Automatic Construction of Surface Model

Miroslav Marić, Filip Marić, Žarko Mijajlović, Boško Jovanović

School of Mathematics, University of Belgrade, Belgrade, Serbia and Montenegro
maricm@matf.bg.ac.yu

Abstract: Our primary goal is surface reconstruction by use of data obtained by 3D scanning. We describe method and its implementation for automatic construction of surface model (triangular mesh) from unorganized set of points. The method doesn't exploit any additional knowledge in specific problem instances like topological type of the surface, structure of the data, orientation information, etc. In our case, set of points can be an unorganized, noisy sample of an unknown surface that can have almost arbitrary geometrical type, and may contain tangent plane discontinuities such as creases and corners.

1 Introduction

Motivation for this work comes from the need to build software for analysis of 3D images obtained from the 3D scanner that we have on our disposal at Faculty of Mathematics.

2 Problem Formulation

In this paper we are solving the following problem: Given a set of points $X = \{x_1, x_2, ..., x_n\} \in R^3$ that lies near a surface U, construct a triangular mesh that approximates surface U.

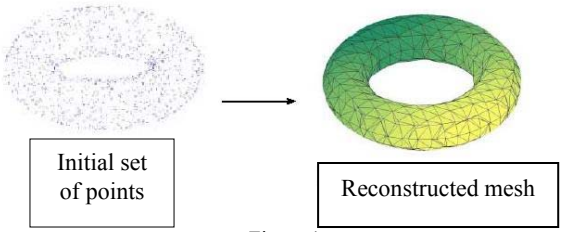

Figure 1
Problem formulation

Most algorithms that are used for solving this problem require additional knowledge such as structure in the data, known surface genus or orientation information. The algorithm we have implemented is more general:

- X can be arbitrary set of points
- X can be noisy sample
- Surface U can be of arbitrary topological type and can contain creases and corners

Still, the surface has to satisfy some constraints:

- It must not be self-intersecting
- It's sheets must not be to close

3 Numeric Parameters of the Problem

Initial set of points is said to be *δ-noisy* i.e. each point $x_i \in X$ could be written as $x_i = y_i + e_i$ where $y_i \in U$ and $\| e_i \| < \delta$

We assume that the sample is *ρ-dense* i.e each ball with center in U contains at least one point of noise-free sample.

Having the parameters δ and ρ we can formulate mentioned constraints on the surface U.

- Details of U which are smaller than ρ i δ can not be successfully reconstructed
- Distance of points from different sheets of surface has to be at least $\rho + \delta$. Distance of sheets has to be at least $\rho + 3\delta$
- p can not lie on surface U if
  $d(p, X) > \rho + \delta$

4 Algorithm

Algorithm has 3 major phases:

1. Initial mesh construction
2. Mesh optimization

3. piecewise smooth surface optimization

In this paper we are describing the first phase – initial mesh construction. Main step in during that phase is approximation of *signed distance function*. Signed distance function represents distance $d_U(p) = \pm d(p,U)$ from each point $p \in R^3$ to the surface *U*, where the sign depends on which side of the surface U the point p lies. Knowing this, the surface *U* can be represented as

$$\{p \in R^3 : d_U(p) = 0\}$$

Since d_U is unknown, we use approximation $\tilde{d}_U$ which is defined only near the set X and represents distance from a given point to the tangent plane estimation of *U*.

Initial surface construction goes in 4 steps:

- Tangent plane construction
- Tangent plane orientation
- Signed distance function approximation
- Mesh construction

As the first step we assign a tangent plane to each point of X. Each plane is represented with its center and normal vector. To each point of X we assign a set of its neigborhood points $K(x_i)$. $K(x_i) = \{x \in X : d(x,x_i) < \rho + \delta\}$ Tangent plane is constructed as the plane that is the best least-squares approximation of the set $K(x_i)$.

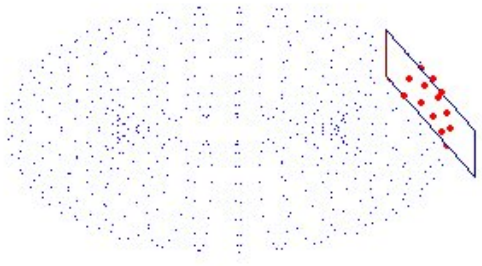

Figure 2
Neighborhood points

To optimize quadratic performance of brute-force neighborhood finding algorithm we use spatial partitioning techniques. Center of the tangent plane is taken to be the centroid of neighborhood $K(x_i)$. The normal is determined as the eigenvector corresponding to the least eigenvector of the matrix

$$\sum_{y \in K(x_i)} (y - c_i) \otimes (y - c_i)$$

After tangent plane estimations, the constructed normals are not consistently oriented (Fig. 3).

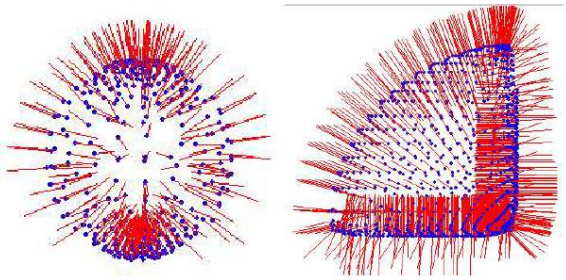

Figure 3
Incosistently oriented normals

If the surface is smooth and the centers cii and cjj of two tangent planes are close normals are almost parallel. i.e. $n_i \cdot n_j \approx \pm 1$

If $n_i \cdot n_j \approx -1$ the orientation of one of the normals has to be changed.

The problem arises near the sharp corners and edges i.e. when

$$| n_i \cdot n_j | \neq 1$$

It is clear that these have to be avoided during orientation of normals.

We use a graph formulation of the orientation problem. Vertices of the constructed graph are tangent plane centers. The edges join the vertices that are geometrically close i.e ones for which: $\| c_i - c_j \| < \rho + \delta$

Normal orientation problem reduces to fining an assignment of a number $b_i = \pm 1$ to each vertex which maximizes the sum $\sum_{i,j \in E} b_i n_i \cdot b_j n_j$

It can be shown that this problem is NP hard. This is the main reason why we use a heuristic “greedy” approach. The orientation of normals is propagated starting from one point (usually the highest point) and orienting each normal in a way that its orientation becomes consistent with the orientation of the previous one. Traversal should avoid sharp edges and corners.and should favorize smooth parts

of the surface. To achieve this, we assign a weight $1-|n_i \cdot n_j|$ to each edge (c_i, c_j). The traversal is determined by a minimal cost spanning tree in this graph.

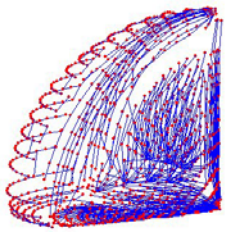

Figure 4
Minimal cost spanning tree

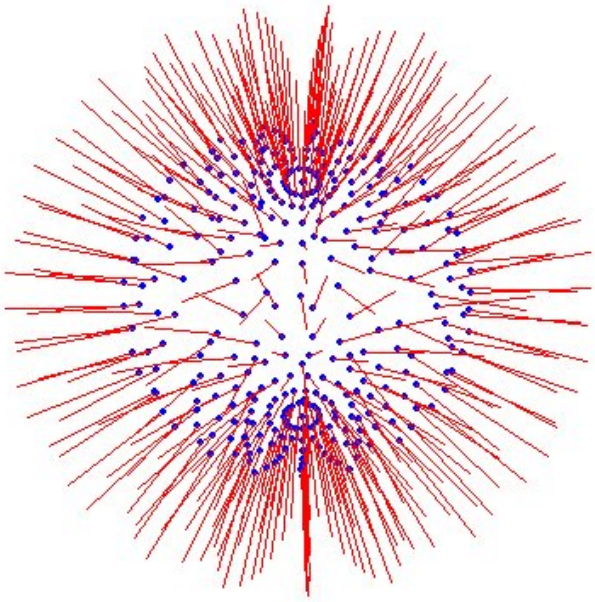

Figure 5
Consistently oriented normals

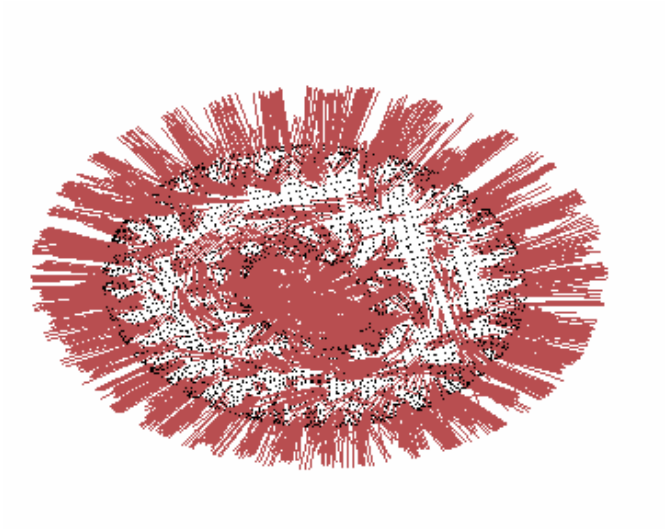

Figure 6
Consistently oriented normals

When the normals are consistently oriented we approximate the signed distance function. For each point p we find a tangent plane whose center is the nearest to p. Signed distance is approximated using the formula:

$$\tilde{d}_U(p) = (p - c_i) \cdot n_i$$

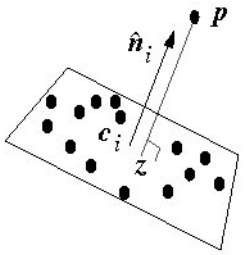

Figure 7
Signed distance function approximation

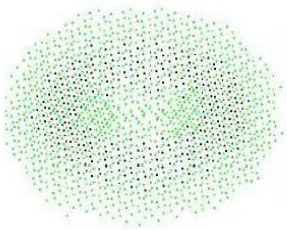

Figure 8
Signed distance function calculated in all points of a cube spatial partitioning

We construct the initial triangular mesh using the famous "marching cubes" algorithm. The space near the set X is partitioned to $(\rho+\delta)$ sized cubes. U each cube vertex p we approximate the signed distance function and the side of this function determines the side of the surface U on which the point p lies. If two cube vertices lie on the opposite sides of the surface there is a point of U on the edge that connects these two points. The position of that point is determined using a simple linear interpolation along the cube edge. The points constructed in this manner become the vertices of a triangular mesh.

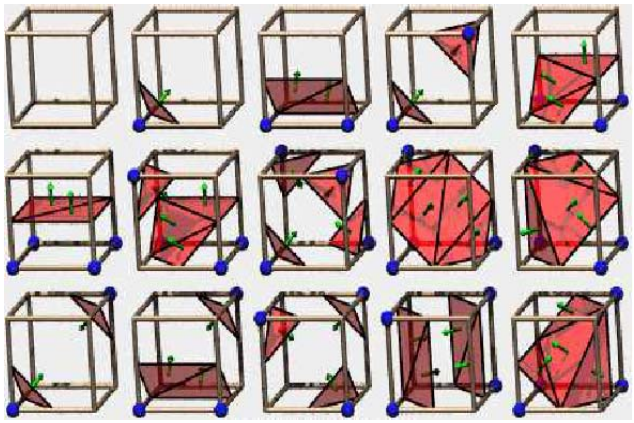

Figure 9
Different cases during"marching cubes" algorithm

5 Implementation

We have implemented described algorithm twice. The first implementation was experimental and it was performed using MATLAB. Since the obtained results were encouraging, we decided to reimplement the algorithm using C++ and

OpenGL library. This has substantially increased the performance and allowed us to increase the number of points we are analyzing (from initial 500 point to more than 100.000).

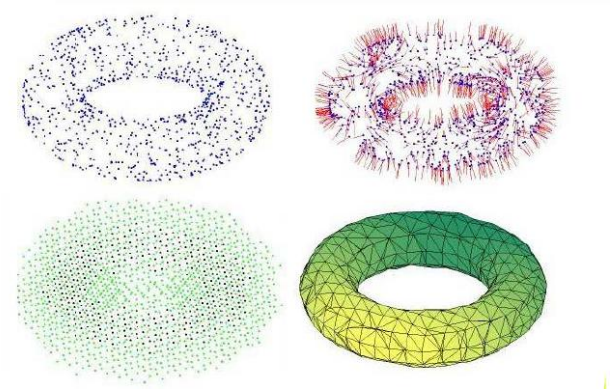

Figure 10
Results

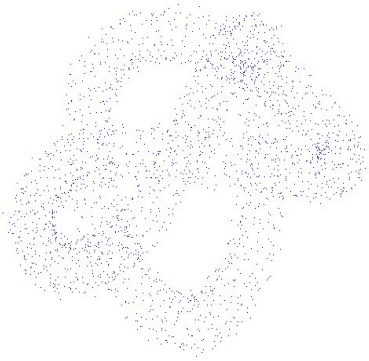

Figure 11

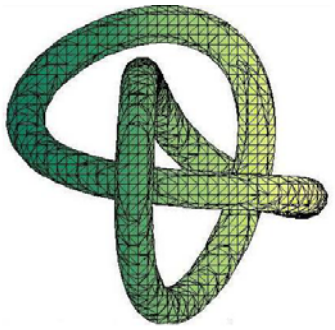

Figure 12

Bibliography

[1] W. E. Lorensen and H. E. Cline. A High Resolution 3D Surface Construction Algorithm, Computer Graphics (Proceedings of SIGGRAPH ’87), Vol. 21, No. 4, pp. 163-169

[2] H. Hoppe. Surface reconstruction from unorganized points, FENS University of Washington, Dissertation 1994

[3] M. J. Turner, J. M. Blackledge, P. R. Andrews, Fractal Geometry in Digital Imaging, Academic Press, 1998

[4] Ž. Mijajlović, D. Urošević. An algorithm for recognition of linear surfaces, Proceeding ”CFT Varna 2002”, workshop ”Multiscale approximations (wavelets, splines, RBF) and applications to Image and Signal Processing”