

# Streaming Surface Reconstruction Documentation

## Overview

---

This software implements the surface reconstruction algorithm presented in "*Multilevel Streaming for Out-of-Core Surface Reconstruction*", M. Bolitho, M. Kazhdan, R. Burns and H. Hoppe, Proceedings of the Symposium on Geometry Processing 2007.

## Release History

---

### Release 689 (21 December 2007)

Version 1.2 Update This release contains a number of bug fixes and new features.

- Compilation on 32bit platforms is now supported
- Compilation on Mac OSX is now supported. Please report any build problems.
- Multithreading support is included on Windows. The second pass (solver) is now multithreaded. The construction and surface extraction passes will be multithreaded in a future release.  
Multithreading support on Linux/Mac OSX is experimental and can be enabled by editing `Reconstructor/Src/Config.hpp` and recompiling from source.

### Release 615 (16 November 2007)

Version 1.1 Update. This release contains a number of bug fixes and improvements in reconstruction quality and performance, especially for larger models. This release fixes a number of program crashes that Linux users were experiencing.

### Release 521 (1 June 2007)

Initial Public Release. Known Issues:

- Multithreading support is disabled in this release. It will be enabled in a future release.
- The only output format supported is PLY. Output to our own streaming format, along with some tools will be made available in a future release.
- Output to the Isenburg streaming format is currently being implemented and will be available in a future release

## License Information

---

The software is provided to the community under the following license:

Copyright (c) 2005-2007, Matthew G Bolitho and Michael Kazhdan All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of the Johns Hopkins University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

## Support

---

The authors will provide limited support of this software. Updates will be provided as they become available. The latest version of the software can always be found at <http://www.cs.jhu.edu/~bolitho/Research/StreamingSurfaceReconstruction>.

Bug reports and feature requests should be directed to [bolitho@cs.jhu.edu](mailto:bolitho@cs.jhu.edu)

## Compilation

---

The source code provided is known to compile under Visual Studio 2008, and GNU G++ 4.x.

To compile on Windows with Visual Studio 2008 or Visual C++ Express Edition, open the solution file (Reconstructor.sln) and build the entire solution. Compilation targets for i386 and amd64 are provided. A Makefile compatible with Microsoft NMake is also available upon request.

To compile on Linux/Unix platforms with GNU G++ 4.x execute the following command:

```
make release
```

To compile on a 32-bit binary on a 64-bit Linux platform, execute the following command:

```
make release ARCH=i386
```

The code is known to compile cleanly on all platforms.

## Usage

---

Because the current implementation uses memory mapped files to access data within the out-of-core data files, a 64-bit operating system and executable is required to perform large reconstructions.

## Preprocessing

The first step in performing a reconstruction is to pre-process the point-set into a format compatible with the streaming algorithms used in this software. The input to this pre-processing is a binary file containing 3D points and normals. Each point/normal is stored in the following format (sometimes referred to as BNPTS format): (6 32-bit precision floating point numbers for a total of 24 bytes per sample) Position X-coordinate, Position Y-coordinate, Position Z-coordinate, Normal X-coordinate, Normal Y-coordinate, Normal Z-coordinate.

To pre-process a BNPTS file, the following command is used:

```
Bin\amd64\Reconstructor Samples\Bunny.Bnpts Samples\Bunny.PointSet  
/Depth:10
```

The extensions of the input and output files must be .Bnpts and .PointSet respectively. The Depth parameter defines the maximum resolution that the pre-processed pointset can be reconstructed at. The following flags may be provided, but are optional:

- **/NoRotate**: Prevents the pointset from being rotated with according to the co-variance matrix.
- **/NoTransform**: Prevents the pointset from being rescaled and translated into a bounding box that fits inside the cube ([0,1], [0,1], [0,1]).

The pre-processing step produces three files. OutputName.PointSet contains all the samples from the input data file, transformed/rotated (if applicable) into a unit cube coordinate system, and completely sorted along the x-axis. OutputName.Transform describes the transformation that was applied to the input samples to place them in the new coordinate system. Output.PointIndex contains a complete binary tree which indexes the samples according to their depth and block.

## Reconstruction

Once a pointset has been pre-processed, it may be used to reconstruct a triangle mesh as follows:

```
Bin\amd64\Reconstructor Samples\Bunny.PointSet Bunny.Ply /Depth:10
```

The extensions of the input and output files must be .PointSet and .Ply respectively. (A version of the reconstructor outputting to our streaming mesh format will be made available in a future release) The depth parameter is required, and defines the maximum depth of the octree (the variable  $h$  in the paper). The following flags and values may be provided, but are optional:

- **/SamplesPerNode:{value}**: Defines the target number of samples that should fall into a tree node (the variable  $k$  from the paper). The value is a floating point number. The minimum value is 1.0 -- higher values can be used when significant noise is present in the input data.
  - **/NoTransform**: Prevents the output mesh being transformed back into the original coordinate system of the .Bnpts file.
  - **/Log:{value}**: Defines a file to write logging information to.
  - **/StreamingFilename:{value}**: Uses a user-defined value to form the prefix for the filenames of all streams created by the reconstructor rather than a temporary filename. Useful when used in conjunctions with **/Step** (see below)
  - **/Step:{value}**: Defines which passes of the reconstructor to execute. 1 = Construction, 2 = Solving, 4 = Extraction, 7 = All
-