COMPUTER-AIDED DESIGN & APPLICATIONS, 2017
VOL. 14, NO. 6, 742–750
https://doi.org/10.1080/16864360.2017.1287676

Automatic construction of watertight manifold triangle meshes from scanned point clouds using matched umbrella facets

Ji Ma [a], Jack Szu-Shen Chen [a], Hsi-Yung Feng [a] and Lihui Wang [b]

[a]The University of British Columbia, Canada; [b]KTH Royal Institute of Technology, Sweden

ABSTRACT

Generation of watertight manifold triangle meshes from scanned point clouds has emerged as a key task in computer-aided design and inspection. An effective algorithm is presented in this paper, targeting the automatic creation of such triangle meshes from unorganized, scanned data points. The algorithm builds on the initial version of the Umbrella Facet Matching (UFM) algorithm developed by the authors. The mesh generation process starts with Delaunay triangulation of the given point cloud to determine the Delaunay triangle set at each data point. The algorithm then seeks to iteratively generate, in parallel, the local 2-dimensional manifold triangle mesh, resembling the shape of an open umbrella, at each data point from its Delaunay triangle set that fully overlaps with its neighboring umbrellas. Particularly, a four-level inheritance priority queuing mechanism is introduced to enhance the prioritization and ordering of the Delaunay triangles at each data point in order to facilitate the iterative establishment of the fully matched umbrella according to the most updated umbrella facet matching results. The presented method has been implemented and validated through a series of minimally post-processed scanned point cloud data sets from physical objects with complex geometry. Improved computational convergence has been observed, which promotes the construction of watertight manifold triangle meshes. The comparison results have demonstrated that the enhanced UFM algorithm outperforms the initial UFM algorithm and an industrial software tool in creating quality triangle meshes from actual scanned point clouds.

KEYWORDS

Watertight manifold mesh; point cloud; Delaunay triangles

1. Introduction

Due to the increasing applications of modern 3D scanning technologies, point clouds have emerged as an effective data source to generate the complete surface model of a scanned object. Converting a discrete point cloud data set into a triangle mesh surface representation, commonly known as mesh surface reconstruction, is one of the most widely employed approaches. One key requirement in the application field of computer-aided design and geometric modeling is for the constructed triangle mesh surface to be a watertight manifold surface with correct topology. However, due to the inevitable measurement noise in the scanned data as well as the surface quality of the original scanned object surface, it is still challenging to reconstruct a watertight manifold mesh surface that is topologically equivalent to the original scanned object surface. An effective and reliable method for mesh surface reconstruction is thus in high demand.

2. Relevant studies

Existing surface reconstruction methods can essentially be classified into three groups: implicit surface, region growing, and Delaunay-based methods. The basic concept of the implicit surface method is to use the input point cloud to form a function in the Euclidean space. The function is formulated to be negative inside the modeled object surface and positive outside the modeled object surface. The desired object surface is then extracted as the zero level set of the formulated function. Typical studies in this group include a method based on a spatial grid [14], a method that considers the measurement noise in the scanned data [9], and methods that employ radial basis functions to form the implicit surfaces [11],[20]. In general, the implicit surface method can output a watertight manifold mesh surface and is robust for noisy input point clouds. Nevertheless, this method can lead to poorly shaped triangle meshes in some cases. More importantly, the generated mesh

CONTACT Ji Ma jima605@gmail.com

surface only approximates the input point cloud and does not normally pass through all the given points. This limits its use in applications such as precision inspection and geometric modeling where the scanned data points are considered to be on or at least very close to the reconstructed surface.

For the region growing method, the mesh reconstruction procedure begins with a seed triangle which is grown incrementally to cover the entire point cloud data set. The outstanding issue with this group of methods remains the identification of appropriate seed triangles. The widely applied ball-pivoting algorithm [5] is a region growing method. In this method, a ball with a user-specified radius pivots around an existing triangle edge. When the ball touches another new data point, the point and the edge form a new triangle. An improved triangle growing process [13] has also been proposed which is able to construct the triangle mesh surface directly without using a pivoting ball. These region growing methods are essentially quite computationally efficient but a common weakness is that the quality of the reconstructed mesh greatly depends on one or more user-specified parameters. Applicable values of these parameters are in general not readily available. Hence, a method that excludes the user-specified parameters has been proposed [18]. More recently, Li et al. [17] proposed a priority-driven region growing method which attempts to construct the surface mesh from flat to sharp regions. Also, Delaunay-based region growing methods have been proposed [8],[15],[16]. These methods grow the mesh by exclusively adding Delaunay triangles, which are the surface triangles of the Delaunay tetrahedron at each data point resulting from the Delaunay triangulation of the entire 3D point cloud data set. For all the existing region growing methods, it is known that the reconstructed mesh much depends on the choice of the initial seed triangle and post-processing is often needed in order to attain a satisfactory watertight manifold triangle mesh.

The Delaunay-based method aims to extract a subset of triangles from the complete set of Delaunay triangles of the point set when constructing the desired triangle mesh surface. Boissonnat [6] appeared to be the researcher introducing the first Delaunay-based mesh reconstruction algorithm. To date, many Delaunay-based algorithms have been introduced [2],[4],[10],[12],[22][24]. The Delaunay-based method is systematic and generally considered to be robust. However, generating a watertight manifold mesh without holes has consistently been a challenge. To address this issue, Amenta and her co-workers [2],[3] have introduced a Delaunay-based method with a theoretical guarantee for topological correctness of the generated mesh for points sampled from a smooth surface. The resulting triangle mesh is guaranteed to be topologically equivalent and geometrically close to the original smooth object surface if the particular sampling condition is met. The applicability of this condition to scanned point clouds with noise to construct a topologically correct mesh surface is, however, uncertain. It should be pointed out that an umbrella filter algorithm [1] has been introduced along with a topological post-processing module for triangle mesh surface reconstruction. In this method, numerical difficulty would occur in non-smooth or under-sampled surface regions.

To address the outstanding issues of the existing methods, a new Delaunay-based method is presented in this paper, which enhances the preliminary Umbrella Facet Matching (UFM) algorithm reported by the authors [19]. The fundamental concept of the UFM algorithm is derived from the fact that the neighborhood of each point in a reconstructed closed surface mesh is homeomorphic to a full disc and resembles the shape of an open umbrella. The umbrella contains neither non-manifold edges or vertices nor self-intersections. The UFM algorithm essentially seeks to iteratively generate, in parallel, a fully matched, local 2-dimensional manifold triangle mesh at each point from its Delaunay triangle set. An umbrella of manifold triangle facets is regarded as a fully matched umbrella when it fully overlaps with its neighboring umbrellas. To establish an umbrella at each point requires sequential removal of unwanted Delaunay triangles from its Delaunay triangle set. A prioritized triangle removal sequence is, thus, of critical importance. To effectively perform this task, a priority queuing mechanism to generate the Delaunay triangle removal sequence at each point is introduced in order to iteratively find the fully matched umbrella based on the current facet matching results. Different from the method of Adamy et al. [1], once the algorithm converges and a fully matched umbrella at each data point is found, the generation of a watertight manifold triangle mesh is guaranteed without the need for additional mesh post-processing. Furthermore, the mesh interpolates all the data points and the quality of the meshing results is not dependent on any user-specified parameters. The preliminary UFM algorithm is seen to work well and outperforms many existing mesh surface reconstruction methods; however, its performance on minimally processed scanned data points still leaves some room for improvement. The minimally processed scanned data are raw point cloud data with only data outliers removed. These data are the most typical input data for a mesh reconstruction algorithm but often cause notable issues in constructing the mesh surfaces. A large amount of processing on the scanned data points is typically required from the user before the current mesh reconstruction methods can work well.

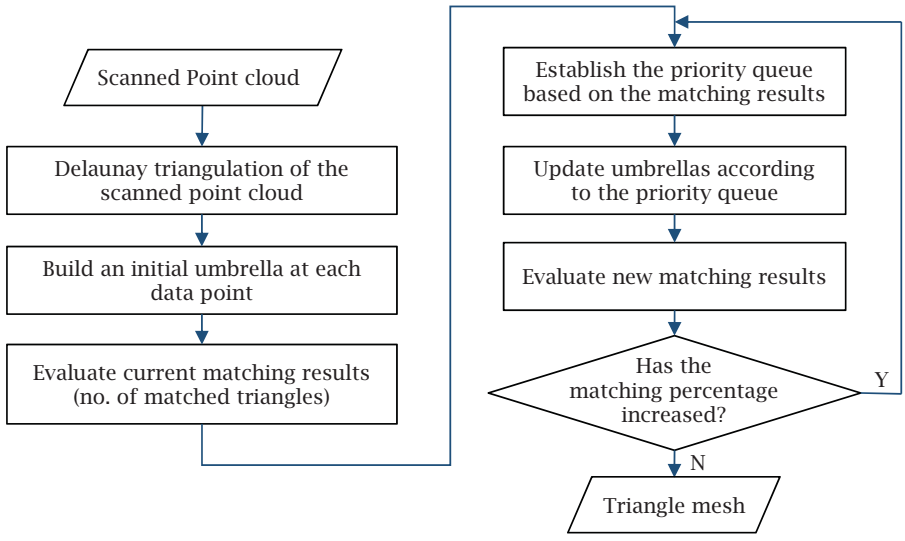

Figure 1. A flowchart for the Umbrella Facet Matching algorithm.

In this paper, an enhanced version of the preliminary UFM algorithm is presented that further promotes numerical convergence. An additional matching index is introduced to the UFM's three tiered priority queuing mechanism. The new matching index adds further insight to each Delaunay triangle regarding its priority to be part of the local umbrella at each data point, leading to a more efficient algorithm and better mesh reconstruction results. A flow chart of the overall algorithm is shown in Fig. 1. The rest of this paper is organized as follows: the fundamentals of the UFM algorithm are outlined in the next section along with the details on the new matching index; Section 4 provides the implementation results; and conclusions are given in the last section.

3. Proposed method

Building an umbrella at each data point in this work is basically a process that sequentially removes all the redundant (non-manifold) triangle facets according to a priority queue [19]. The process will go through three fundamental topological types of Delaunay triangle clusters incident to a data point as depicted in Fig. 2. The original cluster of all the Delaunay triangles is composed of many pockets formed by the triangle facets as shown in Fig. 2(a). Existence of a pocket indicates the existence of a non-manifold triangle facet which has at least one of its two outgoing edges connecting with two or more other triangle facets. After a triangle facet has been removed, another type of non-manifold triangle facets called fins is formed, which has at least one of its outgoing edges not connecting with any other triangle facet as shown in Fig. 2(b). In this work, a redundant triangle facet is a non-manifold facet either as part of a pocket of triangle facets as highlighted in pink in Fig. 2(a), or as a fin as highlighted in pink in Fig. 2(b). The redundant triangle facet removal process starts with removing a non-manifold facet within a pocket, followed by a fin cleaning procedure. This facet removal process ends when there are no more non-manifold edges or vertices in the updated triangle facet cluster, meaning there are no more pockets and fins and thus, no more non-manifold triangle facets to remove. The remaining triangle facets then correctly form an umbrella as depicted in Fig. 2(c).

It is evident that different priority queues for removing the triangle facets would lead to different umbrellas. This means that a specific priority queue at a data point has to be established in order to build the desired umbrella.

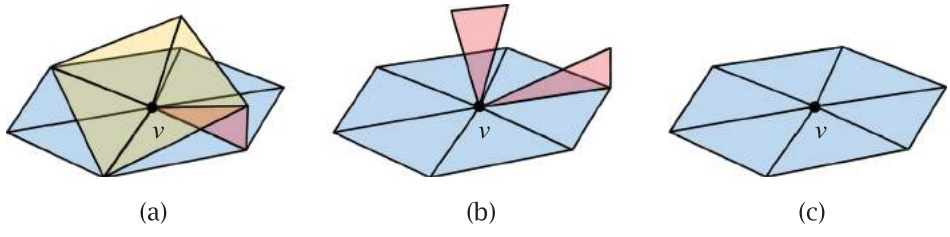

Figure 2. Types of Delaunay triangle clusters: (a) umbrella with pockets, (b) umbrella with fins, and (c) manifold umbrella.

Since the desired umbrellas are the fully matched umbrellas, the corresponding priority queue at each data point should be attainable by updating the priority queue according to the matching results of all the umbrella facets. For an existing umbrella, the matching results of its triangle facets are to be evaluated and then used to establish an updated priority queue via a priority queuing mechanism. The updated priority queue then leads to an updated umbrella. This process repeats until a fully matched umbrella is found. Hence, the priority queuing mechanism and how the umbrella facet matching results are evaluated and used to update the priority queue are clearly the core modules of the automatic mesh reconstruction algorithm in this work.

3.1. Priority queuing mechanism and the initial queue

To construct the desired fully matched umbrella at every data point, a priority queuing mechanism with four-level inheritance is introduced, where a sub-level always inherits the queuing from a super-level. This means that the queuing rules should be prioritized and placed in an ordered sequence starting from the most superior level. For the priority queuing mechanism proposed in this work (Fig. 3), the queuing rule at the first (top) level is the absolute matching index M_f, representing the basic matching result; at the second level is the relative matching index $M_{f(v)}$, representing the refined matching result; at the third level is the neighboring matching index $M_{f(e)}$, representing the extended matching result; and at the fourth (bottom) level is the size of the Delaunay triangle. Details of the matching indices M_f, $M_{f(v)}$ and $M_{f(e)}$ will be presented in the next subsections. As stated in the authors' previous work [19], in the initialization stage of building the initial/first umbrella, only the triangle size information is available (as no matching results exist yet). The diameter of the minimum circumsphere of the triangle is employed to quantify the triangle size. The initial priority queue is then established to remove redundant non-manifold triangles according to their sizes. If an umbrella cannot be constructed, the algorithm then resorts to the complete Delaunay triangle set to ensure that an initial umbrella is established at each data point.

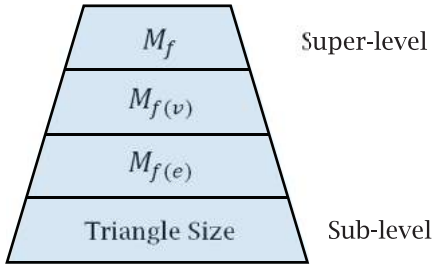

Figure 3. Priority queuing mechanism with four-level inheritance.

3.2. Absolute and relative matching indices

After the initial umbrella at each point is established, three matching indices are evaluated to indicate the degree of overlap among the established umbrellas. In essence, these matching indices are introduced to evaluate how much an umbrella overlaps with its neighboring umbrellas. The first two matching indices were introduced previously: basic and refined [19]. The basic matching result is quantified by the absolute matching index M_f and the refined matching result is quantified by the relative matching index $M_{f(v)}$. The absolute matching index M_f is devised to indicate the degree of matching for a facet f. The relative matching index $M_{f(v)}$ is devised to indicate the degree of matching for the facet f relative to the vertex v.

There are a total of six possible cases of M_f and $M_{f(v)}$. as illustrated in Fig. 4. In principle, $M_{f(v)}$ is a first extension of M_f. In the figure, the facet f has three vertices: v_1, v_2, and v_3. When M_f equals 3, this means that all of the three umbrellas respectively incident to v_1, v_2, and v_3 include the facet f. The last (right most) triangle in Fig. 4 depicts this case. A solid dot for a vertex indicates that the umbrella of this vertex includes the facet f and an empty dot for a vertex indicates that its umbrella does not include the facet f. When M_f equals 2, only two of the three umbrellas respectively incident to v_1, v_2, and v_3 include the facet f. In this case, there exist two

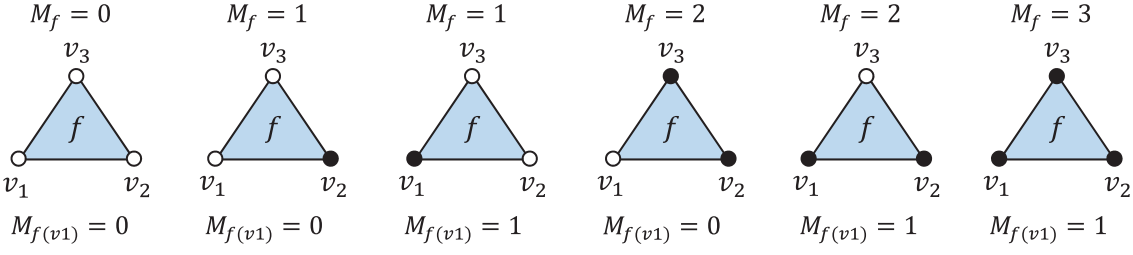

Figure 4. Absolute and relative matching indices of f at v_1.

possible situations. Relative to the vertex v_1, one situation is that the umbrella at v_1 does not include the facet f (the fourth triangle in Fig. 4) and the other situation is that the umbrella at v_1 includes the facet f (the fifth figure in Fig. 4). Their relative matching indices are then respectively expressed as: $M_{f(v1)} = 0$ and $M_{f(v1)} = 1$. The first triangle in Fig. 4 illustrates the situation when all of the three umbrellas incident to v_1, v_2, and v_3 do not include the facet f. As a result, $M_f = 0$ and $M_{f(v1)} = 0$.

As stated previously, the presented UFM algorithm centers on the sequential removal of redundant triangle facets from the candidate triangle facet cluster. This is achieved via the priority queuing mechanism with multi-level inheritance according to the umbrella facet matching results. The priority queue is formed according to the evaluated values of the absolute matching index M_f followed by the relative matching index $M_{f(v)}$. More specifically, for all the triangle facets from the candidate facet cluster at a vertex v, the sequence is formed from $M_f = 0$ to $M_f = 3$ first and then from $M_{f(v)} = 0$ to $M_{f(v)} = 1$. Those facets with the same $M_{f(v)}$ value are then ordered by the neighboring matching index $M_{f(e)}$ as the third-level rule in the priority queuing mechanism (Fig. 3).

3.3. Neighboring matching index

As repeatedly stated in the previous sections, a generated priority queue at a point will lead to a specific umbrella. The objective of the UFM algorithm is to establish the priority queue at each point that would result in fully matched umbrellas for the complete data set. This objective is to be achieved by iteratively updating the priority queue at every point according to the current umbrella facet matching results. As reported in the authors' preliminary work [19], there remain regularly many triangle facets incident to a vertex v which are characterized with the same $M_{f(v)}$. These facets were prioritized according to their triangle sizes (minimum circumsphere radii) in the same way as in the initialization stage. This sequencing much affects the convergence rate, if not the eventual convergence of the algorithm. To promote convergence of the UFM algorithm, an additional matching index has been introduced in order to enable triangle facets with the same $M_{f(v)}$ but larger sizes to be placed behind those with smaller sizes in the priority queue of facet removal if so indicated by the current facet matching results. In other words, the removal priority of facets with the same $M_{f(v)}$ should not be simply based on their sizes. Current matched facets should carry decisive weights to promote neighboring facets that expedite the generation of fully matched umbrellas and the overall watertight manifold triangle mesh surface.

A neighboring matching index, $M_{f(e)}$, is employed as the third-level queuing measure in the priority queuing mechanism presented in this work (Fig. 3). As depicted in Fig. 5, there are three edges e_1, e_2 and e_3 in the triangle facet $f(v1,\ v2,\ v3)$ and each edge has some connected neighboring umbrella triangle facets. Each of these neighboring facets is characterized by an absolute matching index value from the current facet matching result. Let the maximum absolute matching index value among the neighboring facets of edge e_1 be denoted by M_{f1max} and its value could be 3, 2, 1 or 0. To evaluate the neighboring matching results of the facet $f(v1,\ v2,\ v3)$, the following index is devised:

$$M_{f(e)} = \sum_{i=1}^{3} M_{fimax} \tag{1}$$

where the value of $M_{f(e)}$ ranges from 0 to 9. For the triangle facets with the same $M_{f(v)}$ value, those with smaller $M_{f(e)}$ values (not much neighboring support) are to be placed in front of those with larger $M_{f(e)}$ values in the priority queue of facet removal. Those facets with the same $M_{f(e)}$ value are then ordered by their sizes (minimum circumsphere radii) as the bottom-level rule in the priority queuing mechanism. Evidently, the priority queue at each point will be continually updated until all the fully matched umbrellas are successfully found. The success is attributed to the four-level inheritance priority queuing mechanism that allows the proposed UFM algorithm to effectively converge to the desired fully matching of all the umbrellas.

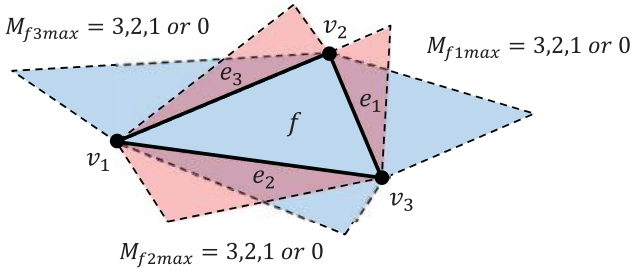

Figure 5. Neighboring matching for facet $f(v1,\ v2,\ v3)$.

4. Implementation results

The enhanced UFM algorithm presented in this paper has been implemented and evaluated using many scanned point cloud data sets. To perform 3D Delaunay triangulation on a point set, the existing codes in the Computational Geometry Algorithms Library CGAL [7] were employed. Also, to effectively manage the topological information of a mesh in the mesh generation process, another open-source template library, the VCG Library

[21], was referenced for manipulating and processing the triangle mesh. The VCG Library supports complete non-manifold and manifold mesh manipulation and processing via an adjacency-indexed data structure, which is different from the traditional half-edge and Weiler's radial edge data structure [23]. This data structure facilitates the fundamental iterative procedure in the UFM algorithm to identify the matched triangle facets. The reported case studies were carried out on a Windows-based PC with a 3.50 GHz processor and 8GB memory.

The objective of this work is to enhance the UFM algorithm so that minimally post-processed scanned point cloud data can be directly used to construct a quality triangle mesh surface. Users prefer not to spend time in preparing well-processed scanned point cloud data for mesh construction. Quite often, the raw scanned data are only processed for outlier removal. Thus, the practical tests in this section will focus on execution of the enhanced UFM method on minimally post-processed scanned point clouds. For tests on highly post-processed scanned data and comparison of the UFM algorithm with other existing algorithms, please refer to the authors' preliminary UFM work [19].

Fig. 6 shows the implementation results of the enhanced UFM algorithm on 7 scanned point cloud data sets with minimal post-processing. The scanned point clouds were obtained by an LDI Surveyor WS3040 3D laser scanner. Only outlier removal was performed on the scanned data sets without any noise reduction processing. It can be seen that for both organic and mechanical objects, the algorithm succeeds in constructing visually well-behaved triangle mesh models. To examine these results further at a quantitative level, Tab. 1 lists and compares the mesh reconstruction results of the preliminary and current/enhanced UFM algorithms as well as the Mesh Doctor module of the Geomagic Studio 12 commercial software. As indicated in the table via the matching percentage, which is defined as the ratio of the number of the data points with fully matched umbrellas to the total number of data points in the input scanned point cloud, the enhanced UFM algorithm in general achieves better convergence. It should be noted that for practical scanned point clouds with minimal post-processing, 100% matching or full convergence is difficult to attain. The enhanced UFM algorithm is better at reaching full convergence but it does not always achieve 100% matching. It is evident that without full convergence, some umbrellas are not fully matched. This is expected since practical scanned point clouds are often quite noisy. The enhanced UFM algorithm attempts to achieve 100% matching results in an iterative manner but it cannot guarantee convergence for minimally processed scanned point cloud data sets. Nevertheless, the matching percentages for the non-converged cases are close to 100% and the number of matched triangle facets is consistently larger for the enhanced UFM algorithm. In particular, compared with the widely used Geomagic software tool, the outputs from the enhanced UFM algorithm are clearly superior. The triangle meshes resulting from the UFM algorithm are guaranteed to be manifold without the triangles crossing each other (self-intersections). Triangle self-intersections in a generated mesh are highly undesirable in manifold mesh surface reconstruction from scanned point clouds since the surface of a physical

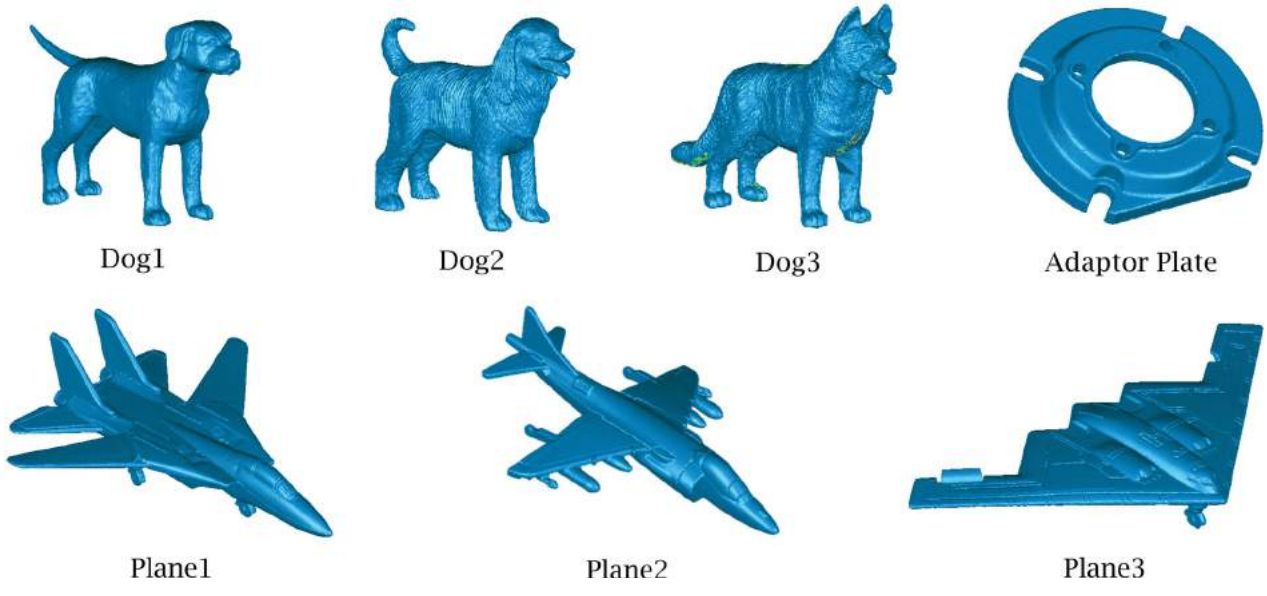

Figure 6. Reconstructed meshes from minimally-processed scanned point cloud data sets.

object cannot possibly cross itself. With the presented UFM algorithm, such a digital artifact is avoided, leading to topological correct reconstructed mesh surfaces. Such reconstructed mesh models represent and model their physical counterparts better even if the matching percentage is not 100%.

It should be pointed out that the topological improvement in the generated triangle mesh is acquired at the

Table 1. Comparison of mesh reconstruction results.

	Geomagic Studio 12	Preliminary UFM	Current/Enhanced UFM
Dog1 (99,753 points)		Matching: 100% No. of Facets: 199,502 Time: 77 seconds	Matching: 100% No. of Facets: 199,502 Time: 76 seconds
Non-Manifold Edges	0	0	0
Self-Intersections	0	0	0
Spikes & Creased Edges	283	1,803	1,949
Small Components	0	0	0
Small Holes	0	0	0
Dog2 (99,925 points)		Matching: 99.98% No. of Facets: 199,843 Time: 135 seconds	Matching: 100% No. of Facets: 199,846 Time: 113 seconds
Non-Manifold Edges	0	0	0
Self-Intersections	0	0	0
Spikes & Creased Edges	2,457	14,247	14,431
Small Components	0	0	0
Small Holes	0	0	0
Dog3 (99,596 points)		Matching: 92.37% No. of Facets: 195,831 Time: 654 seconds	Matching: 94.20% No. of Facets: 198,223 Time: 663 seconds
Non-Manifold Edges	0	0	0
Self-Intersections	563	0	0
Spikes & Creased Edges	13,789	42,022	42,902
Small Components	1	259	86
Small Holes	0	150	129
Plane1 (100,700 points)		Matching: 99.86% No. of Facets: 201,344 Time: 174 seconds	Matching: 99.81% No. of Facets: 201,375 Time: 205 seconds
Non-Manifold Edges	0	0	0
Self-Intersections	30	0	0
Spikes & Creased Edges	2,099	4,853	4,855
Small Components	0	3	1
Small Holes	1	4	1
Plane2 (100,538 points)		Matching: 99.92% No. of Facets: 201,049 Time: 151 seconds	Matching: 99.91% No. of Facets: 201,068 Time: 212 seconds
Non-Manifold Edges	0	0	0
Self-Intersections	20	0	0
Spikes & Creased Edges	1,931	3,617	1,949
Small Components	0	0	0
Small Holes	0	0	0
Plane3 (100,235 points)		Matching: 100% No. of Facets: 200,466 Time: 96 seconds	Matching: 100% No. of Facets: 200,466 Time: 88 seconds
Non-Manifold Edges	0	0	0
Self-Intersections	0	0	0
Spikes & Creased Edges	1,455	5,707	5,627
Small Components	0	0	0
Small Holes	0	0	0
Adaptor Plate (100,013 points)		Matching: 99.93% No. of Facets: 200,038 Time: 183 seconds	Matching: 99.97% No. of Facets: 200,040 Time: 212 seconds
Non-Manifold Edges	0	0	0
Self-Intersections	64	0	0
Spikes & Creased Edges	4,969	15,638	15,588
Small Components	0	0	0
Small holes	0	0	0

expense of surface smoothness. The outputs from the UFM algorithm are inherently "rougher" compared to the Geomagic outputs. Surface spikes and creases are more extensive on the UFM meshes (Tab. 1). Even though the surface roughness is higher for the UFM meshes, they are still favorable as surface roughness reduction is much easier than self-intersection correction. Surface roughness reduction simply involves averaging of the mesh vertex locations of a well-constructed mesh while self-intersection correction typically requires additional and often complex mesh reconstruction steps. It is, thus, much preferable to create a non-smooth mesh than a topologically flawed mesh.

Compared with the authors' preliminary UFM work [19], the current enhanced UFM algorithm with the added neighboring matching index is seen to converge faster, improve the matching percentage, and/or the reconstructed mesh quality, as shown in Tab. 1. For the cases of Dog1 and Plane3, both the preliminary and enhanced UFM algorithms were able to attain full convergence. The enhanced UFM algorithm is seen to converge faster. For the case of Dog2, the added neighboring matching index helped the enhanced UFM algorithm achieve full convergence which was not attainable for the preliminary UFM algorithm. For the cases of Dog3 and Adaptor Plate, the matching percentages from the enhanced algorithm are improved. In particular, the Dog3 point cloud data was very noisy. Hence, the algorithm execution reached the maximum number of iterations, leading to extensive computing time. It should be noted, however, that there are much fewer small components (small detached or isolated mesh patches) and small holes (only a few triangles in size) in the mesh generated by the enhanced UFM algorithm. For the cases of Plane1 and Plane2, although the matching percentages are a bit less for the enhanced UFM algorithm, the overall matched triangle facets are in fact more in number. More importantly, the number of small components and small holes in the Plane1 mesh are reduced for the enhanced UFM algorithm and the number of surface spikes and creases are much reduced in the Plane2 mesh. These mesh reconstruction results clearly demonstrate the effectiveness of the enhanced UFM algorithm presented in this work.

5. Conclusions

By improving the priority queuing mechanism in the authors' preliminary work [19], an enhanced UFM algorithm to automatically generate watertight manifold triangle meshes from scanned point cloud data sets has been presented in this paper. The generated mesh will interpolate all the input data points and be topologically very close to the original scanned object surfaces with no influential or apparent mesh defects. The addition of the neighboring matching index and the resulting four-level inheritance priority queuing mechanism allow the enhanced UFM algorithm to better promote umbrella facet matching convergence, thereby improving the generated mesh quality and the mesh generation speed. From the series of case studies, it is found that the enhanced UFM algorithm in general outperforms the preliminary UFM algorithm in generating the watertight manifold triangle meshes. Still, full convergence of the UFM algorithm for minimally post-processed noisy scanned point cloud data sets cannot be guaranteed. As a possible approach for improvement, geometric heuristics could be introduced as an additional index to control the shape of the umbrella at some specific data points to offer some control over the local geometry. With this additional index, convergence for noisy scanned point cloud data sets could be further improved and the objective of fully automatic construction of watertight manifold triangle meshes from scanned point clouds can be fulfilled.

Acknowledgement

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC).

Funding

This work was supported by Natural Sciences and Engineering Research Council of Canada (NSERC).

ORCID

Ji Ma http://orcid.org/0000-0001-6536-2617
Jack Szu-Shen Chen http://orcid.org/0000-0002-1225-6340
Hsi-Yung Feng http://orcid.org/0000-0001-6189-6910
Lihui Wang http://orcid.org/0000-0001-8679-8049

References

[1] Adamy, U.; Giesen, J.; John, M.: Surface reconstruction using umbrella filters, Computational Geometry, 21(1), 2002, 63–86. http://dx.doi.org/10.1016/S0925-7721(01)00040-2
[2] Amenta, N.; Bern, M.; Kamvysselis, M.: A new Voronoi-based surface reconstruction algorithm, Proceedings of SIGGRAPH '98, 1998, 415–421. http://dx.doi.org/10.1145/280814.280947
[3] Amenta, N.; Bern, M.: Surface reconstruction by Voronoi filtering, Discrete & Computational Geometry, 22(4), 1999, 481–504. http://dx.doi.org/10.1007/PL00009475
[4] Amenta, N.; Choi, S.; Kolluri, R. K.: The power crust, Proceedings of the 6th ACM Symposium on Solid Modeling and Applications, 2002, 249–266.

[5] Bernardini, F.; Mittleman, J.; Rushmeier, H.; Silva, C.; Taubin, G.: The ball-pivoting algorithm for surface reconstruction, IEEE Transactions on Visualization and Computer Graphics, 5(4), 1999, 349–359. http://dx.doi.org/10.1109/2945.817351

[6] Boissonnat, J. D.: Geometric structures for three- dimensional shape representation, ACM Transactions on Graphics, 3(4), 1984, 266–286. http://dx.doi.org/10.1145/357346.357349

[7] CGAL, Computational Geometry Algorithm Library, [Online], Available: http://www.cgal.org

[8] Cohen-Steiner, D.; Da, F.: A greedy Delaunay-based surface reconstruction algorithm, The Visual Computer, 20(1), 2004, 4–16. http://dx.doi.org/10.1007/s00371-003-0217-z

[9] Curless, B.; Levoy, M.: A volumetric method for building complex models from range images, Proceedings of SIGGRAPH '96, 1996, 303–312. http://dx.doi.org/10.1145/237170.237269

[10] Dey, T. K.; Goswami, S.: Tight cocone: A water-tight surface reconstructor, Proceedings of the 8th ACM Symposium on Solid Modeling and Applications, 2003, 127–134. http://dx.doi.org/10.1145/781606.781627

[11] Dinh, H. Q.; Turk, G.; Slabaugh, G.; Reconstructing surfaces by volumetric regularization using radial basis functions, IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(10), 2002, 1358–1371. http://dx.doi.org/10.1109/TPAMI.2002.1039207

[12] Edelsbrunner, H.; Mücke, E. P.: Three-dimensional alpha shapes, ACM Transactions on Graphics, 13(1), 1994, 43–72. http://dx.doi.org/10.1145/174462.156635

[13] Huang, J.; Menq, C. H.: Combinatorial manifold mesh reconstruction and optimization from unorganized points with arbitrary topology, Computer-Aided Design, 34(2), 2002, 149–165. http://dx.doi.org/10.1016/S0010-4485(01)00079-3

[14] Hoppe, H.; DeRose, T.; Duchamp, T.; McDonald, J.; Stuetzle, W.: Surface reconstruction from unorganized points, Proceedings of SIGGRAPH '92, 1992, 71–78. http://dx.doi.org/10.1145/133994.134011

[15] Kuo, C. C.; Yau, H. T.: A Delaunay-based region-growing approach to surface reconstruction from unorganized points, Computer-Aided Design, 37(8), 2005, 825–835. http://dx.doi.org/10.1016/j.cad.2004.09.011

[16] Kuo, C. C.; Yau, H. T.: A new combinatorial approach to surface reconstruction with sharp features, IEEE Transactions on Visualization and Computer Graphics, 12(1), 2006, 73–82. http://dx.doi.org/10.1109/TVCG.2006.2

[17] Li, X.; Han, C. Y.; Wee, W. G.: On surface reconstruction: A priority driven approach, Computer-Aided Design, 41(9), 2009, 626–640. http://dx.doi.org/10.1016/j.cad.2009.04.006

[18] Lin, H. W.; Tai, C. L.; Wang, G. J.: A mesh reconstruction algorithm driven by an intrinsic property of a point cloud, Computer-Aided Design, 36(1), 2004, 1–9. http://dx.doi.org/10.1016/S0010-4485(03)00064-2

[19] Ma, J.; Feng, H. Y.; Wang, L.: Delaunay-based triangular surface reconstruction from points via umbrella facet matching, Proceedings of the 6th IEEE Conference on Automation Science and Engineering, 2010, 580–585.

[20] Turk, G.; O'Brien, J. F.: Shape transformation using variational implicit functions, Proceedings of SIGGRAPH '99, 1999, 335–342. http://dx.doi.org/10.1145/311535.311580

[21] VCG Library, [Online], Available: http://vcg.sourceforge.net

[22] Veltkamp, R. C.: The γ-neighborhood graph, Computational Geometry, 1(4), 1992, 227–246. http://dx.doi.org/10.1016/0925-7721(92)90003-B

[23] Weiler, K.: The radial edge structure: A topological representation for non-manifold geometric boundary modeling, Geometric Modeling for CAD Applications, 1988, 3–36.

[24] Xu, X.; Harada, K.: Automatic surface reconstruction with alpha-shape method, The Visual Computer, 19(7), 2003, 431–443.