane**

Namespaces

Inshape

Inshaping 0.1

include

[Classes](#) | [Namespaces](#)

Poisson.h File Reference

```
#include <pcl/point_types.h> #include <pcl/kdtree/kdtree_flann.h>
#include <pcl/features/normal_3d_omp.h>
#include <pcl/features/normal_3d.h>
#include <pcl/surface/gp3.h>
#include <pcl/surface/poisson.h>
#include <Normal.h>
#include <Common.h>
#include <algorithm>
#include <Convert.h>
```

[Go to the source code of this file.](#)

Classes

```
class Inshape::Poisson
```

Namespaces

Inshape

Inshaping 0.1

include

Namespaces | Functions

RigidRigsteration.h

File Reference

```
#include <TriSparse.h> #include  
<pcl/registration/transformation_estimation_svd.h>  
#include  
<pcl/registration/correspondence_rejection_sample_consensus.h>  
#include <pcl/registration/correspondence_estimation.h>  
#include <pcl/registration/icp.h>  
#include <Normal.h>  
#include <RegistrationTools.h>  
#include <HyperCloud.h>  
#include <iostream>  
#include <unordered_map>
```

[Go to the source code of this file.](#)

Namespaces

Inshape

Functions

Eigen::Matrix4f **Inshape::ICPRigsteration** (pcl::PointCloud< pcl::PointXYZ >::Ptr &data, pcl::PointCloud< pcl::PointXYZ >::Ptr &model)

Eigen::Matrix4f **Inshape::NICPRigsteration** (pcl::PointCloud< pcl::PointXYZ >::Ptr &data, pcl::PointCloud< pcl::PointXYZ >::Ptr &model)

Eigen::Matrix4f **Inshape::SVDRegistration** (pcl::PointCloud< pcl::PointXYZ >::Ptr &sub_data, pcl::PointCloud< pcl::PointXYZ >::Ptr &sub_model)

void **Inshape::AlignWithFace** (pcl::PointCloud< pcl::PointXYZ >::Ptr &body, pcl::PointCloud< pcl::PointXYZ >::Ptr &head_cloud, pcl::PointCloud< pcl::PointXYZ >::Ptr &face)

void **Inshape::rigidRigsterWithoutMatch** (pcl::PointCloud< pcl::PointXYZ >::Ptr &body, MyMesh &head_mesh, pcl::PointCloud< pcl::PointXYZ >::Ptr &face)

void **Inshape::rigidRigsterWithMatch** (pcl::PointCloud< pcl::PointXYZ >::Ptr &body, MyMesh &head_mesh, pcl::PointCloud< pcl::PointXYZ >::Ptr &face, pcl::PointCloud< pcl::PointXYZ >::Ptr &sub_head, pcl::PointCloud< pcl::PointXYZ >::Ptr &sub_body)

Inshaping 0.1

include

[Classes](#) | [Namespaces](#) | [Macros](#)

Setting.h File Reference

```
#include <iostream> #include <fstream>
#include <string>
#include <vector>
#include <anyarg.h>
```

[Go to the source code of this file.](#)

Classes

struct **Inshape::ParamItem**

struct **Inshape::Param**

class **Inshape::Setting**

Namespaces

Inshape

Macros

```
#define HEAD_FILENAME "HEAD_FILENAME"
```

```
#define BODY_FILENAME "BODY_FILENAME"
```

```
#define SUB_HEAD "SUB_HEAD"
```

```
#define SUB_BODY "SUB_BODY"
```

```
#define FACE_FILENAME "FACE_FILENAME"
```

```
#define FINAL_PARAM "FINAL_PARAM"
```

Inshaping 0.1

include

[Classes](#) | [Namespaces](#)

TriSparse.h File Reference

```
#include <string> #include <pcl/kdtree/kdtree_flann.h>
#include <Setting.h>
#include <Normal.h>
#include <Common.h>
#include <Convert.h>
```

[Go to the source code of this file.](#)

Classes

class **Inshape::TriSparse**

Namespaces

Inshape

Inshaping 0.1

Here is a list of all file members with links to the files they belong to:

- BODY_FILENAME : [Setting.h](#)
- FACE_FILENAME : [Setting.h](#)
- FINAL_PARAM : [Setting.h](#)
- HEAD_FILENAME : [Setting.h](#)
- SUB_BODY : [Setting.h](#)
- SUB_HEAD : [Setting.h](#)

Inshaping 0.1

- BODY_FILENAME : [Setting.h](#)
- FACE_FILENAME : [Setting.h](#)
- FINAL_PARAM : [Setting.h](#)
- HEAD_FILENAME : [Setting.h](#)
- SUB_BODY : [Setting.h](#)
- SUB_HEAD : [Setting.h](#)

Inshaping 0.1

Inshape > Cutter >

Inshape::Cutter Member List

This is the complete list of members for **Inshape::Cutter**, including all inherited members.

checkBodyWithHead(pcl::PointCloud< pcl::PointXYZ >::Ptr &bodyshead
Cutter(pcl::PointCloud< pcl::PointXYZ >::Ptr &_body, MyMeshPtr _head)
getCutResult(double threshold, double percent)
setBody(pcl::PointCloud< pcl::PointXYZ >::Ptr &_body)
setHead(MyMeshPtr _head)

Generated by [doxygen](#) 1.8.14

Inshaping 0.1

include

AfterProcess.h

Go to the documentation of this file.

```
1  #pragma once
2  #include<FileIO.h>
3  #include<Plane.h>
4  #include<BoundingBox.h>
5  #include<pcl/common/centroid.h>
6  #include<ConvexHull.h>
7  #include<Poisson.h>
8
9  namespace Inshape
10 {
13     class Cutter
14     {
15     private:
16         pcl::PointCloud < pcl::PointXYZ> ::Ptr
            body_ptr;
17         MyMeshPtr head_ptr;
18
19     public:
20
21         Cutter(pcl::PointCloud<pcl::PointXYZ>::Ptr&
            _body, MyMeshPtr
            _head):body_ptr(_body),head_ptr(_head){}
22
23         void
            setBody(pcl::PointCloud<pcl::PointXYZ>::Ptr&
            _body)
24         {
25             body_ptr = _body;
26         }
```

```
26     void setHead(MyMeshPtr _head)
27     {
28         head_ptr = _head;
29     }
30
31
32     pcl::PointCloud<pcl::PointXYZ>::Ptr
    getCutResult(double threshold, double percent);
33
34     static
    pcl::PointCloud<pcl::PointXYZ>::Ptr
    checkBodyWithHead(pcl::PointCloud<pcl::PointXYZ>
        ::Ptr & bodyshead,
        pcl::PointCloud<pcl::PointXYZ>::Ptr & head,
        double threshold);
35
36
37     };
39 }
```

Inshaping 0.1

Inshape > tri >

Inshape::tri Member List

This is the complete list of members for **Inshape::tri**, including all inherited members.

edge Inshape::tri

from Inshape::tri

to Inshape::tri

Inshaping 0.1

include

BoundingBox.h

Go to the documentation of this file.

```
1 #pragma once
2 #include<Eigen/Dense>
3 #include<vector>
4 #include<iostream>
5 #include<pcl/point_cloud.h>
6 #include<pcl/point_types.h>
7 #include<FileIO.h>
8
9 namespace Inshape
10 {
11
12     struct tri
13     {
14         Eigen::Vector3d edge;
15         int from;
16         int to;
17     };
18
19     bool compareNorm(tri i, tri j);
20
21     class BoundingBox
22     {
23     private:
24         std::vector<Eigen::Vector3d>
25         vertices; // Eigen::Vector3d length;
26         // Eigen::Vector3d width; // Eigen::Vector3d height;
27         Eigen::Vector3d width; // Eigen::Vector3d height;
28         Eigen::Vector3d closer_point; // Eigen::Vector3d farer_point;
29     };
30 }
```

```

public:

BoundingBox(std::vector<Eigen::Vector3d>&
_vertices) :vertices(_vertices) {}

BoundingBox(pcl::PointCloud<pcl::PointXYZ>::Ptr&
cloud) :vertices(getOBB(cloud)) {}
    Eigen::Vector3d getLength();

    Eigen::Vector3d getCloserPoint();

    Eigen::Vector3d
getClosestPoint(Eigen::Vector3d p);

    Eigen::Vector3d getFarerPoint();
    //
    Eigen::Vector3d::Zero()
    Eigen::Vector3d length;
    void computeNormals(Eigen::Vector3d
head_baryCenter);

    void saveOBB(std::string filename);

    static std::vector<Eigen::Vector3d>
getOBB(pcl::PointCloud<pcl::PointXYZ>::Ptr&
cloud);

};
/*@}*/
}
26 | Eigen::Vector3d length; //
    Eigen::Vector3d width; //
    Eigen::Vector3d height; //

```

```

        Eigen::Vector3d closer_point; //
        Eigen::Vector3d farer_point;

    public:

    BoundingBox(std::vector<Eigen::Vector3d>&
        _vertices) :vertices(_vertices) {}

    BoundingBox(pcl::PointCloud<pcl::PointXYZ>::Ptr&
        cloud) :vertices(getOBB(cloud)) {}
        Eigen::Vector3d getLength();

        Eigen::Vector3d getCloserPoint();

        Eigen::Vector3d
        getClosestPoint(Eigen::Vector3d p);

        Eigen::Vector3d getFarerPoint();
        //
        Eigen::Vector3d::Zero()
        void computeNormals(Eigen::Vector3d
        head_baryCenter);

        void saveOBB(std::string filename);

        static std::vector<Eigen::Vector3d>
        getOBB(pcl::PointCloud<pcl::PointXYZ>::Ptr&
        cloud);

    };
    /*@}*/
}
27 | Eigen::Vector3d width; //
Eigen::Vector3d height; //

```

```

        Eigen::Vector3d closer_point; //
        Eigen::Vector3d farer_point;

    public:

    BoundingBox(std::vector<Eigen::Vector3d>&
        _vertices) :vertices(_vertices) {}

    BoundingBox(pcl::PointCloud<pcl::PointXYZ>::Ptr&
        cloud) :vertices(getOBB(cloud)) {}
        Eigen::Vector3d getLength();

        Eigen::Vector3d getCloserPoint();

        Eigen::Vector3d
        getClosestPoint(Eigen::Vector3d p);

        Eigen::Vector3d getFarerPoint();
        //
        Eigen::Vector3d::Zero()
        void computeNormals(Eigen::Vector3d
        head_baryCenter);

        void saveOBB(std::string filename);

        static std::vector<Eigen::Vector3d>
        getOBB(pcl::PointCloud<pcl::PointXYZ>::Ptr&
        cloud);

    };
    /*@}*/
}
    Eigen::Vector3d height; //
    Eigen::Vector3d closer_point; //

```

```

        Eigen::Vector3d farer_point;

    public:

    BoundingBox(std::vector<Eigen::Vector3d>&
        _vertices) :vertices(_vertices) {}

    BoundingBox(pcl::PointCloud<pcl::PointXYZ>::Ptr&
        cloud) :vertices(getOBB(cloud)) {}
        Eigen::Vector3d getLength();

        Eigen::Vector3d getCloserPoint();

        Eigen::Vector3d
        getClosestPoint(Eigen::Vector3d p);

        Eigen::Vector3d getFarerPoint();
        //
        Eigen::Vector3d::Zero()
        void computeNormals(Eigen::Vector3d
        head_baryCenter);

        void saveOBB(std::string filename);

        static std::vector<Eigen::Vector3d>
        getOBB(pcl::PointCloud<pcl::PointXYZ>::Ptr&
        cloud);

};
/*@}*/
}
29 | Eigen::Vector3d closer_point;//
    |
30 | Eigen::Vector3d farer_point;

```

```

31
32     public:
33
34     BoundingBox(std::vector<Eigen::Vector3d>&
    _vertices) :vertices(_vertices) {}
35
36     BoundingBox(pcl::PointCloud<pcl::PointXYZ>::Ptr&
    cloud) :vertices(getOBB(cloud)) {}
37
38     Eigen::Vector3d getLength();
39
40     Eigen::Vector3d getCloserPoint();
41
42     Eigen::Vector3d
    getClosestPoint(Eigen::Vector3d p);
43
44     Eigen::Vector3d getFarerPoint();
45
46     //
47     Eigen::Vector3d::Zero()
    void computeNormals(Eigen::Vector3d head_baryCenter);
48
49     void saveOBB(std::string filename);
50
51     static std::vector<Eigen::Vector3d>
    getOBB(pcl::PointCloud<pcl::PointXYZ>::Ptr&
    cloud);
52
53     };
54     /*@}*/
55
56 }
57
58 void computeNormals(Eigen::Vector3d
    head_baryCenter);
59
60 void saveOBB(std::string filename);
61
62

```

```
48 |
49 |         static std::vector<Eigen::Vector3d>
    |         getOBB(pcl::PointCloud<pcl::PointXYZ>::Ptr&
    |         cloud);
50 |
51 |     };
53 | }
```

Inshaping 0.1

Inshape > BoundingBox >

Inshape::BoundingBox Member List

This is the complete list of members for **Inshape::BoundingBox**, including all inherited members.

| | |
|---|-------------------|
| BoundingBox (std::vector< Eigen::Vector3d > &_vertices) | Inshape::B |
| BoundingBox (pcl::PointCloud< pcl::PointXYZ >::Ptr &cloud) | Inshape::B |
| computeNormals (Eigen::Vector3d head_baryCenter) | Inshape::B |
| getCloserPoint () | Inshape::B |
| getClosestPoint (Eigen::Vector3d p) | Inshape::B |
| getFarerPoint () | Inshape::B |
| getLength () | Inshape::B |
| getOBB (pcl::PointCloud< pcl::PointXYZ >::Ptr &cloud) | Inshape::B |
| saveOBB (std::string filename) | Inshape::B |

Inshaping 0.1

Inshape > Inshaper >

Inshape::Inshaper Member List

This is the complete list of members for **Inshape::Inshaper**, including all inherited members.

getOutputMesh()

Inshaper()

setInputBody(pcl::PointCloud< pcl::PointXYZ >::Ptr _body)

setInputTemplate(MyMeshPtr &_head, pcl::PointCloud< pcl::PointXYZ >

setMatch(pcl::PointCloud< pcl::PointXYZ >::Ptr &_subBody, pcl::PointClo

setParam(Param &_param)

setPreAligned(bool _preAligned)

unSetMatch()

Inshaping 0.1

include

Inshaper.h

[Go to the documentation of this file.](#)

```
1  #pragma once
2  #include<RigidRigsteration.h>
3  #include<FileIO.h>
4  #include<Setting.h>
5  #include<TriSparse.h>
6  #include<AfterProcess.h>
7  #include<iostream>
8  #include<Benchmark.h>
9  #include<Normal.h>
10 #include<Poisson.h>
11 #include<Common.h>
12
13 namespace Inshape
14 {
15     /*
16     Input:
17     MyMeshPtr head;
18     pcl::PointCloud<pcl::PointXYZ>::Ptr
19     face;
20
21
22     pcl::PointCloud<pcl::PointXYZ>::Ptr
23     body;
24
25
26     pcl::PointCloud<pcl::PointXYZ>::Ptr
27     subBody;
28     pcl::PointCloud<pcl::PointXYZ>::Ptr
29     subHead;
```

```

28
29     Param param;
30
31     Output:
32     MyMeshPtr merged;
33
34     */
35
36     class Inshaper
37     {
38     private:
39         MyMeshPtr head;
40         pcl::PointCloud<pcl::PointXYZ>::Ptr
    face;
41
42
43         pcl::PointCloud<pcl::PointXYZ>::Ptr
    body;
44
45
46         pcl::PointCloud<pcl::PointXYZ>::Ptr
    subBody;
47         pcl::PointCloud<pcl::PointXYZ>::Ptr
    subHead;
48
49         Param param;
50
51
52         bool preAligned; //0p0π0t000 true
    000000000000000000
53     public:
54
55         Inshaper() :
56             head(nullptr),
57             face(nullptr),
58             body(nullptr),
59             subBody(nullptr),

```

```

60         subHead(nullptr),
61         preAligned(false)
62     {
63
64     }
65
66     void setMatch
67     (
68
69         pcl::PointCloud<pcl::PointXYZ>::Ptr& _subBody,
70         pcl::PointCloud<pcl::PointXYZ>::Ptr& _subHead)
71     {
72         subBody = _subBody;
73         subHead = _subHead;
74     }
75     //call the function if you don't want
76     to use match to apply rigid registration
77     void unSetMatch()
78     {
79         subBody = nullptr;
80         subHead = nullptr;
81     }
82
83     void
84     setInputBody(pcl::PointCloud<pcl::PointXYZ>::Ptr
85     _body)
86     {
87         body = _body;
88     }
89     void setInputTemplate(MyMeshPtr&
90     _head, pcl::PointCloud<pcl::PointXYZ>::Ptr _face)
91     {
92         head = _head;

```

```
91         face = _face;
92     }
93
94
95     void setParam(Param& _param)
96     {
97         param = _param;
98     }
99
100     //set preAligned to be true if rigid
    registration is not needed
101     void setPreAligned(bool _preAligned)
102     {
103         preAligned = _preAligned;
104     }
105
106
107     MyMeshPtr getOutputMesh();
108
109 };
112 }
```

Inshaping 0.1

Inshape > NormalEstimationPlus >

Inshape::NormalEstimationPlus Member List

This is the complete list of members for **Inshape::NormalEstimationPlus**, including all inherited members.

AddNeighboursToHeap(pcl::PointCloud< pcl::PointNormal >::Ptr cloud_
coherencyPass(pcl::PointCloud< pcl::PointNormal >::Ptr cloud_with_nor
getCoherence()
getCoherencyPointNormals()
NormalEstimationPlus()
setCoherence(int _coherence)

Generated by [doxygen](http://doxygen.org) 1.8.14

Inshaping 0.1

include

Normal.h

[Go to the documentation of this file.](#)

```
1  #pragma once
2  #include<iostream>
3  #include<pcl/visualization/pcl_visualizer.h>
4  #include<pcl/search/kdtree.h>
5  #include<pcl/features/normal_3d.h>
6  #include <pcl/kdtree/kdtree_flann.h>
7
8  namespace Inshape
9  {
12     class NormalEstimationPlus : public
        pcl::NormalEstimation<pcl::PointXYZ,
        pcl::Normal>
13     {
14
15     private:
16         int coherence;
17
18         class WArc
19         {
20         public:
21             struct PointerAndIdx
22             {
23                 pcl::PointNormal* p;
24                 int index;
25
26                 //return a copy of
27                 Eigen::Vector3f normalCopy()
28                 {
29                     return p-
```

```

    >getNormalVector3fMap();
30 |         }
31 |
32 |         void invertNormal()
33 |         {
34 |             p->normal_x = -p-
    >normal_x;
35 |             p->normal_y = -p-
    >normal_y;
36 |             p->normal_z = -p-
    >normal_z;
37 |         }
38 |     };
39 |
40 |     WArc(pcl::PointNormal& _s, int is,
    pcl::PointNormal& _t, int it) :src({ &_s, is }),
    trg({ &_t, it }), w(0)
41 |     {
42 |         w =
    fabs(src.normalCopy().dot(trg.normalCopy()));
43 |     }
44 |
45 |     PointerAndIdx src;
46 |     PointerAndIdx trg;
47 |     float w;
48 |     bool operator< (const WArc &a)
    const { return w < a.w; }
49 |     };
50 |
51 |
52 |     public:
53 |     NormalEstimationPlus()
    :pcl::NormalEstimation<pcl::PointXYZ,
    pcl::Normal>(), coherence(8) {}
54 |
55 |     //????h????f???? void
    setCoherence(int _coherence)//

```

```

    {
        coherence = _coherence;
    }

    //0000h000F000
int getCoherence()
{
    return coherence;
}

    pcl::PointCloud<pcl::PointNormal>::Ptr
getCoherencyPointNormals();

    static void
AddNeighboursToHeap(pcl::PointCloud<pcl::PointNormal>::Ptr cloud_with_normals, int index, int K,
pcl::search::KdTree<pcl::PointNormal>::Ptr
&tree, std::vector<WArc> &heap,
std::shared_ptr<bool>& visited);

    static void
coherencyPass(pcl::PointCloud<pcl::PointNormal>:
:Ptr cloud_with_normals, int coherence = 8);

};
/*@}*/
}
56 void setCoherence(int _coherence)//
57 {
58     coherence = _coherence;
59 }
60
61 //0000h000F000 int getCoherence()
{
    return coherence;
}

```

```

        pcl::PointCloud<pcl::PointNormal>::Ptr
getCoherencyPointNormals();

        static void
AddNeighboursToHeap(pcl::PointCloud<pcl::PointNormal>::Ptr cloud_with_normals, int index, int K,
pcl::search::KdTree<pcl::PointNormal>::Ptr
&tree, std::vector<WArc> &heap,
std::shared_ptr<bool>& visited);

        static void
coherencyPass(pcl::PointCloud<pcl::PointNormal>:
:Ptr cloud_with_normals, int coherence = 8);

};
/*@}*/
}
62         int getCoherence()
63         {
64             return coherence;
65         }
66
67         pcl::PointCloud<pcl::PointNormal>::Ptr
getCoherencyPointNormals();
68
69         static void
AddNeighboursToHeap(pcl::PointCloud<pcl::PointNormal>::Ptr cloud_with_normals, int index, int K,
pcl::search::KdTree<pcl::PointNormal>::Ptr
&tree, std::vector<WArc> &heap,
std::shared_ptr<bool>& visited);
70
71         static void
coherencyPass(pcl::PointCloud<pcl::PointNormal>:
:Ptr cloud_with_normals, int coherence = 8);
72
73     };

```


Inshaping 0.1

Inshape > Plane >

Inshape::Plane Member List

This is the complete list of members for **Inshape::Plane**, including all inherited members.

ABOVE enum value

BELOW enum value

belowPlane(Eigen::Vector3d point, double offset)

D

FromThreePt(std::vector< Eigen::Vector3d > &ptArr)

Normal

operator<<(std::ostream &os, const Plane &plane)

Plane()

Plane(double a, double b, double c, double d)

Plane(Eigen::Vector3d normal, Eigen::Vector3d point)

Plane(std::vector< Eigen::Vector3d > &ptArr)

pointNormForm(Eigen::Vector3d normal, Eigen::Vector3d point)

Position enum name

segmentCloud(pcl::PointCloud< pcl::PointXYZ > &cloud, pcl::PointCloud

segmentMesh(MyMesh &mesh, MyMesh &output, Position position, doul

Inshaping 0.1

include

Plane.h

Go to the documentation of this file.

```
1 #pragma once
2 #include<Eigen/Dense>
3 #include<iostream>
4 #include<vector>
5 #include<algorithm>
6 #include<BoundingBox.h>
7 #include<TriSparse.h>
8
9 namespace Inshape
10 {
11     class Plane
12     {
13     public:
14         enum Position
15         {
16             BELOW, ABOVE
17         };
18
19         Eigen::Vector3d Normal;
20         double D;
21
22         Plane()
23         :Normal(Eigen::Vector3d::Zero()), D(0) {}
24
25         //??'
26         Plane(double a, double b, double c,
27             double d)
28         {
29             Normal.x() = a;
```

```

30         Normal.y() = b;
31         Normal.z() = c;
32         D = d;
33     }
34
35     // 燿' 5 涇  inline
Plane(Eigen::Vector3d normal, Eigen::Vector3d
point) { pointNormForm(normal, point); }

// 得 5
inline
Plane(std::vector<Eigen::Vector3d>& ptArr) {
FromThreePt(ptArr); }

// 0i 0000000h
bool belowPlane(Eigen::Vector3d point,
double offset);

// 000ö 0000000D' match 000x0minMatch
//bool
adjustToTargetMatch(pcl::PointCloud<pcl::PointXY
Z>& body, pcl::PointCloud<pcl::PointXYZ>&
head, int traget_match, double threshold);

// 5 00000 POSITON
0000ç0000j00zj0λ00
void
segmentCloud(pcl::PointCloud<pcl::PointXYZ>&
cloud, pcl::PointCloud<pcl::PointXYZ>& output,
Position position, double offset);

// 5 00000 POSITON
0000ç0000j00zj0λ00
void segmentMesh(MyMesh& mesh, MyMesh&
output, Position position, double offset = 0);

```

```

        //燿
        bool pointNormForm(Eigen::Vector3d
normal, Eigen::Vector3d point);

        //jhh5
        bool
FromThreePt(std::vector<Eigen::Vector3d>&
ptArr);

        inline friend std::ostream&
operator<<(std::ostream& os, const Plane& plane)
        {
            os << "A=" << plane.Normal.x() <<
",B=" << plane.Normal.y() << ",C=" <<
plane.Normal.z() << ",D=" << plane.D <<
std::endl;

            return os;
        }

};
/*@}*/
}
36|         inline Plane(Eigen::Vector3d normal,
Eigen::Vector3d point) { pointNormForm(normal,
point); }
37|
38|         //hh5
39|         inline
Plane(std::vector<Eigen::Vector3d>& ptArr) {
FromThreePt(ptArr); }
40|
41|         //hh          bool
belowPlane(Eigen::Vector3d point, double
offset);

        //ö?D'matchx0minMatch
//bool

```

```
adjustToTargetMatch(pcl::PointCloud<pcl::PointXYZ>& body, pcl::PointCloud<pcl::PointXYZ>& head, int target_match, double threshold);
```

```
        // 5  POSITON  
        Çjzλ
```

```
        void  
segmentCloud(pcl::PointCloud<pcl::PointXYZ>& cloud, pcl::PointCloud<pcl::PointXYZ>& output, Position position, double offset);
```

```
        // 5  POSITON  
        Çjzλ
```

```
        void segmentMesh(MyMesh& mesh, MyMesh& output, Position position, double offset = 0);
```

```
        // 燿'  
        bool pointNormForm(Eigen::Vector3d normal, Eigen::Vector3d point);
```

```
        // jh5  
        bool  
FromThreePt(std::vector<Eigen::Vector3d>& ptArr);
```

```
        inline friend std::ostream&  
operator<<(std::ostream& os, const Plane& plane)  
        {  
            os << "A=" << plane.Normal.x() <<  
            ",B=" << plane.Normal.y() << ",C=" <<  
            plane.Normal.z() << ",D=" << plane.D <<  
            std::endl;  
            return os;  
        }
```

```
};
```

```

        /*@}*/
    }
42 |         bool belowPlane(Eigen::Vector3d point,
        double offset);
43 |
44 |         //000ö0000000D00'match000x0minMatch
45 |         //bool
        adjustToTargetMatch(pcl::PointCloud<pcl::PointXYZ
        Z>& body, pcl::PointCloud<pcl::PointXYZ>&
        head, int target_match, double threshold);
46 |
47 |
48 |         //0000500000 POSITON
        0000ç0000j00zj0λ00
49 |         void
        segmentCloud(pcl::PointCloud<pcl::PointXYZ>&
        cloud, pcl::PointCloud<pcl::PointXYZ>& output,
        Position position, double offset);
50 |
51 |         //0000500000 POSITON
        0000ç0000j00zj0λ00
52 |         void segmentMesh(MyMesh& mesh, MyMesh&
        output, Position position, double offset = 0);
53 |
54 |         //0燦'
55 |         bool pointNormForm(Eigen::Vector3d
        normal, Eigen::Vector3d point);
56 |
57 |         //0000j00h00s00
58 |         bool
        FromThreePt(std::vector<Eigen::Vector3d>&
        ptArr);
59 |
60 |         inline friend std::ostream&
        operator<<(std::ostream& os, const Plane& plane)
61 |         {
62 |             os << "A=" << plane.Normal.x() <<

```

```
" ,B=" << plane.Normal.y() << " ,C=" <<  
plane.Normal.z() << " ,D=" << plane.D <<  
std::endl;
```

```
63 |         return os;
```

```
64 |     }
```

```
65 |
```

```
66 |     };
```

```
68 | }
```

Inshaping 0.1

Inshape > Poisson >

Inshape::Poisson Member List

This is the complete list of members for **Inshape::Poisson**, including all inherited members.

| | |
|---|-----|
| getOutput() | Ins |
| getOutput (pcl::PointCloud< pcl::PointNormal >::Ptr &cloud, int depth) | Ins |
| Poisson() | Ins |
| Poisson (pcl::PointCloud< pcl::PointXYZ >::Ptr &cloud) | Ins |
| setDepth (int _depth) | Ins |
| setInputCloud (pcl::PointCloud< pcl::PointXYZ >::Ptr &cloud) | Ins |

Generated by [doxygen](#) 1.8.14

Inshaping 0.1

include

Poisson.h

Go to the documentation of this file.

```
1 #pragma once
2 //000000, 0n010
3 #include <pcl/point_types.h>
4 //kd00
5 #include <pcl/kdtree/kdtree_flann.h>
6 //000000d
7 #include <pcl/features/normal_3d_omp.h>
8 #include <pcl/features/normal_3d.h>
9 //0ε0
10 #include <pcl/surface/gp3.h>
11 #include <pcl/surface/poisson.h>
12 #include<Normal.h>
13 #include<Common.h>
14 #include<algorithm>
15 #include<Convert.h>
16
17 namespace Inshape
18 {
19
21     class Poisson
22     {
23     private:
24         pcl::PointCloud<pcl::PointXYZ>::Ptr
input;
25         int depth;
26
27     public:
28         Poisson()
29         {
30
```

```

31     }
32
    Poisson(pcl::PointCloud<pcl::PointXYZ>::Ptr&
    cloud) :input(cloud), depth(9)
33     {
34
35     }
36     void setDepth(int _depth)
37     {
38         depth = _depth;
39     }
40     void
    setInputCloud(pcl::PointCloud<pcl::PointXYZ>::Ptr&
    r& cloud)
41     {
42         input = cloud;
43     }
44
45     MyMeshPtr getOutput();
46
47     static MyMeshPtr
    getOutput(pcl::PointCloud<pcl::PointNormal>::Ptr
    & cloud, int depth);
48
49
50
51     };
54 }

```

Inshaping 0.1

CCLib > HyperCloud

CCLib::HyperCloud Member List

This is the complete list of members for **CCLib::HyperCloud**, including all inherited members.

| | |
|--|-----|
| addCloud (CCLib::HyperCloud &cloud) | CCL |
| addPoint (const CCVector3 &P) | CCL |
| addPoint (const Eigen::Vector3d &p) | CCL |
| addPoint (const PointCoordinateType P[]) | CCL |
| applyTransformation (PointProjectionTools::Transformation &trans) | CCL |
| clear () | CCL |
| enableScalarField () | CCL |
| forEach (genericPointAction action) | CCL |
| getBaryCenter () | CCL |
| getBoundingBox (CCVector3 &bbMin, CCVector3 &bbMax) | CCL |
| getEigenVec (unsigned index) | CCL |
| getNextPoint () | CCL |
| getOBB () | CCL |
| getPoint (unsigned index) | CCL |
| getPoint (unsigned index, CCVector3 &P) const | CCL |
| getPointPersistentPtr (unsigned index) | CCL |
| getPointScalarValue (unsigned pointIndex) const | CCL |
| getPointXYZ (unsigned index) | CCL |
| getScalarField () | CCL |
| getScalarField () const | CCL |
| globalIterator | CCL |
| HyperCloud () | CCL |

| | |
|--|-----|
| isScalarFieldEnabled() const | CCL |
| m_points | CCL |
| m_scalarField | CCL |
| m_validBB | CCL |
| placeliteratorAtBeginning() | CCL |
| PointsContainer typedef | CCL |
| Ptr typedef | CCL |
| reserve (unsigned n) | CCL |
| resize (unsigned n) | CCL |
| setPointScalarValue (unsigned pointIndex, ScalarType value) | CCL |
| size() const | CCL |
| ~HyperCloud() | CCL |

Inshaping 0.1

include

HyperCloud.h

[Go to the documentation of this file.](#)

```
1  /*
2  @author:Elliott Zheng
3  @date: 2018/4/12
4  */
5
6  #ifndef HYPER_CLOUD_HEADER
7  #define HYPER_CLOUD_HEADER
8
9  //Local
10 #include "GenericChunkedArray.h"
11 #include "GenericIndexedCloudPersist.h"
12 #include "PointProjectionTools.h"
13 #include <Eigen/Dense>
14 #include<pcl/point_types.h>
15 #include<memory>
16
17 namespace CCLib
18 {
19
20     class PointsContainer;
21     class ScalarField;
22
23
24
25
26
27
28
29     class CC_CORE_LIB_API HyperCloud : public
GenericIndexedCloudPersist
30     {
31     public:
32         typedef std::shared_ptr<HyperCloud>
Ptr;
```

```

33 |
34 |
35 |     HyperCloud();
36 |
37 |
38 |     virtual ~HyperCloud();
39 |
40 |     /**** inherited form GenericCloud
    ****//
41 |         virtual unsigned size() const;
42 |         virtual void
    forEach(genericPointAction action);
43 |
44 |         virtual void getBoundingBox(CCVector3&
    bbMin, CCVector3& bbMax);
45 |         virtual std::vector<Eigen::Vector3d>
    getOBB();
46 |         virtual void
    placeIteratorAtBeginning();
47 |         virtual const CCVector3*
    getNextPoint();
48 |         virtual bool enableScalarField();
49 |         virtual bool isScalarFieldEnabled()
    const;
50 |         virtual void
    setPointScalarValue(unsigned pointIndex,
    ScalarType value);
51 |         virtual ScalarType
    getPointScalarValue(unsigned pointIndex) const;
52 |
53 |         /**** inherited form
    GenericIndexedCloud ****//
54 |         virtual const CCVector3*
    getPoint(unsigned index) { return
    getPointPersistentPtr(index); }
55 |
56 |
57 |         /**** 000L0d00*****//
58 |

```

59

```

//Eigen::Vector3d
inline
virtual const Eigen::Vector3d
getEigenVec(unsigned index);

inline virtual const pcl::PointXYZ
getPointXYZ(unsigned index);

//
Eigen::Vector3d getBaryCenter();

void addCloud(CCLib::HyperCloud& cloud);

virtual void getPoint(unsigned index,
CCVector3& P) const;

//**** inherited form
GenericIndexedCloudPersist ****//
virtual const CCVector3*
getPointPersistentPtr(unsigned index);

//! Clears cloud
void clear();

//! Point insertion mechanism
/** The point data will be duplicated in
memory.
\param P the point to insert
**/
virtual void addPoint(const CCVector3
&P);

virtual void addPoint(const
Eigen::Vector3d& p);

//! Point insertion mechanism
/** The point data will be duplicated in

```

```

memory.
    \param P the point to insert (as a 3-
size array)
    **/

    virtual void addPoint(const
PointCoordinateType P[]);

    //! Reserves some memory for hosting the
points
    /** \param n the number of points
    **/
    virtual bool reserve(unsigned n);

    //! Presets the size of the vector used
to store the points
    /** \param n the number of points
    **/
    virtual bool resize(unsigned n);

    //! Applies a rigid transformation to
the cloud
    /** WARNING: THIS METHOD IS NOT
COMPATIBLE WITH PARALLEL STRATEGIES
    \param trans transformation (scale *
rotation matrix + translation vector)
    **/
    virtual void
applyTransformation(PointProjectionTools::Transf
ormation& trans);

    //! Returns associated scalar field (if
any)
    ScalarField* getScalarField() { return
m_scalarField; }

```

```

        //! Returns associated scalar field (if
any) (const version)
        const ScalarField* getScalarField()
const { return m_scalarField; }

protected:

        //! Point container
        typedef GenericChunkedArray<3,
PointCoordinateType> PointsContainer;

        //! 3D Points container
        PointsContainer* m_points;

        //! The points distances
        ScalarField* m_scalarField;

        //! Iterator on the points container
        unsigned globalIterator;

        //! Bounding-box validity
        bool m_validBB;
};

        /*@}*/
}

#endif //HYPER_CLOUD_HEADER

```

```

60 |         inline virtual const Eigen::Vector3d
        getEigenVec(unsigned index);
61 |
62 |         inline virtual const pcl::PointXYZ
        getPointXYZ(unsigned index);

```

63

64

```
        //????????? Eigen::Vector3d
getBaryCenter();

        void addCloud(CCLib::HyperCloud& cloud);

        virtual void getPoint(unsigned index,
CCVector3& P) const;

        /*** inherited form
GenericIndexedCloudPersist ***/
        virtual const CCVector3*
getPointPersistentPtr(unsigned index);

        //! Clears cloud
        void clear();

        //! Point insertion mechanism
        /** The point data will be duplicated in
memory.
\param P the point to insert
**/
        virtual void addPoint(const CCVector3
&P);

        virtual void addPoint(const
Eigen::Vector3d& p);

        //! Point insertion mechanism
        /** The point data will be duplicated in
memory.
\param P the point to insert (as a 3-
size array)
**/

        virtual void addPoint(const
PointCoordinateType P[]);
```

```

        //!< Reserves some memory for hosting the
points
        /** \param n the number of points
        **/
        virtual bool reserve(unsigned n);

        //!< Presets the size of the vector used
to store the points
        /** \param n the number of points
        **/
        virtual bool resize(unsigned n);

        //!< Applies a rigid transformation to
the cloud
        /** WARNING: THIS METHOD IS NOT
COMPATIBLE WITH PARALLEL STRATEGIES
        \param trans transformation (scale *
rotation matrix + translation vector)
        **/
        virtual void
applyTransformation(PointProjectionTools::Transf
ormation& trans);

        //!< Returns associated scalar field (if
any)
        ScalarField* getScalarField() { return
m_scalarField; }

        //!< Returns associated scalar field (if
any) (const version)
        const ScalarField* getScalarField()
const { return m_scalarField; }

protected:

```

```

        //! Point container
        typedef GenericChunkedArray<3,
PointCoordinateType> PointsContainer;

        //! 3D Points container
        PointsContainer* m_points;

        //! The points distances
        ScalarField* m_scalarField;

        //! Iterator on the points container
        unsigned globalIterator;

        //! Bounding-box validity
        bool m_validBB;
};

        /*@}*/
}

#endif //HYPER_CLOUD_HEADER

```

```

65 |         Eigen::Vector3d getBaryCenter();
66 |
67 |         void addCloud(CCLib::HyperCloud&
        cloud);
68 |
69 |         virtual void getPoint(unsigned index,
        CCVector3& P) const;
70 |
71 |         /*** inherited form
        GenericIndexedCloudPersist ***/
72 |         virtual const CCVector3*
        getPointPersistentPtr(unsigned index);

```

```

73
75     void clear();
76
78
81     virtual void addPoint(const CCVector3
    &P);
82
83     virtual void addPoint(const
    Eigen::Vector3d& p);
84
86
90     virtual void addPoint(const
    PointCoordinateType P[]);
91
92
94
96     virtual bool reserve(unsigned n);
97
99
101    virtual bool resize(unsigned n);
102
104
107    virtual void
    applyTransformation(PointProjectionTools::Transf
    ormation& trans);
108
109
111    ScalarField* getScalarField() { return
    m_scalarField; }
112
114    const ScalarField* getScalarField()
    const { return m_scalarField; }
115
116    protected:
117
119    typedef GenericChunkedArray<3,
    PointCoordinateType> PointsContainer;

```

```
120
122     PointsContainer* m_points;
123
124
126     ScalarField* m_scalarField;
127
129     unsigned globalIterator;
130
132     bool m_validBB;
133 };
134
136 }
137
138 #endif //HYPER_CLOUD_HEADER
139
```

Inshaping 0.1

Inshape > ParamItem >

Inshape::ParamItem Member List

This is the complete list of members for **Inshape::ParamItem**, including all inherited members.

| | | |
|------------------------|---------------------------|---------------|
| eta | Inshape::ParamItem | |
| lamda_max | Inshape::ParamItem | |
| lamda_min | Inshape::ParamItem | |
| theta | Inshape::ParamItem | |
| threshold | Inshape::ParamItem | |
| toString(int i) | Inshape::ParamItem | inline |

Generated by **doxygen** 1.8.14

Inshaping 0.1

include

Setting.h

Go to the documentation of this file.

```
1 #pragma once
2 #include<iostream>
3 #include<fstream>
4 #include<string>
5 #include<vector>
6 #include<anyarg.h>
7
8 namespace Inshape
9 {
12     #define HEAD_FILENAME "HEAD_FILENAME"
13     #define BODY_FILENAME "BODY_FILENAME"
14     #define SUB_HEAD "SUB_HEAD"
15     #define SUB_BODY "SUB_BODY"
16     #define FACE_FILENAME "FACE_FILENAME"
17     #define FINAL_PARAM "FINAL_PARAM"
18
19
20     struct ParamItem
21     {
22         double theta; // 湍流係数 double
lamda_min; // LAMDA, LAMDA_min
        double lamda_max; //
LAMDA, LAMDA_max
        double eta; // ETA ETA_min
        double threshold; // THRESHOLD
        std::string toString(int i)
        {
            return std::to_string(theta) + "_" +
```

```

std::to_string(lamda_min) + "_" +
std::to_string(lamda_max) + "_" +
std::to_string(eta) + "_" +
std::to_string(threshold) + std::to_string(i);
    }
};

struct Param
{
    std::vector<ParamItem> paramItems;
    ParamItem finalParam;

    void clear()
    {
        paramItems.clear();
    }
};

class Setting:public Anyarg
{
public:
    std::string headFilename;
    std::string faceFilename;

    std::string bodyFilename;

    std::string outputFilename;

    std::string subHead;
    std::string subBody;

    Param param;

    bool preAligned; //0p0Π0t000 true
000000000000000000000000

```

```

        bool outputIntermediate;
        bool useMatch;

        bool readInputTemplate(std::string
filename);

        bool readMatch(std::string filename);

        bool readParam(std::string filename);

        Setting();

        // parsing command line, collect command
line arguments
        bool parse_argv(int argc, char** argv);

};

        /*@}*/
    }
23 |         double lamda_min;//
        double lamda_max; //
        double eta;//
        double threshold;//
        THRESHOLD
        std::string toString(int i)
        {
            return std::to_string(theta) + "_" +
std::to_string(lamda_min) + "_" +
std::to_string(lamda_max) + "_" +
std::to_string(eta) + "_" +
std::to_string(threshold) + std::to_string(i);
        }

```

```

};

struct Param
{
    std::vector<ParamItem> paramItems;
    ParamItem finalParam;

    void clear()
    {
        paramItems.clear();
    }
};

class Setting:public Anyarg
{
public:
    std::string headFilename;
    std::string faceFilename;

    std::string bodyFilename;

    std::string outputFilename;

    std::string subHead;
    std::string subBody;

    Param param;

    bool preAligned; //0p0π0:000 true
000000000000000000000000
    bool outputIntermediate;
    bool useMatch;

    bool readInputTemplate(std::string
filename);

```

```

        bool readMatch(std::string filename);

        bool readParam(std::string filename);

        Setting();

        // parsing command line, collect command
line arguments
        bool parse_argv(int argc, char** argv);

};

        /*@}*/
}
24 |         double lamda_max; //
        double eta; //
        double threshold; //
        THRESHOLD
        std::string toString(int i)
        {
            return std::to_string(theta) + "_" +
std::to_string(lamda_min) + "_" +
std::to_string(lamda_max) + "_" +
std::to_string(eta) + "_" +
std::to_string(threshold) + std::to_string(i);
        }
};

struct Param
{
    std::vector<ParamItem> paramItems;
    ParamItem finalParam;
};

```

```

        void clear()
        {
            paramItems.clear();
        }
};

class Setting:public Anyarg
{
public:
    std::string headFilename;
    std::string faceFilename;

    std::string bodyFilename;

    std::string outputFilename;

    std::string subHead;
    std::string subBody;

    Param param;

    bool preAligned; //0p0π0:000 true
0000000000000000000000
    bool outputIntermediate;
    bool useMatch;

    bool readInputTemplate(std::string
filename);

    bool readMatch(std::string filename);

    bool readParam(std::string filename);

    Setting();

```

```

        // parsing command line, collect command
line arguments
        bool parse_argv(int argc, char** argv);

};

```

```

        /*@}*/

```

```

}

```

```

25 |         double eta; // 0.000000 ETA

```

```

ETA 0.000000

```

```

26 |         double threshold; // 0.000000 0.000000000000

```

```

THRESHOLD

```

```

27 |         std::string toString(int i)

```

```

28 |         {

```

```

29 |             return std::to_string(theta) + "_"

```

```

+ std::to_string(lamda_min) + "_" +

```

```

std::to_string(lamda_max) + "_" +

```

```

std::to_string(eta) + "_" +

```

```

std::to_string(threshold) + std::to_string(i);

```

```

30 |         }

```

```

31 |     };

```

```

32 |

```

```

33 |     struct Param

```

```

34 |     {

```

```

35 |         std::vector<ParamItem> paramItems;

```

```

36 |         ParamItem finalParam;

```

```

37 |

```

```

38 |         void clear()

```

```

39 |         {

```

```

40 |             paramItems.clear();

```

```

41 |         }

```

```

42 |     };

```

```

43 |

```

```

44     class Setting:public Anyarg
45     {
46     public:
47         std::string headFilename;
48         std::string faceFilename;
49
50         std::string bodyFilename;
51
52         std::string outputFilename;
53
54
55         std::string subHead;
56         std::string subBody;
57
58         Param param;
59
60
61         bool preAligned; //0p0π0t000 true
        000000000000000000000000
62         bool outputIntermediate;
63         bool useMatch;
64
65         bool readInputTemplate(std::string
        filename);
66
67         bool readMatch(std::string filename);
68
69         bool readParam(std::string filename);
70
71         Setting();
72
73         // parsing command line, collect
        command line arguments
74         bool parse_argv(int argc, char**
        argv);
75
76     };

```

```
77  
78  
79  
80  
82 }
```

Inshaping 0.1

Inshape > Param >

Inshape::Param Member List

This is the complete list of members for **Inshape::Param**, including all inherited members.

clear() Inshape::Param **inline**

finalParam Inshape::Param

paramItems Inshape::Param

Inshaping 0.1

Inshape > Setting >

Inshape::Setting Member List

This is the complete list of members for **Inshape::Setting**, including all inherited members.

add_flag(const char *name, char letter, const char *desc)

add_flag(char letter, const char *desc)

add_option_double(char letter, double v0, const char *desc)

add_option_double(const char *name, char letter, double v0, const char

add_option_int(char letter, int v0, const char *desc)

add_option_int(const char *name, char letter, int v0, const char *desc)

add_option_str(const char *name, char letter, const char *v0, const char

add_option_str(char letter, const char *v0, const char *desc)

Anyarg()

auto_usage()

bodyFilename

faceFilename

get_arg(int i) const

get_argc() const

get_value_double(const char *name) const

get_value_double(char letter) const

get_value_int(const char *name) const

get_value_int(char letter) const

get_value_str(const char *name) const

get_value_str(char letter) const

headFilename

is_true(const char *name) const

is_true(char letter) const
outputFilename
outputIntermediate
param
parse_argv(int argc, char **argv)
preAligned
readInputTemplate(std::string filename)
readMatch(std::string filename)
readParam(std::string filename)
Setting()
subBody
subHead
useMatch

Inshaping 0.1

Inshape > TriSparse >

Inshape::TriSparse Member List

This is the complete list of members for **Inshape::TriSparse**, including all inherited members.

confirmNearestNeighbor(pcl::PointCloud< pcl::PointNormal > &body, My
confirmNearestNeighbor(pcl::PointCloud< pcl::PointXYZ > &from, pcl::P
confirmNearestNeighbor2(pcl::PointCloud< pcl::PointXYZ > &from, pcl:::
construct()
findNearestNeighbor(pcl::PointCloud< pcl::PointNormal > &body, MyMe:
findNearestNeighbor2(pcl::PointCloud< pcl::PointXYZ > &body, pcl::Poi
getBody()
getHead()
getLAMDA(unsigned index)
getMap()
printParam()
setA(Eigen::SparseMatrix< double > &_A)
setB(Eigen::VectorXd &_B)
setInput(pcl::PointCloud< pcl::PointNormal >::Ptr _body, MyMeshPtr _he:
setParam(const double _THRESHOLD, const double _THETA, const dou
setParam(const ParamItem &_param, bool _doubleCheck)
solve(Eigen::VectorXd &x)
solveToCloud(pcl::PointCloud< pcl::PointXYZ >::Ptr &cloud)
solveTriSparse(Eigen::SparseMatrix< double > &A, Eigen::VectorXd &B,
TriSparse(pcl::PointCloud< pcl::PointNormal >::Ptr _body, MyMeshPtr _h
updateHead(pcl::PointCloud< pcl::PointXYZ >::Ptr &head_cloud)
updateMesh(MyMesh &mesh, pcl::PointCloud< pcl::PointXYZ >::Ptr &clo

~TriSparse()

Generated by doxygen 1.8.14

Inshaping 0.1

include

TriSparse.h

[Go to the documentation of this file.](#)

```
1  #pragma once
2
3  #include<string>
4  #include<pcl/kdtree/kdtree_flann.h>
5  #include<Setting.h>
6
7  #include<Normal.h>
8  #include<Common.h>
9
10 #include<Convert.h>
11
12
13 namespace Inshape
14 {
15     class TriSparse
16     {
17     private:
18
19         //-----@#@#@#@#@-----//
20         ParamItem param;
21         bool doubleCheck;
22         unsigned n;
23         std::unordered_map<unsigned, unsigned>
24         mapping;
25         pcl::PointCloud<pcl::PointNormal>::Ptr
26         body;
27         MyMeshPtr head;
28         Eigen::SparseMatrix<double> A;
29         Eigen::VectorXd B;
```

```

30
31 //-----private 0000-----//
32
33 //dimension 0-x,1-y,2-z
34 unsigned getIndex(unsigned old_index,
    int dimension);
35
36 public:
37     TriSparse(
38
    pcl::PointCloud<pcl::PointNormal>::Ptr _body,
39         MyMeshPtr _head
40     ) :body(_body),
41         head(_head),
42         n((*_head).n_vertices()),
43         doubleCheck(false)
44     {
45         assert((*body).size() != 0);
46         assert(n != 0);
47     }
48
49 //0000`0000 void setParam(const
    double _THRESHOLD, const double _THETA, const
    double _LAMDA_MIN, const double _LAMDA_MAX,
    const double _ETA, bool _doubleCheck);

    void setParam(const ParamItem& _param,
    bool _doubleCheck);

    void printParam();

    inline void setInput(
    pcl::PointCloud<pcl::PointNormal>::Ptr _body,
        MyMeshPtr _head)
    {
        body = _body;

```

```

        head = _head;
        n = (*_head).n_vertices();
    }

    void setA(Eigen::SparseMatrix<double>&
_A);

    void setB(Eigen::VectorXd& _B);

    void
updateHead(pcl::PointCloud<pcl::PointXYZ>::Ptr&
head_cloud);

    double getLAMDA(unsigned index);

    MyMesh& getHead() { return *_head; }

    pcl::PointCloud<pcl::PointNormal>&
getBody() { return *_body; }

    std::unordered_map<unsigned, unsigned>&
getMap() { return mapping; }

    bool construct();//?????????養
?????j??Ç??,??setParam()

    bool solve(Eigen::VectorXd& x);//? ????

    bool
solveToCloud(pcl::PointCloud<pcl::PointXYZ>::Ptr
& cloud);//?????????y?e???

    ~TriSparse() {}

    //-----?????-----

```

```

-----//

        bool static
solveTriSparse(Eigen::SparseMatrix<double>& A,
Eigen::VectorXd& B, Eigen::VectorXd& x);

        // 養
        headbodyheadN
        static void
confirmNearestNeighbor(pcl::PointCloud<pcl::PointNormal>& body, MyMesh& head_mesh,
std::unordered_map<unsigned, unsigned>& mapping,
double threshold);

        // 養
        headbodyheadN
        static void
confirmNearestNeighbor(pcl::PointCloud<pcl::PointXYZ>& from, pcl::PointCloud<pcl::PointNormal>&
to, std::unordered_map<unsigned, unsigned>&
mapping, double threshold);

        // body z head
        , h mapping, _ threshold
        static void
findNearestNeighbor(pcl::PointCloud<pcl::PointNormal>& body, MyMesh& head,
std::unordered_map<unsigned, unsigned>& mapping,
double threshold, bool doubleCheck);

        static void
findNearestNeighbor2(pcl::PointCloud<pcl::PointXYZ> & body, pcl::PointCloud<pcl::PointXYZ>&
head, std::unordered_map<unsigned, unsigned>&
mapping, double threshold, bool doubleCheck);

```

```

        static void
confirmNearestNeighbor2(pcl::PointCloud<pcl::PointXYZ> & from,
pcl::PointCloud<pcl::PointXYZ> & to,
std::unordered_map<unsigned, unsigned>& mapping,
double threshold);

```

```

        // 更新 mesh
        static void updateMesh(MyMesh& mesh,
pcl::PointCloud<pcl::PointXYZ>::Ptr& cloud);

```

```

};
/* @} */

```

```

}

```

```

50 |         void setParam(const double _THRESHOLD,
    |         const double _THETA, const double _LAMDA_MIN,
    |         const double _LAMDA_MAX, const double _ETA, bool
    |         _doubleCheck);

```

```

51 |

```

```

52 |         void setParam(const ParamItem& _param,
    |         bool _doubleCheck);

```

```

53 |

```

```

54 |         void printParam();

```

```

55 |

```

```

56 |         inline void setInput(

```

```

57 |

```

```

    |         pcl::PointCloud<pcl::PointNormal>::Ptr _body,
58 |         MyMeshPtr _head)

```

```

59 |         {

```

```

60 |             body = _body;

```

```

61 |             head = _head;

```

```

62 |             n = (*_head).n_vertices();

```

```

63 |         }

```

```

64 |

```

```

65 |         void setA(Eigen::SparseMatrix<double>&
    |         _A);

```

```

66 |

```

```

67         void setB(Eigen::VectorXd& _B);
68
69         void
updateHead(pcl::PointCloud<pcl::PointXYZ>::Ptr&
head_cloud);
70
71         double getLAMDA(unsigned index);
72
73         MyMesh& getHead() { return *head; }
74
75         pcl::PointCloud<pcl::PointNormal>&
getBody() { return *body; }
76
77         std::unordered_map<unsigned,
unsigned>& getMap() { return maping; }
78
79         bool construct();//?????????養
?????j??ç?n,???setParam()
80
81
82         bool solve(Eigen::VectorXd& x);//
? ????
bool
solveToCloud(pcl::PointCloud<pcl::PointXYZ>::Ptr
& cloud);//?????y?e???
~TriSparse() {};
//-----?????-----
-----//
bool static
solveTriSparse(Eigen::SparseMatrix<double>& A,
Eigen::VectorXd& B, Eigen::VectorXd& x);

```



```
        //0ö00zj0000澆mesh0K000
        static void updateMesh(MyMesh& mesh,
        pcl::PointCloud<pcl::PointXYZ>::Ptr& cloud);

        };
        /*@}*/
    }
```

83

84

```
        bool
    solveToCloud(pcl::PointCloud<pcl::PointXYZ>::Ptr
    & cloud);//???y0e000
        ~TriSparse() {};
```

```
        //-----000000-----
    -----//
```

```
        bool static
    solveTriSparse(Eigen::SparseMatrix<double>& A,
    Eigen::VectorXd& B, Eigen::VectorXd& x);
```

```
        //r00y0養
    0000çhead00000body000000000000head000N000
        static void
    confirmNearestNeighbor(pcl::PointCloud<pcl::Poin
    tNormal>& body, MyMesh& head_mesh,
    std::unordered_map<unsigned, unsigned>& mapping,
    double threshold);
```

```
        //r00y0養
    0000çhead00000body000000000000head000N000
        static void
    confirmNearestNeighbor(pcl::PointCloud<pcl::Poin
    tXYZ>& from, pcl::PointCloud<pcl::PointNormal>&
    to, std::unordered_map<unsigned, unsigned>&
```

```

mapping, double threshold);

        // body head
        mapping, threshold
        static void
findNearestNeighbor(pcl::PointCloud<pcl::PointNo
rmal>& body, MyMesh& head,
std::unordered_map<unsigned, unsigned>& mapping,
double threshold, bool doubleCheck);

        static void
findNearestNeighbor2(pcl::PointCloud<pcl::PointX
YZ> & body, pcl::PointCloud<pcl::PointXYZ>&
head, std::unordered_map<unsigned, unsigned>&
mapping, double threshold, bool doubleCheck);

        static void
confirmNearestNeighbor2(pcl::PointCloud<pcl::Poi
ntXYZ> & from, pcl::PointCloud<pcl::PointXYZ> &
to, std::unordered_map<unsigned, unsigned>&
mapping, double threshold);

        // mesh
        static void updateMesh(MyMesh& mesh,
pcl::PointCloud<pcl::PointXYZ>::Ptr& cloud);

};
/*@}*/
}
85 |
86 | ~TriSparse() {};
87 |
88 |
89 | //-----
-----//
90 |
91 | bool static

```

```
solveTriSparse(Eigen::SparseMatrix<double>& A,  
Eigen::VectorXd& B, Eigen::VectorXd& x);
```

```
92
```

```
93
```

```
94
```

```
95
```

```
//_00y0養
```

```
0000çhead00000body000000000000head000N000
```

```
static void
```

```
confirmNearestNeighbor(pcl::PointCloud<pcl::PointNormal>& body, MyMesh& head_mesh,  
std::unordered_map<unsigned, unsigned>& mapping,  
double threshold);
```

```
//_00y0養
```

```
0000çhead00000body000000000000head000N000
```

```
static void
```

```
confirmNearestNeighbor(pcl::PointCloud<pcl::PointXYZ>& from, pcl::PointCloud<pcl::PointNormal>&  
to, std::unordered_map<unsigned, unsigned>&  
mapping, double threshold);
```

```
//0000 body 000z0 head
```

```
000000,00000h00mapping,??????.threshold
```

```
static void
```

```
findNearestNeighbor(pcl::PointCloud<pcl::PointNormal>& body, MyMesh& head,  
std::unordered_map<unsigned, unsigned>& mapping,  
double threshold, bool doubleCheck);
```

```
static void
```

```
findNearestNeighbor2(pcl::PointCloud<pcl::PointXYZ> & body, pcl::PointCloud<pcl::PointXYZ>&  
head, std::unordered_map<unsigned, unsigned>&  
mapping, double threshold, bool doubleCheck);
```

```
static void
```

```
confirmNearestNeighbor2(pcl::PointCloud<pcl::Poi
```



```

mapping, double threshold, bool doubleCheck);

        static void
confirmNearestNeighbor2(pcl::PointCloud<pcl::PointXYZ> & from,
pcl::PointCloud<pcl::PointXYZ> & to,
std::unordered_map<unsigned, unsigned>& mapping, double threshold);

        //zj mesh
        static void updateMesh(MyMesh& mesh,
pcl::PointCloud<pcl::PointXYZ>::Ptr& cloud);

};
/*@}*/
}
99         static void
confirmNearestNeighbor(pcl::PointCloud<pcl::PointXYZ>& from,
pcl::PointCloud<pcl::PointNormal>& to,
std::unordered_map<unsigned, unsigned>& mapping, double threshold);
100
101         // body z head
, mapping, .. threshold
102         static void
findNearestNeighbor(pcl::PointCloud<pcl::PointNormal>& body,
MyMesh& head,
std::unordered_map<unsigned, unsigned>& mapping,
double threshold, bool doubleCheck);
103
104         static void
findNearestNeighbor2(pcl::PointCloud<pcl::PointXYZ> & body,
pcl::PointCloud<pcl::PointXYZ>& head,
std::unordered_map<unsigned, unsigned>& mapping,
double threshold, bool doubleCheck);
105
106         static void
confirmNearestNeighbor2(pcl::PointCloud<pcl::PointXYZ> & from,
pcl::PointCloud<pcl::PointXYZ> & to,
std::unordered_map<unsigned, unsigned>& mapping, double threshold);

```

```

ntXYZ> & from, pcl::PointCloud<pcl::PointXYZ> &
to, std::unordered_map<unsigned, unsigned>&
mapping, double threshold);
107 |
108 |         // mesh static
void updateMesh(MyMesh& mesh,
pcl::PointCloud<pcl::PointXYZ>::Ptr& cloud);

};
/*@}*/
}
109 |         static void updateMesh(MyMesh& mesh,
pcl::PointCloud<pcl::PointXYZ>::Ptr& cloud);
110 |
111 |     };
113 | }

```

Inshaping 0.1

Anyarg Member List

This is the complete list of members for **Anyarg**, including all inherited members.

add_flag(const char *name, char letter, const char *desc)

add_flag(char letter, const char *desc)

add_option_double(char letter, double v0, const char *desc)

add_option_double(const char *name, char letter, double v0, const char

add_option_int(char letter, int v0, const char *desc)

add_option_int(const char *name, char letter, int v0, const char *desc)

add_option_str(const char *name, char letter, const char *v0, const char

add_option_str(char letter, const char *v0, const char *desc)

Anyarg()

auto_usage()

get_arg(int i) const

get_argc() const

get_value_double(const char *name) const

get_value_double(char letter) const

get_value_int(const char *name) const

get_value_int(char letter) const

get_value_str(const char *name) const

get_value_str(char letter) const

is_true(const char *name) const

is_true(char letter) const

parse_argv(int argc, char **argv)

Inshaping 0.1

include

anyarg.h

Go to the documentation of this file.

```
1
38 /*
39 A single letter option begins with a hyphen '-'
  '. The parsing of single letter options follows
  POSIX conventions.
40
41 A long option begins with two hyphens '--'.
  The parsing of long options follows GNU
  conventions.
42
43 Only specify/define a option once, otherwise
  you will incur an error.
44
45 Example:
46 Supposing program \c foo has two flags (-a -
  all, -v --verbose) and two taking-value options
  (-s
47 --buffer-size, -n), you can turn on flag a and
  v, set option s to 100 and n to 50, and pass
  another two
48 non-option arguments (abc and xyz) to foo by:
49
50 foo -av -s 100 -n 50 abc xyz foo -av
51 --buffer-size=100 -n 50 abc xyz
52 foo --all --verbose --buffer-size=100 -n 50
  abc xyz
53
54 POSIX conventions:
55 http://pubs.opengroup.org/onlinepubs/969991979
```

9/basedefs/V1_chap12.html

```
56 |
57 | GNU extensions:
58 | http://www.gnu.org/prep/standards/html\_node/Co
   | mmand\_002dLine-Interfaces.html
59 | */
60 |
61 |
62 | #ifndef __ANYARG_H__
63 | #define __ANYARG_H__
64 |
65 | #include <string>
66 | #include <vector>
67 | #include <cstdlib>
68 | #include <cstring>
69 |
70 | using std::string;
71 | using std::vector;
72 |
73 |
75 | struct Option
76 | {
77 |     char type; // type of option
78 |     char letter; // single-letter label of an
   | option
79 |     string name; // long name of an option
80 | // char valtype; // value type, [BSID],
   | bool: B, string: S, int: I, double: D
81 |     string value_str; // option value as a
   | string
82 |     union { // option value as a
   | bool, int or double value
83 |         bool value_bool;
84 |         int value_int;
85 |         double value_double;
86 |     };
87 |     string desc; // a short sentence to
```

```

describe an option
88 |     string meta;    // a META word to specify
    |     the type of option argument, such as "FILE",
    |     "SIZE"
89 |
90 |     Option():type(0), letter(), name(),
    |     value_str(), value_double(0.0), desc(), meta()
    |     {}
91 |     bool set_value(const char *opt_value);
92 |     bool set_desc_meta(const char *opt_desc);
93 | };
94 |
95 |
97 | class Anyarg
98 | {
99 |     public:
101 |     Anyarg();
102 |
104 |
112 |     bool add_flag(const char *name, char
    |     letter, const char *desc);
113 |     bool add_flag(char letter, const char
    |     *desc);
115 |
117 |
127 |     // Add an option with a string value
128 |     bool add_option_str(const char *name, char
    |     letter, const char *v0, const char *desc);
129 |     bool add_option_str(char letter, const
    |     char *v0, const char *desc);
130 |
131 |     // Add an option with an integer value
132 |     bool add_option_int(char letter, int v0,
    |     const char *desc);
133 |     bool add_option_int(const char *name, char
    |     letter, int v0, const char *desc);
134 |

```

```
135 | // Add an option with an double value
136 | bool add_option_double(char letter, double
    | v0, const char *desc);
137 | bool add_option_double(const char *name,
    | char letter, double v0, const char *desc);
139 |
148 | bool parse_argv(int argc, char **argv);
149 |
151 |
157 | bool is_true(const char *name) const;
158 | bool is_true(char letter) const;
160 |
162 |
170 | // Get the value of an option with string
    | values
171 | const char *get_value_str(const char
    | *name) const;
172 | const char *get_value_str(char letter)
    | const;
173 |
174 | // Get the value of an option with integer
    | values
175 | int get_value_int(const char *name) const;
176 | int get_value_int(char letter) const;
177 |
178 | // Get the value of an option with double
    | values
179 | double get_value_double(const char *name)
    | const;
180 | double get_value_double(char letter)
    | const;
182 |
187 | int get_argc() const;
188 |
194 | const char *get_arg(int i) const;
195 |
200 | const char *auto_usage();
```

```

201 |
202 |     private:
203 |         string prog_name_; // name of the program
204 |         int argc_;        // count of non-option
           arguments
205 |         vector<string> argv_; // vector of non-
           option arguments
206 |
207 |         vector<Option> options_; // vector of
           options
208 |
209 |         string help_;     // formatted help for
           options
210 |
211 |         Anyarg(const Anyarg &); // prevent the
           copy of a anyarg object
212 |
213 |         Anyarg & operator = (const Anyarg &); //
           prevent assignment of anyarg object
214 |
215 |         bool is_new_option(const char *name, char
           letter);
216 |
217 |         int  get_optind(char letter) const;
218 |
219 |         int  get_optind(const char *name) const;
220 |
221 |         // Show information of all options, for
           debugging.
222 |         void show_options() const;
223 |     };
224 |
225 | #endif

```

Inshaping 0.1

Option Member List

This is the complete list of members for **Option**, including all inherited members.

| | | |
|---|---------------|---------------|
| desc | Option | |
| letter | Option | |
| meta | Option | |
| name | Option | |
| Option() | Option | inline |
| set_desc_meta (const char *opt_desc) | Option | |
| set_value (const char *opt_value) | Option | |
| type | Option | |
| value_bool | Option | |
| value_double | Option | |
| value_int | Option | |
| value_str | Option | |

Inshaping 0.1

Inshape > NormalEstimationPlus > WArc > PointerAndIdx >

Inshape::NormalEstimationPlus::WArc::PointerAndIdx Member List

This is the complete list of members for **Inshape::NormalEstimationPlus::WArc::PointerAndIdx**, including all inherited members.

| | |
|-----------------------|---|
| index | Inshape::NormalEstimationPlus::WArc::PointerAndIdx |
| invertNormal() | Inshape::NormalEstimationPlus::WArc::PointerAndIdx |
| normalCopy() | Inshape::NormalEstimationPlus::WArc::PointerAndIdx |
| p | Inshape::NormalEstimationPlus::WArc::PointerAndIdx |

Generated by  1.8.14

Inshaping 0.1

include

Benchmark.h

Go to the documentation of this file.

```
1 #pragma once
2 #include<TriSparse.h>
3
4 namespace Inshape
5 {
6     double avgE1(TriSparse& triSprase);
7
8     double avgE3(TriSparse& triSprase);
9
10
11
14 }
```

Generated by doxygen 1.8.14

Inshaping 0.1

include

Common.h

Go to the documentation of this file.

```
1  #pragma once
2
3  #include<Eigen/Dense>
4  #include<Eigen/Sparse>
5  #include<unordered_map>
6
7  #include<memory>
8  #include <pcl/common/centroid.h>
9  #include<pcl/point_cloud.h>
10 #include<pcl/point_types.h>
11 #include <pcl/surface/gp3.h>
12 #include <pcl/surface/poisson.h>
13 // ----- OpenMesh-----
14 // -----
15 #include <OpenMesh/Core/IO/MeshIO.hh>
16 #include
17 <OpenMesh/Core/Mesh/PolyMesh_ArrayKernelT.hh>
18
19 // -----
20
21 #include<HyperCloud.h>
22 #include<SimpleMesh.h>
23
24 namespace Inshape
25 {
26     typedef OpenMesh::PolyMesh_ArrayKernelT<>
27     MyMesh;
28     typedef std::shared_ptr<MyMesh> MyMeshPtr;
29     typedef std::shared_ptr<CCLib::SimpleMesh>
```

```
SimpleMeshPtr;
```

```
26 | }
```

Generated by [doxygen](#) 1.8.14

Inshaping 0.1

include

Convert.h

Go to the documentation of this file.

```
1 | #pragma once
2 | #include<TriSparse.h>
3 |
4 | namespace Inshape
5 | {
6 |
7 |     namespace Convert
8 |     {
9 |         //????????e??? void
10 |         meshToCloud(MyMesh& head_mesh,
11 |                    pcl::PointCloud<pcl::PointXYZ>& head_cloud);
12 |
13 |         pcl::PointCloud<pcl::PointXYZ>::Ptr
14 |         hyperToPointCloud(CCLib::HyperCloud::Ptr&
15 |                            hyper);
16 |
17 |         CCLib::HyperCloud::Ptr
18 |         cloudToHyper(pcl::PointCloud<pcl::PointXYZ>&
19 |                      cloud);
20 |
21 |         MyMeshPtr
22 |         PolygonToMyMesh(pcl::PolygonMesh & mesh);
23 |     }
24 |     /*@}*/
25 | }
26 |
27 | void meshToCloud(MyMesh& head_mesh,
28 |                 pcl::PointCloud<pcl::PointXYZ>& head_cloud);
29 |
30 | pcl::PointCloud<pcl::PointXYZ>::Ptr
31 | hyperToPointCloud(CCLib::HyperCloud::Ptr&
```

```
hyper);  
14 |  
15 |         CCLib::HyperCloud::Ptr  
    |         cloudToHyper(pcl::PointCloud<pcl::PointXYZ>&  
    |         cloud);  
16 |  
17 |         MyMeshPtr  
    |         PolygonToMyMesh(pcl::PolygonMesh & mesh);  
18 |     }  
20 | }
```

Generated by doxygen 1.8.14

Inshaping 0.1

include

ConvexHull.h

Go to the documentation of this file.

```
1 | #include<Common.h>
2 | #include<DistanceComputationTools.h>
3 | #include<pcl/surface/convex_hull.h>
4 | #include<Convert.h>
5 |
6 | namespace Inshape
7 | {
10 |     namespace ConvexHull
11 |     {
12 |         pcl::PolygonMesh::Ptr
            getConvexHull(pcl::PointCloud<pcl::PointXYZ>::Ptr
            r& cloud);
13 |
14 |         pcl::PointCloud<pcl::PointXYZ>::Ptr
            filterWithHull(pcl::PointCloud<pcl::PointXYZ>::Ptr
            tr & bodyshead,
            pcl::PointCloud<pcl::PointXYZ>::Ptr & head,
            double threshold);
15 |     }
18 | }
```

Inshaping 0.1

include

FileIO.h

Go to the documentation of this file.

```
1 | #pragma once
2 | #include<fstream>
3 | #include<string>
4 | #include<Common.h>
5 | #include<Normal.h>
6 |
7 | namespace Inshape
8 | {
11 |     bool readPly(const std::string & filename,
    |     pcl::PointCloud<pcl::PointXYZ>& cloud);
12 |
13 |     bool savePly(const std::string& filename,
    |     pcl::PointCloud<pcl::PointXYZ>& cloud);
14 |
15 |     bool savePolygonMesh(std::string filename,
    |     pcl::PolygonMesh & mesh);
16 |
18 | }
```



```

NICPRigsteration(pcl::PointCloud<pcl::PointXYZ>:
:Ptr & data, pcl::PointCloud<pcl::PointXYZ>::Ptr
& model);
23 |
24 |     Eigen::Matrix4f
    SVDRegistration(pcl::PointCloud<pcl::PointXYZ>::
Ptr& sub_data,
pcl::PointCloud<pcl::PointXYZ>::Ptr& sub_model);
25 |
26 |     void
    AlignWithFace(pcl::PointCloud<pcl::PointXYZ>::Pt
r& body, pcl::PointCloud<pcl::PointXYZ>::Ptr&
head_cloud, pcl::PointCloud<pcl::PointXYZ>::Ptr&
face);
27 |
28 |     //????????????ICP
29 |     void
    rigidRigsterWithoutMatch(pcl::PointCloud<pcl::Po
intXYZ>::Ptr& body, MyMesh&
head_mesh, pcl::PointCloud<pcl::PointXYZ>::Ptr&
face);
30 |
31 |
32 |     //000000000000face icp
33 |     void
    rigidRigsterWithMatch(pcl::PointCloud<pcl::Point
XYZ>::Ptr& body, MyMesh& head_mesh,
pcl::PointCloud<pcl::PointXYZ>::Ptr& face,
pcl::PointCloud<pcl::PointXYZ>::Ptr& sub_head,
pcl::PointCloud<pcl::PointXYZ>::Ptr& sub_body);
34 |
36 | }

```