PCL :: Filtering

Filtering 25, 2011
Introduction

- irregular density (2.5D)
- occlusions
- massive amount of data
- noise
Modifying the point cloud or point attributes.
Overview Filtering

Modifying the point cloud or point attributes.

- Removing Points:
  - Conditional Removal
  - Radius/Statistical Outlier Removal
  - Color Filtering
  - Passthrough

- Downsampling:
  - Voxelgrid Filter
  - approximate Voxelgrid filtering
Modifying the point cloud or point attributes.

- **Removing Points:**
  - Conditional Removal
  - Radius/Statistical Outlier Removal
  - Color Filtering
  - Passthrough

- **Downsampling:**
  - Voxelgrid Filter
  - approximate Voxelgrid filtering
Modifying the point cloud or point attributes.

- **Removing Points:**
  - Conditional Removal
  - Radius/Statistical Outlier Removal
  - Color Filtering
  - Passthrough

- **Downsampling:**
  - Voxelgrid Filter
  - approximate Voxelgrid filtering

- **Modifying Other Point Attributes:**
  - Contrast
  - Bilateral Filtering
All filters are derived from the \textbf{Filter} base class with following interface:

\begin{verbatim}
template<typename PointT> class Filter : public PCLBase<PointT>
{
public:
    Filter (bool extract_removed_indices = false);
    inline IndicesConstPtr const getRemovedIndices () const;
    inline void setFilterFieldName (const std::string &field_name);
    inline std::string const getFilterFieldName ();
    inline void setFilterLimits (const double &limit_min, const double &limit_max);
    inline void getFilterLimits (double &limit_min, double &limit_max);
    inline void setFilterLimitsNegative (const bool limit_negative);
    inline bool getFilterLimitsNegative ();
    inline void filter (PointCloud &output);
};
\end{verbatim}
Removes points where values of selected field are out of range.

```
// point cloud instance for the result
PointCloudPtr thresholded (new PointCloud);

// create passthrough filter instance
pcl::PassThrough<PointT> pass_through;

// set input cloud
pass_through.setInputCloud (input);

// set fieldname we want to filter over
pass_through.setFilterFieldName ("z");

// set range for selected field to 1.0 - 1.5 meters
pass_through.setFilterLimits (1.0, 1.5);

// do filtering
pass_through.filter (*thresholded);
```
Example: Passthrough Filter

- Original pointcloud: `robot1.pcd`
- Passthrough on the z axis 1.0 m - 1.5m
Divides the space into discrete cells (voxels) and replaces all points within a voxel by their centroids.

```cpp
// point cloud instance for the result
PointCloudPtr downsampled (new PointCloud);

// create passthrough filter instance
pcl::VoxelGrid<PointT> voxel_grid;

// set input cloud
voxel_grid.setInputCloud (input);

// set cell/voxel size to 0.1 meters in each dimension
voxel_grid.setLeafSize (0.1, 0.1, 0.1);

// do filtering
voxel_grid.filter (*downsampled);
```
VoxelGrid Filter

voxelgrid with 0.1m voxel size in each dimension
Removes all points with less than a given number of neighbors within a radius

1  // point cloud instance for the result
2  PointCloudPtr cleaned (new PointCloud);
3
4  // create the radius outlier removal filter
5  pcl::RadiusOutlierRemoval<pcl::PointXYZRGB> radius_outlier_removal;
6
7  // set input cloud
8  radius_outlier_removal.setInputCloud (input);
9
10  // set radius for neighbor search
11  radius_outlier_removal.setRadiusSearch (0.05);
12
13  // set threshold for minimum required neighbors neighbors
14  radius_outlier_removal.setMinNeighborsInRadius (800);
15
16  // do filtering
17  radius_outlier_removal.filter (*cleaned);
Open the file test_filtering.cpp

```cpp
pcl::console::printinfo ("-t mindepth,maxdepth ...... Threshold depthn");
pcl::console::printinfo ("-d leafsize ............... Downsamplen");
pcl::console::printinfo ("-r radius,minneighbors ..... Radius outlier removaln");
pcl::console::printinfo ("-s output.pcd ............... Save outputn");
...
cloud = thresholdDepth (cloud, mindepth, maxdepth);
...
cloud = downsample (cloud, leafsize);
...

cloud = removeOutliers (cloud, radius, (int)minneighbors);
```