```cpp
#include <pcl/point_cloud.h>
#include <pcl/point_types.h>
#include <pcl/io/openni_grabber.h>
#include <pcl/visualization/cloud_viewer.h>

#include <pcl/compression/octree_pointcloud_compression.h>

#include <stdio.h>
#include <sstream>
#include <stdlib.h>

#ifdef WIN32
# define sleep(x) Sleep((x)*1000)
#endif

class SimpleOpenNIViewer
{
public:
  SimpleOpenNIViewer () :
    viewer (" Point Cloud Compression Example")
  {
  }

  void
  cloud_cb_ (const pcl::PointCloud<pcl::PointXYZRGBA>::ConstPtr &cloud)
  {
    if (!viewer.wasStopped ())
    {
      // stringstream to store compressed point cloud
      std::stringstream compressedData;
      // output pointcloud
      pcl::PointCloud<pcl::PointXYZRGBA>::Ptr cloudOut (new
pcl::PointCloud<pcl::PointXYZRGBA> ());

      // compress point cloud
      PointCloudEncoder->encodePointCloud (cloud, compressedData);

      // decompress point cloud
      PointCloudDecoder->decodePointCloud (compressedData, cloudOut);


      // show decompressed point cloud
      viewer.showCloud (cloudOut);
    }
  }

  void
  run ()
  {

    bool showStatistics = true;

    // for a full list of profiles see: /io/include/pcl/compression/
compression_profiles.h
```

```
    pcl::octree::compression_Profiles_e compressionProfile =
pcl::octree::MED_RES_ONLINE_COMPRESSION_WITH_COLOR;

    // instantiate point cloud compression for encoding and decoding
    PointCloudEncoder = new
pcl::octree::PointCloudCompression<pcl::PointXYZRGBA>
(compressionProfile, showStatistics);
    PointCloudDecoder = new
pcl::octree::PointCloudCompression<pcl::PointXYZRGBA> ();

    // create a new grabber for OpenNI devices
    pcl::Grabber* interface = new pcl::OpenNIGrabber ();

    // make callback function from member function
    boost::function<void
    (const pcl::PointCloud<pcl::PointXYZRGBA>::ConstPtr&)> f =
boost::bind (&SimpleOpenNIViewer::cloud_cb_, this, _1);

    // connect callback function for desired signal. In this case its a
point cloud with color values
    boost::signals2::connection c = interface->registerCallback (f);

    // start receiving point clouds
    interface->start ();

    while (!viewer.wasStopped ())
    {
      sleep (1);
    }

    interface->stop ();

    // delete point cloud compression instances
    delete (PointCloudEncoder);
    delete (PointCloudDecoder);

  }

  pcl::visualization::CloudViewer viewer;

  pcl::octree::PointCloudCompression<pcl::PointXYZRGBA>*
PointCloudEncoder;
  pcl::octree::PointCloudCompression<pcl::PointXYZRGBA>*
PointCloudDecoder;

};

int
main (int argc, char **argv)
{
  SimpleOpenNIViewer v;
  v.run ();

  return (0);
```

```
}
```