Surface Reconstruction From Problem Point Clouds

Alexander Emelyanov, Vaclav Skala

ABSTRACT

We present an algorithm for creating a CAD model of an existing physical object from a scanned point cloud containing badly scanned regions. The algorithm has a linear complexity and can be applied to processing big clouds. For the triangulation of well-scanned regions, a method from the group of greedy triangulation algorithms is used. The possibility and efficiency of use of very simple and fast tests for this method are shown. Triangulation of badly scanned regions is done using a projection onto 2D surface. The projection surface is defined locally for each region on the basis of already triangulated well-scanned regions. The presented algorithm can reconstruct a surface when well-scanned regions are represented by isolated "islands".

Keywords: *surface reconstruction, greedy triangulation*.

1. INTRODUCTION

The problem of creating a CAD model for an existing physical object from a given point set from object surface is important for many fields of science and industry. In most cases it is possible to receive the input point cloud having necessary properties (density, linear and angular isotropy). In this case a model can be successfully constructed by the application of existing algorithms. However, in some cases, obtaining of the initial point cloud with good enough characteristics can be complicated or impossible. Typical examples are architectural monuments, the objects explored from distance, e.g., objects in space. For the cloud of points obtained from these objects, there is a typical combination of regions having good allocation of points, and regions with unsatisfactory allocation of points, or without them.

2. PREVIOUS WORK

Generally, there are two types of surface reconstruction methods: interpolation and approximation. Among algorithms of the first type in this paper we cite several typical. The algorithm presented in [EM94] uses α-shapes. In this algorithm α-shape is a heuristic. It works well for uniform sampling. The major drawback of using α-shapes for surface reconstruction is that the optimal value of α depends on the sampling density, which often varies over different parts of the surface. The "crust" [ABK98] is provably correct if the sampling density is high enough everywhere. But the local topology may be changed and holes may appear due to undersampling. The algorithm [DG01] allows to reconstruct a surface from an input, that is not sufficiently sampled. But it is not suitable for clouds having serious problems. Among algorithms of the second type the typical examples are [HDD92], [HDD93], [CL96]. They exploited the fact that a surface can be reconstructed from its normal orientations because we can get tangent planes from normals and the area around the normal on a tangent plane is the first order approximation of the local surface. These algorithms need to estimate normals from the point set very accurately. One more way is approximating by radial-basis functions (RBF) for processing point clouds with local problems [CFB97]. But this and similarly methods are not suitable for processing of big clouds.

3. FIELD OF USE

A proper reconstruction of an actual surface is possible only if it is *sufficiently sampled*. There are three conditions we adduce about a sufficiently sampled surface [Att97, ABK98]. Firstly, the sampling of the data should be *locally uniform*, which means that the distance ratio of the farthest and closest neighbor of a sample in the given sampling of the object is less than a constant value. The second condition is to distinguish points from two close layers of the object. The closest distance between a point P in one layer and another layer is at least $k\eta$, where k is a constant and η is the shortest distance between P and another point in its layer. The third condition is about the smoothness of the underlying object. The normal deviation between any two triangles incident on a vertex should be less than 90°.

For an estimation of a degree of local uniformity at a point P we use the factor μ_N^α, where α is an angular condition and N is a number of the nearest neighbors of P, which satisfy to the given angular condition. We calculate this factor using a below described algorithm (initially all the neighbors of P are unmarked, no one vector is admitted):

do
{
V = vector from P to the nearest unmarked neighbor Q;

if (angle between V and any of the already admitted vectors < α) continue;

admitting V and marking Q;
}
while (number of the admitted vectors < N);

L = the longest admitted vector length;
S = the shortest admitted vector length;

$$\mu_N^\alpha = L / S;$$

Usually, we consider, that N = 6 (it is the most probable number of edges of a point), α = 30° (it is a typical boundary value in the majority of works in the field of a triangulation). In the given paper we shall denote μ_6^{30} as μ.

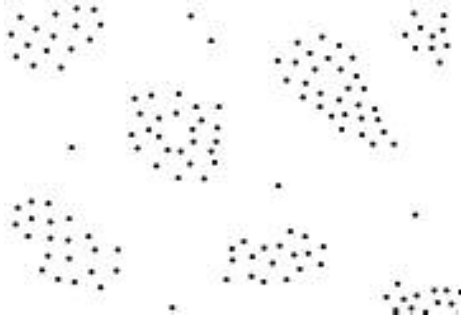

Figure 1: Expected input

As was mentioned above, the algorithm is intended for processing point clouds, which were obtained by physical scanning architectural objects and other objects explored from distance. Generally, point clouds from such objects have follows distinctive properties:

- about 80% of points are contained in well-scanned regions, irrespective of a relation between total areas of well-scanned and problem regions;
- typically, in well-scanned regions $\mu \le 3$, $k > 2\mu$;
- well-scanned regions generally are isolated from each other (let's call such regions as islands), in more opportunity there is one coherent well-scanned region with holes;
- a point cloud can have a big size (more than 5 million points).

Schematically an expected kind of input is shown in Fig. 1. Thus, on the basis of the mentioned above properties of expected point clouds we shall call a region of a point cloud as *sufficiently sampled region (SSR)* if for any point of the region $\mu \le 3$, $k > 2\mu$, and the region satisfies the mentioned above condition of smoothness.

4. ALGORITHM OVERVIEW

Outgoing from the mentioned above properties of input point clouds, for surface reconstruction a strategy of several consecutive steps was chosen.

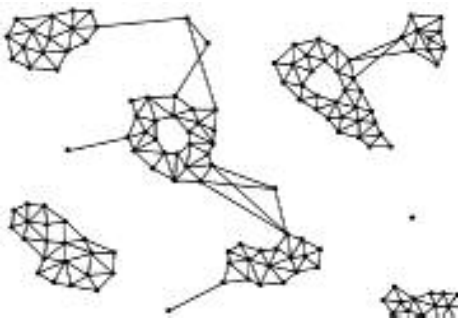

Figure 2

Step 1. Making a draft mesh, that mainly is correct in SSRs only (Fig. 2).

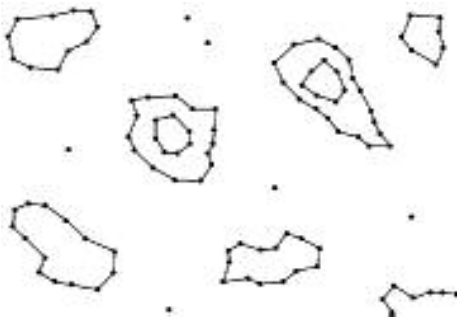

Figure 3

Step 2. Determination of correctly triangulated regions and their boundaries (Fig. 3).

Step 3. Determination of the status (hole or island) of the boundaries obtained at the previous step.

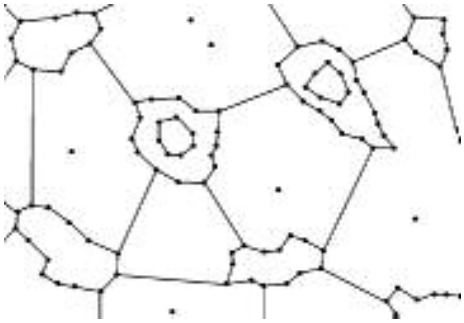

Figure 4

Step 4. Making connections between the islands (let's call them *bridges*) and then extracting new holes, which are made by fragments of the islands boundaries and the bridges (Fig. 4).

Step 5. Decomposition of holes having the complex form.

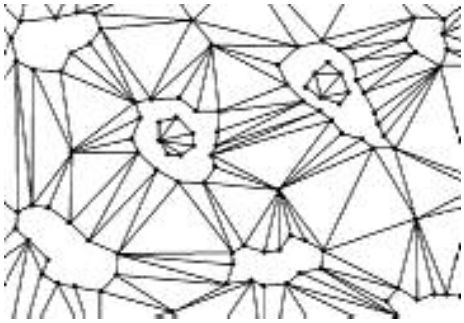

Figure 5

Step 6. Triangulation of the region inside each detected hole (Fig. 5).

Step 7. Postprocessing.

Generally, this algorithm uses the ideas of greedy triangulation (GT) and belongs to the group of interpolating methods. The GT of a point set in the plane is the triangulation obtained by starting with the empty set and at each step adding the shortest compatible edge between two of the points [BE95]. In 2D a compatible edge is defined to be an edge that crosses none of the previously added edges. Step 1 is an extension of 2D GT-strategy for 3D, step 7 uses classical 2D GT algorithm. Steps 3, 4, 5 which provide possibilities of processing problem point clouds, generally lie outside the field of usually considering triangulation problems. But step 4 partially can be also related to the group of GT algorithms.

5. ALGORITHM REALIZATION

Step 1. The basic goal of this step is a fast triangulation of SSRs to obtain the necessary information of a reconstructed surface behavior. As distribution of points and properties of a surface in these regions are good (see section 3), for a triangulation complex algorithms are not required. For using GT-strategy in 3D a special very simple and fast test was designed. This test analyzes a topology of a created mesh at the place of prospective inclusion of an edge, and can be formulated as follows: if insertion of the current tested edge leads to an appearance of the edges having more than two adjacent triangles or leads to appearance of a tetrahedron, the edge is considered as incompatible and compatible otherwise. The given test does not use any floating-point operations, and is passed fast.

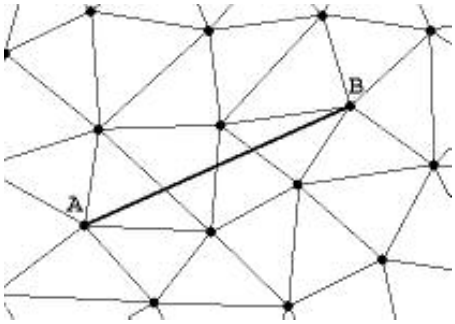

Figure 6: Edge without adjacent triangles

This test can't prevent appearance of edges having less than 2 adjacent triangles (edge AB in Fig. 6). Therefore, at the end of the step we eliminate all such edges. Naturally, process based on such simple test can't provide 100% reliability for triangulation of even SSRs - typically we have about 95% correctly connected points of such regions. However, in a combination with the subsequent test revealing errors in triangulation (at the step 2), application of such test is justified, because:

1) total expenses of the triangulation with the use of the given test and the subsequent step of a verification appear considerably lower, than expenses of triangulation with the use of more powerful tests;
2) we very quickly obtain a correct triangulation of about 80% points, and as a consequence, we obtain a general behavior of the reconstructed

surface; correctly linked points can be excluded from the further consideration, it is especially important thing for the big clouds;

3) additional 2-3% increment of the total area of problem regions by parts of SSRs that were incorrectly triangulated is not essential.

It is necessary to note that the algorithm described above is optimized for processing point clouds having mentioned above properties. For other cases at the given step another algorithm can be used, that is more suitable for a given point cloud.

Step 2. At this step the check of the work of the previous step and determination of boundaries of correctly triangulated regions is made. Also at this step we determine the points, which not belong to a correct triangulation (let's call them *lost points*). For this step we use a variant of the well-known "umbrella" filtering [AGJ02].

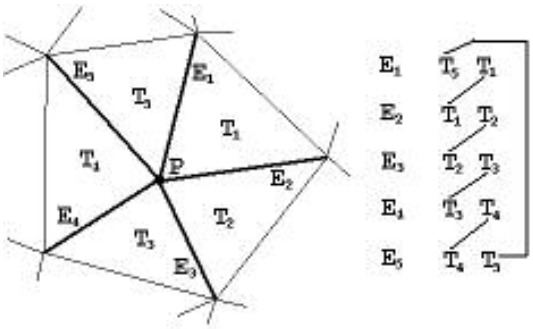

Figure 7: "Umbrella" test

For a given tested point we consider all its edges, and for each edge we determine pair of triangles sharing this edge. In the Fig. 7 a point P with 5 outcoming edges is shown. Edge E1 has the pair of adjacent triangles (T1, T5), edge E2 has the pair (T1, T2) and so on. It can be proved that the point belongs to a correct triangulated area if and only if such pairs of triangles make a correct closed chain. The given condition can be applied also to border points if we put, that the absent triangle is a special kind of "empty" triangle.

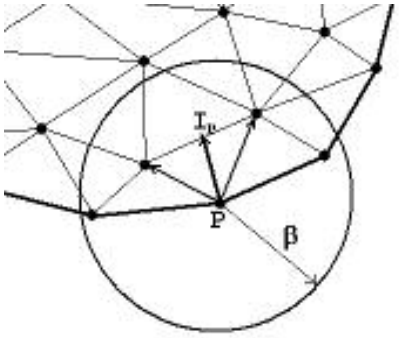

Figure 8: Determination of the boundary vector of a point

After detection of all correctly connected points the boundaries of correctly triangulated regions are found. For each point P (Fig. 8) of a boundary we determine a special *boundary vector* I_p. This vector is average of vectors from P to neighbor points, which are located closer than the given distance β (with exception of the boundary points).

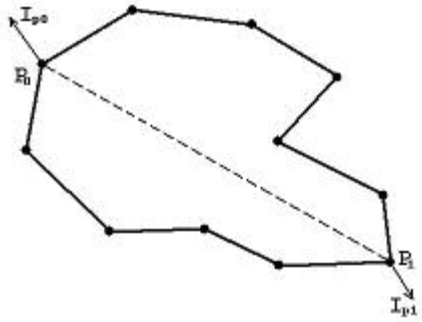

Figure 9: Determination of the status of a boundary

Step 3. For determination of the status (hole or island) of an obtained boundary the test described below is used (Fig. 9). Initially, some point P_0 on the boundary is fixed. Then on the boundary the farthest from P_0 point P_1 is searched. We analyze dot products: between the vector $\mathbf{P_0P_1}$ and the boundary vector of P_1 (I_{p1}), and also between the vector $\mathbf{P_1P_0}$ and the boundary vector of P_0 (I_{p0}). If both dot products are positive - the boundary is considered a hole, if both are negative - the boundary is considered an island. If these scalar products have different signs, we fix a new point P_0 and repeat the test. The testing stops when 3/4 or more dot products have the same sign.

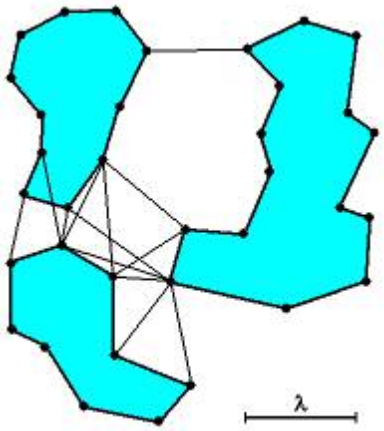

Figure 10a: Set of all the created candidate bridges

Step 4. At this step we make bridges between islands and then extract new holes made by fragments of the island boundaries and the bridges. Initially, for all islands, for all points of an island boundary we make set of the shortest candidate bridges to each of other island boundaries (Fig. 10a). To avoid a quadratic complexity, a candidate bridge length is bounded by an upper limit λ and a spatial decomposition is applied.

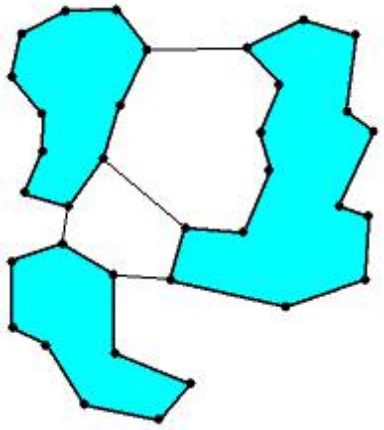

Figure 10b: Set of the admitted bridges

Then all the created candidate bridges are ordered in ascending order of their lengths. After sorting we test bridge by bridge from the sorted list and admit only bridges (Fig. 10b), which meet the conditions given below:

- the sphere circumscribed around a candidate bridge does not contain any other points of any boundary;
- there is no already admitted bridge at the boundary points, connected by the tested candidate bridge.

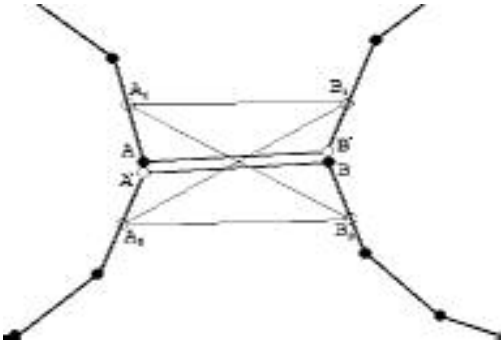

Figure 11: Bridge installation

Then we install all the admitted bridges. Each installed bridge consists of two edges. At a bridge installation (Fig. 11) between point A of a first island boundary and point B of a second island boundary each boundary is disconnected at the corresponding point. For it we insert additional points A' and B' having the same coordinates that points A and B. For determination of a connection order of these points, the auxiliary points laying on boundaries on some small distance from the points A,A' (B,B') are considered. If the angle between the vectors $\mathbf{A_0B_0}$ and $\mathbf{A_1B_1}$ is smaller than the angle between the vectors $\mathbf{A_0B_1}$ and $\mathbf{A_1B_0}$ then we install the edges pair (AB**,** A'B') and the pair (AB'**,** A'B) otherwise.

After installation of all the admitted bridges the extraction of holes formed by fragments of the islands boundaries and edges of the installed bridges is made. Thus, as a result of this step we have only some set of holes (holes obtained at the given step and internal holes of islands) and the set of lost points for consideration at subsequent steps.

Step 5. At this step a recursive decomposition of complex holes is carried out until each hole has an unambiguous projection onto the average plane of the own boundary points (let's call this plane as *hole's average plane*).

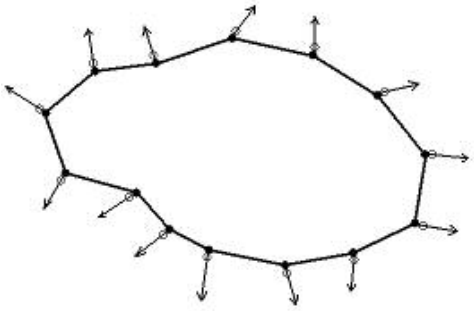

Figure 12: Determination of auxiliary boundary points

For a determination, whether a hole has unambiguous projection onto its average plane or not, for each point of the hole boundary we define the *auxiliary boundary point* (Fig. 12). This point (featured by an empty circle) is the end of the point's boundary vector, that is reduced to the length equal some small value ε.

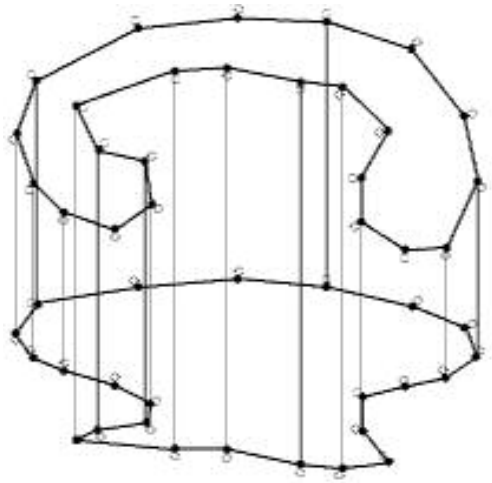

Figure 13: Case of ambiguous projection of a hole

Thus, if the line of hole's boundary projection don't cross itself, and all the projections of auxiliary boundary points lie outside the region, restricted by this line, we consider, that the hole has unambiguous projection. In Fig. 13 the case is shown, when a hole has ambiguous projection (when several projections of auxiliary boundary points are inside the hole projection area).

If a given hole hasn't unambiguous projection, we make its recursive binary subdivision until each resulting hole has unambiguous projection onto own average plane. The subdivision is made by insertion a chord between two points of the hole boundary. The method of a chord insertion is similar to the method of the installation of a bridge between islands (we create two additional points and make two edges for each chord).

Step 6. A triangulation of the region inside each hole is carried out by a projection of the hole boundary and the lost points located inside it onto the hole's average plane. For the triangulation, a variant of the classical GT-algorithm for 2D is applied [BE95], but for determination of the current shortest edge the 3D distance between the points used.

Step 7. At this step we remove auxiliary points and edges, which were added at the installing bridges and chords (steps 4, 5).

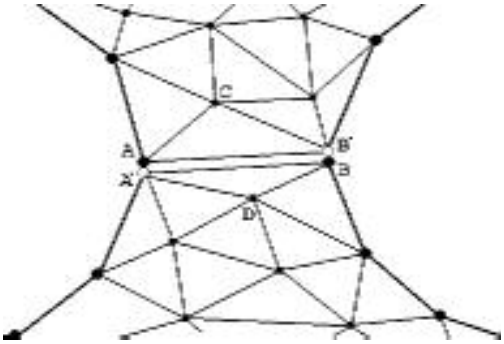

Figure 14a: Triangulation with auxiliary links

For each bridge or chord we initially eliminate auxiliary points A' and B' and reconnect all their edges to the points A and B respectively (Fig. 14a).

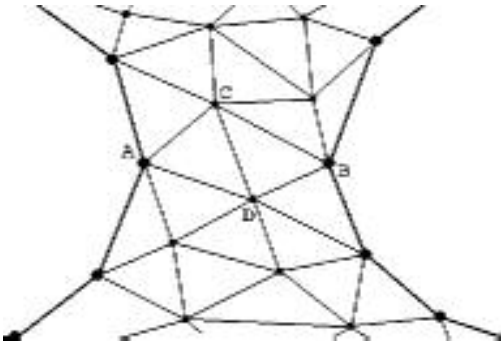

Figure 14b: Triangulation after auxiliary links removing

If the edge AB is longer, than the distance between adjacent triangles vertices C and D, we replace edge AB by edge CD (Fig. 14b).

6. RESULTS AND CONCLUSION

The results of the algorithm application for several samples are shown in Table 1. All the timing measurements were made on a PC with 1500 MHz Athlon and 1GB main memory.

"Bunny" (Fig. 15) and "Bone" (Fig. 16) are widely used for testing models. Also we use two artificially damaged variants of "Bunny" to test. In several regions points were removed and in several regions points density was decreased 10 times. Model "Bunny1" (Fig. 17a) is a little damaged, and model "Bunny'2" – heavy (Fig. 18a). In the Fig. 17b, 18b the same models are shown after surface reconstruction. The "Face" is a fragment of a distance scanned big sculpture. In the Fig. 19a the original point cloud and in the Fig. 19b the reconstructed and lighted surface are shown.

The obtained results allow to make the following conclusions:

- The algorithm shows a better speed (in comparison with [ACD00, DG01]) at processing point clouds, which are almost completely represented by a SSR having a little μ ("Bunny").
- The algorithm shows results comparable to the [DG01] at processing point clouds, which are mainly represented by one continuous SSR, with a small amount of problem regions caused by breaking the conditions of SSR ("Bone"), or problems with sampling ("Bunny1").
- The algorithm can reconstruct the closed surface from point clouds having serious problems, when available algorithms show unsatisfactory results ("Bunny2") or can't be applied ("Face"). However, in especially difficult cases (in Fig. 18b marked by a circle) the algorithm still is not capable to restore an original surface correctly.

As further works we plan to increase abilities of steps 3, 4, 5, which are crucial for processing problem point clouds.

REFERENCES

[ABK98] N. Amenta, M. Bern, M. Kamvysselis. A New Voronoi-Based Surface Reconstruction Algorithm. In Proc. of SIGGRAPH'98, pp.415-421.

[ACD00] N. Amenta, S. Choi, T. K. Dey, N. Leekha. A simple algorithm for homeomorphic surface reconstruction. In Proc. of 16th Sympos. Comput. Geom., 2000, pp.213-222.

[AGJ02] U. Adamy, J. Giesen, M. John. Surface Reconstruction using Umbrella Filters. Computational Geometry Theory & Applications 21(1-2): pp.63-86, January 2002.

[Att97] D.Attali. r-regular shape reconstruction from unorganised points. In ACM Computational Geometry, 1997, pp.248-253.

[BE95] M. Bern, D.Eppstein. Mesh Generation and Optimal Triangulation. Computing in Euclidean Geometry, D.-Z. Du and F.K. Hwang, eds., World Scientific, 1992, 2nd edition, 1995.

[CFB97] J. C. Carr, W. R. Fright, R. K. Beatson. Surface interpolation with radial basis function for medical imaging. IEEE Trans. Medical Imaging, 16(1): pp.96-107, February 1997.

[CL96] B. Curless, M. Levoy. A volumetric method for building complex models from range images. In Proc. of SIGGRAPH'96, pp.303-312.

[DG01] T. K. Dey, J. Giesen. Detecting undersampling in surface reconstruction. Proc. 17th Sympos. Comput. Geom., 2001, pp.257-263.

[EM94] H. Edelsbrunner, D. Mucke, Three-dimensional Alpha Shapes. In ACM Transactions on Graphics, 1994, 13, pp.43-72.

[HDD92] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle. Surface reconstruction from unorganized points. In Proc. of SIGGRAPH'92, pp.71-78.

[HDD93] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle. Mesh Optimization. In Proc. of SIGGRAPH'93, pp.19-26.

About authors:

Department of Computer Science, University of West Bohemia, Plzen, Czech Republic
Alexander Emelyanov: E-mail aiem@kiv.zcu.cz
Vaclav Skala: E-mail skala@kiv.zcu.cz

Acknowledgments

We thank Ivana Kolingerova (UWB) for her helpful comments.
This work was supported by the Ministry of Education of the Czech Republic - project MSM 23500005.

Sample	N points	Average μ in SSRs	Area of SSRs (%)	N points in SSRs (%)	Used steps	Time (sec)	Speed (point/sec)
Bunny	35947		98	98	1, 2, 6	3.6	9985
Bone	68537	2.4	87	94	1, 2, 6	15.1	4539
Bunny1	29491	1.9	82	94	1, 2, 6	14.2	2077
Bunny2	26553	1.9	69	94	1, 2, 5, 6, 7	20.9	1270
Face	232511	1.8	63	82	1, 2, 3, 4, 5, 6, 7	385.8	603

Table 1

Figure 15

Figure 16

Figure 17a

Figure 17b

Figure 18a

Figure 18b

Figure 19a

Figure 19b