Manifold Extraction in Surface Reconstruction

Michal Varnuška[1] and Ivana Kolingerová[2]

Centre of Computer Graphics and Data Visualization
Department of Computer Science and Engineering
University of West Bohemia, Pilsen, Czech Republic
miva@kiv.zcu.cz, kolinger@kiv.zcu.cz

Abstract. Given a finite point set in R^3 scanned by special devices from the object surface, a surface model interpolating or approximating the points set has to be obtained. We use for the reconstruction a CRUST algorithm by Nina Amenta, which selects surface triangles from the Delaunay tetrahedronization using information from the dual Voronoi diagram. This set of candidate surface triangles does not form a manifold, so the manifold extraction step is necessary. We present two improvements for this step, the former is limited to the used algorithm and the latter can be used with any other reconstruction algorithm.

1 Introduction

Many applications from various areas of science or industry need to work with the piecewise interpolation of the real objects. One of often-used ways to obtain the model is the points cloud reconstruction. The task of the reconstruction is not simple, we have only points in 3D space without any additional data (such as normal vectors).

Four kinds of approaches exist based on warping, distance function, incremental surface reconstruction and spatial subdivision. Warping works on the basic idea that we deform some starting surface to the surface that forms the object. The idea of warping is relatively old and is used in Müller's approach [17] or by Muraki [18].

The incremental surface reconstruction is the second huge group of algorithms. Boissonat's approach [8] begins on the shortest edge from all possible edges between points and incrementally appends the edges to create a triangle mesh. Mencl and Müller [19] developed a similar algorithm. It creates an extended minimum spanning tree, extends it to the surface description graph and extracts typical features.

Hoppe [16] presented an algorithm, where the surface is represented by the zero set of a signed distance function. The function sign is plus if the point lies inside the closed surface and minus otherwise, the value is the distance to the surface. Curless and Levoy [9] gave an effective algorithm using the signed distance function on a voxel grid, it is able to reconstruct eventual holes in a post-processing.

The fundamental property of the methods based on spatial subdivision is the space division into independent areas. The simplest division is presented by the voxel grid,

[1] The author was supported by the project FRVŠ G1/1349
[2] The author was supported by the project MSM 235200005

which Algorri and Schmitt [1] use in their effective algorithm. The voxels containing points from the input set are chosen and the surface is extracted. The most often used division is the Delaunay tetrahedronization (DT) because the surface forms a subgraph of the tetrahedronization. Edelsbrunner and Mücke [14, 15] developed an α-shape algorithm for uniform sample sets, Bernardini and Bajaj [6] extended it. They use the binary search on the parameter alpha to find the surface subcomplex. Bernardini [7] presented a very fast and efficient ball pivoting algorithm.

Amenta introduced the concept of CRUST in [2, 3, 4]. Dey extended the ideas of Amenta, giving an effective COCONE algorithm. The extension of COCONE algorithm can handle large data [10], detect boundaries [11], undersampling and oversampling [12]. These methods are based on the observation that the places with a changing of point density can be detected using shape of Voronoi cells in these places. Both authors gave then algorithms for watertight surface creation, Amenta's POWERCRUST [5] and Dey's TightCOCONE [13].

As mentioned in abstract, we use the CRUST algorithm. It works on spatial subdivision achieved by DT. Auxiliary subdivision, Voronoi diagram (VD), is obtained by dualization from DT. There exist two versions of the algorithm based on onepass or twopass tetrahedronization. We have chosen these methods because they have strong theoretical background, are not so much sensitive to the sampling errors (the sampling criterion is based on *local feature size (LFS)*, closer in [3]) and we have a fast and robust implementation of DT. Due the sampling LFS criterion the CRUST algorithm is not sensitive to big changes in sampling density, the data need not to be uniformly sampled, but it has problems with outliers and sharp edges. Then the surface normals estimated using poles (explained bellow) point to bad directions, the reconstruction fails and a lot of bad triangles appear.

The details of these methods can be found in [2, 3, 4], we concentrate only to the information necessary for later understanding. The positive pole $p+$ is the furthest Voronoi vertex (VV) of the Voronoi cell around some point p, the negative pole $p-$ is the farthest VV on the "second side" (the dot product of the vectors $(p-, p)$ and $(p+, p)$ is negative). For successfully sampled surface all Voronoi cells are thin, long, the poles lay on the medial axis and vectors to the poles approximate the normal vectors.

The first shared step of both versions is the DT creation followed by its dualization to VD and poles computation. Then the versions differ, the twopass algorithm takes the poles as an approximation of the medial axis while the onepass takes the poles as the approximation of the normal vectors. We use the onepass version because it is more than three times faster and less memory consuming. Three conditions must hold for the surface triangles: their dual Voronoi edges must intersect the surface, the radius of the circumcircle around the surface triangle is much smaller than the distance to the medial axis at its vertices and the normals of surface triangles make small angles with the estimated surface normals at the vertices.

We can compute the set of surface triangles as follows. For each point p we have an approximation of its normal vector $n = p+ - p$. Each triangle in DT has an edge e dual in VD. For the triangles on the surface, this edge has to pass through the surface P. Let us denote the vertices of the edge e as w_1, w_2, the angles $\alpha = \angle(w_1 p, n)$ and $\beta = \angle(w_2 p, n)$. When the interval $<\alpha, \beta>$ intersects the interval $<\pi/2 - \theta, \pi/2 + \theta>$ and this condition holds for each vertex p of the triangle, then the triangle is on the surface. The parameter θ is the input parameter of the method.

2 Manifold Extraction

The result of the CRUST algorithm is the set of surface triangles T (we call it the primary surface). These triangles passed conditions of the CRUST algorithm but they do not form the manifold yet. There can be more than two triangles incident on some edges or some triangles may miss on the places of local discontinuity. For example, very flat tetrahedra in the smooth part of the surface (Fig. 1a) or the tetrahedra on the surface edge (Fig. 1b) may have all faces marked as surface triangles. The number of overlapped triangles differs from model to model and depends on the surface smoothness. For smooth surface it is in tens percent and when the surface is rough, the rate decreases.

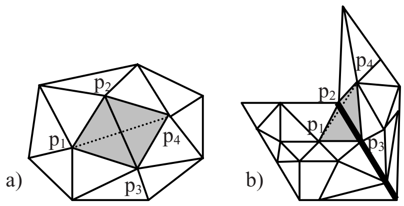

Fig. 1. a) Flat part of the surface, b) the part with the sharp edge (bold line). One pair of triangles is ($p_1 p_2 p_3$) and ($p_2 p_3 p_4$), the second pair is ($p_1 p_2 p_4$) and ($p_2 p_3 p_4$)

That is why the surface extraction step must be followed by a manifold extraction step. The input to the manifold extraction step is just the set of triangles. Manifold extraction step is independent of the reconstruction method, therefore it could be combined with other algorithms than CRUST. We have developed our own algorithm. The reason was that the manifold extraction methods were explained very briefly in the papers, however, this step is important. Our approach uses breadth-first search for appending triangles on free edges and has a linear time complexity. The algorithm is presented in [20], for clarity of the text we will briefly explain it.

The preprocessing step of the extraction is creation of two structures, a list of incident triangles for each point and a multiple neighbors mesh containing for each triangle the list of incident triangles on the edges. More then two triangles sharing one edge can exist as the manifold is not ensured yet (e.g. Fig. 2). First, we have to find the starting triangles using these structures, they will form the root of the searching tree. These triangles form a triangle fan; no triangles from the fan overlap when we project them to the plane defined by the point and the normal at this point.

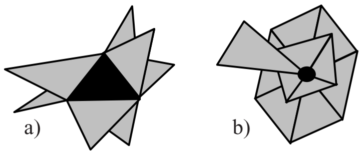

Fig. 2. a) An example of multiple neighbors to some triangle, b) an example of incident triangles to one point

Next, we add to the already extracted triangles their neighbors on the non-connected edges. These neighbors form next level of the tree. Because we can have multiple neighbors, we have to find just one triangle of them. We assume that the triangles must be small to form a correct surface, so we take the one, which has the shortest sum of edge length. We need only 2 levels of the tree at one moment, older levels can be safely deleted. We continue recursively until all edges are processed.

3 Prefiltering

We already mentioned that the CRUST algorithm has very good results for smooth surfaces. However, even with datasets of smooth objects, sometimes small triangle holes appear in the reconstructed surface. It is not a problem to find and fill them in the postprocessing step, but the question is why they appear. Each tetrahedron has four faces – triangles. The CRUST marks them whether they belong to the set of the primary surface *T*. We have found that the triangle holes appear in the smooth places where very flat tetrahedron lies whose three faces are marked as surface triangles. See Fig. 3a) for an example: the dark gray triangles are already extracted and we are looking for the triangle neighbor on the bold edge of the triangle 1. The light gray triangles are marked triangles from one tetrahedron (there are three overlapping triangles), two of them are incident with the bold edge of triangle 1 and we have to choose only one of them. When we select bad triangle then in the next step of extraction the triangle hole occurs (Fig. 3b). Fig. 3c) shows a correct configuration.

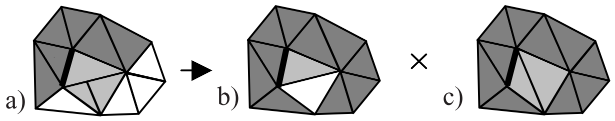

Fig. 3. Two configurations in the manifold extraction of the tetrahedron with three marked surface triangles, a) initial status, b) wrong choice, c) correct choice

In order to avoid such situations it is necessary before the manifold extraction step to detect the tetrahedra, which have three marked faces, and remove one overlapped face. So we take one tetrahedron after another and we mark surface triangles (faces) using the CRUST algorithm. If there are three marked faces on one tetrahedron, we preserve only these two faces whose normals make the smallest angle (the tetrahedron is flat, so the triangles on the other edges make sharp angle together), the third face is deleted. We have to be careful with the orientation of the triangle normals, they have to be oriented in the direction of the tetrahedron centre of gravity (see an example in Fig. 4). The best configuration is in Fig. 4d), the angle between triangles normals incident on the edge is the smallest (the dot product of the normals is close to one, 4b) and 4c) are close to minus one).

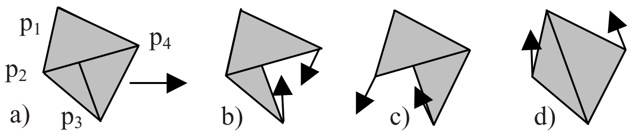

Fig. 4. Tetrahedron with three marked faces ($p_1p_2p_4$, $p_1p_2p_3$, $p_1p_4p_3$) and three possibilities which two triangles to choose. Arrows present the triangle normals

This approach converts tetrahedra with three marked triangles to tetrahedra with two marked triangles. We can use it to filter tetrahedra with four marked triangles too. Besides removal of problematic places, the prefiltering approach reduces the number of triangles in the primary surface. After converting all tetrahedra with four and three good faces to tetrahedra with two good faces, the set of primary surface triangles is ready for extraction.

4 Postfiltering

When we have the data, which are not uniformly sampled, with some noise or some features missing due to undersampling, the manifold extraction may fail because the CRUST selects bad surface triangles and unwanted triangle configurations occur (see Fig. 8a). This detail is taken from a dataset which is not uniformly sampled and contains some noise. The highlighted part presents the erroneous place after the manifold extraction – missing and overlapping triangles.

Missing and overlapping triangles appear there due to bad normal vectors arisen from the incorrect shape of Voronoi cells. We have analyzed triangle fans around the points obtained after the reconstruction. These configurations may be detected using an undirected graph. The nodes of the graph correspond to the fan triangles. A graph edge *e* exists in the graph if the nodes of the edge *e* correspond to neighboring triangles (see Fig. 5a), 5b).

There exist two acceptable configurations of the triangle fan. Fig. 5c) presents a full fan around a point. It can be detected as the graph cycle which contains all nodes. Fig. 5d) is just one single triangle, which can appear, e.g. on the corners of the surface with the boundary. Detection of these configurations is simple.

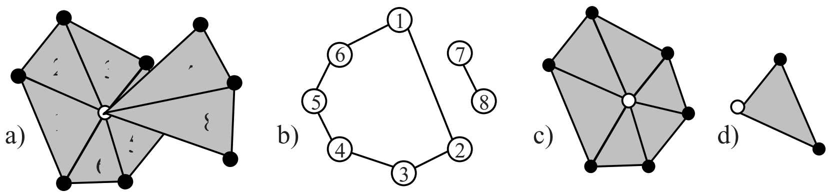

Fig. 5. a) Some fan configuration, b) a graph corresponding to the fan. Pictures c) and d) present acceptable fan configurations, c) a full fan, d) a point with one triangle

Other configurations are incorrect and some triangles have to be deleted. When we are able to find one cycle there, we can delete all triangles whose graph nodes are not included in the cycle. The most common configuration is shown in Fig. 6a), one full triangle fan with one separated triangle. Fig. 6b) is a hypothetic situation with more then one cycle but we did not find any occurrence of this.

The configurations presented in the Fig. 6c), 6d), 6e) are more problematic. When there are only subfans (we denote the fan as subfan if it does not form a cycle), the finding good fan configuration is not so simple and it will be explained in the following text. Here we can not avoid the use of the normal vectors (we are testing these configurations in the projected plane), and it can bring problems. The normal vectors have good estimation only on the smooth parts of the surface, but the places, where these problematic configurations of the fans appear, are on the places where the sampling is not correct.

All the triangles around the fan are projected to the plane given by the point (centre of the fan) and its normal vector (although the normal direction probably has not correct direction). The detection is simpler for the configuration in the Fig. 6c) and 6d) than 6e) because the triangles create only one subfan. When the sum of angles of the projected triangles (angle between two edges incident with the point) has less then 2π (Fig. 6c) the configuration is accepted and no changes in the triangle mesh is done. When it is more (Fig. 6d) we delete triangles from one end of the subfan until the angle is less then 2π. We have implemented just the removing from one end but it is better to remove these triangles in order to choose the sum of angles closer to 2π.

The Fig. 6e) represents the worst case, a set of more subfans. This configuration occurs fortunately very rarely and we remove all tringles except the subfan with the largest sum of angles.

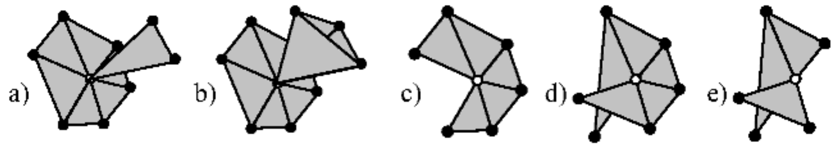

Fig. 6. Illustrations a) and b) present two types of configuration with a full fan, a) full fan with another separated triangle, b) more fans. Illustrations c) d) e) show some fan configurations (in a projection) without fans, c) one subfan, d) one overlapping subfan, e) more subfans

5 Results

The implementation of the CRUST algorithm and all of our improvements was done in Borland Delphi under the Windows XP system running on AMD Athlon XP+ 1500MHz processor with 1GB of memory. We have tested our implemented algorithm (Amenta's CRUST with our manifold extraction) together with Dey's COCONE algorithm [10, 11, 12], which is similar to the CRUST.

When we ran our algorithm without the prefiltering improvements, several triangle holes appeared. The number was not so high but when looking closer to the reconstructed object, it can disturb the visual perception and the object does not form the manifold. The same occurs in the Dey's algorithm (Fig. 7a). After prefiltering, the situation changed and our algorithm was able to reconstruct the surface with much less triangles holes (Fig. 7b), 7c). Some triangle holes still appear but the cause is different, the missing triangles did not pass the surface triangle test (recall Section 1).

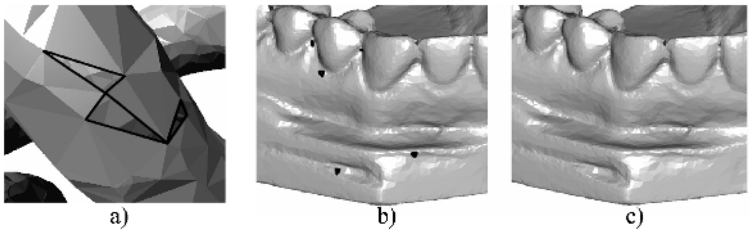

Fig. 7. a) Detail of the reconstructed surface by Dey's COCONE, black are highlighted triangle holes in the surface. The picture b) and c) shows the reconstruction using our approach, b) missing triangles are black, c) the same part of the surface after prefiltering applied

The next consequence of this prefiltering improvement was the reduction of the amount of triangles in the primary surface. We have measured (Table 1) the number of redundant triangles, which it is necessary to remove from the triangulation. The row "without" presents the number of redundant triangles marked as surface triangles without the prefiltering applied. The number of redundant marked surface triangles computed with the help of the prefiltering is in the row "prefilter". The last row presents the rate in percents of the number of marked triangles before applying prefiltering and the number of triangles after prefiltering. It can be seen that 38-99 percent of the redundant triangles are removed by prefiltering.

Table 1. Number of points ("N") in datasets used for testing, Number of triangles marked as surface without prefiltering ("without"), number of triangles with prefiltering ("prefilter") and the percent rate of the removed triangles using the prefiltering ("rem")

	bone	**bunny**	**x2y2**	**engine**	**hypshet**	**knot**	**mann**	**nascar**	**teeth**
N	68537	35947	5000	22888	6752	10000	12772	20621	29166
withou	8106	11937	358	9835	1451	2017	926	992	4642
prefilte	111	71	122	33	898	70	54	297	145
% rem	98	99	65	99	38	96	94	70	96

Now we will present the result of postfiltering. In Fig. 8a) we can see the case where some bad fan (or umbrella) configurations appear, in Fig. 8b) the same part of the surface after applying the postfiltering is shown. The overlapped "flying" triangles disappear and the remaining triangle holes are filled with the triangles.

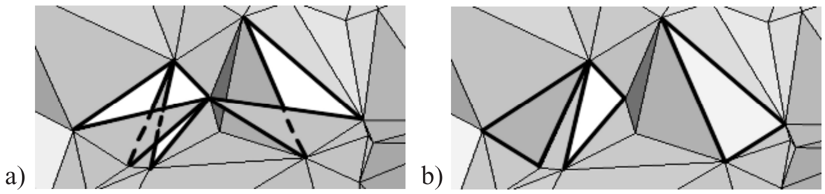

Fig. 8. a) The part of the surface with and b) without bad fans after postfiltering

Our implementation of manifold extraction is able to find all holes in the surface, but the holes filling is now limited to the triangles holes (as presented in Fig. 9). Not all holes are so small, we are planning in the future to develop or apply some simple algorithm for holes triangulation.

The same problem occurs using the Dey's algorithm, we found overlapping triangles on the surface of the reconstructed objects, too (Fig. 9a), 9b). In this case, we were not able to reproduce Fig. 9a), 9b) by our program, because although the algorithms are similar, the code is not the same and the reconstructed meshes differ a little for the same models. Fig. 9c) and 9d) shows the same part of the reconstructed model using our approach and the same part after postfiltering.

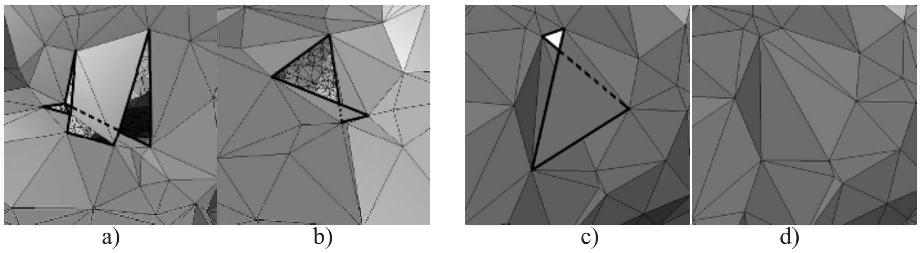

Fig. 9. a), b) The overlapping triangles in the surface reconstructed using COCONE, c) the overlapping triangles from our reconstruction without and d) with postfiltering

6 Conclusion

We have presented two improvements to the manifold extraction step in surface reconstruction problem. When the surface is not well sampled or a noise is present, some surface triangles are missing or other hybrid triangles appear. Our tests show that it is not a problem only of our algorithm. The prefiltering improvement helped us with the missing triangles in the smooth places and it makes the manifold extraction a little faster. The postfiltering improvement prevents from the creation of overlapped triangles, the holes are later triangulated. That would be the next step of our development, to use the existing structures and to better develop this step, or to use some existing algorithm, for a hole retriangulation.

References

1. M. E. Algorri, F. Schmitt. Surface reconstruction from unstructured 3D data. Computer Graphic Forum (1996) 47 - 60
2. N.Amenta, M.Bern, M.Kamvysselis.A new Voronoi-based surface reconstruction algorithm. SIGGRAPH (1998) 415 - 421
3. N.Amenta, M.Bern. Surface reconstruction by Voronoi filtering. Discr. and Comput. Geometry 22 (4), (1999) 481 - 504
4. N. Amenta, S. Choi, T. K. Dey, N. Leekha. A simple algorithm for homeomorphic surface reconstruction. 16th. Sympos. Comput. Geometry (2000)

5. N.Amenta, S.Choi, R.Kolluri. The PowerCrust. Proc. of 6th ACM Sympos. on Solid Modeling (2001)
6. F.Bernardini, C.Bajaj. A triangulation based. Sampling and reconstruction manifolds using a-shapes. 9th Canad. Conf. on Comput. Geometry (1997) 193 - 168
7. F.Bernardini, J.Mittleman, H.Rushmeier, C.Silva, G.Taubin. The ball pivoting algorithm for surface reconstruction. IEEE Trans. on Vis. and Comp. Graphics 5 (4) (1999)
8. J.D.Boissonat. Geometric structures for three-dimensional shape representation. ACM Trans. Graphics 3, (1984) 266 - 286
9. B.Curless, M.Levoy. A volumetric method for building complex models from range images. SIGGRAPH (1996) 302 - 312.
10. T.K.Dey, J.Giesen, J.Hudson. Delaunay Based Shape Reconstruction from Large Data. Proc. IEEE Sympos. in Parallel and Large Data Visualization and Graphics (2001)
11. T.K.Dey, J.Giesen, N.Leekha, R.Wenger. Detecting boundaries for surface reconstruction using co-cones. Intl. J. Computer Graphics & CAD/CAM, vol. 16 (2001) 141 - 159
12. T.K.Dey, J.Giesen. Detecting undersampling in surface reconstruction. Proc. of 17th ACM Sympos. Comput. Geometry (2001) 257 - 263
13. T.K.Dey, S.Goswami. Tight Cocone: A water-tight surface reconstructor. Proc. 8th ACM Sympos. Solid Modeling application (2003) 127 - 134 [27]
14. H.Edelsbrunner, E.P.Mücke. Three-dimensional alpha shapes. ACM Trans. Graphics 13 (1994) 43 - 72
15. H.Edelsbrunner. Weighted alpha shapes. Technical report UIUCDCS-R92-1760, DCS University of Illinois at Urbana-Champaign, Urbana, Illinois (1992)
16. H.Hoppe, T.DeRose, T.Duchamp, J.McDonald, W.Stuetzle. Surface reconstruction from unorganized points. Computer Graphics 26 (2) (1992) 71 - 78
17. J.V.Müller, D.E.Breen, W.E.Lorenzem, R.M.O'Bara, M.J.Wozny. Geometrically deformed models: A Method for extracting closed geometric models from volume data. Proc. SIGGRAPH (1991) 217 - 226
18. S.Muraki. Volumetric shape description of range data using "Blobby model". Comp. Graphics (1991). 217 - 226
19. R.Mencl, H.Müller. Graph based surface reconstruction using structures in scattered point sets. Proc. CGI (1998) 298 - 311
20. M.Varnuška, I.Kolingerová. Improvements to surface reconstruction by CRUST algorithm. SCCG Budmerice, Slovakia (2003) 101-109