VTK/PCL File Conversions

Release 0.00

David Doria and Jaudiel Velez-robles

July 25, 2011

Army Research Laboratory, Aberdeen MD and Army Geospatial Agency

Abstract

This document presents a set of simple conversions between the Point Cloud Library's PCD file format and the Visualization Toolkit's VTP file format.

The code is available here:

`https://github.com/daviddoria/VTK_PCL_Conversions`

Contents

1 Introduction

This document presents a set of simple conversions between the Point Cloud Library's (PCL) Point Cloud Data (PCD) file format and the Visualization Toolkit's (VTK) PolyData (VTP) file format. In PCL, point clouds are represented as a collection of coordinates along with optional additional attributes. PCL provides some standard sets of attributes in the form of structs that include PointXYZ for coordinate-only points,

PointXYZRGB for colored points, and PointXYZRGBNormal for colored points with associated normal vectors. In VTK, these different data attributes are not explicitly grouped, but rather one can add any number of any type of attribute to the base coordinate data. That is, for simple coordinate-only points, a vtkPolyData objects only containing a vtkPoints is used. For colored points, a vtkPoints object with an associated vtkUnsignedCharArray is used. For colored points with associated normal vectors, a vtkPoints object with associated vtkUnsignedCharArray for the colors and a vtkFloatArray for the normals is used. These equivalent representations are summarized in the table below.

PCL	VTK
PointXYZ	vtkPoints
PointXYZRGB	vtkPoints + vtkUnsignedCharArray
PointXYZRGBNormal	vtkPoints + vtkUnsignedCharArray + vtkFloatArray
PointNormal	vtkPoints + vtkFloatArray

2 VTK to PCD

We provide a simple, templated interface to the conversions function.

```
template <typename PointT>
void VTKtoPCL(vtkPolyData* pdata, typename pcl::PointCloud<PointT>::Ptr cloud)
```

The generic template takes any vtkPolyData point-only pcl::PointCloud. This generic template handles the case of PointXYZ. Specializations handle the cases of PointXYZRGB, PointXYZRGBNormal, and PointNormal by additionally adding the extra attributes to the PointT struct.

2.1 Code Snippet

```
// Read the VTP file
vtkSmartPointer<vtkXMLPolyDataReader> reader = vtkSmartPointer<vtkXMLPolyDataReader>::New();
reader->SetFileName(inputFileName.c_str());
reader->Update();

// Create the output point cloud
pcl::PointCloud<pcl::PointXYZ>::Ptr cloud(new pcl::PointCloud<pcl::PointXYZ>);

// Convert the data
VTKtoPCL<pcl::PointXYZ>(reader->GetOutput(), cloud);

// Save the output
pcl::io::savePCDFileASCII (outputFileName.c_str(), *cloud);
```

3 PCD to VTK

We provide a simple, templated interface to the conversions function.

```
template <typename PointT>
void PCLtoVTK(typename pcl::PointCloud<PointT>::Ptr cloud, vtkPolyData* pdata)
```

The generic template takes any PCL point struct that has .x, .y, and .z members available and converts it to a point-only vtkPolyData. This generic template handles the case of PointXYZ. Specializations handle the cases of PointXYZRGB, PointXYZRGBNormal, and PointNormal by additionally adding the extra attributes to the vtkPolyData.

3.1 Code Snippet

```
// Load the PCL PCD file
pcl::PointCloud<pcl::PointXYZ>::Ptr cloud (new pcl::PointCloud<pcl::PointXYZ>);
pcl::io::loadPCDFile<pcl::PointXYZ> (inputFileName.c_str(), *cloud);

// Create a polydata object.
vtkSmartPointer<vtkPolyData> polydata = vtkSmartPointer<vtkPolyData>::New();

// Convert the PCL data to VTK data
PCLtoVTK<pcl::PointXYZ>(cloud, polydata);

// Write the output
vtkSmartPointer<vtkXMLPolyDataWriter> writer =
  vtkSmartPointer<vtkXMLPolyDataWriter>::New();
writer->SetFileName(outputFileName.c_str());
writer->SetInputConnection(polydata->GetProducerPort());
writer->Write();
```