Surface Reconstruction

Proposal for a CGAL package

Pierre Alliez Laurent Saboret

September 7, 2007

Keywords: surface reconstruction, point set, slices, points with normals, points with oriented normals, implicit function, surface meshes.

1 Introduction

The purpose of this document is to define the specification of a CGAL Surface Reconstruction component. This specification will have a 2 sections:

- the specification of the package *concepts*,
- the specification of the package *implementation* (in the first version).

The idea behind the *concepts* specification is to find the common parts among various reconstruction algorithms and to formalize them as concepts. This is a way to define the algorithms that the package will *potentialy* support. This will also ease the implementation of new methods in the package.
As many reconstruction methods are available in the litterature and at GEOMETRICA, we will use several steps of the UML method [BRJ99] to define the concepts:

- list all algorithms that the package will *potentialy* support as "use cases" (as free text),
- group related "use cases" as "scenarios" (still free text), and
- analyse the scenarios to change them to C++ concepts.

In the other hand, the *implementation* specification will simply describe the method(s) that we plan to implement by the end of 2007.
This document is a work in progress. It may be modified by any member of the GEOMETRICA project-team interested by surface reconstruction problems.

2 Proposal

We propose to elaborate upon a CGAL Package which implements some of the state-of-the-art surface reconstruction methods. Priority will be given to methods which take as input unorganized point sets and which compute an implicit function, although we leave the room to Delaunay-based techniques as all building blocks are available in CGAL. A common implicit function is an approximate distance function to the input points or to an estimate tangent plane, or an indicator function.

The package will propose an interface to the CGAL Surface Mesh Generator [RY06, BO05], and we leave the room to other surface contouring techniques such as the Marching Cubes algorithm [LC87] and its variants (a user may prefer using his favorite contouring algorithm).

The input can be an unorganized point set, possibly with attributes such as unoriented normals, oriented normals, confidence values. For piecewise smooth reconstruction, it is also desirable to take not only one attribute per point, but possibly several ones (for example, two normals for a point on a sharp edge, or several normals for a corner, or even a cone of normals). The input can also be a set of arbitrary cross sections, be they defined by point sets localized onto well-defined planes, or as polylines.

Since reconstruction methods often require to estimate and/or to orient the normals of a point set, we plan to implement two components devoted to this task. These components will make use of existing components such as PCA or Jet fitting.

The output can be either an implicit function (ready for evaluation by any contouring algorithm), or a surface mesh [format to be defined].

3 Intended Audience

The intended audience of this package is researchers, developers or students developing algorithms around surface reconstruction. We also target end users and industrial applications.

4 Specification of the Package Concepts

4.1 Use Cases

4.1.1 Use Case U1: Smooth Surface Reconstruction via Natural Neighbor Interpolation of Distance Functions

Boissonnat and Cazals implemented for Dassault a very efficient reconstruction algorithm which is a simplified version of [BC02]. The outline is:

1. Input is a 3D point set.
2. A normal estimator module evaluates the direction of the normal of each input point [Frederic: Principal Components Analysis over k nearest neighbors or pole-based?].
3. A normal orientation module orients the normals [Frederic: please insert details here].
4. Build Delaunay triangulation.
5. For each cell of the dual Voronoi diagram:
   - Evaluate if each pole is inside or outside the surface [Frederic: please insert details here].
6. For each edge of the dual Voronoi diagram:
   - If the edge has 1 pole in and 1 pole out, add the dual triangle to the reconstructed surface.

4.1.2 Use Case U2: Delaunay-based Poisson Reconstruction

The outline of this algorithm is:

1. Input is a 3D point set.
2. A normal estimator module computes the normal direction of each input point using PCA over the k nearest neighbours [not yet implemented].
3. Orient normals with MST algorithm [HDD+92] [not yet implemented].
4. Build Delaunay triangulation.
5. Delaunay refinement (break bad tetrahedra, where "bad" means bdly shaped or too big). The normal of Steiner points is set to zero.
6. Optionally, extrapolate normals.
7. Compute an indicator function f() piecewise-linear over the tetrahedra. We solve the Poisson equation Laplacian(f) = divergent(normals field) at each vertex of the triangulation via the TAUCS sparse linear solver. One vertex must be constrained.
8. CGAL Surface Mesh Generator extracts the isosurface "f() = median of f() over the input points".

4.1.3 Use Case U2b: Same as U2 with average value of the implicit function

Same as U2 with step 7 replaced by:

- CGAL Surface Mesh Generator extracts the isosurface "f() = average of f() over the input points".

4.1.4 Use Case U2c: Same as U2 with several surface extractions

Same as U2 with several steps 7 for different values of f().

4.1.5 Use Case U2d: Same as U2 from 3D points with normals

Same as U2 without steps 2 and 3.

4.1.6 Use Case U3: Kazhdan's Poisson Surface Reconstruction

The Poisson Surface Reconstruction algorithm[KBH05] is similar to U2 except that:

- Input points are stored in an octree.
- The sparse linear system is solved by a solver dedicated to the octree.
- Contouring is computed by a Marching Cubes [LC87] algorithm.

4.1.7 Use Case U4: Voronoi-based Variational Reconstruction of Unoriented Point Sets

The outline of the Voronoi-based Variational Reconstruction of Unoriented Point Sets algorithm[PAD07] is:

1. Input is a 3D point set.
2. Build Voronoi diagram.

3. A normal estimator module evaluates each normal direction as a tensor aligned along the Voronoi cell(s). In a way, the tensor contains the normal direction + a confidence value in the orientation.
4. Compute the implicit function [Pierre: please insert details here].
5. CGAL Surface Mesh Generator extracts an iso-surface [Pierre: please insert details here].

4.1.8 Use Case U5: Reconstruction with Voronoi Centered Radial Basis Functions (1)

Given a set of input points measured on a surface S, the algorithm[MMPM06] constructs a surface S' that approximates S. S' is the zero-level of f() where f(x) is expressed as a weighted sum of basis functions centered at a set of center points.

The outline of the algorithm is:

1. Compute the Delaunay triangulation of the input points (plus bounding points to avoid infinite Voronoi cells).
2. Extract the set of poles from the Voronoi vertices :
   - 2 poles are extracted for each bounded cell Vp. The first pole v1 is the Voronoi vertex in Vp with the largest distance to the sample point p. The second pole is the Voronoi vertex v2 in Vp the further away from p in the opposite half space of v1. If the sampling is dense enough, the vectors p->v1 and p->v2 are a good approximation of the normal to the surface at p.
3. Classify the poles as inside or outside:
   (a) Construct a poles graph.
   (b) The main idea of the labeling algorithm is to associate to each pole temporary labels, in and out, and a probability measure on the temporary labels. Specically, the labels and their probability are propagated through the pole graph.
   (c) In order to initialize the algorithm, the poles of the bounding points are labeled as outer with a probability one.
   (d) Then the pairs (label, probability) are propagated until all poles are labeled. The pole v with the highest probability is popped out of the queue, and gets a final label. Then, the labels of the neighbors of v in the poles graph are updated and the priority queue is updated.
4. Select the center points from the set of poles;
5. Assemble a matrix to solve the problem by minimizing a least squares error.
6. Solve the linear system with TAUCS solver.
7. Mesh the surface with a Boomenthal algorithm.

4.1.9 Use Case U5b: Reconstruction with Voronoi Centered Radial Basis Functions (2)

Same as U5 when replacing the Boomenthal algorithm by CGAL Surface Mesh Generator.

4.1.10 Use Case U6: Shape Reconstruction from Unorganized Cross-Sections

The outline of the Shape Reconstruction from Unorganized Cross-Sections algorithm[BM07] is:

[TO EDIT]

4.1.11 Use Case U7: Robust Cocone

The outline of the Robust Cocone algorithm[NSTKN00, DG04] is:

[TO EDIT]

4.1.12 Use Case U8: Power Crust

The outline of the Power Crust algorithm[ACK01] is:

[TO EDIT]

4.1.13 Use Case U9: Smooth Surface Reconstruction with the improved Surface Mesher

[TO EDIT]

4.2 Scenarios

We can group together the use cases above based on implicit functions. In the other hand, we seem to need one scenario per Delaunay-based reconstruction algorithm. [U7, U8 and U9 are not yet evaluated].

4.2.1 Scenario S1: CGAL Surface Mesh Generator from an implicit function

This scenario groups together the use cases U2, U2b, U2c, U2d, U4, U5b.

1. The input is a point cloud.
2. If points have no normals, an estimation module is called.

3. If normals are not oriented, an orientation module is called.
4. We define an implicit function f().
5. CGAL Surface Mesh Generator meshes a surface as an iso-level of f().
6. Optionaly, CGAL Surface Mesh Generator meshes an another iso-level of f().
7. The ouput format can be:
   (a) a polygon soup, or
   (b) an organized surface like a BGL Graph or CGAL Polyhedron [not implemented], or
   (c) SurfaceMeshComplex_2InTriangulation_3 (Surface Mesh Generator's native output format).
8. Optionaly, the output format may contain the normals computed by steps 2-3.

4.2.2 Scenario S2: Marching Cubes (or a variant) from an implicit function

Same as S1 when replacing CGAL Surface Mesh Generator by the Marching Cubes. The output cannot be a model of SurfaceMeshComplex_2InTriangulation_3.
This scenario groups together the use cases U3, U5.

4.2.3 Scenario S3: Implicit function only

We keep only the steps 1-4 of the scenario S1 to define an implicit function that the user can interrogate.

4.2.4 Scenario S4 = Use Case U1: Smooth Surface Reconstruction via Natural Neighbor Interpolation of Distance Functions

4.2.5 Scenario S5 = Case U6: Shape Reconstruction from Unorganized Cross-Sections

4.3 Overall Specification of the Concepts

4.3.1 Input

- 3D points
- 3D points with unoriented normals
- 3D points with oriented normals
- 3D points with confidence value of the position or of the normal
- Cross-sections (parallel of not) with each cross-section defined as a contour = set of 2D points
- Cross-sections (parallel of not) with each cross-section defined as a contour = polyline

4.3.2 Output

- 2-manifold surfaces vs non-manifold surfaces
- Watertight models vs models with boundaries
- The general case is just: a polygon soup

4.3.3 Features

- Taxonomy of reconstruction methods:
  - Based on an implicit function:
    - Signed approximate distance to points [CL96]
    - Signed approximate distance to tangent plane estimates [HDD+92, BC02]
    - RBF [CBC+01]
    - Poisson [KBH05]
    - ...
  - Delaunay-based:
    - ...
  - Deformable models:
  - Graph cuts
- Properties of reconstruction methods:
  - Interpolation vs approximation
  - Resilient to noise?
  - Resilient to outliers?
  - Resilient to sparse sampling?
  - Resilient to over-sampling?
  - Resilient to anisotropic sampling?
  - Watertight output?
  - 2-manifold output?
  - Smooth vs piecewise smooth reconstruction?
  - Scalability?
  - Out-of-core/streaming/on-line?
- Taxonomy of normal estimation methods:
  - Point-based Principal Components Analysis over the k nearest neighbors
  - Weighted point-based Principal Components Analysis over the k nearest neighbors

  - Jet fitting
  - Pole-based
  - Voronoi-based Principal Components Analysis
- Taxonomy of normal orientation methods:
  - Minimal Spanning Tree [HDD+92]
  - Pole-based [ABK98, BC02]

4.4 Analysis

The objects/classes/concepts that seem to appear in the above scenarios are:

- A Point_3 concept to represent the 3D points which are the input of all use cases.
  Models:
  - Kernel::Point_3
- A PointWithNormal_3 concept representing a 3D point + its normal (oriented or not).
  Models:
  - a Point_with_normal_3 class containing a Point_3<Kernel> + a Vector_3<Kernel> + a boolean coding the normal orientation
  - a tensor
  - a 3D point with an "enriched" normal like two normals for a point on a sharp edge, or several normals for a corner, or even a cone of normals
- A NormalEstimation_3 concept representing a module to estimate the direction of the normals of a point set.
  Input: container of a model of Point_3.
  Output: container of a model of PointWithNormal_3.
  Models: see taxonomy above.
- A NormalOrientation_3 concept representing a module to orient the normals of a point set.
  Input/output: container of a model of PointWithNormal_3.
  Models: see taxonomy above.
- The package entry points will be the set of contouring algorithms: Marching Cubes, Surface Mesh Generator, and each Delaunay-based reconstruction algorithm.
  Marching Cubes and Surface Mesh Generator will be templated by an implicit function.
  Input: container of a model of PointWithNormal_3.
  Output: see below.
  As these algorithms are the entry points of the package and are quite different, we plan to make them a set of independent classes or functions with no common concept:
  - a wrapper around Surface_mesher CGAL package,
  - a Marching Cubes implementation,
  - and a class or function for each Delaunay-based reconstruction algorithm.
- We will define a ReconstructionImplicitFunction concept that provides a common interface to the Marching Cubes and the Surface Mesh Generator (see Surface Mesh Generator's ImplicitFunction concept).
  Note that some of these implicit functions are linked to a specific normals representation (e.g. a tensor for Voronoi-based Variational Reconstruction of Unoriented Point Sets).
- Ideally, we should define a concept for the output of the whole algorithms. This is not obvious as the current implementations have incompatible outputs. In the worst case, we will end up with outputs specific to each contouring algorithm.
  Candidates are:
  - a polygon soup,
  - an organized surface like a BGL Graph or CGAL Polyhedron,
  - SurfaceMeshComplex_2InTriangulation_3 (Surface Mesh Generator's native output format).

5 Specification of the Implementation

We plan to *implement* in the first version of the package:

- the Delaunay-based Poisson reconstruction algorithm, with
- an normal estimation module based on PCA over the k nearest neighbors (available in CGAL),
- a normal orientation module implementing [HDD+92],
- and a contouring using CGAL Surface Mesh Generator.

6 Algorithms References

This section describes algorithms which are key parts of the above use cases.

6.1 Algorithm A1: CGAL 3.3's Surface Mesh Generator

[TO EDIT]

6.2 Algorithm A2: Piecewise Smooth Surface Mesh Generator

Mariette and Laurent R. are working on an improved Surface Mesh Generator dealing with piecewise-smooth surfaces. The Surface Mesh Generator reconstructs a surface by calling 2 "oracles":

- intersection of the surface with a line segment,
- intersection of a surface's sharp edge with a plane.

Note that the implicit function is computed only when requested by the Surface Mesh Generator. There is no need to represent it explicitely.

6.3 Algorithm A3: Marching Cubes

The outline of the Marching Cubes algorithm[LC87] is:

1. The Marching Cubes use 3D medical images as input, thus an implicit function whose value is known on a grid.
2. Read four slices into memory.
3. Scan two slices and create a cube from four neighbors on one slice and four neighbors on the next slice.
4. Calculate an index for the cube by comparing the eight density values at the cube vertices with the surface constant.
5. Using the index, look up the list of edges from a precalculated table.
6. Using the function value at each edge vertex, find the surface edge intersection via linear interpolation.
7. Calculate a unit normal at each cube vertex using central differences. Interpolate the normal to each triangle vertex.
8. Output the triangle vertices and vertex normals.

References

[ABK98] Nina Amenta, Marshall Bern, and Manolis Kamvysselis. A new voronoi-based surface reconstruction algorithm. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 415–421, New York, NY, USA, 1998. ACM Press.

[ACK01] Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. The power crust. In *SMA '01: Proceedings of the sixth ACM symposium on Solid modeling and applications*, pages 249–266, New York, NY, USA, 2001. ACM Press.

[Ale04] M. Alexa. State-of-the-Art-Report Survey Acquisition and Reconstruction. Technical report, AIM@SHAPE, 2004.

[BC02] Boissonnat and Cazals. Smooth surface reconstruction via natural neighbour interpolation of distance functions. *CGTA: Computational Geometry: Theory and Applications*, 22, 2002.

[BM07] Jean-Daniel Boissonnat and Pooran Memari. Shape reconstruction from unorganized cross-sections. In *Symposium on Geometry Processing*, pages 89–98, 2007.

[BO05] Jean-Daniel Boissonnat and Steve Oudot. Provably good sampling and meshing of surfaces. *Graph. Models*, 67(5):405–451, 2005.

[BRJ99] G. Booch, J. Rumbaugh, and I. Jacobson. *The unified modeling language user guide*. Addison-Wesley, 1999.

[CBC+01] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 67–76, New York, NY, USA, 2001. ACM Press.

[CL96] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312, New York, NY, USA, 1996. ACM Press.

[CP03] Frederic Cazals and Marc Pouget. Estimating differential quantities using polynomial fitting of osculating jets. In *Symposium on Geometry Processing*, pages 177–187, 2003.

[DG04] Tamal K. Dey and Samrat Goswami. Provable surface reconstruction from noisy samples. In *SCG '04: Proceedings of the twentieth annual symposium on Computational geometry*, pages 330–339, New York, NY, USA, 2004. ACM Press.

[GHJV95] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns – Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[HDD+92] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 71–78, New York, NY, USA, 1992. ACM Press.

[KBH05] Michael Kazhdan, M. Bolitho, and Hugues Hoppe. Poisson Surface Reconstruction. In *Symp. on Geometry Processing*, pages 61–70, 2005.

[LC87] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 163–169, New York, NY, USA, 1987. ACM Press.

[MMPM06] Samozino Marie, Alexa Marc, Alliez Pierre, and Yvinec Mariette. Reconstruction with Voronoi Centered Radial Basis Functions. In Polthier Konrad and Sheffer Alla, editors, *Eurographics/ACM Siggraph Symposium on Geometry Processing*, pages 51–60, 2006.

[NSTKN00] Amenta N., Choi S., Dey T. K., and Leekha N. A Simple Algorithm for Homeomorphic Surface Reconstruction. In *Proceedings 16th ACM Symposium on Computational Geometry*, pages 213–222. ACM Press, 2000.

[PAD07] Yiying Tong Pierre Alliez, David Cohen-Steiner and Mathieu Desbrun. Voronoi-based variational reconstruction of unoriented point sets. In *Symposium on Geometry Processing*, pages 39–48, 2007.

[RCG+01] C. Rocchini, P. Cignoni, F. Ganovelli, C. Montani, P. Pingi, and R. Scopigno. Marching intersections: An efficient resampling algorithm for surface management. In *SMI '01: Proceedings of the International Conference on Shape Modeling & Applications*, page 296, Washington, DC, USA, 2001. IEEE Computer Society.

[RY06] Laurent Rineau and Mariette Yvinec. A generic software design for Delaunay refinement meshing. Technical Report 5983, INRIA, 2006.