Delaunay Surface Reconstruction from Scattered Points

Angel Rodríguez[1], José Miguel Espadero[1], Domingo López[2], and Luis Pastor[3]

[1] U. Politécnica de Madrid, Dep. de Tecnología Fotónica,
Campus de Montegancedo s/n, 28660 Boadilla del Monte, Madrid, Spain
{arodri,jespa}@dtf.fi.upm.es
[2] Visual Tools S.A., Dep. de I+D.,
C. Xaudaró, 13 bis, 2^a planta. 28034, Madrid, Spain.
dlopez@vtools.es
[3] U. Rey Juan Carlos, Dep. de Ciencias Experimentales y Tecnológicas,
C. Tulipán, s/n, 28933 Móstoles,Madrid, Spain.
lpastor@escet.urjc.es

Abstract. The use of three-dimensional digitizers in computer vision and CAD systems produces an object description consisting of a collection of scattered points in $\mathbb{R}^3$. In order to obtain a representation of the objects' surface it is necessary to establish a procedure that allows the recovering of their continuity, lost during the data acquisition process. A full automatic $O(n^2)$ algorithm is presented. Such algorithm obtains surface representations of free genus objects described from a set of points that belong to the original surface of the object. The only information available about each point is its position in $\mathbb{R}^3$. The achieved surface is a Delaunay triangulation of the initial cloud of points. The algorithm has been successfully applied to three-dimensional data proceeding from synthetic and real free shape objects.

Keywords: Automatic surface reconstruction, 3D Delaunay triangulation, 3D Modeling

1 Introduction

This article deals with the problem of tessellating real objects' surface representations in an automatic way. The free genus objects are exclusively defined by a set of three-dimensional points that belong to their surface. This problem can be found in areas such as computer vision or CAD, that accept object descriptions based on 3D scattered points. The input data is composed by a set of unstructured points which represent the samples digitized from the surface of the objects using passive or active acquisition techniques [23]. Furthermore, they could be obtained from the segmentation of 3D volumetric images, although this approach is more frequently used in medical environments because of its high cost [20]. From this set of points, represented by their 3D coordinates, we are intending to automatically build a mesh of triangular facets produced by the proximity

G. Borgefors, I. Nyström, and G. Sanniti di Baja (Eds.): DGCI 2000, LNCS 1953, pp. 272–283, 2000.

relationships between the points of the surface. Another desirable property of the mesh is to be as regular as possible. So, the extracted mesh under these conditions will be a Delaunay triangulation.

1.1 Related Work

The use of 3D digitizers to capture the shape of the objects is more and more frequent in computer vision and CAD systems. This kind of devices make a discrete sampling over the surface of the object obtaining a description based on a cloud of points. However, we know few works in $\mathbb{R}^3$ that deal with the automatic recovering of the surface continuity without more information that the points' coordinates. Most of the existing techniques manage some type of additional information in order to characterize the cloud of points. The $\alpha-$shapes of Edelsbrunner *et al.* [10] are a good effort to formalize the concept of "shape" for the discussed descriptions. Several solutions require to use some kind of manipulation from a human being, making the process highly interactive [5,21]. In other cases, some limits to the data sampling process are fixed [22,8,1]. For example, Amenta and Bern restrict the sampling distance to avoid the problems that appear with the existence of creases and corners. Other methods need to know the orientation or the position of the sensor during the sampling stage in order to recover the connectivity between the points of the surface [12,26]. Another approach is to use the orientation of the normals associated to each point to establish neighbor relationships between the samples [4]. Hoppe *et al.* [19] and Bajaj *et al.* [3] define a signed distance function to compute its zero-set, considering the surface as an implicit function defined over the input data. Another alternative approach used by some authors consists to perform a deformation over a reference mesh by means of an iterative process which adjust the mesh to the cloud of points [25,27,18]. The only known work in 3D that deals with the stated problem is the Attali's one [2], where the boundary and surface extraction in two and three dimensions is intended to be formalized. In 2D she extracts the boundary from a Delaunay triangulation and uses the existence of topological relationships between neighbor points with the so-called *τ-regular shapes*. However, the extrapolation of this problem to 3D can not be formalized in the same way so she offers a heuristic solution to extract the surfaces, but limited to closed surfaces.

The contents of this paper is as follows. Section 2 describes the algorithm implemented to obtain the Delaunay tetrahedralization. Section 3 presents the algorithm that allows the computation of the objects' surface Delaunay triangulation in an automatic way. Section 4 shows some meshes extracted with the proposed algorithm. Last, section 5 summarizes the paper's main conclusions.

2 3D Delaunay Tetrahedralization

It is well known that there is an equivalence relationship between a Voronoi diagram and the corresponding Delaunay triangulation that allows changing from one type of representation to its dual by means of a $O(n)$ process [12].

For our purposes, we have chosen the incremental algorithm described by Faugueras in [12] to obtain a Delaunay tetrahedralization of the cloud of points, although there are other methods to compute the Delaunay representation [17,11,6,9]. It is a $O(n^3)$ algorithm that can be summarized in the next steps:

1. Compute a Delaunay tetrahedralization of the vertex of a cube $\{V_i,\ i = 1,\ldots,8\}$ that contains all the points $\{M_j \mid M_j \in \mathbb{R}^3 \quad j = 1,\ldots,p\}$ belonging to the surface of the object, and the centers $\{C_i,\ i = 1,\ldots,8\}$ and the radius $\{R_i,\ i = 1,\ldots,8\}$ of the circumscribed spheres of the computed tetrahedra. The centers of the spheres are the Voronoi points.
2. If M_j is the current point to be inserted in the tetrahedralization and k is the current number of tetrahedra, each new point M_j will belong, at least, to one of the current spheres. If it would belong to p spheres, all the tetrahedra of those spheres must be marked in order to be deleted from the tetrahedralization, because the Voronoi region $R(p_i)$ is a convex polyhedron that contains the corresponding p_i generator inside it. E.g., if $d^2(M_j, C_i) - R_i^2 \leq 0\ i = 1,\ldots,k$ then mark the i-th tetrahedron T_i.
3. For all the marked tetrahedra, extract the list of their faces.
4. Remove from this list all the faces that appear twice.
5. For each non-removed face, create a new tetrahedron with M_j and insert it in the tetrahedralization.
6. Delete from the tetrahedralization all the marked tetrahedra and return to 2 until there are no more points to process.

One of the main advantages of this algorithm is the local nature of the operations performed from steps 3 to 6, because the point M_j deals with a reduced number of Delaunay tetrahedra. This allows a local and simple update of the tetrahedralization.

To solve the degeneration cases of the Voronoi diagram computed with this algorithm, produced by numerical errors or by the own location of the points, it is possible to lightly modify the coordinates of the new point until the degeneration disappears and then insert it in the usual way. Another option is to discard this point from the cloud and not to include it in the tetrahedralization.

3 Surface Extraction Algorithm

The purpose of the surface extraction algorithm is first to remove the tetrahedra of the simplicial complex resulting from the Delaunay tetrahedralization described in section 2 that are outside of the object, and then extract the external surface as the list of facets of the simplicial complex that only belong to one tetrahedron. In this process it is necessary to use a heuristic to determine those tetrahedra that are inside of the object, because there is not a deterministic procedure to do this only knowing the Cartesian coordinates of the sampling points. Our proposed heuristic is to start with the tetrahedra placed inside of the convex hull of the objects' cloud and remove those that offers to the outside any facet greatest than some given threshold, repeating the process until the

size of all the exterior facets is under the threshold. There are several ways to measure the size of a facet, for example, the perimeter, the area or the radius of the circumscribed circle. We have chosen the perimeter for two main reasons:

1. From a computational point of view, it is faster and easier to compute than other measures.
2. The tests performed have shown a better response of this measure than the others mentioned. We think that this is due to the fact that our starting simplicial complex is a Delaunay triangulation. As it is well known, the Delaunay triangulation has the property that the partition obtained maximizes the minimum angle of the triangular facets achieved. Although this property approximates the triangular facets to equilateral triangles, sometimes the position of the samples makes very irregular triangles, so the area or the radius are not very good measures to clean the tetrahedralization. With this in mind, we are interested to compute a mesh where the distance between the samples, remember that the samples are the vertices of the triangles, is minimum.

Following is the description of the algorithm that we propose.

Algorithm

1. Begin using the Delaunay tetrahedralization computed from the input cloud of points with the algorithm described in section 2.
2. Discard all the tetrahedra having a vertex $\{V_i, i = 1, \ldots, 8\}$ that belongs to the external cube defined in step 1 of the tetrahedralization algorithm (section 2). The result is a simplicial complex of the convex hull of the input data.
3. Compute the external surface of the complex as the list of facets that only belong to one tetrahedron.
4. Compute the threshold v_u as the mean of the perimeters of the facets in the external surface list. Sometimes it will be interesting to multiply this value by a factor, as will be explained below.
5. If a facet in the external surface list exceeds the threshold v_u, remove from the complex the tetrahedron who owns the facet and insert the remainder three facets in the surface list.
6. Repeat step 5 until all the perimeters of the facets in the surface list are below v_u.
7. Recompute the mean perimeter of the external surface. If the difference with respect the old value is up 1%, return to step 4.

The most time consuming step of this algorithm is, by far, the calculus of the Delaunay tetrahedralization in step 1. The main disadvantage of the method is that it employs a global threshold for all the surface, so if the input points are not registered in a uniform way and there is too much variability in the perimeter measures along the surface, some holes will appear over the areas with less density of points. This could lead to the loss of some of the points from

the original input representation. Sometimes, instead of computing the surface in a fully automatic way, it is possible to achieve better results if we multiply the mean perimeter value by a factor between 1.0 and 2.0 in the step 4 of the algorithm.

The complexity of the algorithm from steps 2 to 7 is $O(n^2)$, where n is the number of tetrahedra of the tetrahedralization. Each iteration treats n tetrahedra, and in the worst case, we need to loop $n-1$ times.

4 Experimental Results

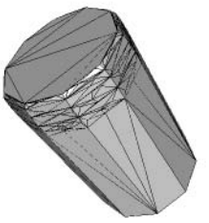
The main purpose of this section is to present some representative objects used in the experiments and the results achieved with the algorithm proposed. The meshes shown in this section have been generated on a Silicon Graphics Origin2000. The main hardware features of this multiprocessor are 2 GB of RAM and eight 250 MHz R10000 MIPS processors, although our actual implementation computes only over one of the eight available processors. The application is implemented in C++ and has been ported successfully to several UNIX based platforms (Linux). Although the development has been performed on a proprietary computer, we have used free development distribution tools to assure the portability of the code. The development tools have been those that provide GNU [16], and the 3D visualization of the objects has been carried out with Geomview [14].

The clouds of points presented with more detail in this paper belong to two synthetic objects, the goblet and the sea shell, proceeding from the GTS Library [24]. GTS is an Open Source Free Software Library intended to provide a set of useful functions to deal with 3D surfaces meshed with interconnected triangles. A third object with higher genus than the previous has been considered too with a similar detail: a human skull extracted from 3D tomographic real data. A fourth object proceeding from the Large Geometric Models Archive of the Georgia Institute of Technology is presented: a human skeleton hand [15]. In this case, the number of available samples is very higher, but the hand presents areas where the density of points is very variable too. The reason to show these examples is that the fourth objects present very different geometric features that allow to illustrate how the algorithm works. The selection of the first two synthetic shapes is motivated by the fact that the available sensors can not capture the internal geometry of the objects, like the details recovered inside the sea shell. Table 1 summarizes some statistics of the examples shown, together with the data of the other meshes extracted from some of the real objects used in the tests. It includes the number of points of the clouds describing the geometry of the objects, the number of vertices and facets of the final meshes and the execution time of each one of the most demanding stages of the whole process: the 3D tetrahedral Delaunay computation, and the final triangular facet extraction. Although there is some loss in the number of vertices with respect the number of input samples, the examples show that the shape, globally and locally, is cor-

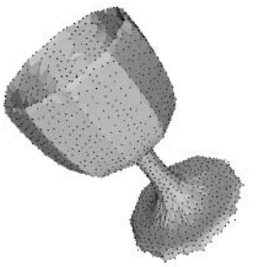
Table 1. Statistic data about some of the objects managed in the experiments

Object	No. of input points	No. of output vertices	No. of output facets	Exec. time(s): 3D Delaunay Tetrahedra	Exec. time(s): Triang. mesh Extraction	Total Execution Time(s)
Bowl	1093	685	1412	6.800	0.988	8.00
Cup	3051	2791	5630	14.773	1.087	16.391
Amphora	6287	6041	16880	45.757	4.781	51.48
Human skull	6391	5707	14158	63.220	3.570	68.172
Tip of an arrow	7609	5405	12342	85.305	5.838	92.561
Sea shell	7782	5379	10906	38.298	2.728	42.259
Hand	52704	47342	144166	1090.171	41.158	1132.884

(a) Input data of the goblet.

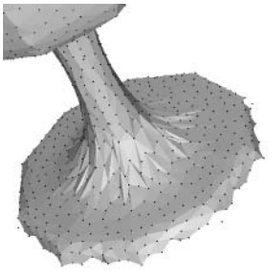


(b) Erosion with the convex hull criterion.



(c) First view of the final mesh.

(d) Second view of the final mesh.



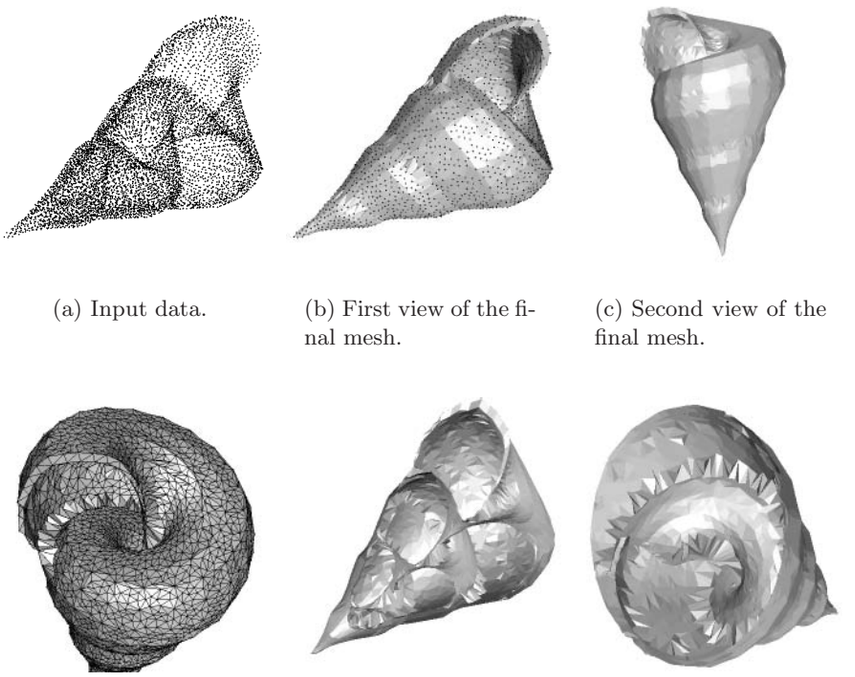
(e) Some detail of the base of the goblet.

(f) Slice view of the final mesh.

Fig. 1. The goblet. No. of points: 3051. No. of facets: 5630

rectly recovered. The most of the points removed are discarded in the Delaunay tetrahedralization step due to numerical errors.

The first object, Fig. 1, is a goblet that proceeds from [24]. The surface description of the goblet is a set of 3051 3D points. Figure 1(a) shows the input



(a) Input data.

(b) First view of the final mesh.

(c) Second view of the final mesh.

(d) Some detail of the output mesh.

(e) Interior view of the final mesh.

(f) Some detail in the inside of the output mesh.

Fig. 2. The sea shell. No. of points: 7782. No. of facets: 10906

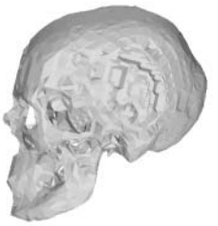
cloud of points that describes the geometry of the goblet. The mesh obtained after the first stage of the algorithm is shown on figure 1(b). This is the stage where the algorithm makes an erosion of the tetrahedra whose circumscribed spheres have centers outside of the convex hull. Figures 1(c)–1(f) show several views of the final mesh. Figure 1(c) overlaps the cloud of points with the output mesh, like Fig. 1(e), but in this one, we have performed a zoom over the base of the goblet to take a closer look at the extracted mesh and the input data. The black dots represent the 3D input points. The extracted mesh is a Delaunay triangulation of a subset of the input data. As it is shown in this and in the next figures, it can be considered that the corners, holes and cavities are satisfactory recovered.

The second object, Fig. 2, is a synthetic sea shell taken from the GTS Library [24]. Fig. 2(a) shows the input data; Fig. 2(b) shows the mesh achieved after applying the algorithm described in section 3 with the superposition of the

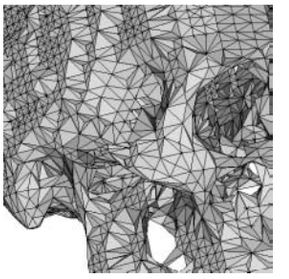
(a) Volumetric data of the human skull.

(b) Cloud of points extracted from the volumetric data of Fig. 3(a).

(c) First view of the human skull.



(d) Second view of the human skull.



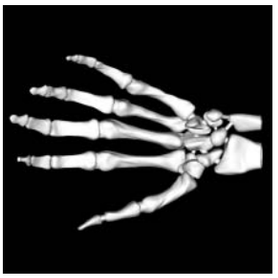
(e) A closer view showing some detail of the human skull.

Fig. 3. The human skull. No. of points: 6391. No. of facets: 14158

input cloud of points; Fig. 2(c) presents another view of the final mesh; we have applied a zoom process over the shell's opening in Fig. 2(d) for a better view of the details of the sea shell hole; in Fig. 2(e) we see a cut of the final mesh in order to see the inside and the cavities of the output mesh; finally, in Fig. 2(f), we present a zoom over the half-bottom interior of the sea shell.

The third object is a human skull (Figure 3). The main problems of this shape are the high complexity and the variable density of the cloud of points, presenting zones where there are very few points, for example in the eye holes, and others where the distribution is more uniform. Like in the previous figures, we present the input data, Fig. 3(b) and several views of the final mesh, Fig. 3(c)–3(e).

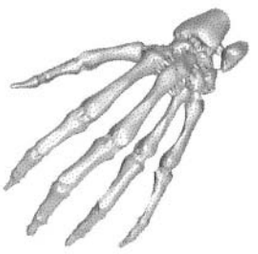
Figure 4 shows the surface extracted from a human hand. It presents some difficulties, like the inter-finger space or the finger tips, but in all the cases the surface recovered can be considered very close to the ideal.



(a) A photograph taken from the original human skeleton hand.

(b) Cloud of points extracted from the human skeleton hand.

(c) Erosion with the convex hull criterion.



(d) First view of the human hand.

(e) Second view of the human hand.

Fig. 4. The human hand. No. of points: 47342. No. of facets: 144166

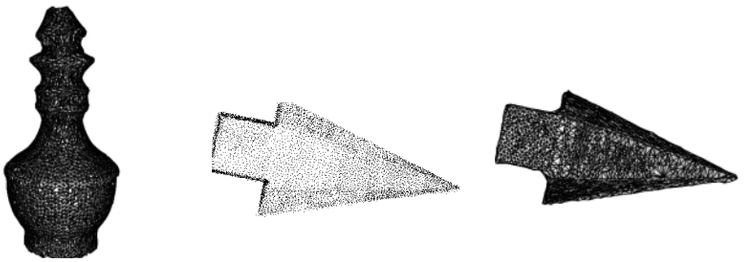
Last, in Fig. 5 we present the shape of the other objects mentioned in Table 1. In this case, the digitization of the bowl and the tip of an arrow has been carried out with a hand-held 3D digitizer. This device is a tactile low-price sensor manipulated by a human operator, which allows the simulation of any other kind of sensor that provides scarce and irregularly sampled measurements. It provides exclusively geometric information (only the points' coordinates are kept). The amphora has been digitized with a laser-based sensor, which provides a more regular sampling than the hand-held digitizer, but its prize is considerably higher too.

The current version of the implemented algorithm allows the specification of a threshold defined by the user. Sometimes, the geometry of the object does not allow to reach satisfactory results in a fully automatic way and the operator has the possibility to decide his preference: reducing the value of the threshold achieves a finer resolution mesh, but may produce some undesirable holes on the

(a) Input data from a bowl.

(b) Surface extracted from Fig. 5(a).

(c) Input data from the amphora.



(d) Surface extracted from Fig. 5(c).

(e) Input data from the tip of an arrow.

(f) Surface extracted from Fig. 5(e).

Fig. 5. Objects digitized with a laser-based sensor (amphora) and with the hand-held digitizer (bowl and tip of an arrow) and meshes obtained

final mesh. Usually, this defect is not present in the meshes generated with a more uniform distribution, but it may be noticeable if the density of points over the object's surface is very variable.

We must notice that all the examples shown have been refined by the user specifying a threshold very similar to that computed in the fully automatic process.

5 Conclusions and Future Work

It has been presented a $O(n^2)$ algorithm that allows the automatic extraction of 3D Delaunay triangulations from free genus real objects. The only input information about the geometry of the objects is a set of 3D points irregularly distributed over the whole objects' surfaces. We must remark that n is the number of tetrahedra achieved by the spatial partition algorithm stated by Faugueras [12], and n depends not only on the number of considered points, but also on their positions.

The use of a topological criteria is not enough to obtain a mesh that correctly fits the shape of the objects without a human interaction, and it has been necessary to include additional restrictions to the process in order to become determinist. In our case, the heuristic chosen has been the perimeter of the triangular facets, mainly, because of its simplicity and good results achieved over other measures. The algorithm has been tested successfully with numerous free shape real objects, as we have shown with some examples presented above.

Thinking in future works, it should be desirable an optimization of the algorithm to reduce the response time by means of a parallel implementation, taking advantage of the local nature of the operations [7].

Acknowledgments

This work has been partially funded by the Spanish Commission for Science and Technology (grants CICYT TIC98-0272-C02-01 and TIC99-0947-C02-01). The authors gratefully acknowledge the help provided by Jaime Gómez (3D amphora data).

References

1. Nina Amenta and Marshall Bern. Surface reconstruction by Voronoi filtering. In 14^{th} *ACM Symposium on Computational Geometry*, pages 39–48. ACM, June 1998. 273
2. D. Attali. τ-regular shape reconstruction from unorganized points. *Computational Geometry*, 10:239–247, 1998. 273
3. C. L. Bajaj, F. Bernardini, and G. Xu. Reconstructing surfaces and functions on surfaces from unorganized three-dimensional data. *Algorithmica*, 19:243–261, 1997. 273
4. Jean-Daniel Boissonnat and Frédéric Cazals. Smooth surface reconstruction via natural neighbour interpolation of distance functions. In *Proc.* 16^{th} *Annu. ACM Symposium on Computational Geometry*, 2000. 273
5. S. Campagna and H.-P. Seidel. Parameterizing meshes with arbitrary topology. In H.Niemann, H.-P. Seidel, and B. Girod, editors, *Image and Multidimensional Signal Processing'98*, pages 287–290, 1998. 273
6. P. Cignoni, C.Montani, R.Perego, and R.Scopigno. Parallel 3D Delaunay triangulation. *Computer Graphics Forum*, 12(3):129–142, 1993. 274
7. P. Cignoni, D. Laforenza, C. Montani, R. Perego, and R. Scopigno. Evaluation of parallelization strategies for an incremental Delaunay triangulator in E^3. *Concurrency: Practice and Experience*, 7(1):61–80, 1995. 282
8. T. K. Dey, K. Mehlhorn, and E. R. Ramos. Curve reconstruction: Connecting dots with good reason. In *Proc.* 15^th *Annu. ACM Sympos. Comptuational Geometry*, pages 197–206, 1999. 273
9. R. Dwyer. A faster divide and conquer algorithm for constructing Delaunay triangulations. *Algorithmica*, 2:137–151, 1989. 274
10. H. Edelsbrunner and E. P. Mücke. Three-dimensional alpha shapes. *ACM Trans. on Graphics*, 12(1):43–72, 1994. 273

11. T. Fang and L. Piegl. Delaunay triangulation using a uniform grid. *IEEE Computer Graphics and Applications*, pages 36–47, 1993. 274
12. Olivier Faugueras. *Three-Dimensional Computer Vision. A Geometric Viewpoint*, chapter 10 Interpolating and Approximating Three-Dimensional Data, pages 403–482. In Olivier Faugueras [13], 1993. 273, 274, 281
13. Olivier Faugueras. *Three-Dimensional Computer Vision. A Geometric Viewpoint*. The Massaachusetts Institute of Technology, 1993. 283
14. The Geometry Center. Geomview. University of Minnesota. www.geom.umn.edu/docs/software/download/geomview.html 276
15. Georgia Institute of Technology. Large Geometrical Models Archive http://http://www.cc.gatech.edu/projects/large_models 276
16. GNU. www.gnu.org 276
17. L. J. Guibas, D. E. Knuth, and M. Sharir. Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica*, 7:381–261, 1992. 274
18. A. Hilton and J. G. M. Gonçalves. 3D scene representation using a deformable surface. In IEEE Computer Society Press, editor, *Proc. on Physics Based Modeling Workshop in Computer Vision*, volume 1, pages 24–30, June 1995. 273
19. H. Hoppe, T. DeRose, T. Duchamp, J. McDonal, and W Stuetzle. Surface reconstruction from unorganized points. In *SIGGRAPH'92*, pages 71–78, July 1992. 273
20. Andrew E. Johnson and Martial Hebert. Control of polygonal mesh resolution for 3-D computer vision. *Graphical Models and Image Processing*, 60:261–285, 1998. 272
21. Leif Kobbelt, Jens Vorsatz, and Hans-Peter Seidel. Multiresolution hierarchies on unstructured triangle meshes. *Computational Geometry*, 14:5–24, November 1999. 273
22. C. Oblonsek and N. Guid. A fast surface-based procedure for object reconstruction from 3D scattered points. *Computer Vision and Image Understanding*, 2(69):185–195, 1998. 273
23. Michael Petrov, Andrey Talapov, Timothy Robertson, Alexeis Lebedev, Alexander Zhilayaev, and Leonid Polonsky. Optical 3D digitizers: Bringing life to the virtual world. *IEEE Computer Graphics and Applications*, 18(3):28–37, May 1998. 272
24. Stéphan Popinet. The GNU Triangulated Surface Library. http://gts.sourceforge.net 276, 277, 278
25. H. Shum, M. Hebert, K. Ikeuchi, and R. Reddy. An integral approach to free-form object modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(19):1366–1370, 1997. 273
26. Hiromi T. Tanaka. Accuracy-based sampling an reconstruction with adaptive meshes for parallel hierarchical triangulation. *Computer Vision and Image Understanding*, 61(3):335–350, May 1995. 273
27. D. Terzopoulos and D. Metaxas. Dynamic 3D models with local and global deformations: Deformable superquadrics. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(7):703–714, 1991. 273