Direct quad-dominant meshing of point cloud via global parameterization

Er Li, Wujun Che, Xiaopeng Zhang*, Yi-Kuan Zhang, Bo Xu
LIAMA-NLPR-Digital Content Technology Research Center, Institute of Automation, CAS, China

A R T I C L E   I N F O
Article history:
Received 10 December 2010
Revised 10 December 2011
Accepted 14 March 2011
Available online 23 March 2011

Keywords:
Point cloud
Global parameterization
Quad-dominant meshing
Surface reconstruction

A B S T R A C T
In this paper, we present a new algorithm for quad-dominant meshing of unorganized point clouds based on periodic global parameterization. Our meshing method is guided by principal directions so as to preserve the intrinsic geometric properties. We use local Delaunay triangulation to smooth the initial principal directions and adapt the global parameterization to point clouds. By optimizing the fairness measure we can find the two scalar functions whose gradients best align with the guided principal directions. To handle the redundant vertices in the iso-lines due to overlapped triangles, an approach is specially designed to clean the iso-lines. Our approach is fully automatic and applicable to a surface of arbitrary genus. We also show an application of our method in curve skeleton extraction from incomplete point cloud data.

1. Introduction

Quad-dominant meshing of a 3D model is preferred over triangulation in many applications, such as texturing, simulation with finite elements and B-spline fitting. Compared with a triangle mesh, a quad mesh is a more compact and more efficient representation of 3D shape surfaces. Most of the existing works on quadrangulation can produce high quality quad meshes, but they exclusively operate on triangle meshes. However, in recent years point cloud has been used more and more widely in CAD and computer graphics communities due to the availability of fast and accurate laser scan devices. Many works focus on reconstructing a triangle mesh from a point cloud but few consider the quadrangulation of a point cloud directly.

Indeed, an immediate method is first performing a surface reconstruction and then applying mesh based quadrangulation methods. However, for those methods based on implicit form, the final triangle surface is obtained by meshing the isosurface. The drawback is that the point in initial data is not preserved during the meshing; in reverse engineering this may induce unnecessary error when using the reconstruction results for spline fitting to the point cloud. And those methods based on computer geometry which strive for an exact meshing of a point cloud often suffer from noise and data missing. The time burden of surface reconstruction also poses a significant obstacle when handling large data or animation point cloud data. To overcome the weakness of these indirect methods, we intend to directly quadrangulate a point cloud by skipping the step of global triangulation.

Although point cloud is a widespread geometry model representation, it is difficult to quadrangulate it for further applications. In this paper, we propose a method to generate feature aligned quadrilateral meshes directly from point cloud data. Our method is based on periodic global parameterization [15] but need not the input model to be a triangle mesh. We show that local Delaunay triangles is sufficient to smooth the principal direction field and define gradient of the scalar function. However, it is challenging to extract iso-lines and construct quad-dominant meshes from these local triangles. Thus, we develop an approach to clean the iso-lines efficiently. By applying the whole algorithm to several sets of registered raw scanned data, we show that our method is robust to noise and can deal with non-uniformly distributed samples.

Compared with the initial method in [15], the major difference of our work is that we can handle point cloud without global connectivity, even with noise and non-uniformly distributed samples. In addition, their iso-line processing method is not suitable for our case. Alternatively, we develop a robust method to remove unnecessary connections in the iso-lines caused by local triangles. In particular, we show that our method produces better quad-dominant meshes on registered point scans than original periodic global parameterization applied to quadrangulation for a reconstructed mesh. Beside adapting the method to quad-dominant meshing of point cloud models, we also present how to utilize our algorithm for point cloud skeleton extraction.

The contributions of our work can be summarized as follows: (1) we develop a method to use local Delaunay triangulation to smooth the cross field and to compute parameterization; (2) we...
propose a new way for cleaning the iso-lines to overcome the overlapped triangles caused by local triangulation; (3) we present a novel application of our algorithm in curve skeleton extraction from incomplete point cloud data.

1.1. Related work

Much work has been done on quadrangulation, especially on triangular meshes, because of their wide appeal in various application fields. More details can be found in [2]. Here we only review those work related to our research.

Alliez et al. [1] integrate the lines of curvature on a triangle mesh in the parameter domain and generate a quad-dominant mesh by intersecting these lines of curvature. Marinov and Kobbelt [14] improve this method by directly integrating the curvature lines on the input model so that a surface of arbitrary genus can be handled. Dong et al. [8] build a harmonic scalar field and the gradient of this field provides a smooth vector field which can be used for quad-dominant remeshing. This method generates more regular results at sacrifice of feature alignment.

Another efficient way of quadrangulation is to decompose a mesh of complex shape into several patches, and then convert these patches into quadrilaterals. Boier-Martin et al. [3] propose a clustering-based method to decompose the surface. Dong et al. [7] construct the Morse–Smale complex of Laplacian eigenfunctions to form a quadrangular base mesh. The functions distribute extremum evenly across a mesh and thus the final quadrilateral mesh is well-shaped with only a few singularities. However, the feature alignment is not ensured. Ref. [18] uses a singularity graph to control the alignment and designs quadrangulations with discrete harmonic forms to create quads. Based on Dong's work, Huang et al. [11] propose a controllable spectral method to parameterize a triangle model. Then a quadrilateral mesh can be obtained by intersecting these lines of curvature. Kobbelt [14] improve this method by directly integrating the curvature lines on the input model so that a surface of arbitrary genus can be handled. Dong et al. [8] build a harmonic scalar field and the gradient of this field provides a smooth vector field which can be used for quad-dominant remeshing. This method generates more regular results at sacrifice of feature alignment.

Global parameterization has proven to be a useful tool for designing quadrangulations. Ray et al. [15] propose periodic global parameterization (PGP) guided by principal directions to parameterize a triangle model. Then a quadrilateral mesh can be acquired by tracking the isolines in the parameter domain. This method generates high quality quad-dominant meshes fully automatically without any user's interaction. In [12], a given frame field is converted into a single vector field on a branched covering space. Then the surface is cut and a global parameterization guided by the frame field is calculated to produce a quadrilateral mesh. Recently, [4] reduces the direction field smoothing and global parameterization to a mixed-integer optimization problem, and their mixed-integer solver produces quadrilateral meshes with fewer singularities than previous approaches.

A new way of quadrangulation is simplification algorithms targeting quad meshes. Ref. [6] proposes an edge-collapsed method to produce level-of-detail quad structure which can also be used to convert triangle meshes into quad only meshes. In [17], local operations are used to incrementally simplify quad mesh, based on the local operations an original Triangle-to-Quad conversion algorithm is developed to obtain a quad mesh from a given triangle mesh. Simplification based algorithms provide a more easy way for quadrangulation but still have limitation on alignment of edges to curvature directions.

While all of the above methods are performed on mesh data, the research about how to design quadrangulations on point cloud is quite few. The most related work is by [13] to extract lines of curvature from noisy point cloud; the lines can be further used for direct quad-dominant mesh reconstruction. Their work can be viewed as an extension of the approach in [14] to point cloud. This method requires an explicit integration for placing a set of streamlines, thus the final quads are hard to be evenly distributed over the entire surface.

2. Background

We extend Ray's parameterization [15] to an unorganized point cloud, and our work will focus on how to modify Ray's method to make it suitable for a point could. To make a better understanding of our work, we first give a simple introduction to PGP and then we will discuss details about our algorithm.

PGP was first proposed in [15] to construct a global parameterization such that the gradients of scalar functions for the parameterization are aligned with two guided vector fields. The main problem of previous global parameterization is that to reduce the distortion of parameterization one has to cut the initial mesh into several disk-like pieces and that this operation induces discontinuities across the cut. PGP solves this problem by treating each triangle as a chart and then defining transition function between two adjacent triangles. The transition function must satisfy two conditions to eliminate the discontinuities: first, the coordinates of translation vectors should be integer multiples of \(2\pi\); second, the rotation angles are constrained to be multiples of \(\pi/2\). As stated in [15], for a triangle surface the global parameterization that align with principal direction field is reduced to minimize an orientation energy:

\[
E = \sum_T (\|\nabla \theta - \omega_i \|_2^2 + \| \nabla \phi - \omega_i \|^2_2) A_T
\]  

where, \(\theta, \phi\) denote 2D parameter coordinates, \(A_T\) is the area of triangle \(T\) and \(\omega\) is a user-defined parameter that controls the size of final quad mesh. The first part of above equation can be further expressed into a more explicit formulation:

\[
E = \sum_{ij} (\theta_{ij} - \theta_{ij} - \bar{K} \cdot \bar{e}_{ij})^2
\]

where \(ij\) is an edge of triangle \(T\). In fact, the energy is edge based and in [15] a triangle based energy is used to avoid bias due to anisotropic triangulation.

For a point cloud, we find a new way to compute the triangle based energy by constructing an edge based energy through the connections only from local Delaunay triangulation. The energy obtained by this way has no difference from the triangle based energy used for mesh, and in our case anisotropic sampling is not a problem since every triangulation is local. And the local Delaunay triangulation is also a useful tool for the processing of principal direction field. All of these pave the way for a quad-dominant meshing algorithm of point cloud.

3. Quad-dominant meshing via global parameterization

We first present the local Delaunay triangulation in Section 3.1. Then in Sections 3.2 and 3.3, respectively, we will describe the details about how to process the principal direction field and construct the parameterization using local Delaunay triangulation. Finally, Section 3.4 tells how to handle undesirable points in the isolines.

3.1. Local Delaunay triangulation

We notice that the global parameterization is based on a set of charts which only come from the local geometry feature, a local Delaunay triangulation may approximate the discrete operators. For every point \(p\) in the cloud, we define a tangent space
according to its normal estimated before. This tangent space provides an approximate of the points in a small neighborhood \( N \) of \( p \). Usually, \( N \) is defined as the \( k \) nearest points of \( p \).

Once the neighbor set \( N \) is defined, we project all points of \( N \) onto the tangent plane \( T_p \) and let \( \bar{N} \) represents the projection points of \( N \). Building the Delaunay triangulation of \( N \) on \( T_p \) yields a 2D triangle mesh and induces its corresponding 3D triangulation on \( N \). Those triangles incident to \( p \) are selected to participate in global parameterization. For every point we store its incident triangles and in the following steps what we call edges and triangles are both defined from the local triangles. The local Delaunay triangulation does not guarantee the overall validity of topological structure since overlapped triangles do exist. The local triangles are used as local charts to perform parameterization and extract iso-lines. The problem of overlapped triangles will be tackled by cleaning the iso-lines; more details are in Section 3.4.

3.2. Global smoothing of principal direction field

Usually, in an anisotropic region we can get relatively accurate principal directions. However, the principal directions are usually ill-defined in an isotropic region and thus the corresponding estimation is meaningless in practice. To obtain a smooth vector field well defined everywhere of the point cloud, we use a global energy minimization to smooth the principal direction vectors. Before the smoothing, we use the method presented in [13] to estimate initial curvature tensor and reject the outliers. After that, similar to the work in [10], we define the energy function measuring smoothness of principal direction field on a point cloud as

\[
E(x_i) = (1 - \rho) \sum_{kj} \sin^2(\alpha_i - \alpha_j^k) - E_{\text{smoothing}}
\]

where \( \alpha_i \) is the unknown angle between the principal directions \( K_i \) and a reference direction \( H_i \) in the tangent plane of the vertex \( i \), and \( \beta_j \) represents the angle between the projection of edge \( e_j \) into the tangent plane and the reference direction \( H_i \). The user-defined parameter \( \rho \) indicates the smooth intensity. The smoothing not only smoothes and simplifies the principal direction field, but also guarantees the global geometry of the point set surface. Fig. 1 shows the principal direction field before and after smoothing. Significant improvement can be observed in the flat region.

Curl correction is then adopted as presented in [15] using local triangles as an alternative of the triangles defined in a manifold surface mesh.

Actually, the key insight is that both the smoothness and the curl of principal direction field is defined through the variance of direction in a local area. So a local triangulation is sufficient to process the guided direction field. This is also true when defining the gradient of parameterization value on each point, thus building the energy function from local triangulation is straightforward.

3.3. Edge-based energy function

The energy is discretized into each edge by projecting the direction vector \( K \) to the edge \( e_j \) and calculating the gradient of \( \partial \) and \( \phi \) along this edge. Assume that \( \hat{e}_j \) is normalized, then the energy based on this edge can be expressed as

\[
E_j = (2\pi \theta - \hat{e}_j - \hat{e}_j)^2
\]

where \( 2\pi \theta \) is introduced to guarantee the translational invariance. This mixed integer problem is avoided by taking the sine and cosine of Eq. (3). Then \( \theta \) is replaced with \( \cos \theta \) and \( \sin \theta \), and the above formula is approximated as

\[
E_j = \left\| U_j - \left( \frac{\cos(K \cdot \hat{e}_j)}{\sin(K \cdot \hat{e}_j)} \right) U_j \right\|^2
\]

Once the neighbor set \( N \) is defined, we project all points of \( N \) onto the tangent plane \( T_p \) and let \( \bar{N} \) represents the projection points of \( N \). Building the Delaunay triangulation of \( N \) on \( T_p \) yields a 2D triangle mesh and induces its corresponding 3D triangulation on \( N \). Those triangles incident to \( p \) are selected to participate in global parameterization. For every point we store its incident triangles and in the following steps what we call edges and triangles are both defined from the local triangles. The local Delaunay triangulation does not guarantee the overall validity of topological structure since overlapped triangles do exist. The local triangles are used as local charts to perform parameterization and extract iso-lines. The problem of overlapped triangles will be tackled by cleaning the iso-lines; more details are in Section 3.4.

The energy \( E_j \) with respect to \( \phi \) is obtained in the same way by replacing \( K \) with its orthogonal direction \( K^\perp \). When the principal directions defined on the two end points of one edge are inconsistent, rotational invariance is required to allow for switches between iso-\( \partial \)’s and iso-\( \phi \)’s. Rotational invariance is added to Eq. (4) by swapping the \( \partial \)’s and \( \phi \)’s according to the direction variance along this edge. Details can be found in [15]. For the point cloud, the normals on the two end points of one edge may be not consistent, so we need to reorient them before determining the direction variance.

The optimization of Eq. (4) is a quadratic minimization problem, however \( U_i \) must satisfy condition \( \|U_i\|^2 = 1 \). We introduce this constraint in the system by adding a weighted term into

\[
\text{Fig. 1. Comparison of principal direction field before and after smoothing. The top figure in (a) shows the initial estimated principal direction and (b) is its zoom-in view. The bottom figure in (a) and (c) show the smoothing results. Attention to the improvement of direction field in the flat region.}
\]
account, the parameterization is constructed by minimizing the following function:

\[ E = \sum_{T} (E_{ij}^0 + E_{ij}^1) + \omega \sum_{T} ((|U|_1^2 - 1)^2 + (|V|_1^2 - 1)^2) \]

(5)

here \( V_i = (\cos \phi, \sin \phi)^T \). We empirically set \( \omega = 0.01 \) in our experiments. We solve the above problem using Newton’s algorithm, the initial value of \( UV \) is set to 0 except for one randomly picked point with initial value \( U = (1,0), V = (1,0) \).

3.4. Extracting iso-lines

To extract iso-lines aligning best with the principal directions, we first reconstruct a local parameterization on each triangle from those \( \sin \theta, \cos \phi \) variables obtained by solving Eq. (5). The basic idea is that for each triangle \( T \) we determine the translational and rotational degrees of freedom by propagating along each edge [15]. Then the individual segments defined in \( T \) are computed by intersecting the triangle \( T \) with iso-lines \( \theta = 2k_1 \pi \) and \( \phi = 2k_2 \pi \). Here the iso-values are decided as follows:

\[ 2k_1 \pi \in \left[ \frac{\min(\theta)}{\max(\theta)} \right], \quad k_1 \in \mathbb{N} \]

\[ 2k_2 \pi \in \left[ \frac{\min(\phi)}{\max(\phi)} \right], \quad k_2 \in \mathbb{N} \]

However, different from that in [15], extracting and processing iso-lines are more complex since we only have local Delaunay triangulation information. In a triangle surface, each edge is shared by two adjacent triangles and the transition function between them ensures that the end of the iso-lines match. However, for local triangulation, some overlapped triangles may exist and thus each edge may be shared with more than two triangles. As a result there may be loops and redundant intersections. In the iso-lines graph which induce undesirable quads in the quad-dominant meshing step. So several steps are required to clean the iso-lines graph. Notice that only the vertices produced by intersections remain as the vertices of quad-dominant mesh, and the other vertices can be removed without any problem as long as correct connections between intersection vertices are preserved. The cleaning proceeds as follows:

1. Remove dangling segments. This is done by recursively removing edges with endpoint of degree one. Dangling segments appear in three cases: near the border, overlapped triangles and singularities of parameterization. Fig. 2(a) depicts the three cases and a detailed description of the overlapped triangles case. \( \Delta ABC \) is among the local triangles but \( \Delta ACD \) is not. So the segment in \( \Delta ABC \) is a dangling segment. After the removing step, each vertex in the iso-line has degree more than one and no singularities exist Fig. 2(b).

2. Merge redundant intersections. Due to the overlapped triangles, two intersections may be very close. As shown in Fig. 3, \( \Delta ACD \) and \( \Delta BCD \) overlap. \( V_i \) is produced by intersection of segments in \( \Delta BCD \) and \( V_j \) is produced by intersection of segments in \( \Delta ACD \). In this case we merge \( V_i \) and \( V_j \) into their middle point. Specifically, for each intersection vertex, if the distance to its nearest intersection vertex is smaller than a certain threshold, then these two intersection vertices are assumed to be candidate points to be merged. Usually, the threshold is selected as half of the distance to its second nearest intersection vertex.

3. Remove irregular vertices. Due to the overlapped triangles, there may be a few irregular vertices, i.e., vertices have degree more than two but are not intersection points. As shown in Fig. 2(c), \( \Delta ABC \), \( \Delta ACD \) and \( \Delta ABD \) are all among the local triangles, and thus two irregular vertices \( V_i \) and \( V_k \) arise. Irregular vertices cause trouble when traversing the iso-line graph. To remove them, for each edge in the iso-line, if one of the end points is irregular vertex, we collapse this edge from this end point to another end point. Dangling segments produced by collapse are also removed at the same time. This operation is iteratively performed until no edges need to be processed as shown in Fig. 2(d).

![Fig. 2](image1.png)

**Fig. 2.** Dangling vertices and irregular vertices. (a) Three kind of dangling vertices, shown in red, blue, green points respectively. Dangling vertices produced by overlapped triangles are shown in detail in the black box area. (b) After removing dangling vertices. (c) illustrates irregular vertices (red points) and in (d) irregular vertices are removed through edge collapse. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

![Fig. 3](image2.png)

**Fig. 3.** Merging two intersection vertices produced in overlapping triangles. (a) Red points are intersection vertices and the segments incident to them are shown in green and red respectively. (b) The intersection vertices in (a) are replaced through a new point (red) through a merging operation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)
met again. Then a mesh is created and the half-edges are marked as visited. Here the half-edges starting from each point \( p \) is sorted by a clockwise sequence of their projection on the tangent plane of \( p \). The operations are repeated until all half-edges are visited. Notice that globally consistent normals of the point cloud are necessary to traverse the half-edges. We arbitrarily choose an orientation of the normal at a starting point in the iso-line, and then propagate this orientation to its neighboring points iteratively.

Due to the orthogonality of principal directions, the resultant meshes show high quality and most of the polygons are nearly squares. But some non-quadrilateral polygons could exist near the singularities of direction field. We tessellate them with a few quads and triangles. In some applications, however, a pure quadrilateral mesh is required, so we further split the quad-dominant mesh using catmull-clark subdivision to eliminate the triangles, as shown in Fig. 10.

4. Results and applications

In this section, we demonstrate results of quad-dominant meshing of point cloud and also present an application of extracting skeleton from incomplete point cloud using the iso-lines produced by our algorithm. Our algorithm is implemented in C++. We use the CGAL library [5] to perform the local Delaunay triangulation and store the incident edges and triangles for each point. During the process of solving the sparse linear equation, the OpenNL linear solver in CGAL is used and proved to be efficient to handle all the equations produced by our experimental models.

4.1. Quad-dominant meshing of point cloud

We first perform our algorithm directly on the rock-arm model as a point cloud, and then compare it with periodic global parameterization method (PGP) as a triangle mesh as shown in Fig. 5. Fig. 5(b) shows the uncleaned iso-lines, except for degree 4 vertex produced by intersection of iso-lines segments, a few degree 3 vertices also exist due to the overlapped local triangles. Despite the existence of those irregular vertices, our cleaning process still guarantees a correct reconstructed quad-dominant mesh. Here we use curl correction for both methods so the edge length varies across the surface which is reflected in the histogram of edge length distribution. It can be seen that even our method only utilizes the point information without an explicit construction of global manifold mesh, the final result has the same quality as PGP. A quantitative evaluation is provided in Fig. 5(e) and (f). Here the result quad-dominant mesh of PGP is produced by Graphite [9].

On the other hand, our algorithm shows advantage over PGP for a point cloud since an absolute manifold mesh is unnecessary in our method. Fig. 9(a) is a reconstructed triangle mesh from the point cloud of a door model in Fig. 10. Due to the noise, non-manifold triangles and small holes exist, incurring undesired holes in the quad-dominant meshes as in Fig. 9(b). Our local triangulation is insensitive to non-manifold triangles and the iso-line cleaning step guarantees the right result in Fig. 9(c).

---

**Fig. 4.** A pipeline for cleaning iso-lines. (a) is initial iso-lines, in (b) dangling segments are removed, in (c) the two redundant intersections in the up left corner are merged and then irregular vertices in the iso-lines are removed in (d). (e) is the final reconstructed quad-dominant meshes. Intersection vertices are shown in red, and irregular vertices are shown in yellow, green points are dangling vertices. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

**Fig. 5.** Comparison of remeshing results of the rock-arm model. (a) Original model. (b) Iso-lines produced by our point cloud based method. The iso-\( \theta \) and iso-\( \phi \) lines are illustrated by red and yellow lines respectively, blue points show the vertices with degree more than two. (c) and (d) are the corresponding quad-dominant mesh results produced by our method and [15], the numbers of triangles and quads are (104,3683) and (92,3207) respectively, (e) and (f) depict the edge length and angle distribution about the quad-dominant mesh above. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

---

E. Li et al. / Computers & Graphics 35 (2011) 452–460
Since noise and overlap are common in real range scans, it is not easy to reconstruct a manifold triangle surface for PGP. Our algorithm eliminates this obstacle by skipping the triangle reconstruction step.

To further demonstrate the robustness of our method, we apply our algorithm to noisy and non-uniform sampling point clouds. In Fig. 6(a) the points in the upper part and lower part are sampled in different rate and in Fig. 6(c) 1% Gaussian noise is added. In both cases, our method gives good results, as shown in Fig. 6(b) and (d).

More experiments on real world point clouds are shown in Figs. 7 and 10. The first row in Fig. 7 is the raw scanned point cloud data and the second row is an ancient tower data obtained by image based reconstruction method from a series of pictures. In Fig. 10, our algorithm still works well in more complex point cloud data with noise and non-uniformly distributed samples by merging several raw scanned data. The first row is a noisy chair model, and the left three ones are all with non-uniform points due to overlaps. Our algorithm produces well shaped quad meshes for all models. During the quad meshing step, small holes of missing data are automatically filled. Then the resultant quad-dominant meshes by our approach are converted to pure quadrilateral meshes using cartmull-clark subdivision.

**Parameters:** The key parameter required for point cloud based PGP is the number of nearest neighbors $k$ used in local Delaunay triangulation. In our experiments we select $k = 20$. Though we only use a 1-ring triangles for the parameterization, this choice works well in most cases. A larger $k$ may be required when the point cloud is sparse and extremely non-uniformly sampled. For the models in Fig. 10, $k$ is chosen to be 30 for the chair, kitten and human body models to overcome the non-uniformly distributed samples in registered scanned data, and 35 for the last car door model.

**Time and complexity:** Compared with triangle mesh based parameterization, the additional time cost of our method mainly lies in the step of local Delaunay triangulation. The cleaning of iso-lines is negligible in the total time cost. Since $k$ is a constant value, the kNN search requires $O(V \log V)$ time and local Delaunay triangulation requires $O(V)$ time, here $V$ is the point number of the point cloud. Another time cost comes from solving the sparse symmetric system. The edges we used to construct the linear solver is from the local Delaunay triangulation, overlapping

![Fig. 6. The presented algorithm is robust to noise and non-uniform sampling. (a) The lower part of the bottle is 50% sampled from the original uniform sampling model and (c) 1% Gaussian noise is added.](image)

![Fig. 7. Our quad-dominant meshing method on raw point cloud data from laser scan device and image based reconstruction.](image)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Mesh statistics</th>
<th>Run time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$N_p$</td>
<td>$N_e$ (K)</td>
</tr>
<tr>
<td>Rock-arm</td>
<td>14413</td>
<td>62</td>
</tr>
<tr>
<td>Fertility</td>
<td>13971</td>
<td>42</td>
</tr>
<tr>
<td>Tooth</td>
<td>21947</td>
<td>66</td>
</tr>
<tr>
<td>Kitten</td>
<td>39677</td>
<td>105</td>
</tr>
<tr>
<td>Chair</td>
<td>209499</td>
<td>760</td>
</tr>
<tr>
<td>Human</td>
<td>329620</td>
<td>1130</td>
</tr>
<tr>
<td>Door</td>
<td>387742</td>
<td>1240</td>
</tr>
<tr>
<td>Tower</td>
<td>455413</td>
<td>1750</td>
</tr>
</tbody>
</table>
triangles cause more edges than a manifold triangle mesh. Through our experiments, the increasing of edge number is marginal and the average number of 1-ring neighbors for each point does not exceeds 6 as shown in Table 1. Table 1 gives the time cost of the main steps of our algorithm without code optimization on a PC of 2.40G Intel QuadCore 2 CPU with 4 GB memory. The statistics show that the most time-consuming steps are processing of principal direction field and global parameterization. Comparing with this, the time spent on local Delaunay triangulation is only a small part of the total time cost.

Limitations: Like most point cloud processing algorithms, our method suffers from close-by structures. The local Delaunay triangulation of k-nearest neighbors may cause topological errors by connecting two nearby surfaces. We plan to investigate a better alternative of k-nearest neighbors to overcome this problem in our future work. Another limitation of our method is that a few triangles still appear in the resultant quad-dominant meshing result which inherits from PGP [15]. The state-of-the-art algorithms [4,19] produce pure quadrilateral mesh with less singularities, which may be promising directions of the future work.

4.2. Curve skeleton from iso-lines

The skeleton of a 3D model has been used in various applications since it effectively represents the shape abstractions and much work has been done on mesh skeletonization. Recently, algorithms of extracting curve skeletons from incomplete point cloud have been developed to overcome the problems caused during the surface reconstruction when handling point cloud with missing data. In [16], generalized rotational symmetry axis (ROSA) is used to extract skeletons from point cloud composed of generally cylindrical regions.

Based on the observation that the orientation of an optimal cutting plane can be approximated by principal directions for a cylindrical shape, we directly compute skeletal points from the iso-lines. For each intersection vertex of the iso-line we only need to select one optimal cutting plane orientation from the two principal directions at this point (\(\vec{K}\) and \(\vec{K}\)’). Specifically, taking \(\vec{K}\) for example, we determine the relevant neighborhood for the cutting plane corresponding to \(\vec{K}\), i.e., the plane cross the intersection vertex with normal direction \(\vec{K}\), and calculate the variance of angles between point normals from the neighborhood and \(\vec{K}\). The variance corresponding to \(\vec{K}\)’ is calculated in the same way. Then the direction with smaller variance is chosen as the orientation of the optimal cutting plane. Next, starting from this intersection vertex, we traverse the iso-lines along its orthogonal direction and utilize the vertex encountered to compute the corresponding center of rotational symmetry. Then we take the measure in [16] to handle joint and extract skeleton.

We show our result in Fig. 8. The incomplete point cloud is obtained by merging several virtual scanned data as done in [16]. Although almost half of the data is missing, our method can extract reasonable curve skeletons from the iso-lines. Here we only use the point in the iso-lines, more accurate position of the center of rotational symmetry can be obtained by increasing the density of iso-lines or utilizing all the points in the relevant neighborhood.

![Fig. 8. Application of our method in curve skeleton extraction from incomplete point cloud data.](image)

![Fig. 9. Comparison of quad-dominant meshes produced by PGP and our method on a non-manifold mesh. From left to right are the mesh with non-manifold faces in the black region, the quad-dominant mesh of PGP with large holes, and the quad-dominant mesh by our method. Borders are highlighted in red lines. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)](image)
5. Conclusion and future work

We have presented an algorithm for reconstructing quad-dominant meshes from unstructured point sets. The reconstruction aims to find two scalar functions that optimize the alignment between their gradients and a guided direction field. Local Delaunay triangulation is used to smooth the direction field and design parameterization on discrete point cloud data. An iso-line extracting method for point cloud is also introduced. The whole process is fully automatic and the final result is of high quality. Besides the limitations described in Section 4.1, some further considerations are in our future work. A robust statistical detection of the crest lines may help to preserve the geometry feature more effectively. Additionally, we notice that even after smoothing there are still some triangles in the final quad-dominant meshes. How to control the position of singularities...
to reduce the number of irregular meshes is also worth investigating in the future. We also would like to compare a quad-
dominant reconstruction of one of those clay models of Pixar's characters using our proposed techniques, versus the Pixar artists
and engineer’s mesh they produce from the clay model.

Acknowledgments

The authors thank anonymous reviewers for their constructive comments. We also thank Bruno Lévy for the discussions and
thank AIM@SHAPE for providing the datasets. Thanks to Yihong
Wu for the data tower in Fig. 7. This work is supported in part by
National Natural Science Foundation of China under Nos.
60970093, 60902078, and 60872120; and in part supported by
French Systematic Paris-Region (CSDL Project).

References

’04: Proceedings of the computer graphics and applications, 12th Pacific
[16] Tagliasacchi A, Zhang H, Cohen-Or D. Curve skeleton extraction from
[18] Tong Y, Alliez P, Cohen-Steiner D, Desbrun M. Designing quadrangulations
with discrete harmonic forms. In: Symposium on geometry processing.