Normal Vector Estimation for Point Clouds via Local Delaunay Triangle Mesh Matching

Ji Ma[1], Hsi-Yung Feng[2] and Lihui Wang[3]

[1]The University of British Columbia, jma84@mail.ubc.ca
[2]The University of British Columbia, feng@mech.ubc.ca
[3]University of Skövde, lihui.wang@his.se

ABSTRACT

Reliable estimation of normal vectors for point clouds is of practical importance in computer-aided geometric modeling and inspection. This paper introduces a new normal vector estimation method for point clouds based on the matching results of the local Delaunay triangle mesh formed at each point. The local mesh is a manifold patch of Delaunay triangles and resembles an open umbrella. According to the matching results of these umbrellas, the local Delaunay neighbors at each point can be reliably identified, which leads to accurate normal vector calculations. Compared with the existing methods, the proposed method yields normal vectors of notably improved accuracy, especially for points near edge or corner features. The improvement has been demonstrated using both simulated and scanned point cloud data sets.

Keywords: normal vector, point cloud, Delaunay neighbors, local mesh matching.
DOI: 10.3722/cadaps.2013.399-411

1 INTRODUCTION

To date, a point cloud is becoming a new data format to represent 3D surface geometry due to the increasing application of 3D scanning systems. Reliable and accurate estimation of normal vectors of a point cloud is important in many practical applications of computer-aided geometric modeling and inspection. For instance, surface reconstruction from a point cloud with reliable normal vectors is a much easier problem to solve than surface reconstruction from a point cloud alone. For many existing surface reconstruction algorithms, the quality of a reconstructed surface heavily relies on how closely the estimated normal vectors approximate the true normals on the scanned physical object surface. In fact, normal vector estimation is often the very first data processing task in a surface reconstruction algorithm. This is not only true for Delaunay-based and region-growing approaches [4],[7],[12],[31], but also for implicit-surface oriented approaches [1],[3],[5],[8],[14],[17]. Other applications requiring

accurate estimated normal vectors include segmentation of the point cloud [16],[18],[20],[29],[30] and point-based surface rendering [1],[15],[25].

Many normal vector estimation methods for point clouds have been proposed in the literature. These methods mainly fall into two dominant categories [13]: numerical optimization based on plane or surface fitting [17],[21],[24] and geometric analysis based on Voronoi diagram / Delaunay triangulation [4],[12],[14],[31]. In all of the reported methods, the normal vector estimation procedure generally involves two main steps. The first step is to identify a set of local neighborhood points for each point in the point cloud data. The second step is to determine the desired normal vector using the identified local neighborhood points. Details on the existing normal vector estimation methods and their main features are presented in the next section.

2 EXISTING METHODS

Normal vector is a local geometric property of a surface and specific to each surface point. As a result, it is sensible that reliable estimation of the normal vector at a point in a point cloud would depend significantly on the correct identification of the point's local neighborhood points. Using too many neighboring points for normal vector estimation can lead to inaccuracy in the estimated normal vector, especially for points near sharp or high-curvature features. Using too few neighboring points may not represent the local geometry adequately, which again compromises the estimation accuracy. A well-chosen set of neighboring points is thus essential for reliable normal vector estimation.

The first normal estimation method appears to be developed by Hoppe et al. [17] in the context of surface reconstruction. The method finds the k-nearest neighbors of a given point v, for which the entire neighboring point set is denoted as $N_k(v)$, and takes the normal of the least-squares best-fitted plane to $N_k(v)$ as the surface normal at v. Such an normal estimation method has been called the plane fitting (PF) method [13]. The k-nearest neighbors $N_k(v)$ can also be used to fit a local quadric surface for normal vector estimation [30]. Pauly et al. [24] improved the original PF method by a weighted least-squares formulation with different weight assigned to each neighboring point based on its distance to v. If the distance of a neighboring point p_i to v is small, a large weight is assigned via a Gaussian function formula. The primary issue for the k-nearest neighbors is the assumed consistency in point distribution. So, when the ideal $N_k(v)$ is non-uniform in size, the selected uniform-in-size $N_k(v)$ at each point may not provide reliable local geometric information throughout the point cloud data set. Methods to select the neighboring points of v according to a fixed or adaptive distance r from v have also been applied. Mitra et al. [21] proposed a plane fitting method based on an adaptive distance r to estimate the normal vectors in a point cloud. It is evident that to determine the optimal r at each point would then become the challenging issue.

An alternative concept to adaptively select the local neighboring points is through the construction of a local polygonal mesh surface at each point. This has been done based on the geometric analysis of the Voronoi diagram/Delaunay triangulation of the input point cloud. Voronoi diagram and Delaunay triangulation are closely associated global geometric structures of a point set and can be built for any point cloud data set with arbitrary non-degenerative point distribution and density. They provide a powerful means to approximate the local neighborhood at each point in a point cloud – the local neighboring points at each point can always be extracted from the local Delaunay triangle set incident to each point. Adamy et al. [2] proposed to build an umbrella using a set of connected Delaunay triangles at each point, taken from the Gabriel subset of the complete Delaunay triangle set. The building of the umbrella is an incremental triangle-adding process based on the proposed λ-interval

concept, which often requires an additional manifold post-processing step. The resulting umbrella is in fact a local manifold triangle mesh. OuYang and Feng [23] proposed to build the local Delaunay triangle mesh at each point based on the region growing concept, which is similar to the ball-pivoting algorithm [7]. A set of quadric curves were then constructed using the identified neighboring points from the built local triangle mesh to calculate the normal vector. The main benefit of this algorithm is that the required number of neighboring points could be only three. It should be emphasized that all of the methods mentioned above do not evaluate the quality of the built local triangle mesh (thus the reliability of the identified local neighboring points). Also, the procedure to estimate the normal vector from the identified neighboring points for these existing methods is assumed to be applicable across the overall surface region, smooth or non-smooth. In practical situations, however, the local triangle mesh at a point cannot always be built reliably. It is thus much desirable to apply differentiated normal vector calculation procedures to points with varied degree of reliability for their respective identified neighboring points. In this work, a new algorithm has been developed to address the issues described above in order to provide better normal vector estimates for point clouds.

3 PROPOSED METHOD

As discussed in the previous sections, existing normal vector estimation methods in general follow a distinct two-step approach: identifying local neighborhood points first and then calculating the desired normal vector from the identified neighboring points. Unlike these existing methods, an improved and integrated method is proposed in this paper to link the quality (reliability) of the identified local neighborhood points to the specific numerical procedure to calculate the desired normal vector. More specifically, a matching scheme is developed to quantify the reliability of each plausible neighboring point candidate. When a good percentage of the neighboring point candidates are evaluated as reliable, the point of interest is deemed to be in a region without much geometric variation and data inconsistency. An accurate normal vector estimate can thus be obtained from the reliably identified neighboring points. On the other hand, when only a small number of reliable neighboring points can be attained, it is likely that the point of interest is in a region with large geometric variation or data inconsistency. In this case, a more inclusive procedure that would accept contributions from all the plausible neighboring point candidates to estimate the normal vector is more appropriate. Details of the proposed method are presented in the following subsections.

3.1 Local Neighborhood

The neighborhood at each point in a point cloud P has been widely studied and applied in the computational geometry community [22]. Typical employed point neighborhoods include the k-nearest neighbors $N_k(v)$, Euclidean minimum spanning tree $EMST(P)$, Gabriel graph $GG(P)$, and Delaunay triangulation $D(P)$. These geometric structures all have specific properties that are of benefit in establishing the neighborhood at a point. These geometric properties have also been used by researchers in formulating the initial step for 2D curve reconstruction. Let the edge set of Delaunay triangulation in 2D be denoted by $ED(P)$. The relationship of the relevant graphs of a 2D point cloud P can be expressed as:

$$EMST(P) \subseteq GG(P) \subseteq ED(P) \tag{3.1}$$

Graph is a geometric representation in which pairs of points are linked by an edge. For example, as an initial graph, $EMST(P)$ guarantees that the resulting edges are the shortest possible. Therefore, points close to one another in a point cloud are likely to be linked together. $GG(P)$ gives some clue about the

best interconnection among points in reconstructing the boundary of a 2D point cloud [27],[28]. Nonetheless, in most cases, $EMST(P)$ provides too few neighboring points while $GG(P)$ and $ED(P)$ provide too many neighboring points for local neighborhood estimation.

In some existing 3D surface reconstruction approaches, the reconstructed triangle mesh surface is extracted from the Delaunay triangle set $D(P)$ of a point cloud P. The Gabriel triangle set $GT(P)$, a subset of $D(P)$, is often applied at the initial step to help identify the Delaunay triangle candidates for reconstructing the triangle mesh surface [2],[6],[19]. For a point v in P, the relationship $GT(P) \subseteq D(P)$ could also be locally described at v as:

$$GT_v \subseteq DT_v \tag{3.2}$$

The concept of local mesh neighbors does avoid the bias issue of adopting the k-nearest neighbors in normal vector estimation. However, as noted in the previous section, these existing local mesh construction methods do not provide information on the quality of the constructed local mesh.

3.2 Reliable Local Mesh Neighbor Determination

In the Umbrella Facet Matching algorithm presented in the authors' previous work [19], a full umbrella at each point is guaranteed to be generated. In particular, based on the built umbrella at each point, a novel methodology to evaluate the matching results of all the umbrella facets is proposed with the matching resulting quantified using a set of matching indices. This in effect provides a way to evaluate the reliability of the established local mesh neighbors at each point. The matching index M_f was devised to indicate the degree of overlap among the generated full umbrellas. For example, for a triangle facet f in an umbrella with three vertices v_1, v_2 and v_3, when M_f equals three, this means that all the three neighboring umbrellas incident to v_1, v_2 and v_3 include the triangle f. A fully matched triangle facet is denoted as $\bar{f}$. When M_f equals two, only two of the three umbrellas incident to v_1, v_2 and v_3 include the triangle f. If only one of the neighboring umbrellas incident to v_1, v_2 and v_3 include the triangle f, M_f equals one. Thus, the matching index M_f of a triangle f reflects the degree of overlap among all of its three neighboring umbrellas. The triangle f with a larger M_f value means it is more reliable to be part of the local triangle mesh for determining the local neighborhood points to estimate the desired normal vector. The constructed local mesh at each point is in fact a subset of Delaunay triangulation. For a point v, the following relationship exists:

$$U(\bar{f})_v \subseteq U(f)_v \subseteq DT(U)_v \subseteq DT_v \tag{3.3}$$

where DT_v denotes all the Delaunay triangles incident to v, $DT(U)_v$ all the (Delaunay) triangles incident to point v that belong to a umbrella at any point, $U(f)_v$ all the Delaunay triangles in the umbrella at v, and $U(\bar{f})_v$ all the fully matched Delaunay triangles at v.

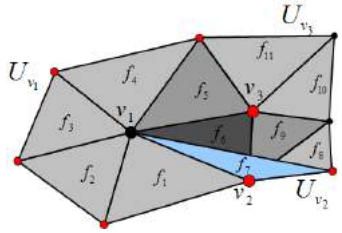
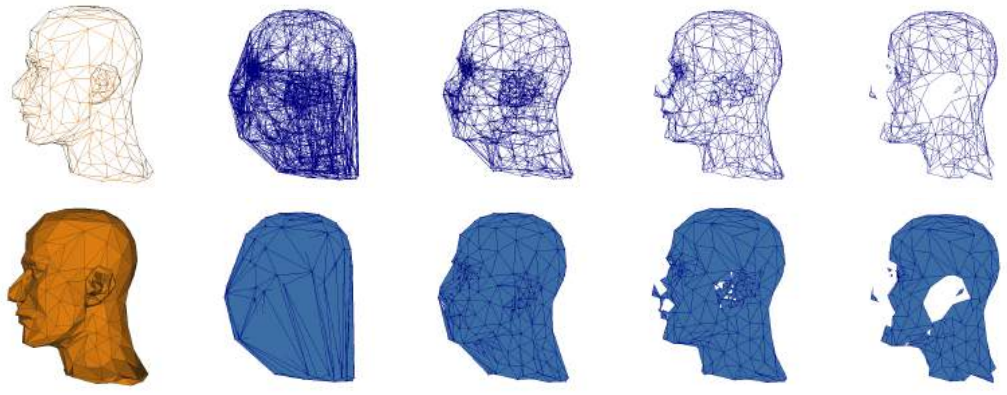


Fig. 1: Neighboring points and their umbrellas.

An example is illustrated in Fig. 1, where three input points v_1, v_2 and v_3 and their umbrellas U_{v_1}, U_{v_2} and U_{v_3} are shown with $U_{v_1} = \{f_1, f_2, f_3, f_4, f_5, f_6\}$, $U_{v_2} = \{f_6, f_7, f_8, f_9\}$ and $U_{v_3} = \{f_5, f_6, f_9, f_{10}, f_{11}\}$. In these Delaunay triangles incident to v_1, v_2 and v_3, there exists one fully matched triangle f_6 ($M_{f_6} = 3$). The matching index of triangle f_5 and f_9 is two ($M_{f_5} = 2$ and $M_{f_9} = 2$). The matching index of the other triangles is one. According to Eqn. (3.3), different local mesh neighbors can be attained for point v_1: $U(\overline{f})_{v_1} = \{f_6\}$, $U(f)_{v_1} = \{f_1, f_2, f_3, f_4, f_5, f_6\}$ and $DT(U)_{v_1} = \{f_1, f_2, f_3, f_4, f_5, f_6, f_7\}$. For clarity, all the associated neighboring points in $DT(U)_{v_1}$ of point v_1 are shown in red in Fig. 1.

Fig. 2 compares the resulting meshes based on the employment of different local mesh neighbors in order to shed some light on the reliability of these different options. Fig. 2(a) is the original Mannequin triangle mesh surface for a point cloud P (downloaded from the CGAL website), Fig. 2(b) the global Delaunay triangulation of P, and Fig. 2(c) the Gabriel triangles subset. Figs. 2(d) and 2(e) respectively illustrate $U(f)_P$, which is equivalent to $DT(U)_P$, and $U(\overline{f})_P$ based on the umbrella facet matching results at all the points in P. It can be seen that the triangle mesh in Fig. 2(d) is the best approximation of the original triangle mesh surface. Fig. 2(e) perhaps depicts the most reliable local mesh neighbors but some points/triangles are clearly missing. In this work, the desired normal vectors will be computed based on a detailed analysis of the local Delaunay neighbors shown in Figs. 2(d) and 2(e), as will be presented in the next subsection.



(a) (b) (c) (d) (e)

Fig. 2: Combined meshes resulting from different local mesh neighbors.

3.3 Normal Vector Computation

The well-known Euler formula [10] describes the relationship between the numbers of vertices V, edges E and faces F in a closed 2-manifold polygonal mesh:

$$V - E + F = 2(1 - G) \tag{3.4}$$

where G is the genus of the meshed object and intuitively represents the number of through-holes in the object. As the genus of a typical mesh is generally small compared to the number of mesh elements, the right-hand side of Eqn. (3.4) can be assumed to be close to zero. For a closed 2-manifold triangle mesh, each triangle is bounded by three edges and each edge is incident to two triangles, the following triangle mesh statistics can thus be derived [9]:

- The number of edges is about three times the number of vertices: $E \approx 3V$;
- The number of triangles is about twice the number of vertices: $F \approx 2V$; and
- The average number of edges or triangles incident to a vertex/point is 6.

The average number of incident edges to a point v is also called the average vertex degree or valence. As stated previously, the triangles in $U(\bar{f})_v$ likely provide the most reliable local neighborhood information. If the number of triangles in $U(\bar{f})_v$ is more than half of the average valence, the estimated normal vector at v based on the normals of the identified incident triangles is considered to be of high confidence. In other words, when the number of fully matched umbrella facets at v is equal to or greater than three, the normal vector N_v is to be estimated as a weighted average of the normal vectors of all the k triangles in $U(f)_v$:

$$N_v = \frac{\sum_{i=1}^{k} w_i n_i}{\sum_{i=1}^{k} w_i} \qquad w_i = M_{f_i} \cdot \angle p_i v p_{i+1} = M_{f_i} \cdot \cos^{-1}\left(\frac{\overrightarrow{vp_i} \cdot \overrightarrow{vp_{i+1}}}{\left|\overrightarrow{vp_i}\right| \left|\overrightarrow{vp_{i+1}}\right|} \right) \tag{3.5}$$

where n_i and w_i are the normal vector and weight of the ith triangle f_i in $U(f)_v$, respectively. The weight w_i is a product of the matching index M_{f_i} and $\angle p_i v p_{i+1}$ for f_i. When the number of fully matched umbrella facets at v is less than three, this means that the local neighborhood triangles at v cannot be reliably attained. More specifically, this is generally an indication that large geometric variation or data inconsistency is present in the neighborhood. To reduce the estimation errors introduced by the associated data uncertainty, the normal vector N_v is thus best estimated via the weighted plane fitting method by Pauly et al. [24] using all the neighboring points in the $DT(U)_v$ triangle set (all the red points in Fig. 1). Also, the estimated normal vectors should always be consistently oriented with one another on the same side of the surface (inside or outside). Finding a globally consistent orientation for the estimated normal vectors is not simple, especially for point clouds that are of low density or containing sharp features. In this work, the widely-used method proposed by Hoppe et al. [17] is employed to orient all the estimated normal vectors.

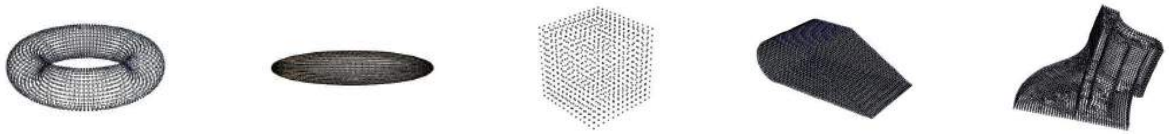
4 IMPLEMENTATION RESULTS AND DISCUSSION

Numerous case studies have been performed to validate the performance of the proposed normal vector estimation method. The computed results were analyzed and compared with those generated by the plane fitting (PF) method of Hoppe et al. [17], the weighted plane fitting (WPF) method of Pauly et al. [24] and the local Delaunay neighbors (LDN) method of OuYang and Feng [23].

4.1 Employed Point Cloud Data Sets

In order to compare the estimated normal vectors from different methods, it is necessary to quantify the deviation of an estimated normal vector from its true normal vector. For scanned point cloud data sets, such ideal reference normal vectors are not available. As a result, synthesized or simulated point cloud data sets, for which the reference normal vectors are known, have been employed. A few conditioned scanned point cloud data sets available in the Internet have also been employed in the comparison. The comparison is thus made on three different types of point clouds.

The first type is simulated point cloud data generated from parametric mathematical expressions. The true normal vector at each point is readily available. Two simulated point cloud data sets of uniform point distribution were generated for the Torus and Ellipsoid model, shown in Figs. 3(a) and 3(b), respectively. Since the local surface shape at a point on a smooth surface can be approximated by a quadric surface, the point can be considered as a parabolic, an elliptical or a hyperbolic point according to its curvature tensor [26]. All these three types of points can be found on a Torus surface. The point cloud from the Ellipsoid surface is examined as it contains high-curvature regions.



(a) Torus (b) Ellipsoid (c) Cube (d) SimulationSolid (e) Fandisk

Fig. 3: Simulated/Synthesized point cloud data sets.

The second type is also simulated point cloud data extracted from well-known idealized uniform-mesh models. The reference normal vector at each (mesh) point can be computed as an area-weighted average of the normal vectors of its local incident mesh triangles. Although the reference normal vectors in this type of point cloud data are not the true surface normal vectors as those in the first type, they are considered applicable reference normal vectors as the involved meshes are uniform and of high point resolution. The point clouds of Cube, SimulationSolid and Fandisk are members of this type and shown in Figs. 3(c), 3(d) and 3(e), respectively.

The third type is actual scanned point cloud data that have been conditioned and made available in the Internet. A proven mesh surface reconstruction algorithm [11] was first used to produce a triangle mesh surface for each point cloud data set. The area-weighted average of the normal vectors of the triangles incident to a point v was then taken as the reference normal vector at v. Such a calculated reference normal vector is not the same as the true normal vector but offers a reasonable approximation for the actual scanned point cloud data.

4.2 Results and Comparison

The error e of an estimated normal vector is quantified as the angle between the estimated normal vector N and the reference normal vector N_R as:

$$e = \cos^{-1}\left(\frac{N \cdot N_R}{|N||N_R|}\right) \tag{4.1}$$

The number of k-nearest neighbors $N_k(v)$ employed for the PF and the WPF method was set as 30 and 40, respectively, as recommended [32]. The computed results, including mean errors, standard deviations (variation about the mean error), and computational time, on the simulated point cloud data sets shown in Fig. 3 using the various methods are listed in Tab. 1.

Model		Torus	Ellipsoid	Cube	Simulation-Solid	Fandisk
No. of Points		3,600	9,950	866	6,988	6,475
Mean Error (radian)	Current	0.0011	0.0013	0.0312	0.0225	0.0179
	LDN	0.0059	0.0020	0.0753	0.0237	0.0255
	PF	0.0158	0.0041	0.1564	0.0730	0.1365
	WPF	0.0082	0.0034	0.1678	0.0702	0.1372
Standard Deviation (radian)	Current	0.0006	0.0017	0.1023	0.1236	0.0777
	LDN	0.0035	0.0013	0.1651	0.1209	0.0828
	PF	0.0087	0.0040	0.1532	0.1675	0.1669
	WPF	0.0101	0.0056	0.1548	0.1598	0.1609
Computational Time (sec.)	Current	2.94	5.67	1.44	5.10	4.51
	LDN	3.07	7.12	0.65	5.15	5.08
	PF	0.36	1.34	0.17	0.50	0.55
	WPF	0.51	2.64	0.23	0.58	0.82

Tab. 1: Comparison of normal vector estimation results.

Since the simulated point cloud data sets are noise-free, all the examined methods produced good normal vector estimates as indicated by the small mean errors and standard deviations in Tab. 1. It can be seen that the current method works better than the other three methods. The corresponding error maps are shown in Fig. 4, where blue indicates small e and red indicates large e. In the first row of Fig. 4, both PF and WPF generate large e at the top of the Torus. This is because the distribution of $N_k(v)$ for each point in this area is much biased. The fitted plane is thus not capable of approximating the ideal tangent plane well, which leads to relatively large e in this area than in other areas. Likewise, for a point in high-curvature areas in Ellipsoid, $N_k(v)$ do not lie consistently about a plane; hence, the fitted plane computed by PF or WPF cannot approximate the ideal tangent plane satisfactorily. For point clouds containing sharp features, such as Cube, SimulationSolid and Fandisk, the current method offers a clear advantage over the plane fitting methods, especially in areas around sharp features. This can be attributed to the reliable determination of local mesh neighbors by the current method.

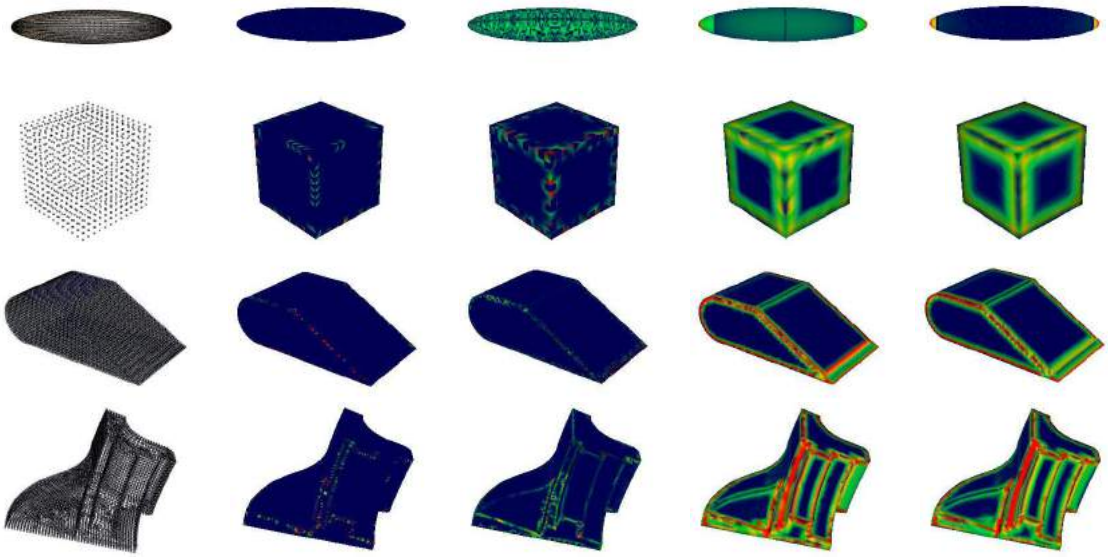
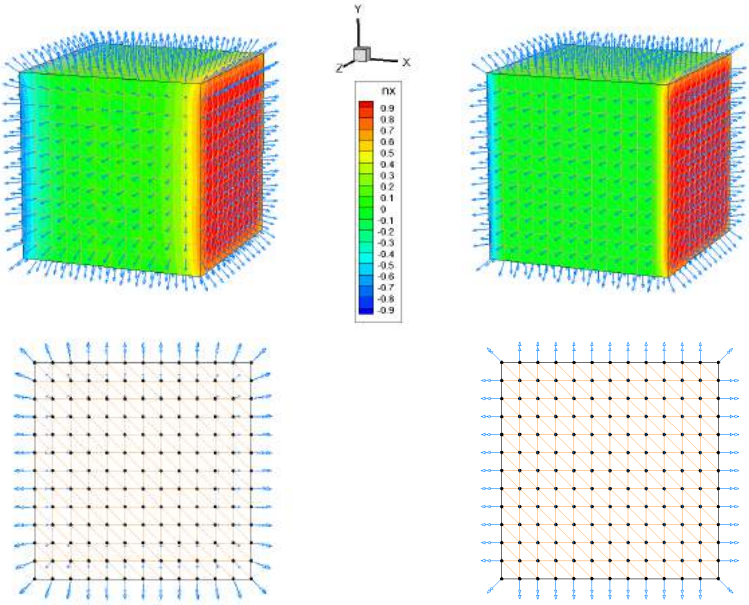


Fig. 4: Color maps of normal vector estimation errors for the simulated point clouds.



(a) WPF (b) Current

Fig. 5: Detailed comparison of the estimated normal vectors for the Cube point cloud.

Fig. 5 gives a more detailed analysis of the estimated normal vectors in areas near sharp features for the Cube point cloud data set. The normal vectors were estimated by the WPF method, shown in Fig. 5(a), and the current method, shown in Fig. 5(b). The top-row 3D color maps depict the direction cosine in the X-axis direction nx of the estimated normal vector at a point. So, if the estimated normal vector is exactly in the $+X$ direction, $nx = 1$ and the normal vector footing is marked in red. If the estimated normal vector is in the ïX direction, $nx = -1$ and its footing is marked in blue. For normal vectors perpendicular to the X axis, $nx = 0$ and their footing is marked in green. The bottom row shows the projected view of the estimated normals on one of the six Cube faces. It can be seen that the current method yields much improved normal vector estimations in areas near the sharp features.

Fig. 6 illustrates the normal vector estimation error maps for some scanned point cloud data set samples downloaded from the Internet. As stated previously, the reference normal vector at each point is not the true surface normal vector and is itself an estimate. Nonetheless, compared to the current method, the plots shown in Fig. 6 do provide a clear indication of the relatively large normal vector deviations of the existing methods in areas of high curvature or near sharp features.

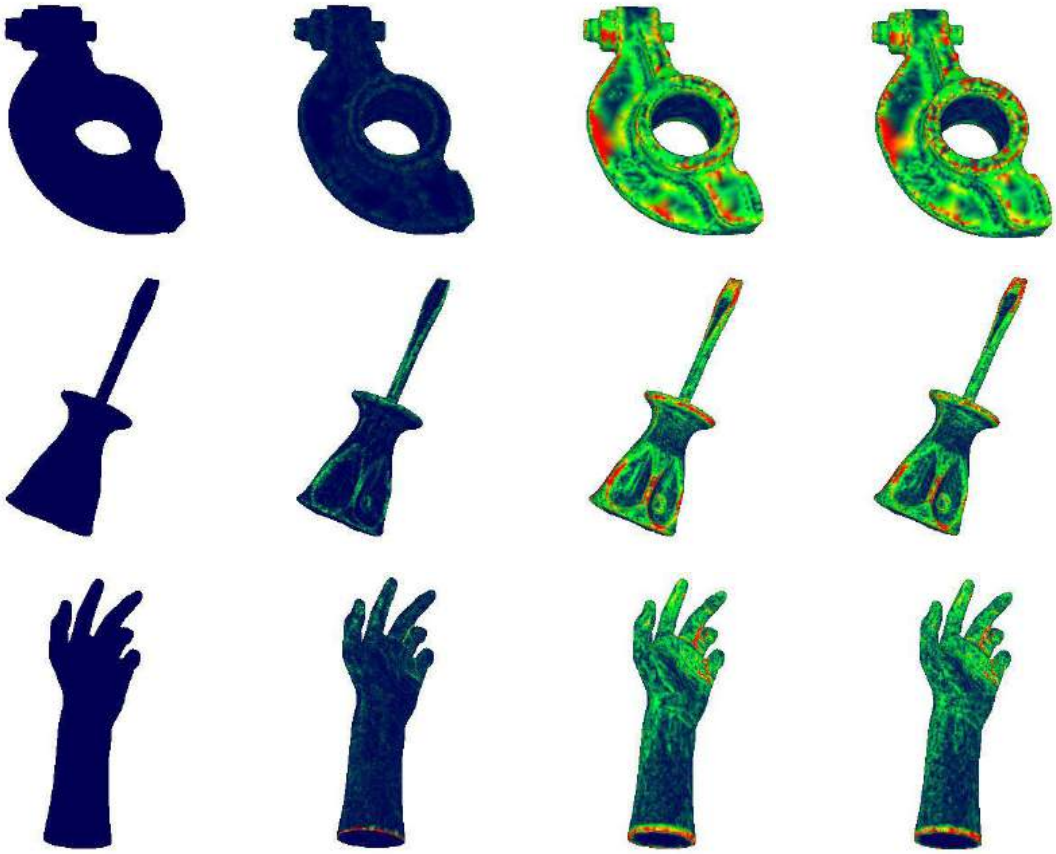


Fig. 6: Color maps of normal vector estimation errors for the scanned point clouds.

5 CONCLUSIONS

A new normal vector estimation method for point clouds has been presented in this paper. This new method uses the matching results of the local manifold Delaunay triangle mesh established at each point to reliably identify the neighboring points for calculating the normal vector. The well-identified neighboring points make it possible to produce reliable normal vector estimates, especially for points near sharp features. The calculation of the normal vector is in fact via a hybrid scheme. When the local mesh matching results at a point are deemed of high confidence (the number of fully matched triangles is equal to or more than three), the normal vector is to be approximated as a matching-based weighted average of the normal vectors of all the triangles in the local manifold mesh. Otherwise, large geometric variation or data inconsistency appears to be present and the normal vector is to be approximated via the existing weighted plane-fitting technique [24] from an increased number of neighboring points in order to reduce the estimation errors due to data uncertainty.

Unlike the existing method that is based on the similar idea of adaptively selecting the neighboring points for normal vector estimation, the current method uses local mesh matching to improve the reliability in identifying the neighboring points. This in turn leads to improved normal vector estimation accuracy. Compared with the existing plane-fitting methods, the current method yields much more reliable results, especially for points near sharp features, although increased computational time is involved. Accurate normal vector estimates for points near sharp features would greatly facilitate the sharp-feature preservation issue in surface reconstruction. The related work is underway in our research group and will be reported later.

ACKNOWLEDGEMENTS

This work was funded in part by the Natural Sciences and Engineering Research Council of Canada (NSERC). The data sets of Cube and Fandisk were obtained from the Surface Reconstruction Testbed webpage of Sophia-Inria; the data sets of SimulationSolid, Rocker-arm and Screwdriver were obtained from the website of AIM@SHAPE Shape Repository; and the data set of Hand was obtained from the website of RapidForm.

REFERENCES

[1] Adamson, A.; Alexa, M.: Ray tracing point set surfaces, Proceedings of Shape Modeling International, 2003, 272-279.

[2] Adamy, U.; Giesen, J.; John, M.: Surface reconstruction using umbrella filters, Computational Geometry: Theory and Applications, 21(1-2), 2002, 63-86. DOI: 10.1016/S0925-7721(01)00040-2

[3] Alexa, M.; Behr, J.; Cohen-Or, D.; Fleishman, S.; Levin, D.; Silva, C. T.: Point set surfaces, Proceedings of the conference on Visualization '01, San Diego, California, 2001, 21-28.

[4] Amenta, N.; Bern, M.: Surface reconstruction by Voronoi filtering, Discrete and Computational Geometry, 22(4), 1999, 481-504. DOI: 10.1007/PL00009475

[5] Amenta, N.; Kil, Y. J.: Defining point-set surfaces, ACM Transactions on Graphics, 23(3), 2004, 264-270. DOI: 10.1145/1015706.1015713

[6] Attene, M.; Spagnuolo, M.: Automatic surface reconstruction from point sets in space, Computer Graphics Forum, 19(3), 2000, 457-465. DOI: 10.1111/1467-8659.00438

[7] 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. DOI: 10.1109/2945.817351

[8] Boissonnat, J.-D.; Cazals, F.: Smooth surface reconstruction via natural neighbour interpolation of distance functions, Proceedings of the Sixteenth Annual Symposium on Computational Geometry, Hong Kong, 2000, 223-232. DOI: 10.1145/336154.336208
[9] Botsch, M.; Pauly, M.; Kobbelt, L.; Alliez, P.; Levy, B.; Bischoff, S.; Rössl, C.: Geometric modeling based on polygonal meshes, ACM SIGGRAPH 2007 Courses, San Diego, California, 2007, Course 23. DOI: 10.1145/1281500.1281640
[10] Coxeter, H. S. M.: Introduction to Geometry, 2nd ed., Wiley, 1989.
[11] Dey, T. K.; Goswami, S.: Tight cocone: A water-tight surface reconstructor, Proceedings of the 8th ACM Symposium on Solid Modeling and Applications, Seattle, Washington, 2003, 127-134. DOI: 10.1145/781606.781627
[12] Dey, T. K.; Goswami, S.: Provable surface reconstruction from noisy samples, Proceedings of the Twentieth Annual Symposium on Computational Geometry, Brooklyn, New York, 2004, 330-339. DOI: 10.1145/997817.997867
[13] Dey, T. K.; Li, G.; Sun, J.: Normal estimation for point clouds: A comparison study for a Voronoi based method, Proceedings of the Eurographics Symposium on Point-Based Graphics, 2005, 39-46. DOI: 10.1109/PBG.2005.194062
[14] Dey, T. K.; Sun, J.: An adaptive MLS surface for reconstruction with guarantees, Proceedings of the Third Eurographics Symposium on Geometry Processing, Vienna, Austria, 2005, 43-52. DOI: 10.2312/SGP/SGP05/043-052
[15] Gross, M.; Pfister, H.: Point-Based Graphics, Morgan Kaufmann, 2007.
[16] Hoffman, R.; Jain, A. K.: Segmentation and classification of range images, IEEE Transactions on Pattern Analysis and Machine Intelligence, 9(5), 1987, 608-620. DOI: 10.1109/TPAMI.1987.4767955
[17] Hoppe, H.; DeRose, T.; Duchamp, T.; McDonald, J.; Stuetzle, W.: Surface reconstruction from unorganized points, Proceedings SIGGRAPH '92, 1992, 71-78. DOI: 10.1145/133994.134011
[18] Huang, J.; Menq, C.-H.: Automatic data segmentation for geometric feature extraction from unorganized 3-D coordinate points, IEEE Transactions on Robotics and Automation, 17(3), 2001, 268-279. DOI: 10.1109/70.938384
[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. DOI: 10.1109/COASE.2010.5584049
[20] Milroy, M. J.; Bradley, C.; Vickers, G. W.: Segmentation of a wrap-around model using an active contour, Computer-Aided Design, 29(4), 1997, 299-320. DOI: 10.1016/S0010-4485(96)00058-9
[21] Mitra, N. J.; Nguyen, A.; Guibas, L.: Estimating surface normals in noisy point cloud data, International Journal of Computational Geometry and Applications, 14(4-5), 2004, 261-276. DOI: 10.1142/S0218195904001470
[22] O'Rourke, J.: Computational Geometry in C, Cambridge University Press, 1994.
[23] OuYang, D.; Feng, H. Y.: On the normal vector estimation for point cloud data from smooth surfaces, Computer-Aided Design, 37(10), 2005, 1071-1079. DOI: 10.1016/j.cad.2004.11.005
[24] Pauly, M.; Keiser, R.; Kobbelt, L. P.; Gross, M.: Shape modeling with point-sampled geometry, ACM Transactions on Graphics, 22(3), 2003, 641-650. DOI: 10.1145/882262.882319
[25] Schaufler, G.; Jensen, H. W.: Ray tracing point sampled geometry, Proceedings of the Eurographics Workshop on Rendering Techniques 2000, 2000, 319-328.
[26] Struik, D. J.: Lectures on Classical Differential Geometry, 2nd ed., Addison-Wesley, 1961.
[27] Veltkamp, R. C.: 3D computational morphology, Computer Graphics Forum, 12(3), 1993, 115-127. DOI: 10.1111/1467-8659.1230115
[28] Veltkamp, R. C.: Closed Object Boundaries from Scattered Points, Springer-Verlag, 1994.

[29] Woo, H.; Kang, E.; Wang, S.; Lee, K. H.: A new segmentation method for point cloud data, International Journal of Machine Tools and Manufacture, 42(2), 2002, 167-178. DOI: 10.1016/S0890-6955(01)00120-1

[30] Yang, M.; Lee, E.: Segmentation of measured point data using a parametric quadric surface approximation, Computer-Aided Design, 31(7), 1999, 449-457. DOI: 10.1016/S0010-4485(99)00042-1

[31] Yau, H.-T.; Kuo, C.-C.; Yeh, C.-H.: Extension of surface reconstruction algorithm to the global stitching and repairing of STL models, Computer-Aided Design, 35(5), 2003, 477-486. DOI: 10.1016/S0010-4485(02)00078-7

[32] Zurich, E., http://graphics.ethz.ch/pointshop3d/, Pointshop3D.