Filtering a PointCloud using a PassThrough filter

In this tutorial we will learn how to perform a simple filtering along a specified dimension – that is, cut off values that are either inside or outside a given user range.

The code

First, create a file, let's say, `passthrough.cpp` in your favorite editor, and place the following inside it:

```
  1 #include <iostream>
  2 #include <pcl/point_types.h>
  3 #include <pcl/filters/passthrough.h>
  4
  5 int
  6  main ()
  7 {
  8   pcl::PointCloud<pcl::PointXYZ>::Ptr cloud (new pcl::PointCloud<pcl::PointXYZ>);
  9   pcl::PointCloud<pcl::PointXYZ>::Ptr cloud_filtered (new
pcl::PointCloud<pcl::PointXYZ>);
 10
 11   // Fill in the cloud data
 12   cloud->width  = 5;
 13   cloud->height = 1;
 14   cloud->points.resize (cloud->width * cloud->height);
 15
 16   for (auto& point: *cloud)
 17   {
 18     point.x = 1024 * rand () / (RAND_MAX + 1.0f);
 19     point.y = 1024 * rand () / (RAND_MAX + 1.0f);
 20     point.z = 1024 * rand () / (RAND_MAX + 1.0f);
 21   }
 22
 23   std::cerr << "Cloud before filtering: " << std::endl;
 24   for (const auto& point: *cloud)
 25     std::cerr << "    " << point.x << " "
 26                         << point.y << " "
 27                         << point.z << std::endl;
 28
 29   // Create the filtering object
 30   pcl::PassThrough<pcl::PointXYZ> pass;
 31   pass.setInputCloud (cloud);
 32   pass.setFilterFieldName ("z");
 33   pass.setFilterLimits (0.0, 1.0);
 34   //pass.setNegative (true);
 35   pass.filter (*cloud_filtered);
 36
 37   std::cerr << "Cloud after filtering: " << std::endl;
 38   for (const auto& point: *cloud_filtered)
 39     std::cerr << "    " << point.x << " "
 40                         << point.y << " "
 41                         << point.z << std::endl;
 42
 43   return (0);
 44 }
```

The explanation

Now, let's break down the code piece by piece.

In the following lines, we define the Point Clouds structures, fill in the input cloud, and display its content to screen.

```
// Fill in the cloud data
cloud->width  = 5;
cloud->height = 1;
cloud->points.resize (cloud->width * cloud->height);

for (auto& point: *cloud)
{
  point.x = 1024 * rand () / (RAND_MAX + 1.0f);
  point.y = 1024 * rand () / (RAND_MAX + 1.0f);
  point.z = 1024 * rand () / (RAND_MAX + 1.0f);
}

std::cerr << "Cloud before filtering: " << std::endl;
for (const auto& point: *cloud)
  std::cerr << "    " << point.x << " "
                      << point.y << " "
                      << point.z << std::endl;
```

Then, we create the PassThrough filter object, and set its parameters. The filter field name is set to the z coordinate, and the accepted interval values are set to (0.0;1.0).

```
pcl::PassThrough<pcl::PointXYZ> pass;
pass.setInputCloud (cloud);
pass.setFilterFieldName ("z");
pass.setFilterLimits (0.0, 1.0);
//pass.setNegative (true);
pass.filter (*cloud_filtered);
```

Finally we show the content of the filtered cloud.

```
std::cerr << "Cloud after filtering: " << std::endl;
for (const auto& point: *cloud_filtered)
  std::cerr << "    " << point.x << " "
                      << point.y << " "
                      << point.z << std::endl;
```

Compiling and running the program

Add the following lines to your CMakeLists.txt file:

```
cmake_minimum_required(VERSION 3.5 FATAL_ERROR)

project(passthrough)

find_package(PCL 1.2 REQUIRED)

include_directories(${PCL_INCLUDE_DIRS})
link_directories(${PCL_LIBRARY_DIRS})
add_definitions(${PCL_DEFINITIONS})

add_executable (passthrough passthrough.cpp)
target_link_libraries (passthrough ${PCL_LIBRARIES})
```

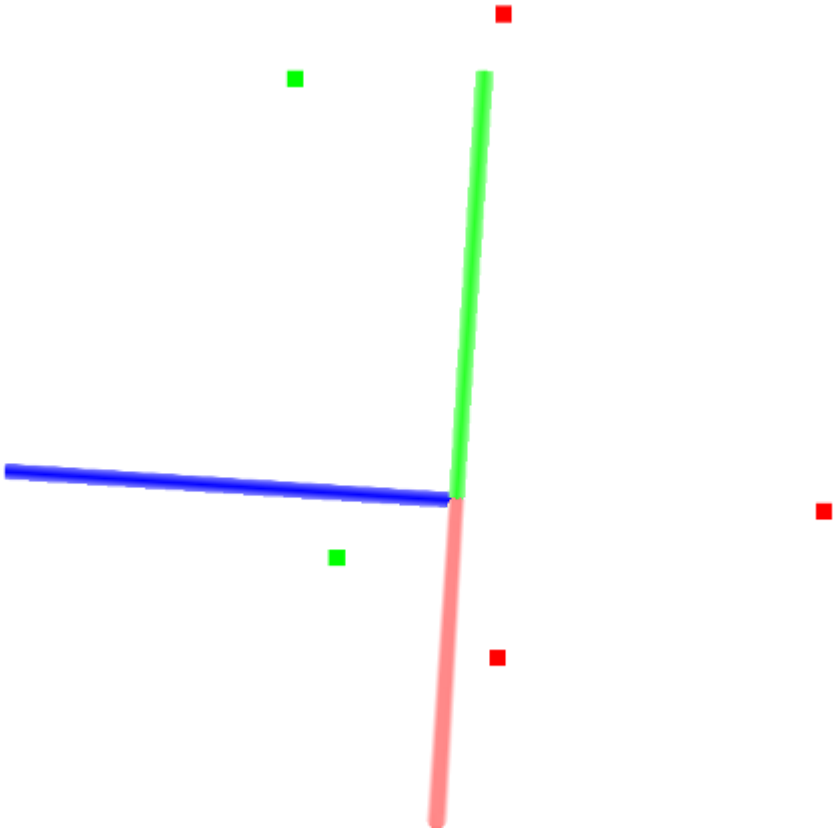
After you have made the executable, you can run it. Simply do:

```
$ ./passthrough
```

You will see something similar to:

```
Cloud before filtering:
    0.352222 -0.151883 -0.106395
    -0.397406 -0.473106 0.292602
    -0.731898 0.667105 0.441304
    -0.734766 0.854581 -0.0361733
    -0.4607 -0.277468 -0.916762
Cloud after filtering:
    -0.397406 -0.473106 0.292602
    -0.731898 0.667105 0.441304
```

A graphical display of the filtering process is shown below.



Note that the coordinate axes are represented as red (x), green (y), and blue (z). The five points are represented with green as the points remaining after filtering and red as the points that have been removed by the filter.

As an exercise, try uncommenting this line

```
//pass.setNegative (true);
```

and run the program again.