Practica 2 VAR - Mapeado PCL

1.0

Generated by Doxygen 1.7.6.1

Sun May 22 2016 15:01:38

Contents

Chapter 1

File Index

1.1 File List

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

Chapter 2

File Documentation

2.1 common.h File Reference

```
#include <ros/ros.h>   #include <pcl_ros/point_cloud.h> ×
#include <pcl/point_types.h>    #include <boost/foreach.-
hpp> #include <pcl/visualization/pcl_visualizer.h> #include
<pcl/visualization/cloud_viewer.h> #include <pcl/filters/voxel-
_grid.h> #include <pcl/keypoints/iss_3d.h> #include <pcl/impl/point-
_types.hpp>  #include <pcl/features/shot_omp.h>  #include
<pcl/features/normal_3d.h> #include <pcl/features/normal-
_3d_omp.h>    #include <pcl/kdtree/impl/kdtree_flann.hpp>
#include <pcl/correspondence.h> #include <pcl/recognition/cg/geometric-
_consistency.h> #include <pcl/common/transforms.h> #include
<pcl/console/parse.h>     #include <pcl/filters/extract_-
indices.h> #include <pcl/io/pcd_io.h> #include <pcl/sample-
_consensus/ransac.h> #include <pcl/sample_consensus/sac_-
model_plane.h>  #include <pcl/sample_consensus/sac_model-
_sphere.h>     #include <pcl/recognition/cg/hough_3d.h> ×
#include <pcl/features/board.h> #include <pcl/common/time.-
h> #include <pcl/registration/sample_consensus_prerejective.-
h> #include <pcl/registration/icp.h> #include <boost/thread/thread.-
hpp> #include <iostream> #include <vector> #include <string> ×
#include <sys/resource.h> #include <pcl/registration/correspondence-
_rejection_sample_consensus.h> #include <pcl/features/fpfh-
_omp.h> #include <pcl/keypoints/sift_keypoint.h> #include
<pcl/keypoints/harris_3d.h> #include <pcl/features/cvfh.-
h>
```

Defines

- #define **EIGEN_YES_I_KNOW_SPARSE_MODULE_IS_NOT_STABLE_YET**
- #define **SHOW_TIME** 1
- #define **DEBUG_MSG** 1

- #define KeypointsMethod 2
- #define DescriptorMethod 2
- #define **DescriptorType** pcl::FPFHSignature33
- #define **PointType** pcl::PointXYZRGB

Functions

- double get_cpu_time ()

Variables

- pcl::PointCloud< PointType >::Ptr **final_cloud**
- pcl::PointCloud< PointType >::Ptr **last_cloud**
- pcl::PointCloud< PointType >::Ptr **last_keypoints**
- pcl::PointCloud < DescriptorType >::Ptr **last_descriptors**
- pcl::PointCloud< pcl::Normal >::Ptr **last_normals**
- Eigen::Matrix4f **transformation**
- double **actual_res**

2.1.1 Detailed Description

Parametrizacion comun necesaria

Ismael Piñeiro Ramos

Definition in file common.h.

2.1.2 Define Documentation

2.1.2.1 #define DescriptorMethod 2

1 == SHOT352 2 == FPFH 3 == CVFH No olvidar ajustar DescriptorType

Definition at line 66 of file common.h.

2.1.2.2 #define KeypointsMethod 2

1 == ISSKeypoints3D 2 == SIFTKeypoint 3 == HarrisKeypoint3D

Definition at line 59 of file common.h.

2.1.3 Function Documentation

2.1.3.1 double get_cpu_time (void)

Computa el tiempo de CPU actual

Returns

tiempo actual de CPU

Definition at line 95 of file node.cpp.

2.2 descriptors.cpp File Reference

```
#include "descriptors.h"
```

Functions

- void SHOT352_descriptors (const pcl::PointCloud< PointType >::ConstPtr &keypoints, const pcl::PointCloud< PointType >::ConstPtr &cloud, pcl::PointCloud< pcl::SHOT352 >::Ptr &descriptors)
- void FPFH_descriptors (const pcl::PointCloud< PointType >::ConstPtr &keypoints, pcl::PointCloud< pcl::FPFHSignature33 >::Ptr &descriptors)
- void CVFH_descriptors (const pcl::PointCloud< PointType >::ConstPtr &keypoints, pcl::PointCloud< pcl::VFHSignature308 >::Ptr &descriptors)

2.2.1 Detailed Description

Algoritmos de obtenicion de descriptores mediante PCL implementacion

Ismael Piñeiro Ramos

Definition in file descriptors.cpp.

2.2.2 Function Documentation

2.2.2.1 void CVFH_descriptors (const pcl::PointCloud< PointType >::ConstPtr & *keypoints*, pcl::PointCloud< pcl::VFHSignature308 >::Ptr & *descriptors*)

Computa los descriptores mediante el algoritmo CVFH

Parameters

keypoints	nube de puntos de la que obtener los descriptore
descriptors	nube de puntos de salida obteniendo los descriptores

Definition at line 50 of file descriptors.cpp.

2.2.2.2 void FPFH_descriptors (const pcl::PointCloud< PointType >::ConstPtr & *keypoints,* pcl::PointCloud< pcl::FPFHSignature33 >::Ptr & *descriptors*)

Computa los descriptores mediante el algoritmo PFH utilizando su vertiente "FAST" FPFH

Parameters

keypoints	Keypoints de los que calular los descriptores
descriptors	nube de puntos de salida obteniendo los descriptores

Definition at line 30 of file descriptors.cpp.

2.2.2.3 void SHOT352_descriptors (const pcl::PointCloud< PointType >::ConstPtr & *keypoints,* const pcl::PointCloud< PointType >::ConstPtr & *cloud,* pcl::PointCloud< pcl::SHOT352 >::Ptr & *descriptors*)

Computa los descriptores mediante el algoritmo SHOT352 de manera paralelizada mediante OpenMP

Parameters

keypoints	Keypoints de los que calcular los descriptores
cloud	Nube de puntso de la que calcular los descriptores
descriptors	nube de puntos de salida obteniendo los descriptores

Definition at line 10 of file descriptors.cpp.

2.3 descriptors.h File Reference

```
#include "common.h" #include "mapping.h"
```

Defines

- #define **SHOT352_RADIUS_SEARCH** 0.05
- #define **FPFH_RADIUS_SEARCH** 0.05
- #define **CVFH_EPS_ANGLE_THRESHOLD** 5.0 / 180.0 ∗ M_PI
- #define **CVFH_CURVATURE_THRESHOLD** 1.0
- #define **CVFH_NORMALIZE_BINS** false

Functions

- void SHOT352_descriptors (const pcl::PointCloud< PointType >::ConstPtr &keypoints, const pcl::PointCloud< PointType >::ConstPtr &cloud, pcl::PointCloud< pcl::SHOT352 >::Ptr &descriptors)

- void FPFH_descriptors (const pcl::PointCloud< PointType >::ConstPtr &keypoints, pcl::PointCloud< pcl::FPFHSignature33 >::Ptr &descriptors)
- void CVFH_descriptors (const pcl::PointCloud< PointType >::ConstPtr &keypoints, pcl::PointCloud< pcl::VFHSignature308 >::Ptr &descriptors)

2.3.1 Detailed Description

Algoritmos de obtenicion de descriptores mediante PCL cabecera

Ismael Piñeiro Ramos

Definition in file descriptors.h.

2.3.2 Function Documentation

2.3.2.1 void CVFH_descriptors (const pcl::PointCloud< PointType >::ConstPtr & *keypoints*, pcl::PointCloud< pcl::VFHSignature308 >::Ptr & *descriptors*)

Computa los descriptores mediante el algoritmo CVFH

Parameters

keypoints	nube de puntos de la que obtener los descriptore
descriptors	nube de puntos de salida obteniendo los descriptores

Definition at line 50 of file descriptors.cpp.

2.3.2.2 void FPFH_descriptors (const pcl::PointCloud< PointType >::ConstPtr & *keypoints*, pcl::PointCloud< pcl::FPFHSignature33 >::Ptr & *descriptors*)

Computa los descriptores mediante el algoritmo PFH utilizando su vertiente "FAST" FPFH

Parameters

keypoints	Keypoints de los que calular los descriptores
descriptors	nube de puntos de salida obteniendo los descriptores

Definition at line 30 of file descriptors.cpp.

2.3.2.3 void SHOT352_descriptors (const pcl::PointCloud< PointType >::ConstPtr & *keypoints*, const pcl::PointCloud< PointType >::ConstPtr & *cloud*, pcl::PointCloud< pcl::SHOT352 >::Ptr & *descriptors*)

Computa los descriptores mediante el algoritmo SHOT352 de manera paralelizada mediante OpenMP

Parameters

keypoints	Keypoints de los que calcular los descriptores
cloud	Nube de puntso de la que calcular los descriptores
descriptors	nube de puntos de salida obteniendo los descriptores

Definition at line 10 of file descriptors.cpp.

2.4 keypoints.cpp File Reference

```
#include "keypoints.h"
```

Functions

- void iss_keypoints (const pcl::PointCloud< PointType >::ConstPtr &cloud, pcl::-PointCloud< PointType >::Ptr &keypoints)
- void sift_keypoints (const pcl::PointCloud< PointType >::ConstPtr &cloud, pcl::-PointCloud< PointType >::Ptr &keypoints)
- void harris_keypoints (const pcl::PointCloud< PointType >::ConstPtr &cloud, pcl-::PointCloud< PointType >::Ptr &keypoints)

2.4.1 Detailed Description

Algoritmos de obtenicion de keypoints mediante PCL implementacion

Ismael Piñeiro Ramos

Definition in file keypoints.cpp.

2.4.2 Function Documentation

2.4.2.1 void harris_keypoints (const pcl::PointCloud< PointType >::ConstPtr & *cloud*, pcl::PointCloud< PointType >::Ptr & *keypoints*)

Calcula los keypoints de una nube de puntos, utilizando el algoritmo HarrisKeypoint3D

Parameters

cloud	Nube de puntos de entrada de la que calcular los keypoints
keypoints	Nube de puntos de salida con los keypoints calculados

Definition at line 59 of file keypoints.cpp.

2.4.2.2 void iss_keypoints (const pcl::PointCloud< PointType >::ConstPtr & *cloud*, pcl::PointCloud< PointType >::Ptr & *keypoints*)

Calcula los keypoints de una nube de puntos, utilizando el algoritmo de ISSKeypoint3D

Parameters

cloud	Nube de puntos de entrada de la que calcular los keypoints
keypoints	Nube de puntos de salida con los keypoints calculados

Definition at line 10 of file keypoints.cpp.

2.4.2.3 void sift_keypoints (const pcl::PointCloud< PointType >::ConstPtr & *cloud*, pcl::PointCloud< PointType >::Ptr & *keypoints*)

Calcula los keypoints de una nube de puntos, utilizando el algoritmo SIFT

Parameters

cloud	Nube de puntos de entrada de la que calcular los keypoints
keypoints	Nube de puntos de salida con los keypoints calculados

Definition at line 33 of file keypoints.cpp.

2.5 keypoints.h File Reference

```
#include "common.h"
```

Defines

- #define **ISS_SALIENT_RADIUS** 6
- #define **ISS_NON_MAX_RADIUS** 4
- #define **SIFT_MIN_SCALE** 0.01f
- #define **SIFT_N_OCTAVES** 3
- #define **SIFT_N_SCALES_OCTAVE** 4
- #define **SIFT_MINIMUM_CONTRAST** 0.0001f
- #define **HARRIS_NON_MAX_SUPRESSION** true
- #define **HARRIS_THRESHOLD** 1e-9

Functions

- void iss_keypoints (const pcl::PointCloud< PointType >::ConstPtr &cloud, pcl::PointCloud< PointType >::Ptr &keypoints)
- void sift_keypoints (const pcl::PointCloud< PointType >::ConstPtr &cloud, pcl::PointCloud< PointType >::Ptr &keypoints)

- void harris_keypoints (const pcl::PointCloud< PointType >::ConstPtr &cloud, pcl::PointCloud< PointType >::Ptr &keypoints)

2.5.1 Detailed Description

Algoritmos de obtenicion de keypoints mediante PCL cabecera

Ismael Piñeiro Ramos

Definition in file keypoints.h.

2.5.2 Function Documentation

2.5.2.1 void harris_keypoints (const pcl::PointCloud< PointType >::ConstPtr & *cloud*, pcl::PointCloud< PointType >::Ptr & *keypoints*)

Calcula los keypoints de una nube de puntos, utilizando el algoritmo HarrisKeypoint3D

Parameters

cloud	Nube de puntos de entrada de la que calcular los keypoints
keypoints	Nube de puntos de salida con los keypoints calculados

Definition at line 59 of file keypoints.cpp.

2.5.2.2 void iss_keypoints (const pcl::PointCloud< PointType >::ConstPtr & *cloud*, pcl::PointCloud< PointType >::Ptr & *keypoints*)

Calcula los keypoints de una nube de puntos, utilizando el algoritmo de ISSKeypoint3D

Parameters

cloud	Nube de puntos de entrada de la que calcular los keypoints
keypoints	Nube de puntos de salida con los keypoints calculados

Definition at line 10 of file keypoints.cpp.

2.5.2.3 void sift_keypoints (const pcl::PointCloud< PointType >::ConstPtr & *cloud*, pcl::PointCloud< PointType >::Ptr & *keypoints*)

Calcula los keypoints de una nube de puntos, utilizando el algoritmo SIFT

Parameters

cloud	Nube de puntos de entrada de la que calcular los keypoints
keypoints	Nube de puntos de salida con los keypoints calculados

Definition at line 33 of file keypoints.cpp.

2.6 mapping.cpp File Reference

```
#include "mapping.h"
```

Functions

- pcl::PointCloud< PointType >::Ptr **final_cloud** (new pcl::PointCloud< PointType >())
- pcl::PointCloud< PointType >::Ptr **last_cloud** (new pcl::PointCloud< PointType >())
- pcl::PointCloud< PointType >::Ptr **last_keypoints** (new pcl::PointCloud< PointType >())
- pcl::PointCloud < DescriptorType >::Ptr **last_descriptors** (new pcl::PointCloud< DescriptorType >())
- pcl::PointCloud< pcl::Normal >::Ptr **last_normals** (new pcl::PointCloud< pcl::Normal >())
- double get_cloud_resolution (const pcl::PointCloud< PointType >::ConstPtr &cloud)
- void remove_nan (pcl::PointCloud< PointType >::Ptr cloud)
- void filter_voxel_grid (const pcl::PointCloud< PointType >::ConstPtr &cloud, pcl::PointCloud< PointType >::Ptr cloud_filtered)
- void estimate_normals (const pcl::PointCloud< PointType >::ConstPtr &cloud, pcl::PointCloud< pcl::Normal >::Ptr normals)
- bool ransac_alignment (const pcl::PointCloud< PointType >::ConstPtr &cloud, const pcl::PointCloud< DescriptorType >::ConstPtr &descriptors, pcl::PointCloud< PointType >::Ptr cloud_aligned)
- void iterative_closest_point (const pcl::PointCloud< PointType >::ConstPtr &cloud)
- void ransac_correspondences (const pcl::PointCloud< PointType >::ConstPtr &keypoints, pcl::CorrespondencesPtr bestCorrespondences)

Variables

- Eigen::Matrix4f **transformation**
- double **actual_res** = 0

2.6.1 Detailed Description

Funciones necesarias para el mapeado, que no son de obtencion de keypoints ni de descriptores. Implementacion

Author

Ismael Piñeiro Ramos

Definition in file mapping.cpp.

2.6.2 Function Documentation

2.6.2.1 void estimate_normals (const pcl::PointCloud< PointType >::ConstPtr & *cloud,* pcl::PointCloud< pcl::Normal >::Ptr *normals*)

Estima las normales locales de una nube de puntos, utiliza paralelizacion mediante OpenMP

Parameters

cloud	Nube de puntos de entrada de la que estimar las normales
normals	Nube de puntos de salida de las normales estimadas

Definition at line 72 of file mapping.cpp.

2.6.2.2 void filter_voxel_grid (const pcl::PointCloud< PointType >::ConstPtr & *cloud,* pcl::PointCloud< PointType >::Ptr *cloud_filtered*)

Realiza un filtrado VoxelGrid sobre la nube de puntos de entrada, aproximando los puntos mediante sus centroides.

Parameters

cloud	Nube de puntos de entrada a filtrar
cloud_filtered	Nube de puntos de salida ya filtrada

Definition at line 58 of file mapping.cpp.

2.6.2.3 double get_cloud_resolution (const pcl::PointCloud< PointType >::ConstPtr & *cloud*)

Calcula la resolución espacial de una nube de puntos dada, mediante la media de la distancia entre cada punto y su vecino mas cercano

Parameters

cloud	Nube de puntos de la que calcular la resolución

Definition at line 23 of file mapping.cpp.

2.6.2.4 void iterative_closest_point (const pcl::PointCloud< PointType >::ConstPtr & *cloud*)

Minimiza las distancias entre los puntos de dos nubes y las transforma utilizando el algoritmo IterativeClosestPoint

Parameters

cloud	Nube de puntos de entrada

Definition at line 114 of file mapping.cpp.

2.6.2.5 bool ransac_alignment (const pcl::PointCloud< PointType >::ConstPtr & *cloud*, const pcl::PointCloud< DescriptorType >::ConstPtr & *descriptors*, pcl::PointCloud< PointType >::Ptr *cloud_aligned*)

Realiza el alineamiento de la nube de puntos actual con la anterior utilizando el algoritmo SampleConsensusPrerejective

Parameters

cloud	Nube de puntos de entrada a alinear
descriptors	Descriptores de la nube de puntos de entrada
cloud_aligned	Nube de puntos de salida alineada

Definition at line 92 of file mapping.cpp.

2.6.2.6 void ransac_correspondences (const pcl::PointCloud< PointType >::ConstPtr & *keypoints*, pcl::CorrespondencesPtr *bestCorrespondences*)

Obtiene las correspondencias y las filtra para quedarnos unicamente con las mejores correspondencias mediante el uso de RANSAC

Parameters

keypoints	nubde de puntos de entrada de la que calcular correspondencias
bestCorrespondences	correspondencias de salida calculadas

Definition at line 133 of file mapping.cpp.

2.6.2.7 void remove_nan (pcl::PointCloud< PointType >::Ptr *cloud*)

Elimina los "Not a Number" de una nube de puntos

Parameters

cloud	Nube de puntos de la que eliminar los NaN

Definition at line 45 of file mapping.cpp.

2.7 mapping.h File Reference

```
#include "common.h"
```

Defines

- #define **NORMALS_RADIUS_SEARCH** 0.01f
- #define **ICP_MAX_ITERATIONS** 50
- #define **ICP_MAX_CORRESPONDENCE_DISTANCE** 0.05
- #define **ICP_TRANSFORMATION_EPSILON** 1e-8
- #define **ICP_EUCLIDEAN_FITNESS_EPSILON** 1
- #define **RANSAC_MAX_ITERATIONS** 1000
- #define **RANSAC_INLIER_THRESHOLD** 0.01

Functions

- double get_cloud_resolution (const pcl::PointCloud< PointType >::ConstPtr &cloud)
- void remove_nan (pcl::PointCloud< PointType >::Ptr cloud)
- void filter_voxel_grid (const pcl::PointCloud< PointType >::ConstPtr &cloud, pcl::PointCloud< PointType >::Ptr cloud_filtered)
- void estimate_normals (const pcl::PointCloud< PointType >::ConstPtr &cloud, pcl::PointCloud< pcl::Normal >::Ptr normals)
- bool ransac_alignment (const pcl::PointCloud< PointType >::ConstPtr &cloud, const pcl::PointCloud< DescriptorType >::ConstPtr &descriptors, pcl::PointCloud< PointType >::Ptr cloud_aligned)
- void iterative_closest_point (const pcl::PointCloud< PointType >::ConstPtr &cloud)
- void ransac_correspondences (const pcl::PointCloud< PointType >::ConstPtr &keypoints, pcl::CorrespondencesPtr bestCorrespondences)

2.7.1 Detailed Description

Funciones necesarias para el mapeado, que no son de obtencion de keypoints ni de descriptores. Cabecera

Author

Ismael Piñeiro Ramos

Definition in file mapping.h.

2.7.2 Function Documentation

2.7.2.1 void estimate_normals (const pcl::PointCloud< PointType >::ConstPtr & *cloud*, pcl::PointCloud< pcl::Normal >::Ptr *normals*)

Estima las normales locales de una nube de puntos, utiliza paralelizacion mediante OpenMP

Parameters

cloud	Nube de puntos de entrada de la que estimar las normales
normals	Nube de puntos de salida de las normales estimadas

Definition at line 72 of file mapping.cpp.

2.7.2.2 void filter_voxel_grid (const pcl::PointCloud< PointType >::ConstPtr & *cloud*, pcl::PointCloud< PointType >::Ptr *cloud_filtered*)

Realiza un filtrado VoxelGrid sobre la nube de puntos de entrada, aproximando los puntos mediante sus centroides.

Parameters

cloud	Nube de puntos de entrada a filtrar
cloud_filtered	Nube de puntos de salida ya filtrada

Definition at line 58 of file mapping.cpp.

2.7.2.3 double get_cloud_resolution (const pcl::PointCloud< PointType >::ConstPtr & *cloud*)

Calcula la resolución espacial de una nube de puntos dada, mediante la media de la distancia entre cada punto y su vecino mas cercano

Parameters

cloud	Nube de puntos de la que calcular la resolución

Definition at line 23 of file mapping.cpp.

2.7.2.4 void iterative_closest_point (const pcl::PointCloud< PointType >::ConstPtr & *cloud*)

Minimiza las distancias entre los puntos de dos nubes y las transforma utilizando el algoritmo IterativeClosestPoint

Parameters

cloud	Nube de puntos de entrada

Definition at line 114 of file mapping.cpp.

2.7.2.5 bool ransac_alignment (const pcl::PointCloud< PointType >::ConstPtr & *cloud,* const pcl::PointCloud< DescriptorType >::ConstPtr & *descriptors,* pcl::PointCloud< PointType >::Ptr *cloud_aligned*)

Realiza el alineamiento de la nube de puntos actual con la anterior utilizando el algoritmo SampleConsensusPrerejective

Parameters

cloud	Nube de puntos de entrada a alinear
descriptors	Descriptores de la nube de puntos de entrada
cloud_aligned	Nube de puntos de salida alineada

Definition at line 92 of file mapping.cpp.

2.7.2.6 void ransac_correspondences (const pcl::PointCloud< PointType >::ConstPtr & *keypoints,* pcl::CorrespondencesPtr *bestCorrespondences*)

Obtiene las correspondencias y las filtra para quedarnos unicamente con las mejores correspondencias mediante el uso de RANSAC

Parameters

keypoints	nubde de puntos de entrada de la que calcular correspondencias
bestCorrespondences	correspondencias de salida calculadas

Definition at line 133 of file mapping.cpp.

2.7.2.7 void remove_nan (pcl::PointCloud< PointType >::Ptr *cloud*)

Elimina los "Not a Number" de una nube de puntos

Parameters

cloud	Nube de puntos de la que eliminar los NaN

Definition at line 45 of file mapping.cpp.

2.8 node.cpp File Reference

```
#include "node.h"
```

Functions

- void simpleVis ()
- void cloud_visualizer (const std::string &name, const pcl::PointCloud< PointType >::ConstPtr &cloud)
- void callback (const pcl::PointCloud< PointType >::ConstPtr &msg)
- double get_cpu_time (void)
- int **main** (int argc, char **argv)

2.8.1 Detailed Description

Implementacion del nodo que se encarga de procesar los mensajes recibidos de la kinect implementacion

Author

Ismael Piñeiro Ramos

Definition in file node.cpp.

2.8.2 Function Documentation

2.8.2.1 void callback (const pcl::PointCloud< PointType >::ConstPtr & *msg*)

Callback donde realizaremos los calculos correspondientes para obtener el mapeado en un visualizador

Parameters

msg	Nube de puntos recibida en el callback desde la kinect

Definition at line 30 of file node.cpp.

2.8.2.2 void cloud_visualizer (const std::string & *name*, const pcl::PointCloud< PointType >::ConstPtr & *cloud*)

Lanza un visualizador de PCL con la nube de puntos pasada por parametro

Parameters

name	Nombre de la ventana
cloud	Nube de puntos a visualizar

Definition at line 20 of file node.cpp.

2.8.2.3 double get_cpu_time (void)

Computa el tiempo de CPU actual

Returns

tiempo actual de CPU

Definition at line 95 of file node.cpp.

2.8.2.4 void simpleVis ( )

Lanza el visualizador por defecto donde se muestra el mapeado

Definition at line 10 of file node.cpp.

2.9 node.h File Reference

```
#include "common.h" #include "keypoints.h" #include "descriptors.-
h" #include "mapping.h"
```

Functions

- void callback (const pcl::PointCloud< PointType >::ConstPtr &msg)
- void simpleVis ()
- void cloud_visualizer (const std::string &name, const pcl::PointCloud< PointType >::ConstPtr &cloud)

2.9.1 Detailed Description

Implementacion del nodo que se encarga de procesar los mensajes recibidos de la kinect

Author

Ismael Piñeiro Ramos

Definition in file node.h.

2.9.2 Function Documentation

2.9.2.1 void callback (const pcl::PointCloud< PointType >::ConstPtr & *msg*)

Callback donde realizaremos los calculos correspondientes para obtener el mapeado en un visualizador

Parameters

msg	Nube de puntos recibida en el callback desde la kinect

Definition at line 30 of file node.cpp.

2.9.2.2 void cloud_visualizer (const std::string & *name,* const pcl::PointCloud< PointType >::ConstPtr & *cloud*)

Lanza un visualizador de PCL con la nube de puntos pasada por parametro

Parameters

name	Nombre de la ventana
cloud	Nube de puntos a visualizar

Definition at line 20 of file node.cpp.

2.9.2.3 void simpleVis ( )

Lanza el visualizador por defecto donde se muestra el mapeado

Definition at line 10 of file node.cpp.