Planar Patch Construction

Release 0.00

David Doria

July 27, 2011

Rensselaer Polytechnic Institute

Abstract

This document presents code to create a mesh of a piece of a plane that is defined by a reasonably planar point set. This is useful when visualizing the output of point cloud segmentation algorithms.
The code is available here: https://github.com/daviddoria/vtkPlanarPatch

Contents

1 Introduction

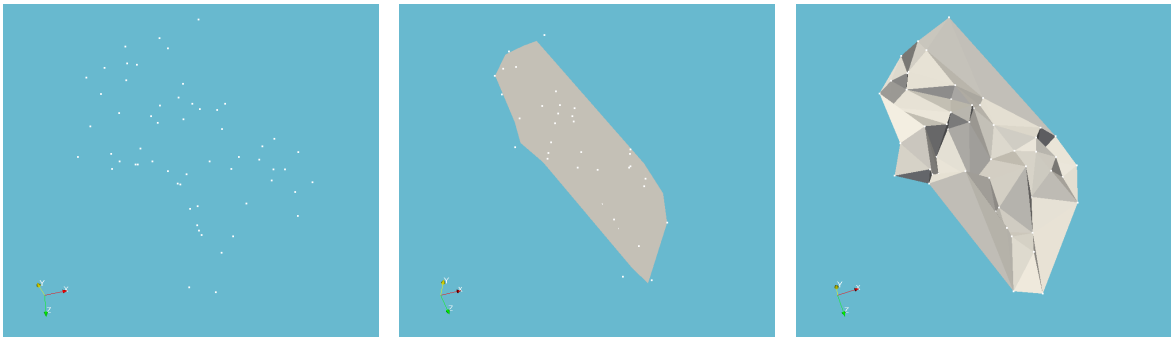
This document presents code to create a mesh of a piece of a plane that is defined by a reasonably planar point set.

2 Algorithm

- Project the approximately planar 3D points onto their best fit plane
- Find a coordinate system on the best fit plane. This is done using the following procedure:
  - Find one vector in the plane, v_0, by subtracting any two points: $v_0 = p_1 - p_0$. Normalize this vector.
  - Find a second vector in the plane, v_1, by computing the cross product of the plane's normal and v_0. Normalize this vector.
  - Create a set of three points that define a 2D coordinate system (origin and two points unit distance along each of the axes) for the local coordinate system that was just found and for the world coordinate system. For the world coordinate system, these points are simply $(0,0,0)$, $(1,0,0)$, and $(0,1,0)$. For the local coordinate system, they are p_0, (p_0+v_0), and (p_0+v_1).
- Compute the transform from the local coordinate system to the world coordinate system (using a vtkLandmarkTransform).
- Compute the 2D Delaunay triangulation on the transformed points. It is not necessary to actually transform the points, the transform can simply be passed to vtkDelaunay2D.
- If the user has specified the FlatOutput flag, the topology of the resulting triangulation will be applied to the projected points (the original 3D points projected on their best fit plane) and passed to the output.
- If the user has not specified the FlatOutput flag, the topology of the triangulation will be applied to the original 3D points and passed to the output.

3 Demonstration

In Figure 1, we show the result of the planar patch computation for both FlatOutput and non FlatOutput.



(a) Original, reasonably planar points. (b) The flat output patch. (c) The non flat output patch.

Figure 1: Planar patch computation demonstration.

4 Code Snippet

```
// ... compute the best fit plane of the points ...

vtkSmartPointer<vtkPlane> bestPlane =
  vtkSmartPointer<vtkPlane>::New();
bestPlane->SetNormal(bestNormal);
bestPlane->SetOrigin(bestOrigin);

vtkSmartPointer<vtkPlanarPatch> planarPatch =
    vtkSmartPointer<vtkPlanarPatch>::New();
planarPatch->SetPlane(bestPlane);
planarPatch->SetFlatOutput(false);
planarPatch->SetInputConnection(inputPoints->GetProducerPort());
planarPatch->Update();

// Write the output
vtkSmartPointer<vtkXMLPolyDataWriter> writer =
  vtkSmartPointer<vtkXMLPolyDataWriter>::New();
writer->SetFileName("output.vtp");
writer->SetInputConnection(planarPatch->GetOutputPort());
writer->Write();
```