Poisson Surface Reconstruction for VTK

Release 0.00

David Doria and Arnaud Gelas

March 5, 2010

Abstract

This document presents an implementation of the Poission surface reconstruction algorithm in the VTK framework. (This code was adapted directly from the original implementation by Misha Kazhdan [1], with permission). We present a class, *vtkPoissonReconstruction*, which produces a surface from an oriented point set. A Paraview plugin interface is provided to allow extremely easy experimentation with the new functionality. We propose these classes as an addition to the Visualization Toolkit.

Latest version available at the Insight Journal [http://hdl.handle.net/10380/3149]

Contents

1 Introduction

There are several data acquisition methods which produce 3D points as output. Examples include Light Detection and Ranging (LiDAR) scanners, Structure From Motion (SFM) algorithms, and Multi View Stereo (MVS) algorithms. It is often producing a mesh from these points if often of interest. This paper presents a simple algorithm for producing a surface from a set of 3D points which have an associated normal vector. This method was originally published in [1], we have simply implemented it in the VTK framework.

2 vtkPoissonReconstruction

2.1 Options

Depth

This integer controls the reconstruction depth; the maximum depth of the tree that will be used for surface reconstruction. Running at depth d corresponds to solving on a voxel grid whose resolution is no larger than $2^d \times 2^d \times 2^d$. Note that since the reconstructor adapts the octree to the sampling density, the specified reconstruction depth is only an upper bound. The default value for this parameter is 8.

This value can be set using:

```
reconstructionFilter->SetDepth(10);
```

Scale

This floating point value specifies the ratio between the diameter of the cube used for reconstruction and the diameter of the samples' bounding cube. The default value is 1.25.

This value can be set using:

```
reconstructionFilter->SetScale(1.1);
```

SolverDivide

Solver subdivision depth; This integer argument specifies the depth at which a block Gauss-Seidel solver is used to solve the Laplacian equation. Using this parameter helps reduce the memory overhead at the cost of a small increase in reconstruction time. (In practice, we have found that for reconstructions of depth 9 or higher a subdivide depth of 7 or 8 can greatly reduce the memory usage.) The default value is 8.

This value can be set using:

```
reconstructionFilter->SetSolverDivide(7);
```

IsoDivide

Iso-surface extraction subdivision depth; This integer argument specifies the depth at which a block iso-surface extractor should be used to extract the iso-surface. Using this parameter helps reduce the memory

overhead at the cost of a small increase in extraction time. (In practice, we have found that for reconstructions of depth 9 or higher a subdivide depth of 7 or 8 can greatly reduce the memory usage.) The default value is 8.

This value can be set using:

```
reconstructionFilter->SetIsoDivide(9);
```

SamplesPerNode

Minimum number of samples; This floating point value specifies the minimum number of sample points that should fall within an octree node as the octree construction is adapted to sampling density. For noise-free samples, small values in the range [1.0 - 5.0] can be used. For more noisy samples, larger values in the range [15.0 - 20.0] may be needed to provide a smoother, noise-reduced, reconstruction. The default value is 1.0.

This value can be set using:

```
reconstructionFilter->SetSamplesPerNode(1.0);
```

Confidence

Enabling tells the reconstructor to use the size of the normals as confidence information. When the flag is not enabled, all normals are normalized to have unit-length prior to reconstruction.

This value can be set using:

```
reconstructionFilter->SetConfidence(0);
```

Verbose

Enabling this flag provides a more verbose description of the running times and memory usages of individual components of the surface reconstructor.

This value can be set using:

```
reconstructionFilter->SetVerbose(0);
```

2.2 Demonstration

To demonstrate surface reconstruction, we use points sampled from a sphere. We computed the normals of these points using a previously submitted class, vtkPointSetNormalEstimation. We oriented these normals using another previously submitted class, vtkPointSetNormalOrientation. The resulting oriented point set is shown in Figure **??**. The signed distance volume, an intermediate step, is shown for clarity in **??**. In Figure **??**, we show the final reconstructed surface.

2.3 Code Snippet

3 Future Work

The algorithm implemented in this paper is very sensitive to noise error in the normal computation, the grid spacing, and the density of the points. There are several newer techniques, such as Poisson surface reconstruction, that we intend to implement for VTK in the future.

4 Paraview Plugin

For convenience, this code is shipped with a Paraview filter plugin. The plugin provides an easy way to set the parameters as well as integrate the new code into your workflow. A screenshot of the plugin interface is shown in Figure **??**.

References

[1] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. *Eurographics Symposium on Geometry Processing*, 2006. (document), 1