Object Pose Estimation

0.1

Generated by Doxygen 1.8.2-20121118

Wed Apr 10 2013 12:25:39

Contents

Chapter 1

Namespace Index

1.1 Namespace List

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

Namespace Documentation

4.1 ope Namespace Reference

Namespace *ope* that contains all the functions and types relevant for estimating an object's pose.

Classes

- class MPoint
  Defines a 3D point that would be used by the minimization routines herein.
- class OPESettings
  User-customizable program settings.
- class ObjectPoseEstimator
  Performs object pose estimation using the parametric superquadric shape model.
- class Plane
  Defines the properties of a plane.
- class PointCloudCapture
  Captures XYZRGB point clouds from the Kinect.
- struct SQFittingThreadInfo
  Holds information relevant for multi-threaded SQ processing.
- struct ParameterLimits
  Defines the upper and lower limits for superquadric parameters.
- struct MinimizationParameters
  Defines the minimization parameters that would be used for minimization during SQ fitting.
- class SQParameters
  Defines the Superquadric parameters used for object pose estimation.
- class TableObjectDetector
  Detects clusters lying on a flat table.
- class BoundingBox
  Defines a 3D bounding box around a point cloud.
- class TableObjectModel
  Models object hypotheses found on a table.
- class ObjectModelGenerator
  Generate point cloud models for objects detected on a table.
- class Utils
  Performs frequently used utility functions.

Typedefs

- typedef double **glmma** [mma]
- typedef glmma **glnparam**
- typedef int **gllista** [mma]
- typedef double **glcovar** [mma][mma]
- typedef glcovar **glnalbynal**
- typedef glcovar **glncabynca**
- typedef glcovar **glnpbynp**
- typedef glcovar **glnpbymp**
- typedef std::vector< double > **glndata**
- typedef std::vector< MPoint > **glndata2**
- typedef gllista **glnp**
- typedef double **glmartrix** [4][4]
- typedef struct ope::SQFittingThreadInfo SQFittingThreadInfo

  Holds information relevant for multi-threaded SQ processing.
- typedef enum ope::ParameterType ParameterType

  Defines the numerical property of a superquadric parameter.
- typedef struct ope::ParameterLimits ParameterLimits

  Defines the upper and lower limits for superquadric parameters.
- typedef struct ope::MinimizationParameters MinimizationParameters

  Defines the minimization parameters that would be used for minimization during SQ fitting.

Enumerations

- enum ParameterType {
  UNCHANGED = 1, **BOUNDED**, **UNLIMITED**, **NOT_USED**,
  OFFSET }

  Defines the numerical property of a superquadric parameter.

Functions

- double mylog (double x)

  Calculates the base-10 logarithm of x.
- double mypow (double x, double y)

  Raises x to the power of y.
- double sqr (double x)

  Calculates the square root of x.
- double errorFunc (const pcl::PointCloud< pcl::PointXYZ > &cloud, SQParameters &sqParams)

  Calculates the average error of fit of superquadric parameters.
- double qualityOfFit (pcl::PointCloud< pcl::PointXYZ > &cloud, SQParameters &sqParams)

  Determines the total error (quality) of fit of superquadric parameters.
- double **funcs** (double x, double y, double z, double rotx[11], double roty[11], double rotz[11], glnparam a, glnparam dFda)
- void **precomp_rot** (glnparam a, double rotx[11], double roty[11], double rotz[11])
- double **mrqcof** (const glndata2 &x, glndata F, glndata sig, int ndata, glmma a, gllista lista, int mfit, glcovar alpha, glmma beta, int *n_model, int *n_model_acc, double addnoise)

- int **gaussj** (glcovar a, int n, glcovar b)
- double **mrqmin_init** (const glndata2 &x, glndata F, glndata sig, int ndata, glmma a, gllista lista, int mfit, glcovar alpha, int nca, int ∗n_model_acc)
- double mrqmin (SQParameters &prm, const glndata2 &x, glndata F, glndata sig, int ndata, glmma a, gllista lista, int mfit, glcovar covar, glcovar alpha, double alamda, int ∗n_model_acc)

  Levenberg Marquadt minimization.
- void initializeMinimizationParameters (SQParameters &sqParams)

  Initializes the Levenberg-Marquadt minimization parameters used for sq estimation.
- int estimateParameters (const pcl::PointCloud< pcl::PointXYZ > &cloud, SQParameters &sqParams, int eigenVector)

  Estimate the parameters of a superquadric from the given point cloud.
- int estimateParametersNew (const pcl::PointCloud< pcl::PointXYZ > &cloud, SQParameters &sqParams, int eigenVector)

  Estimate the parameters of a superquadric from the given point cloud.
- void estimateInitialParameters (pcl::PointCloud< pcl::PointXYZ > &cloud, SQParameters &sqParams)

  Estimate the initial SQ parameters based on eigen analysis.
- double recoverParameters (const pcl::PointCloud< pcl::PointXYZ > &cloud, SQParameters &prm)

  Recover the parameters of the superquadric that best fits the given point cloud.
- void performShapeFitting (const pcl::PointCloud< pcl::PointXYZ > &cloud, SQParameters &initParams, SQParameters &bestParams)

  Executes the superquadric shape fitting process on a point cloud and gets the best parameters.

Variables

- static float minimumTgtDepth = 0.2f

  Minimum depth value for target scene area. Used by the `PassThrough` *filter.*
- static float maximumTgtDepth = 1.8f

  Maximum depth value for target scene area.
- static float minimumObjHeight = 0.01f

  Minimum height of object hypotheses in the scene.
- static float maximumObjHeight = 0.30f

  Maximum height of object hypotheses in the scene.
- static bool verbose = true

  Determines whether status updates are output.
- static int minimumIterations = 30

  Determines the amount of error minimization iterations for superquadric fitting.
- static float objectVoxelSize = 0.003f

  Size of object voxels used by the Table Object Detector.
- static bool allowTapering = false

  Determines whether superquadric shape tapering is allowed when estimating pose.
- double **glochisq**
- glmma **glatry**
- glmma **glbeta**
- int **i_am_in_trouble**

4.1.1 Detailed Description

Namespace *ope* that contains all the functions and types relevant for estimating an object's pose. Namespace where all the Object Pose Estimation functionality resides

4.1.2 Function Documentation

4.1.2.1 double ope::errorFunc (const pcl::PointCloud< pcl::PointXYZ > & *cloud,* SQParameters & *sqParams*)

Calculates the average error of fit of superquadric parameters.

The error of fit is the mean of the algebraic distance calculated by the inside-outside function.

4.1.2.2 int ope::estimateParameters (const pcl::PointCloud< pcl::PointXYZ > & *cloud,* SQParameters & *sqParams,* int *eigenVector*)

Estimate the parameters of a superquadric from the given point cloud.

Returns

the eigen vector index with the largest variance

4.1.2.3 int ope::estimateParametersNew (const pcl::PointCloud< pcl::PointXYZ > & *cloud,* SQParameters & *sqParams,* int *eigenVector*)

Estimate the parameters of a superquadric from the given point cloud.

Returns

the eigen vector index with the largest variance

4.1.2.4 void ope::initializeMinimizationParameters (SQParameters & *sqParams*)

Initializes the Levenberg-Marquadt minimization parameters used for sq estimation.

Assign size parameter values based on their bounds properties

4.1.2.5 double ope::mrqmin (SQParameters & *prm,* const glndata2 & *x,* glndata *F,* glndata *sig,* int *ndata,* glmma *a,* gllista *lista,* int *mfit,* glcovar *covar,* glcovar *alpha,* double *alamda,* int * *n_model_acc*)

Levenberg Marquadt minimization.

Adding Noise

4.1.2.6 void ope::performShapeFitting (const pcl::PointCloud< pcl::PointXYZ > & *cloud,* SQParameters & *initParams,* SQParameters & *bestParams*)

Executes the superquadric shape fitting process on a point cloud and gets the best parameters.

Parameters

<*initParams*>	the initial superquadric parameters for the given cloud
<*bestParams*>	the final superquadric parameters recovered after processing

Chapter 5

Class Documentation

5.1 ope::BoundingBox Class Reference

Defines a 3D bounding box around a point cloud.

`#include <TableObjectModeler.h>`

Public Member Functions

- **BoundingBox** (float _minX, float _maxX, float _minY, float _maxY, float _minZ, float _maxZ, float _w, float _h, float _d)
- bool **isEmpty** () const
- pcl::PointXYZ **centroid** () const
- bool **isPointInside** (const pcl::PointXYZ &p) const
- bool **isPointInside2D** (const int &x, const int &y)
- float **volume** () const
- void **update** (float x, float y, float z)
- int **xPos** () const
- int **yPos** () const

Public Attributes

- float **minX**
- float **maxX**
- float **minY**
- float **maxY**
- float **minZ**
- float **maxZ**
- float **width**
- float **height**
- float **depth**

Friends

- std::ostream & **operator**<< (std::ostream &os, const BoundingBox &b)

5.1.1 Detailed Description

Defines a 3D bounding box around a point cloud.

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

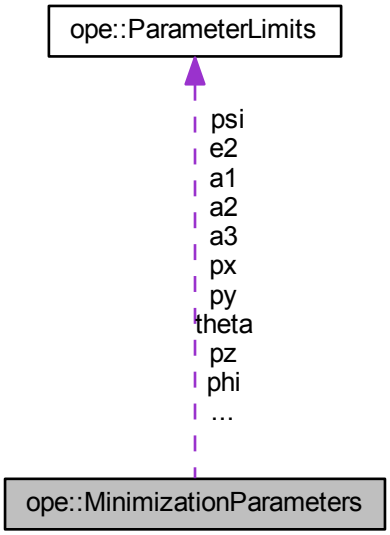
- ObjectPoseEstimation/TableObjectModeler.h

5.2 ope::MinimizationParameters Struct Reference

Defines the minimization parameters that would be used for minimization during SQ fitting.

`#include <SQTypes.h>`

Collaboration diagram for ope::MinimizationParameters:



Public Attributes

- int **iterations**
- ParameterLimits **a1**
- ParameterLimits **a2**
- ParameterLimits **a3**
- ParameterLimits **e1**
- ParameterLimits **e2**
- ParameterLimits **phi**
- ParameterLimits **theta**
- ParameterLimits **psi**

- ParameterLimits **px**
- ParameterLimits **py**
- ParameterLimits **pz**
- ParameterLimits **kx**
- ParameterLimits **ky**

5.2.1 Detailed Description

Defines the minimization parameters that would be used for minimization during SQ fitting.

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

- ObjectPoseEstimation/SQTypes.h

5.3 ope::MPoint Class Reference

Defines a 3D point that would be used by the minimization routines herein.

`#include <Minimization.h>`

Public Member Functions

- double & **operator[]** (const int &idx)
- double **operator[]** (const int &idx) const

Public Attributes

- double **point** [3]

5.3.1 Detailed Description

Defines a 3D point that would be used by the minimization routines herein.

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

- ObjectPoseEstimation/Minimization.h

5.4 ope::ObjectModelGenerator< PointType > Class Template Reference

Generate point cloud models for objects detected on a table.

`#include <TableObjectModeler.h>`

Public Types

- typedef PointCloud::Ptr **PointCloudPtr**
- typedef PointCloud::ConstPtr **PointCloudConstPtr**

Public Member Functions

- **ObjectModelGenerator** (PointCloudConstPtr ptCloudPtr)
- void **setSourceCloud** (PointCloudConstPtr ptCloudPtr)
- bool **generateObjectModels** ()
- std::vector< BoundingBox > **getBoundingBoxes** ()

Public Attributes

- EIGEN_MAKE_ALIGNED_OPERATOR_NEW
  typedef pcl::PointCloud
  < PointType > **PointCloud**
- std::vector< TableObjectModel
  < PointType >
  , Eigen::aligned_allocator
  < TableObjectModel< PointType > > > objects

  List of object models.

Private Attributes

- PointCloudConstPtr **srcCloudPtr**

5.4.1 Detailed Description

template<class PointType>class ope::ObjectModelGenerator< PointType >

Generate point cloud models for objects detected on a table.

5.4.2 Member Data Documentation

5.4.2.1 template<class PointType > std::vector< TableObjectModel< PointType >, Eigen::aligned_allocator< TableObjectModel< PointType > > > ope::ObjectModelGenerator< PointType >::objects

List of object models.

Note

This declaration is rather messy but it is required if we desire to use ANYTHING Eigen. All Eigen objects are 16-byte aligned. Therefore we must use Eigen's memory allocator when using standard containers.

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

- ObjectPoseEstimation/TableObjectModeler.h

5.5 ope::ObjectPoseEstimator Class Reference

Performs object pose estimation using the parametric superquadric shape model.

`#include <ObjectPoseEstimator.h>`

Static Public Member Functions

- static static __declspec(dllexport)
  SQParameters
  calculateObjectPose(pcl __declspec (dllexport) SQParameters run(const OPESettings &opeSettings)

  Calculate the pose of the object represented by the point cloud provided.

Static Private Member Functions

- static void init (const OPESettings &opeSettings)

  Initializes program-wide OPE properties.

5.5.1 Detailed Description

Performs object pose estimation using the parametric superquadric shape model.

Superquadrics are a family of parametric shapes that include superellipsoids, supertoroids, and superhyperboloids with one and two parts. They are appealing for robotic applications by nature of their definition. We focus on the superellipsoid which is useful for a volumetric part-based description. Given the parameters that define a superquadric, the shape and pose information can be easily extracted as well as volumes and moments of inertia. They are compact in shape and have a closed surface. Moreover, superquadrics exhibit tri-axis symmetry, which is a characteristic well approximated by many household objects.

Superquadrics can be defined in an object centered coordinate system with five variables and in a general coordinate system by eleven independent variables. The variables a_1, a_2, a_3 are the scaling dimensions along the x, y, and z axes of the superquadric, $e1, e2$ are the factors which determine the superquadric's shape ranging from from 0.1 to 1, and $(n_x, n_y, n_z, o_x, o_y, o_z, a_x, a_y, a_z, p_x, p_y, p_z)$ are the twelve parameters of the homogeneous transformation matrix that is a result of a rotation and translation of the world coordinate frame.

5.5.2 Member Function Documentation

5.5.2.1 static static __declspec (dllexport) SQParameters calculateObjectPose(pcl ope::ObjectPoseEstimator::__declspec (dllexport) const [static]

Calculate the pose of the object represented by the point cloud provided.

The pose is estimated by using the algorithm outlined in the ICRA 2013 paper **"Multi-scale Superquadric Fitting for Efficient Shape and Pose Recovery of Unknown Objects"** by Kester Duncan http://www.cse.usf.edu/~kkduncan Run the pose estimation algorithm

This function executes as follows: Capture a point cloud, extract the table objects from the point cloud, present a viewing window to the user in order for them to select the target object point cloud using its index, perform pose estimation on the object point cloud, and finally return the results in a *SQParameters* object. *

Returns

An *SQParameters* object containing the 13 superquadric parameters.

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

- ObjectPoseEstimation/ObjectPoseEstimator.h
- ObjectPoseEstimation/ObjectPoseEstimator.cpp

5.6 ObjectSelector Class Reference

Selects an object point cloud from the screen for further processing.

`#include <ObjectSelector.h>`

Private Member Functions

- void **mouseEventOccurred** (const pcl::visualization::MouseEvent &event, void *viewer_void)

5.6.1 Detailed Description

Selects an object point cloud from the screen for further processing.

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

- ObjectPoseEstimation/ObjectSelector.h
- ObjectPoseEstimation/ObjectSelector.cpp

5.7 ope::OPESettings Class Reference

User-customizable program settings.

`#include <ObjectPoseEstimator.h>`

Public Member Functions

- OPESettings ()

  Constructor.

Public Attributes

- float minTgtDepth

  Minimum depth value for target scene area. Used by the `PassThrough` filter.
- float maxTgtDepth

  Maximum depth value for target scene area.
- float minObjHeight

  Minimum height of object hypotheses in the scene.
- float maxObjHeight

  Maximum height of object hypotheses in the scene.
- bool verbose

  Determines whether status updates are output.
- int minIterations

  Determines the amount of error minimization iterations for superquadric fitting.
- bool allowTapering

  Determines whether superquadric shape tapering is allowed when estimating pose.
- float objVoxelSize

  Object voxel size.

5.7.1 Detailed Description

User-customizable program settings.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 ope::OPESettings::OPESettings ( ) [inline]

Constructor.

Postcondition

Initializes all member variables to their default values

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

- ObjectPoseEstimation/ObjectPoseEstimator.h

5.8 ope::ParameterLimits Struct Reference

Defines the upper and lower limits for superquadric parameters.

`#include <SQTypes.h>`

Public Attributes

- ParameterType **type**
- float **value**
- float **lowerBound**
- float **upperBound**

5.8.1 Detailed Description

Defines the upper and lower limits for superquadric parameters.

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

- ObjectPoseEstimation/SQTypes.h

5.9 ope::Plane Class Reference

Defines the properties of a plane.

`#include <Plane.h>`

Public Member Functions

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

- Plane (const pcl::PointXYZ &normal, const pcl::PointXYZ &p)

  Construct a plane from a normal vector and a point.
- Plane ()

  Default Constructor.
- bool isValid () const

  Determines whether or not this is a valid plane.
- pcl::PointXYZ normal () const

  Gets the normal vector that defines this plane.
- void set (double a_, double b_, double c_, double d_)

  Sets the parameters of the plane.
- pcl::PointXYZ intersectionWithLine (const pcl::PointXYZ &p1, const pcl::PointXYZ &p2) const

  Determines whether this plane intersects with the specified line given by the parameters.
- float distanceToPlane (const pcl::PointXYZ &p) const

  Determines a point's distance from the plane.

Private Attributes

- double a

  Plane parameters.
- double **b**
- double **c**
- double **d**

5.9.1 Detailed Description

Defines the properties of a plane.

5.9.2 Member Function Documentation

5.9.2.1 float Plane::distanceToPlane (const pcl::PointXYZ & *p*) const

Determines a point's distance from the plane.

Returns

Distance from the plane

5.9.2.2 pcl::PointXYZ Plane::intersectionWithLine (const pcl::PointXYZ & *p1*, const pcl::PointXYZ & *p2*) const

Determines whether this plane intersects with the specified line given by the parameters.

Parameters

<*p1*>	the first point that defines the line
<*p2*>	the second point that defines the line

5.9.2.3 bool Plane::isValid ( ) const

Determines whether or not this is a valid plane.

Returns

true if it is valid

5.9.2.4 pcl::PointXYZ Plane::normal ( ) const

Gets the normal vector that defines this plane.

Returns

the x, y, & z values of the plane's normal

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

- ObjectPoseEstimation/Plane.h
- ObjectPoseEstimation/Plane.cpp

5.10 PointCloud< PointXYZRGB > Class Template Reference

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

- ObjectPoseEstimation/ObjectPoseEstimator.h

5.11 ope::PointCloudCapture Class Reference

Captures XYZRGB point clouds from the Kinect.

`#include <PointCloudCapture.h>`

Public Member Functions

- void run (pcl::PointCloud< pcl::PointXYZRGB > &ptCloud)

  Captures an XYZRGBA point cloud from the Kinect and stores an XYZRGB cloud.
- void cloudCallback (const pcl::PointCloud< pcl::PointXYZRGBA >::ConstPtr &cloud)

  Callback function that is used to read XYZRGB point clouds from the Kinect.

Private Attributes

- pcl::PointCloud
  < pcl::PointXYZRGBA >
  ::ConstPtr **ptCloudPtr**
- pcl::PointCloud
  < pcl::PointXYZRGBA >::Ptr **filteredPtCloudPtr**

5.11.1 Detailed Description

Captures XYZRGB point clouds from the Kinect.

Note

The coordinate system for the captured point clouds is as follows: x-axis -> right, y-axis -> down, z-axis -> points into scene.

Also, a PassThrough filter is applied to the captured point cloud so that the target area is within a specified range which is adjustable

5.11.2 Member Function Documentation

5.11.2.1 void PointCloudCapture::run (pcl::PointCloud< pcl::PointXYZRGB > & *ptCloud*)

Captures an XYZRGBA point cloud from the Kinect and stores an XYZRGB cloud.

Parameters

<*ptCloud*>	- holds the captured PointXYZRGB point cloud

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

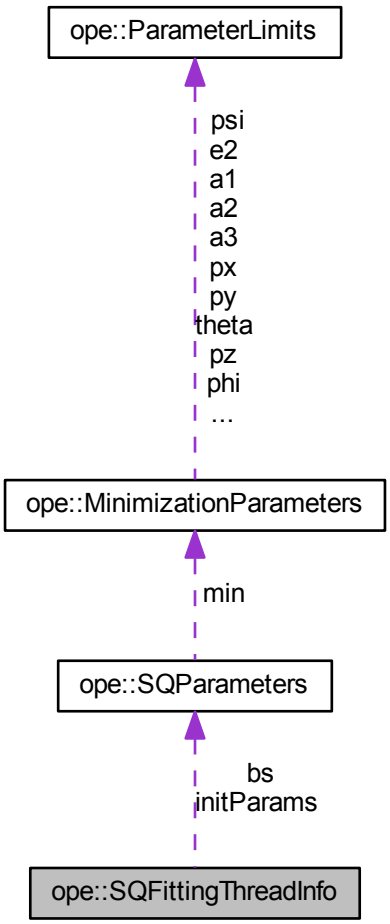
- ObjectPoseEstimation/PointCloudCapture.h
- ObjectPoseEstimation/PointCloudCapture.cpp

5.12 ope::SQFittingThreadInfo Struct Reference

Holds information relevant for multi-threaded SQ processing.

`#include <SQFitting.h>`

Collaboration diagram for ope::SQFittingThreadInfo:



Public Attributes

- int **threadId**
- pcl::PointCloud< pcl::PointXYZ > **cloud**
- SQParameters **bs**
- SQParameters **initParams**

5.12.1 Detailed Description

Holds information relevant for multi-threaded SQ processing.

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

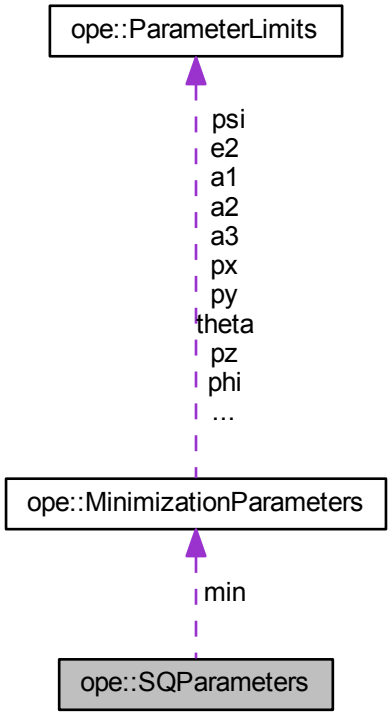
- ObjectPoseEstimation/SQFitting.h

5.13 ope::SQParameters Class Reference

Defines the Superquadric parameters used for object pose estimation.

`#include <SQTypes.h>`

Collaboration diagram for ope::SQParameters:



Public Member Functions

- SQParameters ()

  Default Constructor.
- void copyTo (SQParameters &sqParams)

  Deep copy of this class' properties.

Public Attributes

- float a1

  The shape dimension for the x-axis.
- float a2

  The shape dimension for the y-axis.

- float a3

  The shape dimension for the z-axis.
- float e1

  The north-south superquadric shape parameter.
- float e2

  The east-west superquadric shape parameter.
- float px

  The x-axis location of the centroid of this superquadric.
- float py

  The y-axis location of the centroid of this superquadric.
- float pz

  The z-axis location of the centroid of this superquadric.
- float phi

  Euler rotation angle along the x-axis.
- float theta

  Euler rotation angle along the y-axis.
- float psi

  Euler rotation angle along the x-axis.
- float kx

  Tapering parameter along the x-axis.
- float ky

  Tapering parameter along the y-axis.
- int principalAxis

  Index of the principal axis in the calculated rotation matrix.
- int **majorAxis**
- int **minorAxis**
- MinimizationParameters min

  Minimization properties.

Friends

- std::ostream & operator<< (std::ostream &os, const SQParameters &sq)

  Prints the superquadric parameters to an output stream.

5.13.1 Detailed Description

Defines the Superquadric parameters used for object pose estimation.

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

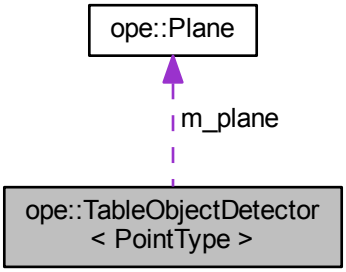
- ObjectPoseEstimation/SQTypes.h

5.14 ope::TableObjectDetector< PointType > Class Template Reference

Detects clusters lying on a flat table.

`#include <TableObjectDetector.h>`

Collaboration diagram for ope::TableObjectDetector< PointType >:



Public Types

- typedef pcl::PointCloud < PointType > **PointCloud**
- typedef PointCloud::ConstPtr **PointCloudConstPtr**
- typedef pcl::search::KdTree < PointType >::Ptr **KdTreePtr**
- typedef std::vector< PointType, Eigen::aligned_allocator < PointType > > **PointVector**

Public Member Functions

- TableObjectDetector ()

  Constructor.
- void initialize ()

  Initialize the properties of the TableObjectDetector.
- float voxelSize () const

  Gets the voxel size used for downsampling.
- void setObjectVoxelSize (float s=0.003)

  Sets the voxel size used for object downsampling. Default size is 0.3 cm.
- void setBackgroundVoxelSize (float s=0.01)

  Sets the voxel size used for background downsampling. Default size is 1.0 cm.
- void setDepthLimits (float min_z=0.4, float max_z=2.0)

  Sets the depth limits for processing.
- void setObjectHeightLimits (float min_h=0.01f, float max_h=0.5f)

Sets the height limits for detected objects.

- void setMaxDistToPlane (float d)

  Sets the threshold for an object's distance from the plane (table)
- bool **detect** (PointCloudConstPtr cloud)
- const Plane & plane () const

  Gets a constant reference to the plane properties.
- const std::vector< PointVector > & objectClusters () const

  Gets a constant reference to the list of objects detected on the table.
- PointCloudConstPtr tableInliers () const

  Gets a constant pointer to the table points.

Public Attributes

- EIGEN_MAKE_ALIGNED_OPERATOR_NEW
  typedef PointType **Point**

Private Attributes

- KdTreePtr **normals_tree_**
- KdTreePtr **clusters_tree_**
- pcl::PassThrough< Point > **pass_**
- pcl::VoxelGrid< Point > **grid_**
- pcl::VoxelGrid< Point > **grid_objects_**
- pcl::NormalEstimation< Point,
  pcl::Normal > **n3d_**
- pcl::SACSegmentationFromNormals
  < Point, pcl::Normal > **seg_**
- pcl::ProjectInliers< Point > **proj_**
- pcl::ConvexHull< Point > **hull_**
- pcl::ExtractPolygonalPrismData
  < Point > **prism_**
- pcl::EuclideanClusterExtraction
  < Point > **cluster_**
- double **downsample_leaf_**
- double **downsample_leaf_objects_**
- int **k_**
- double **min_z_bounds_**
- double **max_z_bounds_**
- double **sac_distance_threshold_**
- double **normal_distance_weight_**
- double **object_min_height_**
- double **object_max_height_**
- double **object_cluster_tolerance_**
- double **object_cluster_min_size_**
- double **m_max_dist_to_plane**
- PointCloudConstPtr **cloud_**
- PointCloudConstPtr **cloud_filtered_**
- PointCloudConstPtr **cloud_downsampled_**
- pcl::PointCloud< pcl::Normal >
  ::ConstPtr **cloud_normals_**

- pcl::PointIndices::ConstPtr **table_inliers_**
- pcl::ModelCoefficients::ConstPtr **table_coefficients_**
- PointCloudConstPtr **table_projected_**
- PointCloudConstPtr **table_hull_**
- PointCloudConstPtr **cloud_objects_**
- PointCloudConstPtr **cloud_objects_downsampled_**
- Plane m_plane

  Represents the table plane.
- std::vector< PointVector > m_object_clusters

  The object cluster which are found after detection.

5.14.1 Detailed Description

template<class PointType>class ope::TableObjectDetector< PointType >

Detects clusters lying on a flat table.

Note

This class is adapted from ntk's TableObjectDetector

Warning

Ensure that the table is large enough to be distinguished from the ground plane

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

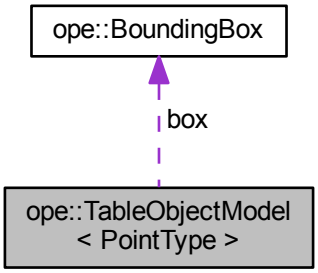
- ObjectPoseEstimation/TableObjectDetector.h
- ObjectPoseEstimation/TableObjectDetector.hpp

5.15 ope::TableObjectModel< PointType > Class Template Reference

Models object hypotheses found on a table.

`#include <TableObjectModeler.h>`

Collaboration diagram for ope::TableObjectModel< PointType >:



Public Member Functions

- void create (const TableObjectDetector< PointType > &detector, const size_t &clusterId)

  Create the object model.

Public Attributes

- EIGEN_MAKE_ALIGNED_OPERATOR_NEW
  pcl::PointCloud< PointType > **objectCloud**
- BoundingBox **box**
- size_t objectId

  A cluster object's unique id according to the order it was detected.
- std::string name

  Cluster's identifier.

5.15.1 Detailed Description

template<class PointType>class ope::TableObjectModel< PointType >

Models object hypotheses found on a table.

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

- ObjectPoseEstimation/TableObjectModeler.h

5.16 ope::Utils Class Reference

Performs frequently used utility functions.

`#include <Utils.h>`

Static Public Member Functions

- static void convertPointCloud (const pcl::PointCloud< pcl::PointXYZRGBA > &src, pcl::PointCloud< pcl::PointXYZRGB > &tgt)

  Convert between pcl point cloud types.
- static void transformPointCloud (pcl::PointCloud< pcl::PointXYZRGB > &cloud)

  Transform an XYZRGB point cloud from the Kinect optical frame to world coordinate frame.
- static void printCurrentDateTime ()

  Prints the current local time to the output stream.
- static size_t getDesiredObject (pcl::PointCloud< pcl::PointXYZRGB >::ConstPtr ptCloudPtr, const std::vector< BoundingBox > &boxes)

  Displays a pointcloud with highlighted objects in order to determine the user's object of choice.

5.16.1 Detailed Description

Performs frequently used utility functions.

Author

Kester Duncan

5.16.2 Member Function Documentation

5.16.2.1 void Utils::convertPointCloud (const pcl::PointCloud< pcl::PointXYZRGBA > & *src*, pcl::PointCloud< pcl::PointXYZRGB > & *tgt*) [static]

Convert between pcl point cloud types.

Converts from pcl::PointXYZRGBA to pcl::PointXYZRGB

Parameters

<*src*>	- input `pcl::PointXYZRGBA` point cloud
<*tgt*>	- output `pcl::PointXYZRGB` point cloud

5.16.2.2 void Utils::transformPointCloud (pcl::PointCloud< pcl::PointXYZRGB > & *cloud*) [static]

Transform an XYZRGB point cloud from the Kinect optical frame to world coordinate frame.

Parameters

<*cloud*>	- the cloud to be transformed

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

- ObjectPoseEstimation/Utils.h
- ObjectPoseEstimation/Utils.cpp

Chapter 6

File Documentation

6.1 ObjectPoseEstimation/Main.cpp File Reference

Main entry point for Object Pose Estimation.

6.1.1 Detailed Description

Main entry point for Object Pose Estimation.

Author

Kester Duncan

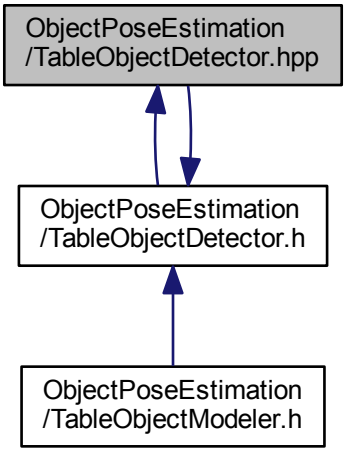
This file is the main entry point for using the object pose estimator using superquadrics by Kester Duncan

6.2 ObjectPoseEstimation/TableObjectDetector.hpp File Reference

```
#include "TableObjectDetector.h"
#include <boost/make_shared.hpp>
```

Include dependency graph for TableObjectDetector.hpp:

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



Namespaces

- namespace ope

  Namespace ope that contains all the functions and types relevant for estimating an object's pose.

6.2.1 Detailed Description

Author

Nicolas Burrus, altered by Kester Duncan

Note

Adapted from ntk's TableObjectDetector

This implementation file is used to separate the declared templates from the implementation but at the same time, make the implementation of the templates visible to the compiler. This is so because if the implementation is done in a $*$.cpp file, the compiler cannot explicitly instantiate a new class given the template argument. The implementation must be found with the declaration, which is a stupid but necessary C++ limitation.

Index