Proceedings of the IMProVe 2011
International conference on Innovative Methods in Product Design
June 15th – 17th, 2011, Venice, Italy

A fast algorithm for manifold reconstruction of surfaces

L. Di Angelo [(a)], L. Giaccari [(a)]

[(a)] Department of Industrial Engineering, University of L'Aquila, Italy

Article Information

Keywords:
Surface reconstruction triangular meshes, reverse engineering.

Corresponding author:
Luca Di Angelo
Tel: 00390862434310
e-mail: luca.diangelo @univaq.it
Address: Via G. Gronchi 18, L'Aquila, Italy

Abstract

Purpose:
In a previous paper these authors presented a new mesh-growing approach based on the Gabriel 2 – Simplex (G2S) criterion. If compared with the Cocone family and the Ball Pivoting methods, G2S demonstrated to be competitive in terms of tessellation rate, quality of the generated triangles and defectiveness produced when the surface to be reconstructed was locally flat. *Nonetheless, its major limitation was that, in the presence of a mesh which was locally non – flat or which was not sufficiently sampled, the method was less robust and holes and non – manifold vertices were generated. In order to overcome these limitations, in this paper, the performance of the G2S mesh-growing method is fully improved in terms of robustness.*

Method:
For this purpose, an original priority queue for the driving of the front growth and a post processing to efficiently erase the non–manifold vertices are proposed.

Result:
The performance of the new version of the *G2S approach has been compared with that of the* old one*, and that of the* Cocone *family and the* Ball Pivoting *methods in the tessellation of some benchmark point clouds and artificially noised test cases. The results derived from these experiments show that the improvements being proposed and implemented prevent the generation of non–manifold vertices and render the new version more robust than* the old one*. This performance improvement is achieved by a small reduction of the tessellation rate as opposed to the old version; the rate is still, however, at least an order of magnitude higher than the other methods here considered (the Cocone family and the Ball Pivoting methods).*

Discussion & Conclusion:
The results obtained show that the use of the new version of G2S is advantageous, as opposed to the other methods here considered, even in the case of noised point clouds. In fact, since it does not perform the smoothing of points, not even in the presence of very noised meshes, the new version of G2S, while producing more holes than the Robust Cocone and the Ball Pivoting, nonetheless manages to preserve the manifoldness and important details of the object.

1 Introduction

The process of constructing a triangular mesh from a point cloud which has been acquired from a 3D real object is known as surface tessellation or reconstruction. The applications of triangular meshing of 3D point clouds are wide–ranging and include Reverse Engineering, Collaborative Design, Inspection, Computer Vision, Dissemination of Museum artefacts, Medicine, Special Effects, Games and Virtual Worlds. In many of the previously mentioned applications, robustness, reliability and triangulation speed are important factors that are required in any tessellation method.

For the last few years the 3D scanning system has been offering high resolutions with a measuring accuracy as high as 10 μm, which, on the one hand, make it possible to capture the smallest surface features but, on the other hand, generate very large data sets. When analysing the related literature, it is evident that faster, more reliable and robust methods have a triangulation speed below 50K triangles per second with the typical computing power. This speed may not be enough in the tessellation of a cloud with over one million points. Furthermore, by analysing the peak memory usage of these algorithms, it is clear that most of them cannot be run on personal computers.

These authors have recently proposed a new mesh-growing method based on the *Gabriel 2 – Simplex* (*G2S*) criterion. The results obtained are very promising since they demonstrate that this method makes it possible to tessellate quickly clouds with over one million points even by using a laptop. Its major limitation is however that, in the presence of a mesh which is locally non–flat or which is not sufficiently sampled, the method is less robust and therefore holes and non–manifold vertices are generated. We should bear in mind the fact that the mesh may be unusable without the deletion of this kind of vertices, which has to be carried out without compromising the areas which have been correctly reconstructed.

In order to overcome these limitations, and as it is going to be shown in this paper, the performance of the *G2S* mesh-growing method is fully improved in terms of robustness. To this end, an original *priority queue* for the driving of the front growth and a post-processing to efficiently erase the non–manifold vertices are proposed.

The improved method has been tested for the tessellation of some benchmark point clouds and artificially noised test cases. The results derived from these experiments are critically discussed hereinafter.

2 Related works

It is on account of the importance of the problem referred to that numerous algorithms to tessellate point clouds are presented in literature. The more recent exhaustive overviews of 3D surface reconstruction methods are presented in [1] and [2]. Surface reconstruction algorithms are generally divided into two categories: implicit and explicit.

2.1 Implicit methods

Implicit methods attempt to reconstruct an implicit function f(**p**)=0, where **p** is either the whole point cloud or only a part of it. The final triangulation is obtained by extracting the isosurface for the f(**p**)=0. The most common methods define the implicit function as:

- the sum of radial basis functions (RBF) centred at the points [3], [4];
- a set of constraints that force the function to assume given values at the sample points and also force its upward gradient to match the assigned normals at the sample points (Moving Least Squares) [5], [6];
- a Poisson problem [7].

These implicit methods carry out a watertight surface reconstruction also in the case of sparse and noised data. However, these methods may generally require many computations and, sometimes, they may even need the surface normal at each data point. Moreover, the triangles of the final surface may not pass through all the points, and this in turn may imply the loss of some details of the shape of the original model. The more points we use to compute the implicit function, the higher will be the fitting accuracy and the longer will be the computation time.

2.2 Explicit methods

Explicit methods attempt to triangulate the points directly. In contrast with implicit methods, they are less robust against noise but they are generally faster. The most common explicit methods can be classified into two main groups: Voronoi/Delaunay-based and mesh growing approaches.

2.2.1 Voronoi/Delaunay-based methods

The first group includes algorithms that compute a volume tetrahedralisation by a 3D Delaunay triangulation of the sample points. The most important methods presented in the related literature ([8], [9], [10], [11], [12], [13] and [14]) essentially differ in the way of removing the tetrahedra and building the external triangular mesh. These approaches often provide theoretical guarantees of a good reconstruction as the sampling density increases. Gopi and al. in [15] proposed a different approach which, for each sample point, provides the projection of the neighbouring points onto the approximating tangent plane and the tessellation of the projections by means of a 2-D Delaunay triangulation. The 2D edges obtained are then applied to 3D space.

2.2.2 Mesh growing – based methods

The mesh-growing approaches generate the tessellated surface from a seed triangle and grow the meshed area pushing the fronts ahead by using some criteria. For the last few years a number of algorithms have been presented. Thus Bernardini et al. [17] introduced the *Ball Pivoting* algorithm, whose front grows as a ball of user-defined radius pivots around the front edge. When the ball touches three points a new triangle is formed. Generally speaking, this method affords a correct triangulation for a uniform data point, which is also noised, but which does not have concave sharp features. Huang and Menq in [18] proposed an algorithm which, for each front edge, projects the k nearest points of two endpoints onto the plane that is defined by the triangle adjacent to the front edge. Any point generating triangles whose edges intersect edges of already existing triangles is discarded. Among the points which are retained, the point showing the minimum sum of distances from the front edge endpoints is chosen. Nonetheless, and as pointed out by Lin et al. in [19], this algorithm presents some shortcomings. In order to overcome them, Lin et al. in [19] introduced the *Intrinsic Property Driven* (IPD) algorithm, which improves the way of searching for the point to be triangulated. As stated by Chang at al. in [2], all the methods based on mesh growing approaches are fast, efficient and simple to implement but they, however, fall short whenever two surfaces are either close together or near sharp features. More recently, Li et al. in [1] proposed a method based on a *Priority Driven* approach that evaluates shape changes from an estimation of the original surface that is made at the front of the mesh-growing area. The experimental results in [1] evidenced that the triangulation speed of this method is higher than that of *Ball Pivoting* and *Cocone*. However, no reckoning is made of the defectiveness it generates. Finally, in [20] these authors put forth a new mesh growing approach based on the *Gabriel 2 – Simplex* (*G2S*) criterion: *A triangle is a G2S if its smallest circumscribing ball is empty*.

3 The Gabriel 2 – simplex criterion based method

The algorithm presented in [20] can be summarised in the following steps:

- Import of the point cloud;
- Building of a specific data structure to speed up the nearest point search;
- Seed triangle search;
- G2S criterion based triangulation.

The first step entails importing the point cloud, which pertains to a continuous surface, in the form of the coordinates x, y, z. Each point of the point cloud is kept in a hash table data structure for both point indexing and nearest neighbour searching. In the proposed algorithm the data structure used is an improvement of those proposed by Hoppe et al. in [21] and Turk et al. in [22].

3.1 Selection of the seed triangle

In order to select an appropriate seed triangle, a new method is employed. First of all, it makes a random choice of a point from the cloud. Then, its nearest neighbour point is searched for, and an edge is formed between these two points. A range search is performed inside a sphere which is centred at the midpoint of the edge and whose radius is k – times the length of the edge. For every point in the range, the method manages to build a triangle that is formed by the edge and the concerning point. That triangle is selected as seed if it is *G2S* and if the points contained in an infinite cylinder passing through the triangle vertices and having its axis parallel to the normal of the triangle are either all above or all under the triangle. This procedure is repeated until a triangle

satisfying the two previously mentioned conditions is found.

3.2 G2S criterion based triangulation

The edges of the seed triangle constitute the initial advancing front of the growing-mesh method. For each *free edge* ($\mathbf{e_f}$) (which are edges pertaining to only one triangle) of the growing front a triangle is generated according to the following procedure. Firstly, the candidate points near $\mathbf{e_f}$ are identified as those inside a sphere having its centre on the axis of the *free edge* which lie on the plane of the front triangle, in the growing direction, and having a radius equal to the *free edge's* length (figure 1). If no points are found inside the *search region*, the *free edge* is classified as *boundary edge*. When more than one point is inside the search region ($\mathbf{cp}_1$, $\mathbf{cp}_2$ and $\mathbf{cp}_3$ in figure 2*a)*), a point is randomly chosen ($\mathbf{cp}_2$ in figure 2*b)*) and the smallest sphere circumscribing that point and the front edge's points is constructed (the *first search sphere* in figure 2b). Then, if this sphere is not empty ($\mathbf{cp}_1$ and $\mathbf{cp}_3$ in figure 2c), another point inside the region is picked ($\mathbf{cp}_3$) and the procedure is repeated (the *second search sphere* in figure 2*c)*). The process ends when the sphere passing through the chosen *reference point* and the extremes of $\mathbf{e_f}$ is empty. The triangle defined by the front edge and the *reference point* is a *candidate triangle* which needs to be verified in the next step (figure 2*d)*).

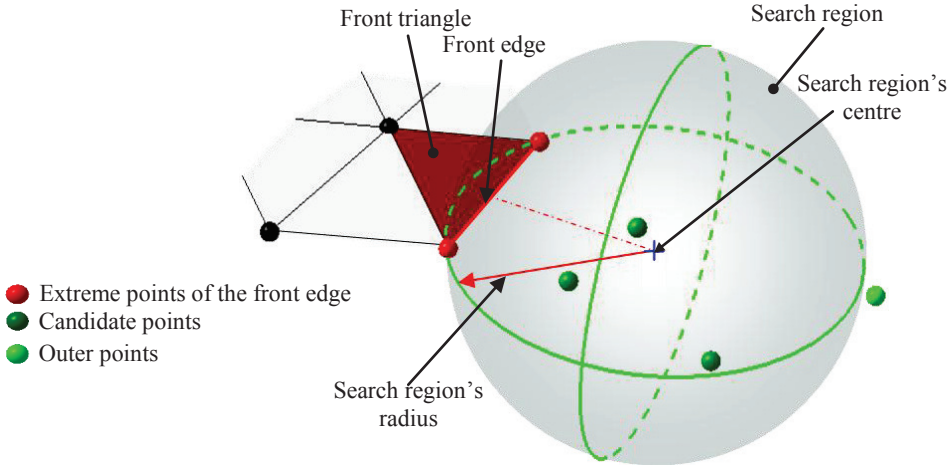

Fig. 1 The search region's definition terms

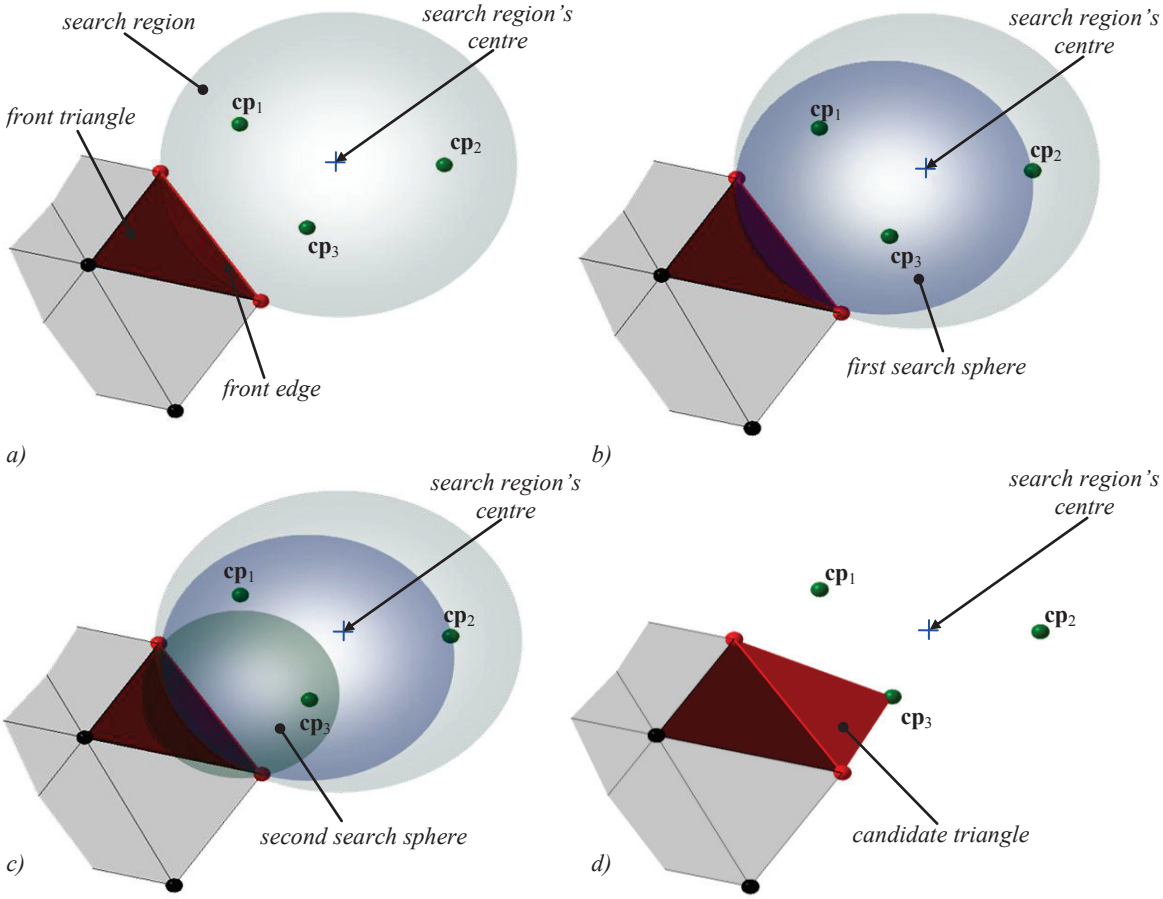

Fig. 2 Explanation of the strategy for the selection of a reference point with a view perpendicular to the front triangle

In order to speed up the triangulation process, if only one point is found inside the search region, that point identifies a candidate triangle with the *free edge* ($\mathbf{e_f}$) without verifying whether or not it is a *G2S*. If no point is found inside the search region, $\mathbf{e_f}$ is removed from the front queue. Each of the new triangles retained is formed by the front edge ($\mathbf{e_f}$) and two further edges ($\mathbf{e_1}$ and $\mathbf{e_2}$). In order to check efficiently whether these two edges ($\mathbf{e_1}$ and $\mathbf{e_2}$) are really new or they already belong to other triangles, a data structure called *Point Edge Map* (*PEM*) is proposed which relates every point to its edges. For either edge ($\mathbf{e_1}$ and $\mathbf{e_2}$), the following conditions should be verified:

- If the edge already pertains to another triangle, the consistency of the orientation of the new triangle with the triangle sharing the edge must be verified.
- If this edge is new, it is added to the Point Edge Map and to the front queue and $\mathbf{e_f}$ is removed from the front queue.

The procedure ends when the free edges' queue is empty.

The theoretical basis of *G2S* is taken for granted by accepting the fact that point clouds can be considered locally flat, or, in other words, that the surface to be reconstructed is locally oriented, smooth, manifold, well sampled and not self-intersecting. Under this hypothesis, the *G2S* criterion works like a 2D Delaunay tessellation through which surface reconstruction is guaranteed. These requirements are not so restrictive anymore, especially since the advent of high-resolution non–contact scanners which produce noise-free points clouds. More generally, as pointed out by Dyer et al. in [23], a *Gabriel mesh* (a mesh for which each triangle verifies the *G2S* criterion) is a *Delaunay mesh*. In [20] these authors already demonstrated it by analysing the typical benchmarks presented in the related literature:

- the triangulation speed of G2S is comparable with a traditional 2D Delaunay-based mesher and it is at least an order of magnitude higher than the other methods here considered;
- G2S produces triangles whose quality is similar to that of those triangles obtained by the Cocone methods and slightly better than the quality of the triangles obtained by the Ball Pivoting one;
- G2S can reproduce even the smallest details of well sampled surfaces, similarly to Cocone methods, also in concave areas of strongly non–uniform point clouds where the Ball Pivoting method shows some problems;
- G2S does not produce non–manifold edges, self intersecting triangles or slivers;
- as regards non–manifold vertices, holes and boundary edges, the quantity and the extension of defectiveness generated by the G2S tessellation are on average similar to those produced by the Cocone and the Tight Cocone;
- in the presence of a mesh which is locally non–flat or which is not sufficiently sampled, G2S is less robust and holes and non–manifold vertices are generated.

4 Critical aspects in the G2S method and improvements

As mentioned in the previous section, the *G2S* version proposed in [20] presents some critical aspects. In particular, in any area of a point cloud that is not locally flat or is not sufficiently sampled, *G2S* can generate:

- holes, which identify unmeshed area;
- non–manifold vertices, which are vertices for which the incident triangles form more than one fan (see figure 3);
- a twisting of the surface.

This paper focuses on the improvement of the *G2S* performance as regards the generation of *non–manifold vertices* and *twisted surfaces*.

4.1 Non – manifold vertices

In figure 3, the *non–manifold vertex* (**v**) is classified as *type I* if at least one fan is complete (figure 3a), and *type II* otherwise (figure 3b). In what follows, the triangles with at least one *boundary edge* are referred to as *boundary triangles*. In order to automatically remove the triangles (depicted in red in the figure) which make the vertex non–manifold, in this paper a post–processing approach is proposed. This approach is based on a data structure that takes advantage of the fact that adjacent triangles have a congruent orientation and consists of:

- the dynamic_edge_queue that initially has n_e rows (n_e is the number of edges that are not boundary) and six columns: edge_label (e_l), first_point (p_f), last_point (p_l), first_triangle (t_f) and last_triangle (t_l);
- the matrix point_to_triangles constituted by n_v rows (n_v is the number of vertices) and four columns: vertex_label (v_l), first_triangle_of_loop (*front*), last_triangle_of_loop (*back*) and $n_{t,a}$ (which is the number of triangles added in the loop).

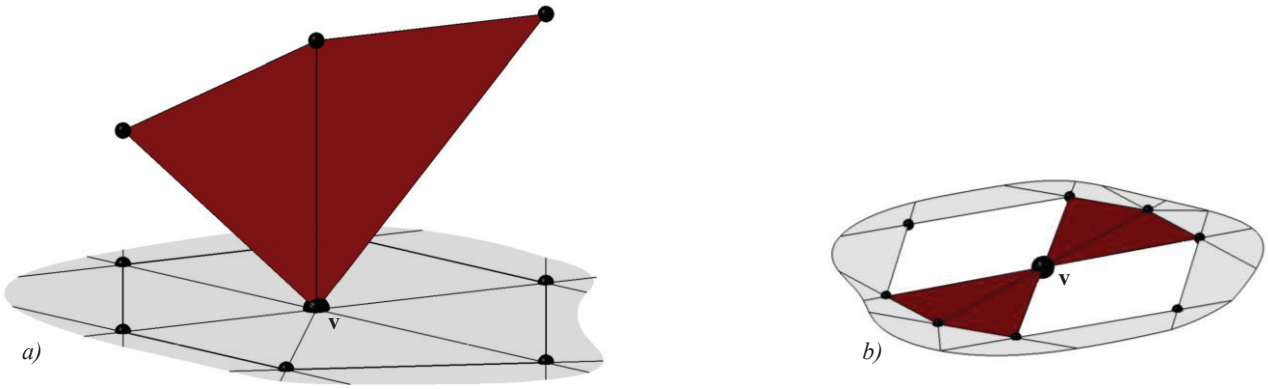

Fig. 3 Types of non – manifold vertices

In order to explain the method here presented, let us consider the mesh depicted in figure 4a with the labels of the vertices, edges and triangles superimposed. First, the *dynamic_edge_queue* is filled with the edges of the mesh, except for the boundary ones (figure 4b) and in the *point_to_triangles* table, the labels of all vertices are added to the first column (figure 4c). The process starts by popping the first element off the queue ($\mathbf{e_3}$) and the

corresponding labels of t_f ($\mathbf{t}_1$) and t_l ($\mathbf{t}_2$) are added to the related lines of the matrix (figure 4d). Then, the first element of the new queue (figure 4e) is popped off ($\mathbf{e}_6$). Since the triangles' labels associated with $\mathbf{v}_1$ for $\mathbf{e}_6$ ($\mathbf{t}_3$ and $\mathbf{t}_8$) are different from those reported in the corresponding row of the matrix, this edge is pushed to the front of the queue (figure 4f). If once the queue has been scanned through, no edges of intersection of one of two extreme triangles of the loop ($\mathbf{t}_1$ and $\mathbf{t}_2$) have been found for the vertex under examination ($\mathbf{v}_1$), the loop is defined as *open* since $t_f \neq t_l$. In the case that there are other edges incident to $\mathbf{v}_1$ in the queue, $\mathbf{v}_1$ is non–manifold of type II and the triangles of the loop are erased (figure 4g). Again, the first element of the queue is popped off ($\mathbf{e}_8$); the triangles $\mathbf{t}_1$ and $\mathbf{t}_6$ are added to the corresponding rows of the matrix of the vertices $\mathbf{v}_1$ and $\mathbf{v}_6$ (figure 4h). Once more, the first element of the new queue (figure 4i) is popped off ($\mathbf{e}_9$). Since one of the two triangles incident to the edge ($\mathbf{t}_4$) is a terminal point (front) of the loop, in the corresponding row of *point_to_triangles*, this triangle is substituted with the other ($\mathbf{t}_3$) (figure 4l). This procedure is iterated until the triangles of the *front* and the *back* column are the same for a vertex (figure 4m), or, in other words, the loop is *closed*; in the case that in the *dynamic_edge_queue* there are edges incident to that vertex (figure 4n), the vertex **v** is non–manifold of type II; these edges and the corresponding triangles are erased (figure 4o). The procedure ends when the *dynamic_edge_queue* is empty.

4.2 The twisting of the surface

In this paper, the *twisting of the surface* identifies the generation on the same body of different tessellated surfaces not having congruent normals (figure 5). This in turn generates holes with extended boundary edges since adjacent patches not having a congruent orientation cannot be merged. In order to solve this problem, an original *priority queue* is proposed. The main idea at the basis of the priority approach being presented is to mesh first those areas for which the front grows in the flattest way in the neighbourhood.

Since a *priority queue* based on a continuous *priority value* can really slow down the algorithm, a set of discrete priority values is adopted. The strategy used involves the definition of *n* priority levels for the search region radius (the smallest radius having priority 1 and the greatest one having priority *n*) and *m* priority levels for the flatness, measured as the angle (β) between the normal of the *front triangle* and the *candidate* triangle (the highest priority being assigned to β=0° and the lowest to β=180°). Since experience shows that the radius of the search region mainly affects a good reconstruction, the *priority value* (*PV*) is defined according to the following expression:

$$PV = m \cdot (pl_r - 1) + pl_f \qquad (1)$$

where pl_r is the priority level of the search region radius (from 1 to *n*) and pl_f is the priority level of the flatness (from 1 to *m*). At the beginning all the edges in the *free queue* start with *PV* = 1 (maximum value of the priority). Then, these edges are positioned in the queue by sorting in ascending order the value of *PV* calculated for the corresponding *candidate triangle*.

5 Experimental Results

The methodology described in the previous sections has been implemented in an original software, coded in C++, by using a library dedicated to the processing of tessellated geometric models, which has been developed at the University of L'Aquila. The method herein proposed has been tested for the tessellation of several scanned point clouds characterised by some critical geometrical features which add to the difficulty in their tessellation. Other characteristic aspects of the analysed point clouds are the number and spatial density distribution of points as well as noise level. Most of the test cases used are typical benchmarks taken from the related literature. The tests have been run on a laptop with 2.4 GHz Intel Core Duo 7700 Processor and 2 GB RAM.

In this paper, the performance of the new version of *G2S* (henceforth, *G2S_new*) has been compared with that of the old one [20] (henceforth *G2S_old*), that of the *Cocone* methods (*Cocone* [9], *Tight Cocone* [10] and *Robust Cocone* [11], whose .exe files were kindly provided by the authors) and with that of the *Ball Pivoting* [17] (whose implementation is based on the *vcg library*: http://vcg/sourceforge.net) in terms of tessellation rate (expressed as thousands of triangles created per second [kΔ/s]) and defectiveness generated. The times have been measured since the import of the point cloud up to the .stl file generation. In what follows, n_{nmv}, n_{holes} and n_{be} are, respectively, the number of non–manifold vertices, holes and boundary edges. The above-mentioned methods are verified in the tessellation of 12 point clouds of closed surfaces and 5 point clouds of open surfaces acquired with different sampling rates from objects having different geometries. Some of these point clouds are very large data sets (*Turbine Blade*, *Nicolò da Uzzano, Neptune* and *Asian Dragon*). As far as the *Cocone* methods are concerned, the point clouds of closed surfaces are tessellated with the *Tight Cocone* [10], whereas those of open surfaces are reconstructed with the simple *Cocone* [9]. Some of the resulting renderings of the surfaces tessellated by the *G2S_new* method are shown in figure 6. The testing results are reported in tables 1 and 2, where NV is the number of points in the cloud and NT is the number of triangles generated.

By analysing the results obtained it is evident that the algorithm here proposed and implemented correctly erases all the non–manifold vertices. Furthermore, when using the priority queue in the new version of G2S, in most cases there is a reduction of holes and boundary edges. This performance improvement is achieved by a small reduction of the tessellation rate, which is still, however, at least an order of magnitude higher than the other methods considered. In some cases, such as the *raptor,* the marked reduction in boundary edges is due to the elimination of the problem of twisting surface generation. Figure 7 shows the renderings of the tessellation obtained for the *Raptor with both the G2S_old (a) and G2S_new*. In the same figure, the outside of triangles is coloured blue whereas the inside is coloured yellow.

In order to verify the performance of the *G2S_new* in the tessellation of noised point cloud data, specific experiments are carried out. The performance of the new version of *G2S* is compared with that of the old version [20], that of the Robust Cocone method [11] and that of the Ball Pivoting one [17].

The first experiment aims at comparing the four methods in the tessellation of the *Stanford Bunny* with different levels of noise added. Noise is randomly generated according to a Gaussian probability density distribution with different values of standard deviation (σ). Figure 8 illustrates the renderings of the results obtained, whereas table 3 shows a comparison between their defectiveness. It is evident that *G2S_new* is more robust than *G2S_old* in the presence of noised point clouds. Since it does not perform the smoothing of any points, even the new

version of *G2S* produces more defectiveness than the Robust Cocone and the Ball Pivoting for σ > 0.00025. However, as highlighted in figure 8, the methods which seem to be less sensitive to noise do not preserve important details of the object.

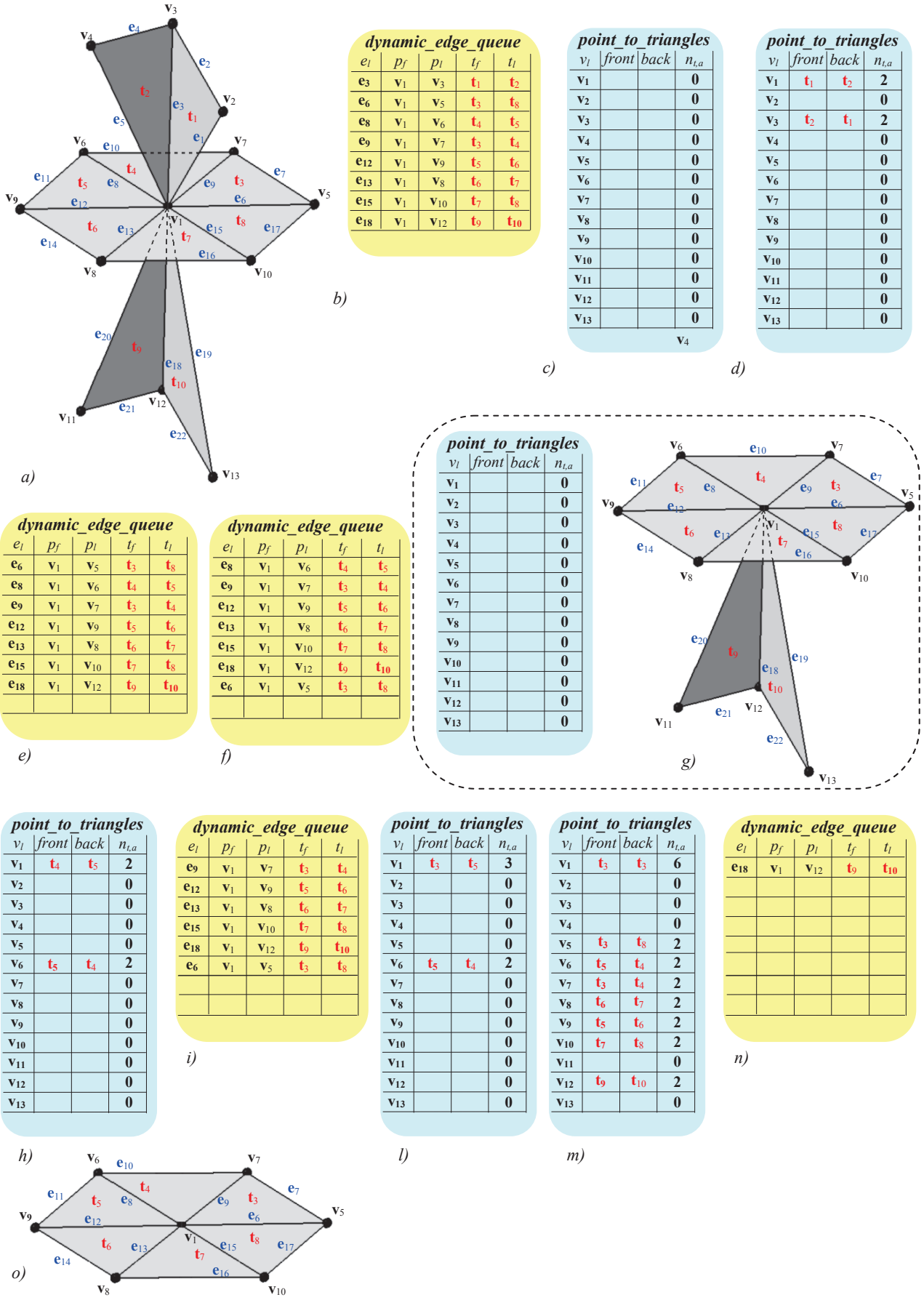

Fig. 4 Explanation of the post processing used to erase the non – manifold vertices

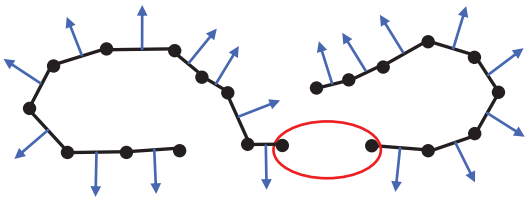

Fig. 5 The twisting of the surface

The second experiment is carried out in order to compare the four methods in the tessellation of the Stanford Bunny with different levels of outliers being added (5%, 10% and 20% of the total number of points). By analysing the resulting renderings (depicted in figure 9) and the generated defectiveness (reported in table 4), the Robust Cocone seems to be inadequate to tessellate point clouds with this type of noise. Neither does the new version of G2S generate any non – manifold vertices in this case. With a low percentage, up to 5%, the tessellation obtained by means of the *G2S_new* is not significantly affected by the presence of outliers. For greater percentages, the defectiveness generated in terms of holes and boundary edges are comparable with those produced by the Ball Pivoting method.

6 Conclusion

In a previous paper [20] these authors had already presented a new-mesh growing approach based on the Gabriel 2 – Simplex (G2S) criterion. The results obtained proved that the G2S is competitive in terms of tessellation rate, quality of the generated triangles and low defectiveness, especially when compared with the *Cocone* family and the *Ball Pivoting* methods. Its major limitation was that, in the presence of a mesh which was locally non – flat or was not sufficiently sampled, it proved to be less robust and holes and non – manifold vertices were generated.

In order to improve the robustness of the *G2S* mesh-growing method, this paper proposes an original *priority queue* for the driving of the front growth and a post processing to efficiently erase the non – manifold vertices. The performance of *G2S_new* has been compared with that of *G2S_old*, and that of the *Cocone* family and the *Ball Pivoting* methods in the tessellation of some benchmark point clouds and artificially noised test cases. The results derived from these experiments show that the improvements proposed and implemented prevent the generation of non – manifold vertices and make the *G2S_new* more robust than *G2S_old* in terms of generation of defects such as holes and boundary edges, also in presence of noised point clouds. This performance improvement is achieved by a small reduction of the tessellation rate which is still, however, at least an order of magnitude higher than in the other methods here considered. In the case of much noised meshes, *G2S_new* produces more holes and boundary edges than the *Robust Cocone* and the *Ball Pivoting* methods, but the last named ones do not preserve important details of the object. Finally, in the presence of meshes with outliers, the number of holes and boundary edges produced by *G2S_new* can be said to be comparable with those produced by the *Ball Pivoting* method.

References

[1] Li X., Han C.Y., Wee W. G., 2009. On surface reconstruction: A priority driven approach. Computer-Aided Design, 41 (9), 626-640.

[2] Chang M. C., Leymarie F. F., Kimia B. B., 2009. Surface reconstruction from point clouds by transforming the medial scaffold. Computer Vision and Image Understanding, 113 (11), 1130 – 1146.

[3] Carr J., Beatson R., Cherrie H., Mitchel T., Fright W., Mccallum B., Evans T.: Reconstruction and representation of 3D objects with radial basis functions. SIGGRAPH (2001), 67–76.

[4] Turk G., O'Brien J.: Modelling with implicit surfaces that interpolate. In TOG (2002), 855–873.

[5] Tamal K. Dey , Jian Sun, An adaptive MLS surface for reconstruction with guarantees, Proceedings of the third Eurographics symposium on Geometry processing, July 04-06, 2005, Vienna, Austria .

[6] Kolluri R. Provably good moving least squares. ACM Transactions on Algorithms (TALG), 4 (2).

[7] Kazhdan, M., Bolitho, M., and Hoppe, H. 2006. Poisson surface reconstruction. In Symposium on Geometry Processing, 61–70.

[8] Amenta N., Bern M., Kamvysselis M., 1998. A new Voronoi-Based Surface Reconstruction Algorithm. In the Proceeding of Computer Graphics (SIGGRAPH '98), 415 – 421.

[9] Amenta N., Choi S., Dey T. K., Leekha N., 2002. A simple algorithm for homeomorphic surface reconstruction. International Journal of Computational Geometry & Applications, 12 (1 & 2), 125 – 141.

[10] Dey T.K. and Goswami S., 2003. Tight cocone: A watertight surface reconstructor. Journal of Computing and Information Science in Engineering, 3 (4), 302–307.

[11] Dey T.K. and Goswami S., 2006. Provable surface reconstruction from noisy samples. Computational Geometry, 35 (1 – 2), 124 –141.

[12] Dey T.K., Giesen J., Hudson J., 2001. Delaunay based shape reconstruction from large data. In the Proceeding of the IEEE Symposium on Parallel and Large-Data Visualization and Graphics, 19–27.

[13] Amenda, N., Choi, S., and Kolluri, R., 2001. The Power Crust, In the proceeding of the ACM Symposium on Solid Modeling and Applications, 249-260.

[14] Cohen-Steiner D., Da F., 2004. A greedy Delaunay-based surface reconstruction algorithm. The Visual computer, 20 (1), 4-16.

[15] Gopi M, Krishnan S, Silva C., 2000. Surface reconstruction using lower dimensional localized delaunay triangulation. In the Proceeding of Eurographics, 19 (3), 467 - 478.

[16] Cazals F., Giesen J., 2006. Delaunay triangulation based surface reconstruction. In Effective Computational Geometry for Curves and Surfaces, Boissonnat J., Teillaud M., (Eds.). Springer- Verlag, Math. and Visualization, 231–276.

[17] Bernardini F, Mittleman J, Rushmeier H, Silva C, Taubin G., 1999. The ball-pivoting algorithm for surface reconstruction. IEEE Transactions on Visualization and Computer Graphics, 5(4), 349-59.

[18] Huang J, Menq C. H., 2002. Combinatorial manifold mesh reconstruction and optimization from unorganized points with arbitrary topology. Computer – Aided Design, 34 (2), 149–65.

[19] Lin H. W., Tai C. L., Wang G.-J., 2004. A mesh reconstruction algorithm driven by an intrinsic property of point cloud. Computer-Aided Design, 36 (1), 1–9.

[20] Di Angelo L., Di Stefano P., Giaccari L., "A new mesh-growing algorithm for fast surface reconstruction". Computer – Aided Design, vol. 43 (6), 2011, p. 639-650, ISSN: 0010-4485.
[21] Hoppe H., Derose T., Duchamp T., McDonald J., Stuetzle W., 1992. Surface reconstruction from unorganized point clouds. In ACM SIGGRAPH, 71-78.
[22] Turk G., Levoy M., 1994. Zippered polygon meshes from range images. In ACM SIGGRAPH, 311-318.
[23] Dyer, R., Zhang, H., Möller, T., 2008. Observations on Gabriel meshes and Delaunay edge flips. Tech. Rep. TR 2008-22, Simon Fraser University. SFU-CMPT.

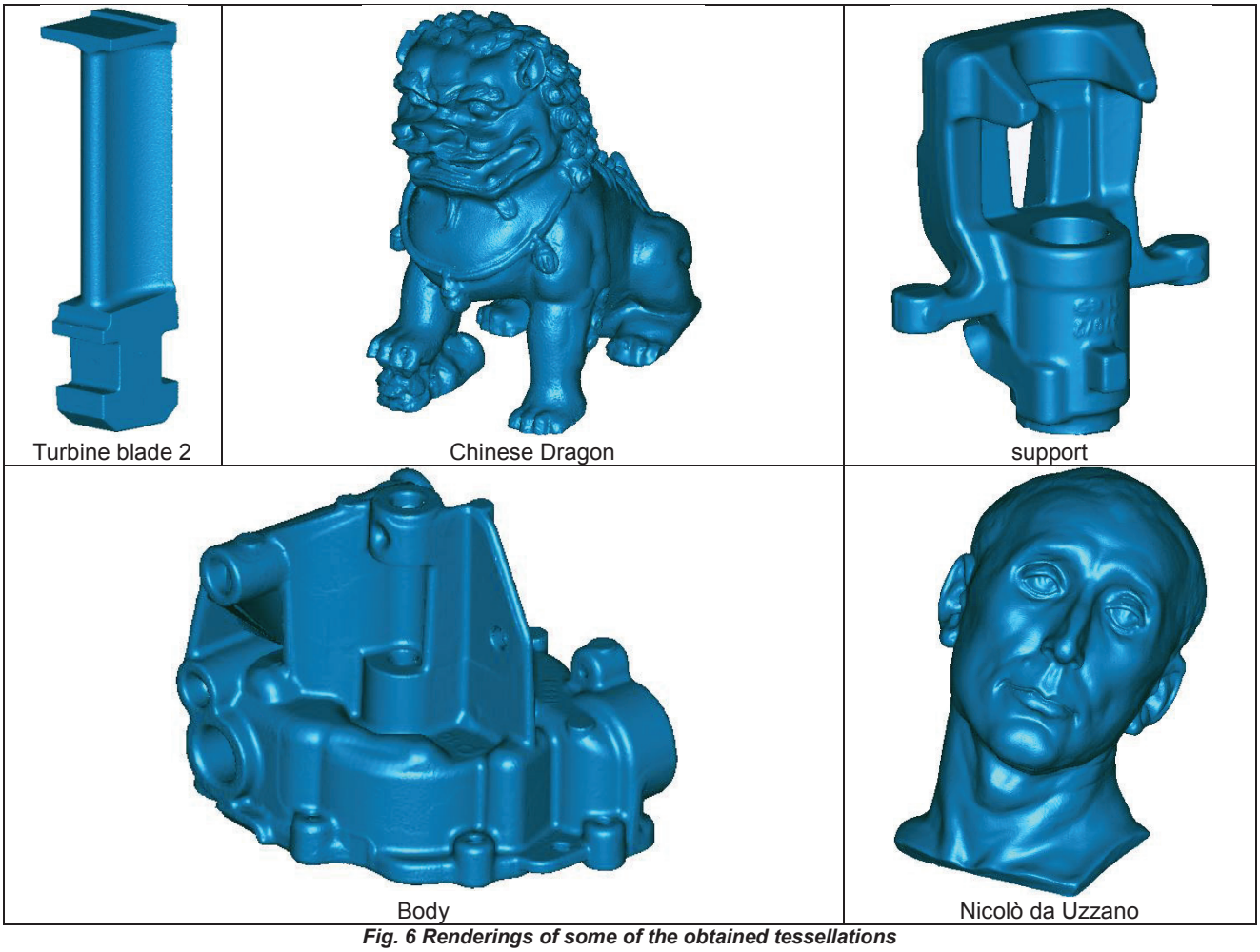

Fig. 6 Renderings of some of the obtained tessellations

Data point		New G2S Method		Old G2S Method [20]		Cocone methods [9], [10] (▫)		Ball Pivoting method [17]	
Model name	**NV**	**NT**	**[kΔ/s]**	**NT**	**[kΔ/s]**	**NT**	**[kΔ/s]**	**NT**	**[kΔ/s]**
Pulley (**)	293,672	587,266	328.5	587,181	371.8	587,312	0.67	571,738	52.82
Turbine Blade 2 (*)	396,104	791,916	288.5	792,041	377.3	791,873	1.72	736,685	43.69
Dragon (***)	435,545	834,771	304.5	805,376	348.1	867,282	0.62	782,185	35.46
Bimba (**)	502,694	1,005,246	366.2	1,005,172	432.5	1,005,088	0.82	953,618	23.82
Happy Buddha (***)	543,652	1,038,953	338.0	1,004,540	351.0	1,081,232	0.51	809,539	25.36
Support (*)	549,007	1,096,742	322.5	1,097,412	397.6	1,097,538	1.82	1,074,677	49.65
Rolling Stage (**)	596,903	1,190,806	319.7	1,193,303	373.5	1,193,688	1.49	1,168,744	57.07
Chinese Dragon (**)	655,980	1,311,307	322.0	1,311,296	475.2	1,310,435	0.99	966,266	25.28
Body (*)	675,049	1,349,076	299.7	1,349,609	279.7	1,344,039	1.2	1,326,963	59.47
Turbine Blade (***)	882,954	1,740,362	351.9	1,759,357	364.4	1,759,514	1.11	1,630,254	47.28
Nicolò da Uzzano (**)	946,760	1,891,949	367.0	1,891,992	464.5	1,891,669	1.93	1,795,917	40.33
Raptor ()	1,000,080	1,685,915	349.6	1,716,226	439.6	--	--	1,378,599	43.48
Neptune (**)	2,003,933	4,007,522	261.8	4,007,628	362.8	--	--	3,119,149	20.01
Asian Dragon (*)	3,609,601	7,217,980	362.9	7,218,442	418.8	--	--	6,715,376	26.22

(▫) The Cocone method [9] is used for point cloud of open surfaces whereas the Tight Cocone method [10] for point cloud of closed ones.
(*) http://www.scansystems.it
(**) http://shapes.aimatshape.net/
(***) http://www.lodbook.com/models/

Tab. 1 Comparison between the performance of the two versions of the G2S and that of the Cocone methods ([9], [10]) and the Ball Pivoting [17].

Model name	Defectiveness generated											
	New G2S Method			Old G2S Method [20]			Cocone methods [9], [10] (▫)			Ball Pivoting method [17]		
		holes			holes			holes			holes	
	n_{nm}	n_{holes}	n_{be}	n_{nm}	n_{holes}	n_{be}	n_{nm}	n_{holes}	n_{be}	n_{nm}	n_{holes}	n_{be}
Pulley	0	1	4	1	2	10	0	0	--	0	0	--
Turbine Blade 2	0	2	9	10	12	77	1	1	11	0	1	3
Dragon	0	40	579	11	12	220	23	24	166	2	31	115
Bimba	0	6	117	11	12	220	8	5	54	0	0	--
Happy Buddha	0	54	508	32	47	462	39	11	93	0	8	41
Support	0	7	43	85	21	57	14	2	8	0	13	159
Rolling Stage	0	1	5	3	6	28	3	5	36	0	0	--
Chinese Dragon	0	19	103	47	35	928	18	13	119	0	12	40
Body	0	8	49	166	25	336	87	30	161	0	50	354
Turbine Blade	0	164	2089	42	66	1054	295	109	864	3	49	180
Nicolò da Uzzano	0	1	4	1	0	--	98	284	1289	41	12	94
Raptor	0	36	207	269	91	1508	--	--	--	0	7	37
Neptune	0	4	34	13	19	107	--	--	--	0	7	37
Asian Dragon	0	31	193	41	88	721	--	--	--	0	7	92

(▫) The Cocone method [9] is used for point clouds of open surfaces whereas the Tight Cocone method [10] is used for point clouds of closed ones.

Tab. 2 Comparison between the reconstruction quality shown by the two versions of the G2S and that of the Cocone methods ([9], [10]) and the Ball Pivoting [17].

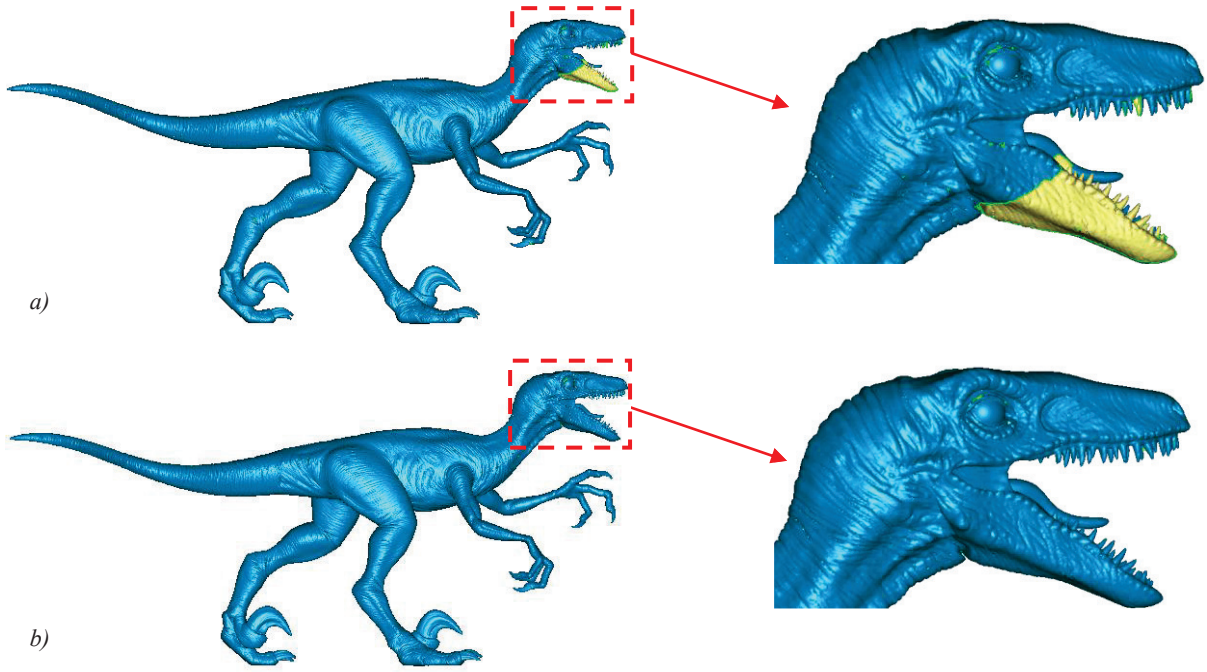

Fig. 7 Renderings of the tessellations obtained for the Raptor with the old (a) and the new versions of the G2S criterion

	Defectiveness generated											
	New G2S Method			Old G2S Method [20]			Robust Cocone method [11]			Ball Pivoting method [17]		
		holes			holes			holes			holes	
	n_{nm}	n_{holes}	n_{be}	n_{nm}	n_{holes}	n_{be}	n_{nm}	n_{holes}	n_{be}	n_{nm}	n_{holes}	n_{be}
σ=0.0001	0	0	--	1	1	6	0	0	--	0	0	--
σ=0.00025	0	17	30	15	22	71	1	0	--	0	3	9
σ=0.0005	0	241	1474	720	457	2967	1	0	--	0	6	36

Table 3. Comparison of defectiveness generated by Robust Cocone [11] and Ball Pivoting [17] in the tessellation of noise added point clouds.

	Defectiveness generated											
	New G2S Method			Old G2S Method [20]			Robust Cocone method [11]			Ball Pivoting method [17]		
		holes			holes			holes			holes	
	n_{nm}	n_{holes}	n_{be}	n_{nm}	n_{holes}	n_{be}	n_{nm}	n_{holes}	n_{be}	n_{nm}	n_{holes}	n_{be}
5%	0	0	--	0	2	4	--	--	--	7	11	149
10%	0	20	244	114	17	725	--	--	--	13	34	206
20%	0	31	457	123	24	757	--	--	--	12	45	326

Table 4. Comparison of defectiveness generated by Robust Cocone [11] and Ball Pivoting [17] in the tessellation of point clouds with outliers added.

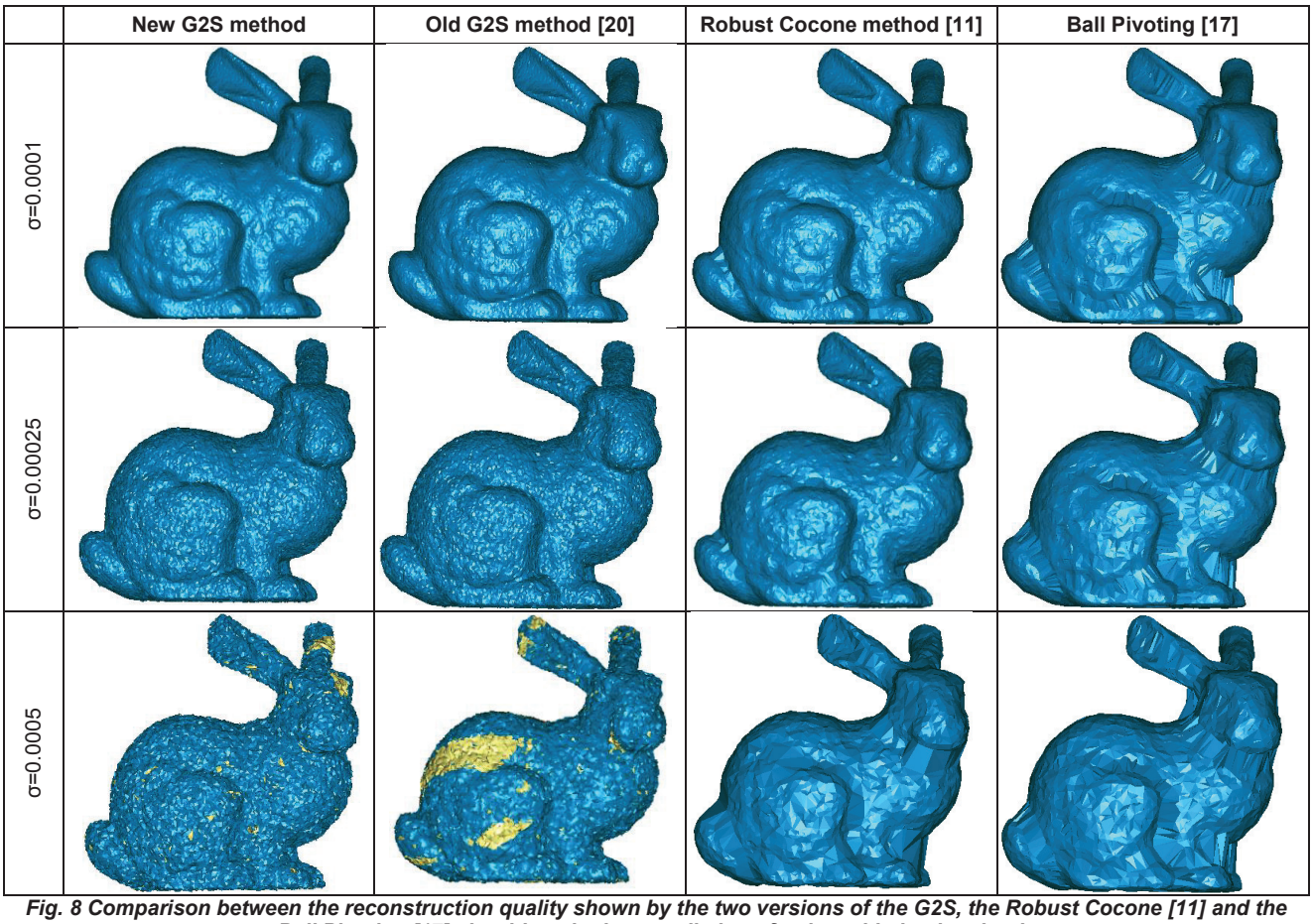

Fig. 8 Comparison between the reconstruction quality shown by the two versions of the G2S, the Robust Cocone [11] and the Ball Pivoting [17] algorithms in the tessellation of noise added point clouds

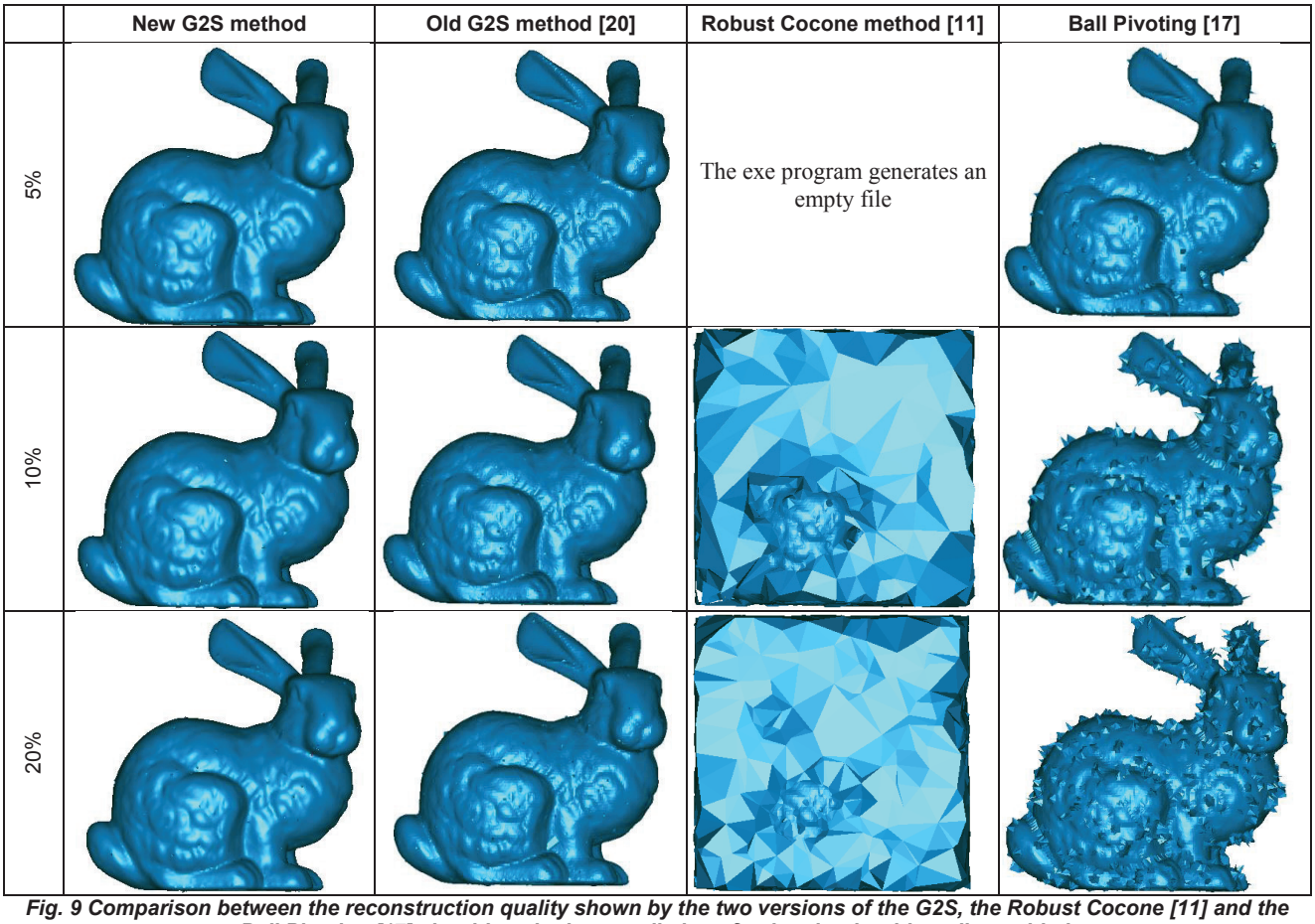

Fig. 9 Comparison between the reconstruction quality shown by the two versions of the G2S, the Robust Cocone [11] and the Ball Pivoting [17] algorithms in the tessellation of point clouds with outliers added.