

# Mesh Reconstruction by Meshless Denoising and Parameterization

Lei Zhang<sup>1†</sup>, Ligang Liu<sup>2‡</sup>, Craig Gotsman<sup>3§</sup>, Hua Huang<sup>1¶</sup>

<sup>1</sup>School of Electronics and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China

<sup>2</sup>Department of Mathematics, Zhejiang University, Hangzhou 310027, China

<sup>3</sup>Department of Computer Science, Technion-Israel Institute of Technology, Haifa 32000, Israel

**Abstract**—We present a new approach to simultaneously denoise and parameterize unorganized point cloud data. This is achieved by minimizing an appropriate energy function defined on the point cloud and its parameterization. An iterative algorithm to minimize the energy is described. The key ingredient of our approach is an “as-rigid-as-possible” meshless parameterization to map a point cloud with disk topology to the plane without building the connectivity of the point cloud. Then 2D triangulation method can be applied to the planar parameterization to provide triangle connectivity for the 2D points, which can be transferred back to the 3D point cloud to form a triangle mesh surface. We also show how to generalize the approach to meshes with closed topology of any genus. Experimental results have shown that our approach can effectively denoise the point cloud and our meshless parameterization can preserve local distances in the point cloud, resulting in a more regular 3D triangle mesh, compared to other methods.

**Keywords**—point clouds; denoising; meshless parameterization; triangulation; surface reconstruction

## 1. INTRODUCTION

Geometry reconstruction from point clouds, also known as *surface reconstruction* or *reverse engineering*, has been extensively studied in the last two decades (e.g. [17], [8]). The objective is that the resulting piecewise-linear surface closely approximates the underlying surface from which the point cloud was sampled.

Typically the point clouds are acquired by a laser scanner or by photometric stereo methods, thus are often unorganized, noisy, and lack orientation information. Oriented normals at the points play a crucial role in reconstructing the surface but augmenting the points with orientation is a difficult task.

We take as input an unorganized point cloud which may contain noise and has no orientation information. A novel approach is presented to remove the noise in the point data and obtain a distortion-minimizing planar

parameterization. The meshing of the point cloud is then easily obtained by triangulating the points in the parameterization.

A key component of our algorithm is *meshless parameterization* of the point cloud to the plane. Surface parameterization is an important component in many digital geometry processing algorithms. Parameterization means to map a given surface to a suitable (usually planar) domain, such that mapping has some natural properties, e.g. smoothness, bijectivity and small distortion. Parameterization has been studied extensively in the past decade, and many elegant approaches address this problem. However, most of the parameterization algorithms are designed specifically for triangle mesh data, as opposed to so-called meshless parameterization, which applies to simple point sets.

In this paper, we present a novel solution to meshing point clouds based on meshless parameterization, as first proposed by Floater and Reimers [13]. The main idea is to embed (parameterize) the points in the plane, triangulate them in the plane using any reasonable triangulation algorithm, e.g. Delaunay, and then use the same triangulation edge structure to mesh the original 3D point cloud. In order for the resulting mesh to have decent quality, i.e. not have too many ill-shaped triangles, the 2D parameterization should preserve the local geometry of the 3D point cloud as much as possible. Since only developable surfaces can be flattened onto the plane with no distortion at all, the main challenge in parameterization is to minimize the shape distortion incurred during the mapping. Shape distortion may be measured in many different ways. Since a point cloud is nothing more than a collection of unorganized points, a practical measure of shape distortion is the change in local inter-point distances as a result of the parameterization. Thus the goal is to minimize this metric distortion — sometimes called stress or strain energy. We adopt a technique due to Zhang *et al.* [37], originally developed for “localizing” a sensor network in the plane (i.e. computing the positions of the sensors) based on measured inter-sensor distances. Although it does not strictly try to minimize stress, this “as-rigid-as-possible” (ARAP) algorithm embeds 3D points in the plane with small metric distortion. As a result, the quality of the triangulation does not significantly deteriorate when transferred from 2D to 3D.

Broadly speaking, our approach belongs to the family

† cgzhanglei@gmail.com

‡ ligangliu@zju.edu.cn

§ gotsman@cs.technion.ac.il

¶ huanghua@xjtu.edu.cn

of Delaunay-based surface reconstruction algorithms, which employs the Delaunay/Voronoi structure of sample points to extract the implied surface. The most renowned algorithms in this family are *Powercrust* [2] and *Cocone* ([3], [9], [10]). The strength of those algorithms is that they provide a theoretical guarantee on the correctness of the reconstructed geometry and topology if the sampling density meets some criteria. However, these approaches are complicated, and real-world input data rarely meets the sampling density requirements. When presented with a noisy input, these approaches might generate ugly-shaped triangles during reconstruction. In contrast, our meshing algorithm is easy to implement and can handle point clouds with noise, sharp features and flexible sampling rates.

Our main contribution is a novel meshless parameterization algorithm for point clouds which aims at minimizing metric distortion during parameterization. This meshless parameterization may be combined with a denoising component, which can filter out the noise, resulting in a smooth triangular mesh. So our approach achieves both denoising and reconstruction. Additionally, due to its inherent distance preserving property, the parameterization better suits surface reconstruction, resulting in higher quality triangulations. Being invariant to rigid transformations, it also permits an extension to treat closed meshes of arbitrary topology. Moreover, our approach is not dependent on the coherent orientation of the normal vector of each point as the orientation is handled automatically. Experimental results have shown that our approach performs much better than existing meshless parameterization algorithms.

## 2. RELATED WORK

Parameterization is a key component of our approach. There are a large number of practical approaches to mesh parameterization in the literature, and to review them all is beyond the scope of this paper. We refer the interested reader to the surveys of Floater and Hormann [14] and Sheffer *et al.* [34] for the state of the art.

As previously mentioned, most of the mesh parameterization approaches are difficult to apply to point clouds directly, since they rely on the underlying mesh connectivity information and the triangle geometry. However, Floater and Reimers [13] were able to extend the so-called “shape-preserving” (SP) mesh parameterization algorithm, based on convex combinations of one-ring neighborhood vertices [12], to a point cloud. A point is represented as a convex combination of its neighbors in a proximity graph, typically a  $k$ -nearest neighbor (kNN) graph, and the resulting “Laplace-type” linear system is solved for the planar positions of the vertices, subject to a fixed convex planar boundary. While this yields a reasonable parameterization, which can be triangulated and transferred to the original 3D point cloud to form a manifold triangle mesh, the

somewhat artificial convex boundary introduces significant distortion into the parameterization, resulting in distorted 3D triangles, which may also intersect each other. In addition, the presence of a boundary implies that the method is applicable only to point clouds whose underlying surface has disk topology. Realizing that a planar disk parameter domain may not be the most suitable, Zwicker and Gotsman [39] used a more sophisticated spherical parameterization algorithm, modified to handle a point cloud, to parameterize the point cloud of genus 0 to the sphere, and triangulated the result on the sphere. This yields better results, but they are still somewhat distorted, due to the significant difference between the unit sphere geometry and the underlying surface geometry. Hormann and Reimers [18] proposed to handle a closed topology by partitioning the point cloud into disk-like patches, meshing each separately, and then “stitching” them back together to form a closed mesh. While in principle this could work (even for higher genus surfaces), their particular method seems to yield rather distorted results, particularly at the patch seams. Tewari *et al.* [36] showed how, in the special case of a genus 1 underlying surface, to locally parameterize the point cloud to the plane in a seamless manner using the mathematical theory of discrete one-forms. This method however, has rather high complexity (because it involves detecting in  $O(n^3)$  the so called non-trivial cycles of the mesh), and is not applicable to surfaces of genus other than 1.

Other approaches that can be easily adapted to perform meshless parameterization are based on recent advances in manifold learning and dimension reduction. These techniques are designed to embed high-dimension points in low-dimension spaces with minimal metric distortion. The Multi-Dimensional Scaling (MDS) technique [7] embeds the points in the plane by preserving all pairs of distances through strain minimization, while Local Linear Embedding (LLE) [33] and Local Tangent Space Alignment (LTSA) ([38], [5]) try to preserve the affine relationships between local neighbors. The Local Rigid Embedding (LRE) [35] is a variant of LLE which tries to preserve distances within the point cloud, and is a particularly good candidate for meshless parameterization in our application. A more elaborate version of LRE is the PATCHWORK method of Koren *et al.* [21].

Broadly speaking, the meshless approach is close to the surface reconstruction algorithms originating in the computational geometry community. These come with theoretical guarantees on the geometry or topology of the reconstructed surfaces [4]. Amenta *et al.* [1] were the first to propose a Voronoi-based reconstruction approach, called *Crust*, which interpolates the sampled points using a Delaunay complex. This method however, requires a “good” sample from a smooth surface. Later, Amenta *et al.* [3] improved the *Crust* algorithm using the inverse medial axis transformation. The reconstructed surface is obtained as the boundary of power diagram

cells instead of the usual Voronoi diagram. However, this approach introduces many extra points in the reconstruction and does not always generate a triangulated surface. The *Cocone* method [2] was developed to improve *Crust* in both theory and practice. This approach uses so-called co-cones to extract the manifold triangles from the Delaunay triangulation. Subsequently, a series of *Cocone*-type approaches, e.g. *Tight Cocone* [9], *Robust Cocone* [10] were proposed to handle water-tight surface reconstruction and noisy input data.

Removing noise from point cloud has also been studied before. Mitra *et al.* [25] use local least squares fitting to analyze normal estimation in the presence of noise. Based on this, noise can be filtered out efficiently. Dey and Sun [11] denoise point clouds by projecting the points onto an underlying smooth surface defined as a moving least squares surface. The projected points are then used in subsequent processing. Lange and Polthier [22] employ an anisotropic mean curvature flow to remove noise from point clouds. Their approach can effectively remove noise as well as detect and enhance geometric features of the sampled surface. The Locally Optimal Projection (LOP) operator is a versatile tool to consolidate point clouds ([29], [19]). It projects an arbitrary point-set onto the point cloud, which can also reduce noise in a sense by replacing the noisy point cloud with the smoothly distributed points in the projection.

The meshless parameterization algorithm that we use is a member of the so-called As-Rigid-As-Possible (ARAP) family of parameterization algorithms. This family was proposed by Liu *et al.* [27], who present an efficient “local/global” algorithm for parameterizing a triangle mesh to the plane in a manner which preserves distances. However, their approach is designed for a triangle mesh with connectivity information, and takes advantage of the triangle structure of the mesh. When dealing with point clouds, lacking triangle connectivity, more caution must be exercised. Recently, Zhang *et al.* [37] adapted the basic ARAP approach to solve the localization problem for sensor networks. This involves the natural neighbors for each point. It first embeds local neighborhoods in “patches” onto the plane in a distance-preserving manner. Then, these patches are stitched together coherently by rigid alignment. We will use a variant of this algorithm here for meshless parameterization.

Furthermore, we propose a novel approach for obtaining the denoised point data and its parameterization simultaneously. In fact, ARAP meshless parameterization minimizes an energy function which contains a global measure of point cloud smoothness. Hence, to denoise point clouds, we simply allow each point in the cloud to move in its vicinity to obtain a better ARAP meshless parameterization result. We define a global Laplacian energy to minimize the noise during the optimization.

### 3. MESHLESS DENOISING AND PARAMETERIZATION

We start by formulating the problem of meshless denoising and parameterization.

#### 3.1 Problem statement

Given a collection of unorganized points  $\mathbf{X} = \{x_1, x_2, \dots, x_n\} \subset \mathbb{R}^3$ , which are sampled from a surface with disk topology and might contain noise, the meshless denoising and parameterization problem is to find new points  $\mathbf{P} = \{p_1, p_2, \dots, p_n\}$  in  $\mathbb{R}^3$  close to  $\mathbf{X}$  and their corresponding positions  $\mathbf{V} = \{v_1, v_2, \dots, v_n\}$  in  $\mathbb{R}^2$ . We call  $\mathbf{P}$  the denoised version of  $\mathbf{X}$  and  $\mathbf{V}$  the parameterization of  $\mathbf{P}$ . Generally we expect that  $\mathbf{P}$  is as close as possible to  $\mathbf{X}$  and that the local distances of  $\mathbf{P}$  are preserved in  $\mathbf{V}$  as much as possible.

We call  $\mathbf{X}$  an oriented point cloud if a normal  $\mathbf{n}_i$  is provided for each point  $x_i$ , otherwise it is *unoriented*. For each  $i$ , we define the local neighbors of point  $x_i \in \mathbf{X}$  as  $N_i = \{j : x_j \text{ is neighbor of } x_i\}$ ,  $X_i = \{x_j : j \in N_i\}$  and its cardinality as  $n_i = |N_i|$ .

#### 3.2 The global optimization problem

The problem of meshless denoising and parameterization is to find the denoised points  $\mathbf{P}$  and its parameterization  $\mathbf{V}$  simultaneously. We formalize it as a global optimization minimizing the following energy function:

$$E(\mathbf{P}, \mathbf{V}) = E_{Para} + E_{Fidelity} + \lambda E_{Smooth} \quad (1)$$

where  $E_{Fidelity}$  measures the distance of  $\mathbf{P}$  from  $\mathbf{X}$  and is defined as

$$E_{Fidelity} = \sum_{i=1}^n \|p_i - x_i\|^2 \quad (2)$$

$E_{Smooth}$  is the global Laplacian smoothing energy ([30], [26]) on  $\mathbf{P}$ :

$$E_{Smooth} = \sum_{i=1}^n \|p_i - \sum_{j \in N_i} w_{ij} p_j\|^2 \quad (3)$$

$w_{ij}$  are weights describing the affine relationship between  $p_i$  and its local neighbors,  $E_{Para}(\mathbf{P}, \mathbf{V})$  is a minimal metric distortion parameterization energy which tries to preserve the local distances of  $\mathbf{P}$  in  $\mathbf{V}$ , and  $\lambda$  is a scalar weight.

The main novelty of our approach lies in the parameterization energy  $E_{Para}(\mathbf{P}, \mathbf{V})$ , which computes a meshless parameterization  $\mathbf{V}$  of  $\mathbf{P}$  by rigid alignment. Given  $\mathbf{P}$ , the idea in computing the meshless parameterization  $\mathbf{V}$  is to preserve the distances between a point and its neighbors as much as possible throughout the process, which finally achieves a minimal metric distortion parameterization. Specifically, we use the local neighborhoods  $P_i$  of all points  $p_i$  as “building blocks” to construct the final meshless parameterization. Each point  $p_i$  in the input undergoes two “local” phases and one “global” phase, as shown in Fig. 1:

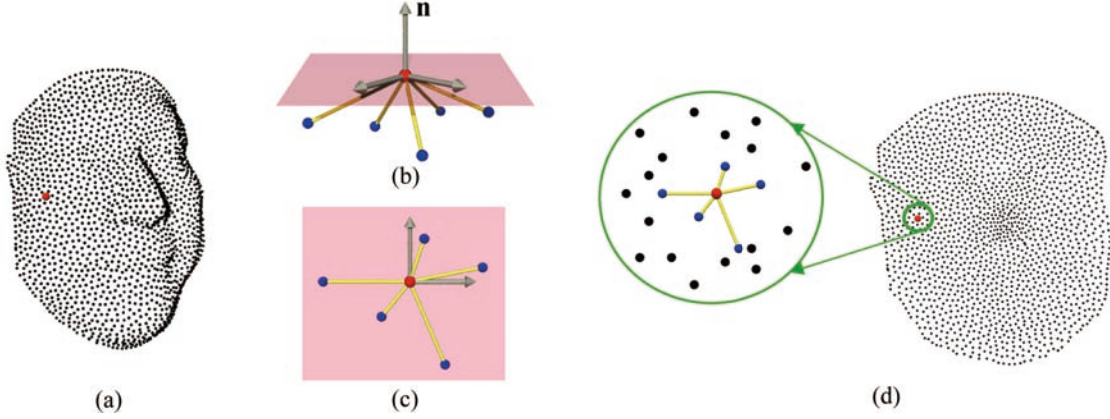


Fig. 1. Given a 3D point cloud (a), any given 3D point (in red) undergoes two local and one global phase in our algorithm: (b) finding its local neighbors (in blue) in the 3D point cloud and (pink) tangent plane; (c) flattening the neighborhood into the 2D tangent plane; (d) “stitching” the 2D neighborhoods into a global meshless 2D parameterization.

- 1) Finding its local neighbors  $P_i$  in the point cloud, and its tangent plane;
- 2) Flattening  $P_i$  into the tangent plane to form  $U_i$ ;
- 3) “Stitching” all the different  $U_i$ ’s into a global meshless parameterization, forming  $V_i$ .

We describe each phase in detail in Section 4 and then give the solution to the global optimization of (1) in Section 5.

#### 4. MESHLESS PARAMETERIZATION BY RIGID ALIGNMENT

In this section, we elaborate on our meshless parameterization scheme, namely, how to compute a parameterization  $V$  from a given point set  $P$ .

##### 4.1 Local flattening

We rely on a proximity graph defined on the point cloud. There are two possible ways to construct such a proximity graph: 1) An  $\varepsilon$ -radius graph, defined by  $P_i = \{p_j : |p_j - p_i| < \varepsilon\}$ ; 2) a  $k$ -nearest-neighbor (kNN) graph, where  $P_i$  is defined as the  $k$  nearest points to  $p_i$ . We adopt the second approach. A kNN graph on  $n$  points may be constructed in  $O(n \log n)$  time using an appropriate spatial data structure [6].

Once the  $P_i$ ’s are established, they should be flattened to the plane, to form  $U_i$ , in a distance-preserving manner, i.e., the distortion of the distances between points in  $U_i$  and the corresponding original distances in  $P_i$  should be minimal. It is most convenient to flatten  $P_i$  to the “tangent plane” of  $p_i$ . If  $P$  is an oriented point cloud, the tangent plane of  $p_i$  is that passing through  $p_i$  and perpendicular to  $\mathbf{n}_i$ . The advantage of using these planes is that they maintain a consistent orientation, which means they can be “stitched” together without foldovers. If  $P$  is an unoriented point cloud, it is possible to estimate the normal directions using standard methods, like PCA [17] or local least squares fitting [25], on each of the  $P_i$ . These typically will have inconsistent orientations, which is inconvenient for

processing. However, as we will see in Section 4.3, a small modification of our basic approach can handle also unoriented point clouds.

To flatten  $P_i$ , we first project the points in  $P_i$  orthogonally onto the tangent plane. Then, following Koren *et al.* [21], we improve the embedding to preserve the 3D distances as much as possible using a “stress majorization” procedure. This involves iteratively updating the position of  $u_{ij} \in U_i$  to be

$$u_{ij} \leftarrow u_i + d_{ij}(u_{ij} - u_i) \text{inv}(\|u_{ij} - u_i\|)$$

where

$$d_{ij} = \|p_{ij} - p_i\|$$

$$\text{inv}(x) = \begin{cases} 1/x, & : x \neq 0 \\ 0, & : x = 0 \end{cases}$$

##### 4.2 Rigid alignment

Once we have a distance-preserving flattening  $U_i$  of each of the separate neighborhoods  $P_i$  to their tangent planes, we would like to align them all to one global 2D embedding. Again, we would like to distort distances minimally. This is possible if we strive to transform each  $P_i$  to the unique parameterization plane by some rigid transformation, which preserves distances.

As stated above, the tangent planes of oriented points have consistent orientations, hence the rigid transformations should be rotation and translation only (without reflections) to prevent any “folding” during the parameterization. For each  $i$ , we want to find a rigid transformation (rotation and translation) for  $U_i$ , (from its local tangent plane to the global parameterization plane) such that the positions of  $P_i$  in the final parameterization ( $V_i$ ) are

$$V_i = U_i \mathbf{R}_i + \mathbf{t}_i \quad (4)$$

where  $\mathbf{R}_i$  is a rotation,  $\mathbf{t}_i$  is a translation and  $V_i = (v_{i1}, \dots, v_{iN_i})^T$ , where  $v_j = (v_j^x, v_j^y)^T$ , are the coordinates of the points in  $P_i$  in the global parameterization. To keep it simple, we assume that  $u_i$  coincides with  $v_i$ , which means the translation  $\mathbf{t}_i$  can be eliminated

from (4). Then, the coordinates of the points in  $N_i$  are redefined as

$$\begin{aligned} U_i' &= U_i - \mathbf{1}^T u_i^T \\ V_i' &= V_i - \mathbf{1}^T v_i^T \end{aligned}$$

where  $\mathbf{1} = (1, 1, \dots, 1)^T$ . In sequel, we will use  $U_i$  and  $V_i$  to denote the coordinates after this translation. Eq. (4) then becomes

$$V_i = U_i \mathbf{R}_i \quad (5)$$

If each of the  $V_i$  are indeed a pure rotation of  $U_i$ , then the final parameterization of the entire point cloud will have the same metric distortion as that present in the  $U_i$ 's. However, since there is some overlap between the different  $U_i$ , multiple rotations will apply to every point  $p_i$ . Thus, in practice, the metric distortion will increase during alignment of the neighborhoods, and we must solve the following optimization problem, where we simultaneously seek the global parameterization  $\mathbf{V}$  (from which the local  $V_i$  are extracted) and the rotations  $\mathbf{R}_i$ :

$$E_{ARAP} = \sum_{i=1}^N \|V_i - U_i \mathbf{R}_i\|_F^2 \quad (6)$$

where  $\|\cdot\|_F$  is the Frobenius norm, and  $\mathbf{R}_i$  satisfies

$$\mathbf{R}_i^T \mathbf{R}_i = \mathbf{I}, \det(\mathbf{R}_i) = 1$$

Problem (6) is a nonlinear least-squares problem. Luckily, we may resort to the alternating least squares (ALS) method to solve this optimization problem, which is simple and efficient. Observe that if all  $\mathbf{R}_i$  are known, then the unknown optimal  $\mathbf{V} = \{v_1, v_2, \dots, v_N\}$  can be obtained as the solution of a quadratic optimization problem; conversely, if  $\mathbf{V} = \{v_1, v_2, \dots, v_N\}$  is known, then the unknown optimal  $\mathbf{R}_i$  can be computed separately for every  $i$ , as we will soon see. Thus we may alternate between these two stages until convergence. We now discuss the details of the iterative solver and its initialization.

#### 4.2.1 The “local-global” alternating least-squares solver

Suppose the optimal  $\mathbf{V} = \{v_1, v_2, \dots, v_N\}$  of Problem (6) is known. For each  $i$ , we may compute the optimal rotation matrix  $\mathbf{R}_i$ , in a least squares sense, as the solution of the following small minimization problem:

$$\mathbf{R}_i = \underset{\mathbf{R}}{\operatorname{argmin}} \{ \|V_i - U_i \mathbf{R}\|_F^2 : \mathbf{R}^T \mathbf{R} = \mathbf{I}, \det(\mathbf{R}) = 1 \} \quad (7)$$

This is a Rotation Orthogonal Procrustes Problem (ROPP), which can be solved by Procrustes analysis [16]. Specifically, the optimal rotation matrix is  $\mathbf{R}_i = A_i Z_i B_i^T$ , where  $A_i$  and  $B_i$  are obtained from the Singular Value Decomposition (SVD) [32] of  $U_i^T V_i$ :

$$U_i^T V_i = A_i Z_i B_i^T \quad (8)$$

and  $Z_i = \operatorname{diag}(1, \dots, 1, \sigma_i)$ , where  $\sigma_i = \det(A_i B_i^T)$ , effectively excluding the orientation-reversing rigid transformation. Computing all the  $\mathbf{R}_i$  is the so-called “local” step of the ALS solver.

Next, if all the  $\mathbf{R}_i$  are known, we can solve for  $\mathbf{V}$  by setting the gradient of (6) to zero. This yields the following Poisson-type linear system for the  $v_i$ :

$$\begin{aligned} & \sum_{j \in N_i} (v_i - v_j) + \sum_{i \in N_j} (v_i - v_j) \\ &= \sum_{j \in N_i} \mathbf{R}_i (u_i - u_{ij}) + \sum_{i \in N_j} \mathbf{R}_j (u_{ji} - u_j) \end{aligned} \quad (9)$$

This is the so-called “global” step of the ALS solver. The entries of the associated matrix depend only on the local neighborhood  $N_i$ , so the coefficient matrix is sparse and fixed throughout the iterative algorithm. Thus we can pre-factor it (e.g. with Cholesky decomposition), and reuse the factorization to solve the linear system, whose right hand side changes in each iteration.

#### 4.2.2 Initialization

Choosing a suitable initial parameterization  $\mathbf{V}$  is important for the stability and convergence rate of the ALS solver described in the previous section. Ideally, the initial parameterization should not be too far from the solution of (6). Essentially, the nonlinearity of the problem is caused by the orthogonality of the rotation matrices  $\mathbf{R}_i$ , namely, if

$$\mathbf{R}_i = \begin{pmatrix} a_i & b_i \\ -b_i & a_i \end{pmatrix} \quad (10)$$

then the entries of  $\mathbf{R}_i$  are nonlinearly constrained by  $a_i^2 + b_i^2 = 1$ .

If we remove these nonlinear constraints on  $\mathbf{R}_i$ , i.e., allow a general similarity transformation matrix  $\mathbf{S}_i$  instead of  $\mathbf{R}_i$ :

$$\mathbf{S}_i = \begin{pmatrix} a_i & b_i \\ -b_i & a_i \end{pmatrix} \quad (11)$$

then (6) will become a simple linear least squares problem with respect to  $\mathbf{V}$ :

$$\mathbf{V} = \underset{V_i \subset V, a, b}{\operatorname{argmin}} \sum_{i=1}^N \left\| V_i - U_i \begin{pmatrix} a_i & b_i \\ -b_i & a_i \end{pmatrix} \right\|_F^2 \quad (12)$$

Note one caveat: The problem (12) has a trivial solution, i.e.,  $V_i \equiv 0$  and  $a_i = b_i = 0$ . To eliminate this solution, we fix two points with longest distance in the plane and solve the resulting non-homogeneous linear system of normal equations with these constraints. When applied to triangle meshes, this solution has been shown by Liu *et al.* [27] to be equivalent to the Least-Squares Conformal Mapping (LSCM) parameterization method [23]. We use this 2D configuration as our initial parameterization for the ALS solver. Fig. 2(b) shows the final rigid alignment parameterization using this initialization, shown in Fig. 2(a).

#### 4.3 Treating unoriented point clouds

If  $\mathbf{P}$  is an unoriented point cloud, the rigid alignment algorithm described in Section 4.2 will probably not work, as the estimated normal orientations may not

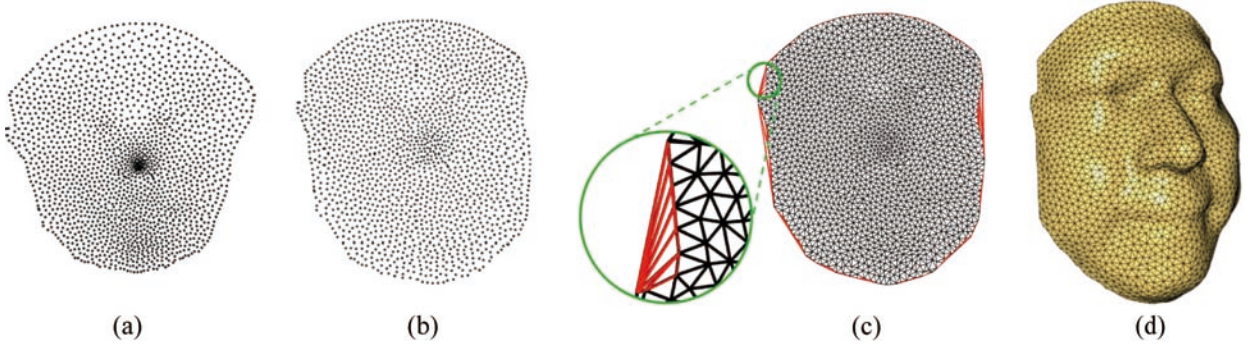


Fig. 2. (a) Initial parameterization of mesh of Fig. 1(a) by similarity alignment. (b) Meshless parameterization generated by the ‘‘As-Rigid-As-Possible’’ ALS algorithm, starting from (a). (c) Delaunay triangulation of the parameterization in (b). Redundant triangles marked in red should be removed. (d) Reconstructed 3D triangle mesh.

be consistent. Since the rigid transformation in (6) is restricted to be a pure rotation, this will resist orientation reversals in the global alignment, which may have rectified the situation. However, we can circumvent the problem by allowing general rigid transformations, i.e., including also reflection. Thus, given  $U_i$  and  $V_i$ , the best rigid transformation between them is

$$\mathbf{R}_i = \operatorname{argmin}_{\mathbf{R}} \{ \|V_i - U_i \mathbf{R}\|_F^2 : \mathbf{R}^T \mathbf{R} = \mathbf{I} \} \quad (13)$$

The solution is  $\mathbf{R}_i = A_i B_i^T$ , where  $A_i$  and  $B_i$  are obtained by SVD, as in Section 4.2.1.

However, the initialization procedure proposed in Section 4.2.2 will not work here, since a similarity transformation is also orientation-preserving as its determinant is always non-negative. Fig. 3(a) shows the results using similarity transformations to generate the initial parameterization for an unoriented point cloud, which introduces foldovers. Fortunately, we can fix this problem by allowing  $\mathbf{R}_i$  to be a general affine transformation. We define

$$\mathbf{A}_i = \begin{pmatrix} a_i & b_i \\ c_i & d_i \end{pmatrix} \quad (14)$$

and the optimization problem (6) becomes

$$\mathbf{V} = \operatorname{argmin}_{V_i \subset V, a, b, c, d} \sum_{i=1}^N \left\| V_i - U_i \begin{pmatrix} a_i & b_i \\ c_i & d_i \end{pmatrix} \right\|_F^2 \quad (15)$$

which is also a linear least squares problem with respect to  $\mathbf{V}$ .

Like the linear system (9), (15) also has the trivial solutions when  $v_i = 0$  and  $a_i = b_i = c_i = d_i = 0$ , so we have to add at least three constraints by fixing the positions of the points in one local neighborhood, e.g., the points in  $U_0$ . Fig. 3(c) shows the rigid alignment parameterization obtained using affine alignment as the initial parameterization.

#### 4.4 The boundary

Once the point cloud has been embedded in the plane, we triangulate it using planar Delaunay triangulation. This results in a set of triangles with quite uniform

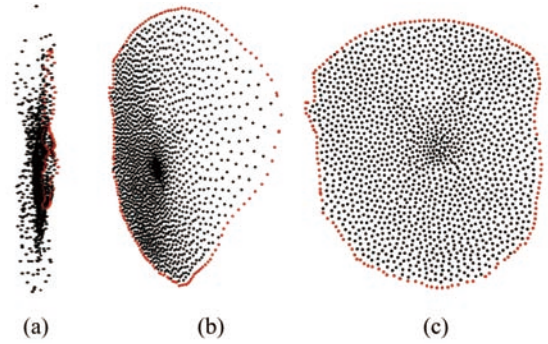


Fig. 3. For the point cloud model in Fig. 1(a), without normals, the solver run with similarity transformations will generate a parameterization with local neighbors folding as shown in (a). Assuming affine transformation will generate the parameterization shown in (b). (c) shows the final parameterization by rigid alignment with (b) as initialization. Red points denote boundary.

shapes. The natural boundary of the point cloud is found in a manner similar to that employed by Floater and Reimers [13]. All Delaunay triangles outside this boundary are subsequently removed. See Fig. 2(c) for an example. The remaining triangles are used to triangulate the input point cloud.

## 5. GLOBAL OPTIMIZATION

The global minimization (1) is a nonlinear function with respect to  $\mathbf{P}$  and  $\mathbf{V}$ , with no closed-form solution. Therefore, we use an iterative approach to solve this optimization. In each iteration, the optimization can be cast as a sparse linear system.

By iteration, we compute a series of points  $\mathbf{P}^{(k)}$  and  $\mathbf{V}^{(k)}$ ,  $k = 0, 1, 2, \dots$ . Initially, we set  $\mathbf{P}^{(0)} = \mathbf{X}$ . The iteration consists of two phases:

(1) Compute  $\mathbf{V}^{(k)}$  from  $\mathbf{P}^{(k)}$ :

From the point data  $\mathbf{P}^{(k)}$ , we compute its parameterization  $\mathbf{V}^{(k)}$  as in Section 4 by minimizing

$$E(\mathbf{V}) = E_{Para} \quad (16)$$

(2) Compute  $\mathbf{P}^{(k+1)}$  from  $\mathbf{V}^{(k)}$ :

The new positions of  $\mathbf{P}^{(k+1)}$  can be computed from  $\mathbf{V}^{(k)}$  by minimizing

$$E(\mathbf{P}) = E_{Fidelity} + \lambda E_{Smooth} \quad (17)$$

where the coefficients  $w_{ij}$  in  $\mathbf{E}_{Smooth}$  (see Eq. (3)) are calculated from the corresponding neighbors in parameterization  $\mathbf{V}^{(k)}$  as:

$$\begin{aligned} \{w_{ij}\} &= \underset{w_{ij}}{\operatorname{argmin}} \sum_{j \in N_i} \|v_i^{(k)} - w_{ij}v_j^{(k)}\|^2, \\ \text{s.t. } &\sum_{j \in N_i} w_{ij} = 1 \end{aligned} \quad (18)$$

for each  $i$ . These may be computed by solving a least-squares problem [33]. Since both  $\mathbf{E}_{Fidelity}$  and  $\mathbf{E}_{Smooth}$  are quadratic energies with respect to  $p_i$ , by setting the gradient of (17) to be zero, we obtain a linear system for  $p_i$ :

$$(p_i - x_i) + \lambda(p_i - \sum_{j \in N_i} w_{ij}p_j) + \lambda \sum_{i \in N_j} w_{ji}(w_{ji}p_i - p_j) = 0 \quad (19)$$

The solution of (19) will be  $\mathbf{P}^{(k+1)}$ .

## 6. APPLICATION: MESHING POINT CLOUDS

In this section, we show that our meshless denoising and parameterization provides a good method for surface reconstruction.

Once the planar parameterization is available, 2D Delaunay triangulation can be applied to the planar positions to provide triangle connectivity for the 2D points, which can be transferred back to the 3D point cloud to form a triangle mesh. Because our ARAP meshless parameterization approach flattens the point cloud into the plane, it is applicable only to an open point cloud (whose underlying surface has disk topology) and the result will be an open 3D triangle mesh, as in Fig. 2(d).

We now show how to modify the basic method, in order to apply it to point clouds sampled from an underlying closed manifold surface. Following Hormann and Reimers [18], this is done by applying the basic approach to small overlapping patches segmented from the point cloud. A closed triangular mesh is then obtained by triangulating each patch separately, yet consistently. As opposed to the results of Hormann and Reimers [18], due to the distance-preservation of ARAP parameterization, and the invariance of the Delaunay triangulation to rigid transformations, the triangulations of two patches will mostly agree in their overlap. This will result in an almost clean manifold triangular mesh which can be perfected by simple post processing. See examples in Figs. 4 and 5.

### 6.1 Patch segmentation and optimization

The goal now is to segment the point cloud into overlapping patches which are topologically equivalent to a disk. Let  $S = \{S_i | i = 1, \dots, K\}$  be the patches. To obtain a closed reconstruction from these patches,  $S$  should satisfy the following conditions: 1)  $S$  should cover the entire point cloud, i.e.,  $\bigcup S_i = \mathbf{P}$ ; 2) two adjacent patches have some overlap. The following scheme generates such a segmentation.

First, choose  $K$  seed points from the point cloud. These seed points should be uniformly distributed on the point cloud. An easy way to achieve this is by  $k$ -means clustering to segment the point cloud into  $K$  patches, denoted by  $S'_i$ . Then the seed point  $s_i$  for patch  $S'_i$  is selected as the point nearest to the geometric center of this patch. Note that the intersection of any two patches is now empty, but we have  $\bigcup S'_i = \mathbf{P}$ . Then each patch  $S'_i$  is extended to a larger patch  $S_i$ , which means  $S'_i \subset S_i$ . Now adjacent patches  $S_i$  have non-empty overlaps. It is easy to get  $S_i$  from  $S'_i$ , e.g., we can simply extend  $S'_i$  by absorbing the points in a small  $\mu$ -neighbor of boundary points of  $S'_i$ .

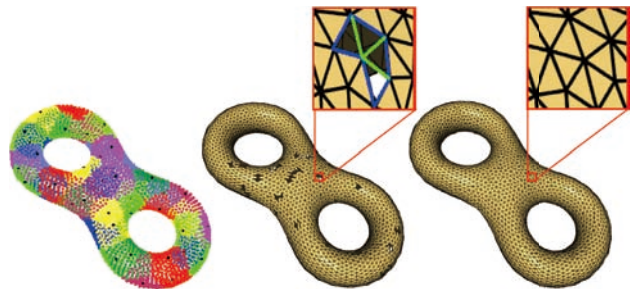


Fig. 4. Meshing a point cloud sampled from a closed surface of genus 2. **(Left)** Input point cloud (with 4,000 points) is segmented into 40 patches by  $k$ -means clustering. The patches are color-coded, black points are the seeds. **(Center)** Reconstructed triangle mesh by ARAP meshless parameterization of each patch and Delaunay triangulation. There may be holes and inconsistent local triangulations. **(Right)** Final triangle mesh after inconsistent triangle removal and hole filling. The holes (bounded by blue lines in Center) are filled by triangulating the polygon (green lines).

We now apply the meshless approach on each patch  $S_i$  to get a planar point cloud. Then we apply Delaunay triangulation and map the connectivity back to point cloud.

It is worthwhile to point out that the orientations of all patches do not have to be consistent because our approach can handle unoriented point clouds automatically, as demonstrated in Section 4.3.

### 6.2 Reconstructing a closed manifold

There may be inconsistent triangulations in the overlap of two patches, for two reasons: First, the Delaunay triangulation on the patch boundary points will not agree with each other; Second, ARAP meshless parameterization cannot guarantee that the distances are not distorted at all during parameterization, and this will affect the structure of the Delaunay triangulation. We can fix the first problem by ignoring triangles close to the triangulation boundary, in effect adopting only triangles which are a subset of  $S_i$ . The parameter  $0 < \alpha < \mu$  indicates to adopt only triangles that have its vertices in  $S_i$  and the  $\alpha$ -neighbor of boundary points of  $S_i$ . The second problem may be addressed by examining the edges after triangulation. If more than two triangles share one edge, we remove the redundant triangles. Fortunately, due to the distance-preservation properties of our ARAP

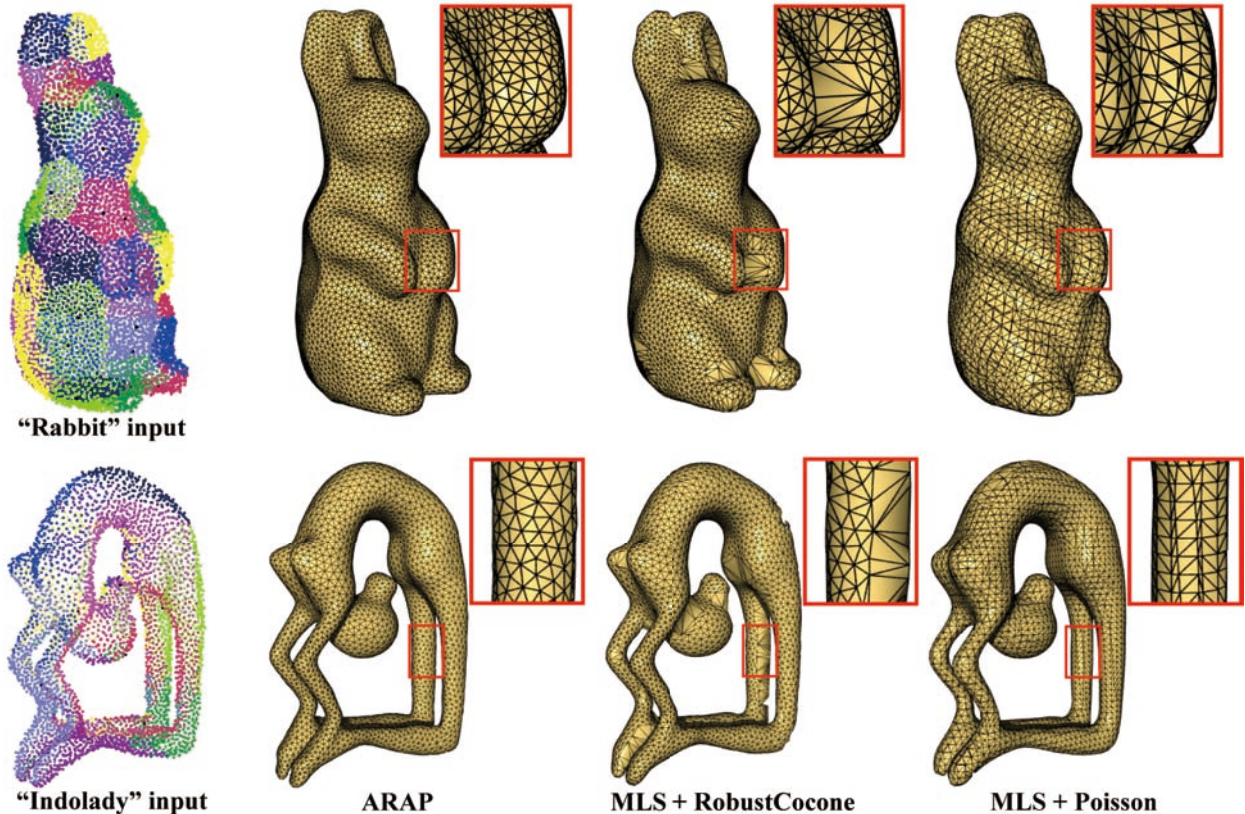


Fig. 5. Meshing noisy point clouds sampled from closed surfaces of different genus. **(Top)** Input point cloud (6,000 points), genus 0, is segmented into 30 patches, and reconstructed triangle mesh. **(Bottom)** Input point cloud (4,817 points), genus 2, is segmented into 10 patches, and reconstructed triangle mesh. Noise rate is  $2\%D$ , where  $D$  is the diameter of point cloud. For RobustCocone [10] and Poisson reconstruction [20], MLS projection [11] is used to denoise the point clouds.

meshless parameterization algorithm, disagreement of triangulations is very infrequent. Table 1 details the amount of inconsistent triangulations that occurred in our experimental examples.

In some rare cases, there may exist holes after reconstruction. To form a closed mesh, the holes are filled by simply triangulating the polygon formed by the hole boundaries. See Fig. 4 for an example.

## 7. EXPERIMENTAL RESULTS

We now present some results of our meshless denoising and parameterization algorithm applied to point clouds obtained by sampling 3D models. Our approach can effectively obtain a denoised point cloud and its parameterization and then construct a high quality triangular mesh.

Fig. 6 shows the results of applying our approach on a data cloud with two levels of noise. The weight  $\lambda$  in (1) controls the smoothness of the result. Larger values of  $\lambda$  will generate smoother results. It can be seen that our approach can achieve both smooth and high quality triangular mesh. The figures in the bottom row show the behavior of the energy in (1), indicating that it decreases during iteration.

In Fig. 9, we compare our basic ARAP approach (on a point cloud with disk topology) with the analogous algorithm using the meshless parameterizations SP [13],

LLE [33], and LRE [35]. SP requires a fixed boundary, as opposed to the others. To make a fair comparison, the point cloud used in this example contains no noise.

It is not surprising that SP incurs large distortion, especially in the vicinity of the boundary, resulting in bad triangle shapes. The other three approaches can generate fairly good results when the boundary of the point cloud is somewhat regular. However, the differences between these algorithms show when the boundary is irregular. LLE and LRE now incur distortion, whereas ARAP generates higher quality parameterizations, and the resulting 3D triangle mesh contains more regular connectivity and uniform-shaped triangles.

Non-uniform sampling is a significant issue when meshing point clouds. Fig. 7 shows that our approach can handle this case well. From the results generated with both  $\epsilon$ -radius and kNN graphs, we see the parameterization preserves the points' distribution, and this is reflected by the resulting triangle mesh.

We also compare our approach with the *Cocone*-type algorithms and Poisson surface reconstruction [20], the latter based on building a characteristic function for the cloud volume. Software for the *Cocone*-type algorithms, including *Cocone* [2], *TightCocone* [9] and *Robust Cocone* [10], were kindly provided by the authors of those algorithms, and Poisson reconstruction is implemented in the MeshLab software package [24].

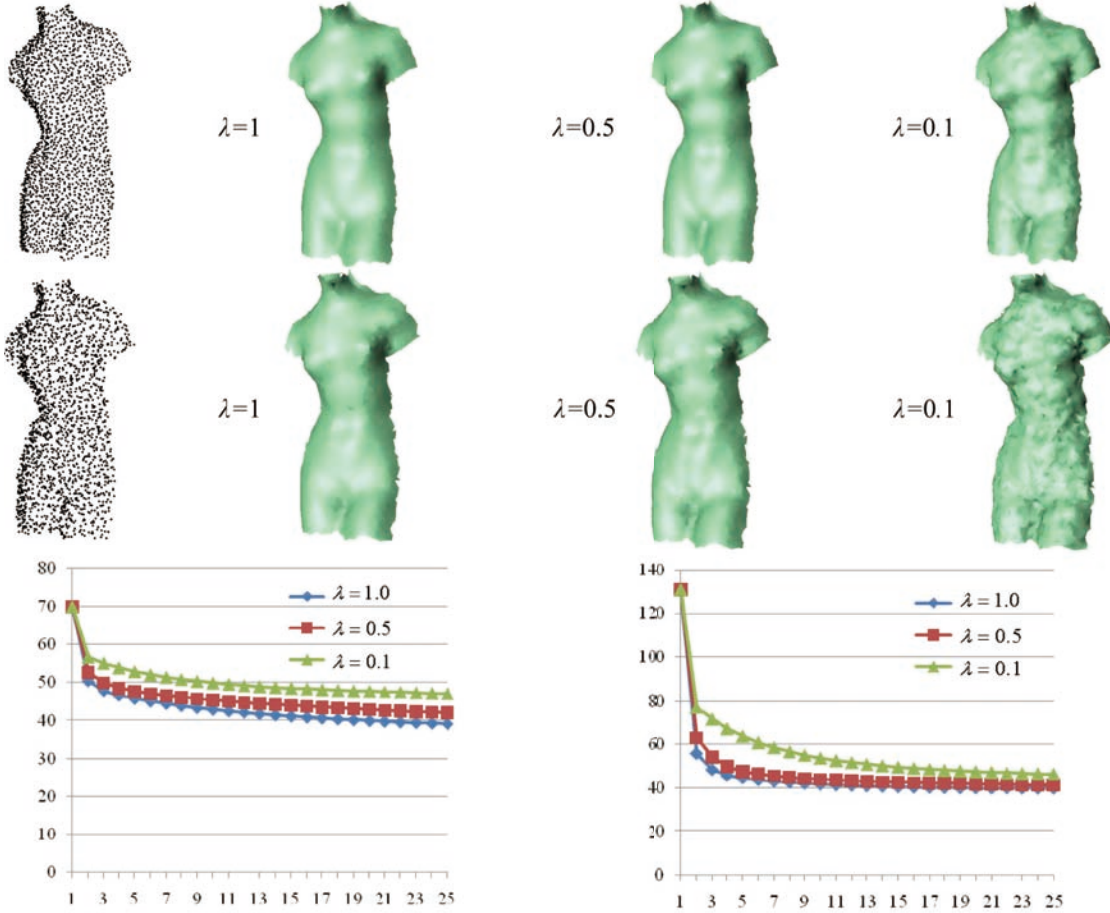


Fig. 6. Denoising and triangulation of the Venus point cloud with different noise levels and values of  $\lambda$ . **Top row**: noise rate is  $1\%D$ ; **Middle row**: noise rate is  $2\%D$ , where  $D$  is the diameter of point cloud. **Bottom row**: energy change of Equation (1) during iteration when noise rates are  $1\%D$  (left) and  $2\%D$  (right) respectively.

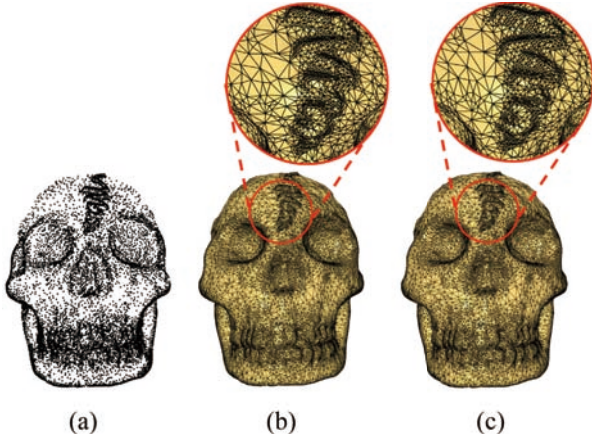


Fig. 7. Meshing a non-uniformly sampled point cloud with different proximity graphs. (a) The point cloud. (b) Meshing using kNN with  $k = 12$ ; (c) Meshing using  $\epsilon$ -radius with  $\epsilon = 3\%$ .

Fig. 5 shows two examples of reconstructing closed mesh surfaces in the presence of noise. For *RobustCocone* and Poisson reconstruction we use moving least squares surface projection [11] to denoise the sample points before reconstruction. *RobustCocone* interpolates through a subset of smooth sample points, so the trian-

gulation might contain non-uniform triangles. Poisson surface reconstruction generates the final triangulation using Marching Cubes, which also has irregular triangles. Our approach can not only directly denoise the raw data, but also produce a high quality triangulation.

As our approach interpolates the sample points, sharp features may be preserved during reconstruction. Fig. 8 shows two such examples. We assume the data contains no noise and compare our results with those of *Cocone* and *TightCocone*. *Cocone* is suitable for reconstructing surfaces with boundaries, but might incur holes when sampling is insufficient. *TightCocone* interpolates the sample points to produce a water-tight surface. We observe that our approach generates a reconstruction no worse than (*Tight*)*Cocone*, but with less computational effort (see Table 1).

### 7.1 Complexity

The complexity of our method is dominated by the  $O(n \log n)$  construction of the proximity graph on  $n$  points, and the ALS iterative solver. Each iteration of the solver requires  $O(n)$  time for the local step (SVD of  $n \times 2 \times 2$  matrices), and  $O(n)$  time for the global step (solution of a sparse linear system having  $O(n)$  non-zero

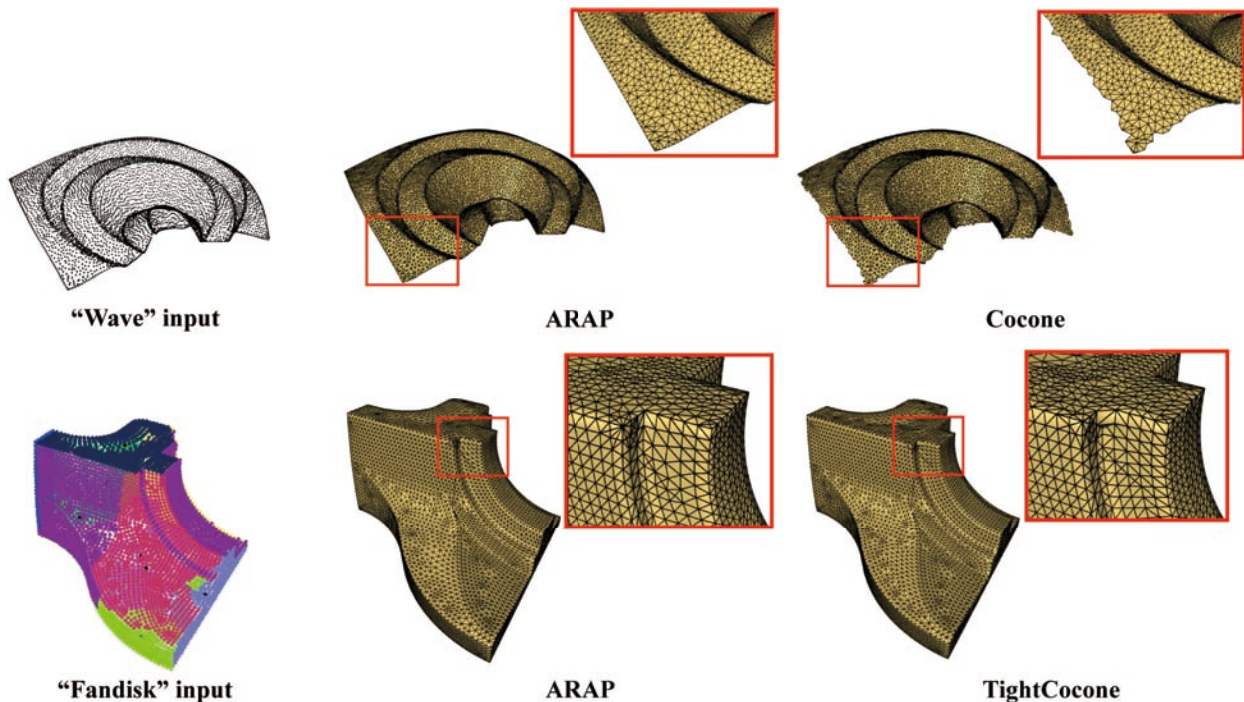


Fig. 8. Meshing point clouds sampled from closed surfaces with sharp features. (**Top**) “Wave” input point cloud (9,274 points), reconstruction by Cocone [2] and ARAP. (**Bottom**) “Fandisk” input point cloud (7,583 points) is segmented into 10 patches, reconstructed by TightCocone [9] and ARAP.

TABLE 1  
STATISTICS FOR MESHLESS ARAP AND THE RUNTIME OF *Cocone*, *TightCocone*, *RobustCocone* AND POISSON RECONSTRUCTION USED IN THE EXPERIMENTS.

Model	Meshless ARAP				Holes	Inconsistent triangles	(T,R)Cocone (sec)	Poisson (sec)
	ARAP parameterization		Global optimization					
	Iterations	Time (sec)	Iterations	Time (sec)				
Rabbit	8	0.4	10	<b>2.1</b>	11	104	<b>3.2</b>	<b>3.5</b>
Indolady	7	0.35	9	<b>1.7</b>	7	82	<b>2.5</b>	<b>2.7</b>
Fandisk	8	0.4	9	<b>2.1</b>	9	50	<b>5.4</b>	<b>3.9</b>
Wave	10	0.9	10	<b>4.2</b>	0	0	<b>6.4</b>	—

entries in a  $n \times n$  matrix). It is difficult to bound the number of iterations required until convergence, but our experiments seem to indicate that this is a very small number. Thus the method is very fast. Table 1 details some of these statistics.

## 8. CONCLUSION

We have presented a new meshless denoising and parameterization approach for point clouds. The denoised point cloud and its parameterization are obtained simultaneously by a global optimization based on an iterative solver. The meshless parameterization consists of a local flattening and a global aligning operation. The key idea is to use rigid transformations to align all the local neighbors together, while preserving their shapes. The point set may then be meshed by triangulating the parameterization result. An extension of the basic idea allows to mesh point clouds sampled from manifold surfaces having arbitrary topologies.

Our future work includes methods to improve the density of the points using our framework. This could

possibly be done by incorporating an energy that encourages the movement of the points along their tangent planes.

## ACKNOWLEDGMENTS

We thank the anonymous reviewers for their helpful comments and Professor Tamal K. Dey for providing the software of *Cocone*, *TightCocone* and *RobustCocone*. Lei Zhang and Hua Huang are supported by the Key Project of Chinese Ministry of Education (No. 109142). Ligang Liu is partly supported by the 973 National Key Basic Research Foundation of China (No. 2009CB320801) and the National Natural Science Foundation of China (60776799).

## REFERENCES

- [1] N Amenta, M Bern, M Kamvysselis. A new Voronoi-based surface reconstruction algorithm. *Siggraph* 98, 1998, 415-421.
- [2] N Amenta, S Choi, T K Dey, N Leekha. A simple algorithm for homeomorphic surface reconstruction. *International Journal of Computational Geometry and Applications* 2002, 12: 125-141.

- [3] N Amenta, S Choi, R Kolluri. The power crust. Proc. Symposium on Solid Modeling and Applications 2001, 249-260.
- [4] F Cazal, J Giesen. Delaunay triangulation based surface reconstruction. In: Efficient Computational Geometry for Curves and Surfaces 2006. Springer-Verlag, 231-276.
- [5] Z G Chen, L G Liu, Z Y Zhang, G J Wang. Surface parameterization via aligning optimal local flattening. Proc. Symposium on Solid and Physical Modeling 2007, 291-296.
- [6] K Clarkson. Fast algorithms for the all nearest neighbors problem. Proc. IEEE FOCS 1983, 226-232.
- [7] T F Cox, M A A Cox. Multidimensional scaling. London: Chapman & Hall, 1994.
- [8] T K Dey. Curve and surface reconstruction: Algorithms with mathematical analysis. Cambridge University Press, 2006.
- [9] T K Dey, S Goswami. Tight Cocone: A watertight surface reconstructor. Journal of Computing and Information Science in Engineering 2003, 3: 302-307.
- [10] T K Dey, S Goswami. Provable surface reconstruction from noisy samples. Computational Geometry: Theory and Applications 2006, 35(1): 124-141.
- [11] T K Dey, J Sun. Adaptive MLS surfaces for reconstruction with guarantees. Proc. Symposium on Geometry Processing 2005, 43-52.
- [12] M S Floater. Parameterization and smooth approximation of surface triangulations. Computer Aided Geometry Design 1997, 14: 231-250.
- [13] M S Floater, M Reimers. Meshless parameterization and surface reconstruction. Computer Aided Geometric Design 2001, 18(2): 77-92.
- [14] M S Floater, K Hormann. Surface parameterization: a tutorial and survey. In: Advances in Multiresolution for Geometry Modeling 2005, 157-186.
- [15] C Gotsman, X F Gu, A Sheffer. Fundamentals of spherical parameterization for 3D meshes. ACM Transactions on Graphics 2003, 22(3):358-363.
- [16] J C Gower, G B Dijkstra. Procrustes problems. Oxford: Oxford University Press 2004.
- [17] H Hoppe, T DeRose, T Duchamp, J McDonald, W. Stuetzle. Surface reconstruction from unorganized points. In: Proceedings of SIGGRAPH 1992, 71-78.
- [18] K Hormann, M Reimers. Triangulating point clouds with spherical topology. In: Proceedings of Curve and Surface Design 2002.
- [19] H Huang, D Li, H Zhang, U Ascher, D Cohen-Or. Consolidation of unorganized point clouds for surface reconstruction. In: Proceedings of ACM SIGGRAPH Asia 2009.
- [20] M Kazhdan, M Bolitho, H Hoppe. Poisson surface reconstruction. In: Proc. Symposium on Point-Based Graphics. 61-70, 2006.
- [21] Y Koren, C Gotsman, M Ben-Chen. PATCHWORK: Efficient localization for sensor networks by distributed global optimization. Technical Report 2005, available at <http://www.cs.technion.ac.il/~gotsman/AmendedPubl/Koren/patchwork-tr.pdf>.
- [22] C Lange, K Polthier. Anisotropic smoothing of point sets. Computer Aided Geometric Design 2005, 22(7):680-692.
- [23] B Levy, S Petitjean, N Ray, J Maillot. Least squares conformal maps for automatic texture atlas generation. ACM Transactions on Graphics 2002, 21(3):362-371.
- [24] MeshLab. Visual Computing Lab-ISTI-CNR. <http://meshlab.sourceforge.net>. 2009
- [25] N J Mitra, A Nguyen, L Guibas. Estimating surface normals in noisy point cloud data. International Journal of Computational Geometry and Applications 2004, 14: 261-276.
- [26] L G Liu, C L Tai, Z P Ji, G J Wang. Non-iterative approach for global mesh optimization. Computer Aided Design 2007, 39(9):772-782.
- [27] L G Liu, L Zhang, Y Xu, C Gotsman, S J Gortler. A local/global approach to mesh parameterization. Computer Graphics Forum 2008, 27(5):1495-1504.
- [28] Y S Liu, P Q Yu, M C Du, J H Yong, H Zhang, J C Paul. Mesh parameterization for an open connected surface without partition. In: Proceedings of Computer Aided Design and Computer Graphics 2005, 306-312.
- [29] Y Lipman, D Cohen-Or, D Levin, H Tal-Ezer. Parameterization-free projection for geometry reconstruction. ACM Transactions on Graphics 2007, 26(3):221-226.
- [30] A Nealen, T Igarashi, O Sorkine, M Alexa. Laplacian mesh optimization. In: Proceedings of ACM GRAPHITE 2006, 381-389.
- [31] M Pauly, L Kobbelt, M Gross. Multiresolution modeling of point sampled geometry. CS Technical Report#378 2002, ETH Zurich.
- [32] W H Press, S A Teukolsky, W T Vetterlilng, B P Flannery. Numerical Recipes in C++: The Art of Scientific Computing. 2002.
- [33] S T Roweis L K Saul. Nonlinear dimensionality reduction by locally linear embedding. Science 2000, 290:2323-2326.
- [34] A Sheffer, E Praun, K Rose. Mesh parameterization methods and their applications. Foundations and Trends in Computer Graphics and Vision 2006, 2(2):105-171.
- [35] A Singer. A remark on global positioning from local distances. Proceedings of the National Academy of Sciences 2008, 105(28):9507-9511.
- [36] G Tewari, C Gotsman, S J Gortler. Meshing genus-1 point clouds using discrete one-forms. Computer and Graphics 2006, 20(6):917-926.
- [37] L Zhang, L G Liu, C Gotsman, S J Gortler. An as-rigid-as-possible approach to sensor network localization. ACM Trans. Sensor Network, 6(4), 2010.
- [38] Z Y Zhang, H Y Zha. Principle manifolds and nonlinear dimensionality reduction via tangent space alignment SIAM Journal on Scientific Computing 2005, 26(11):313-338.
- [39] M Zwicker, C Gotsman. Meshing point clouds using spherical parameterization. Proc. Symposium on Point-Based Graphics 2004, 173-180.

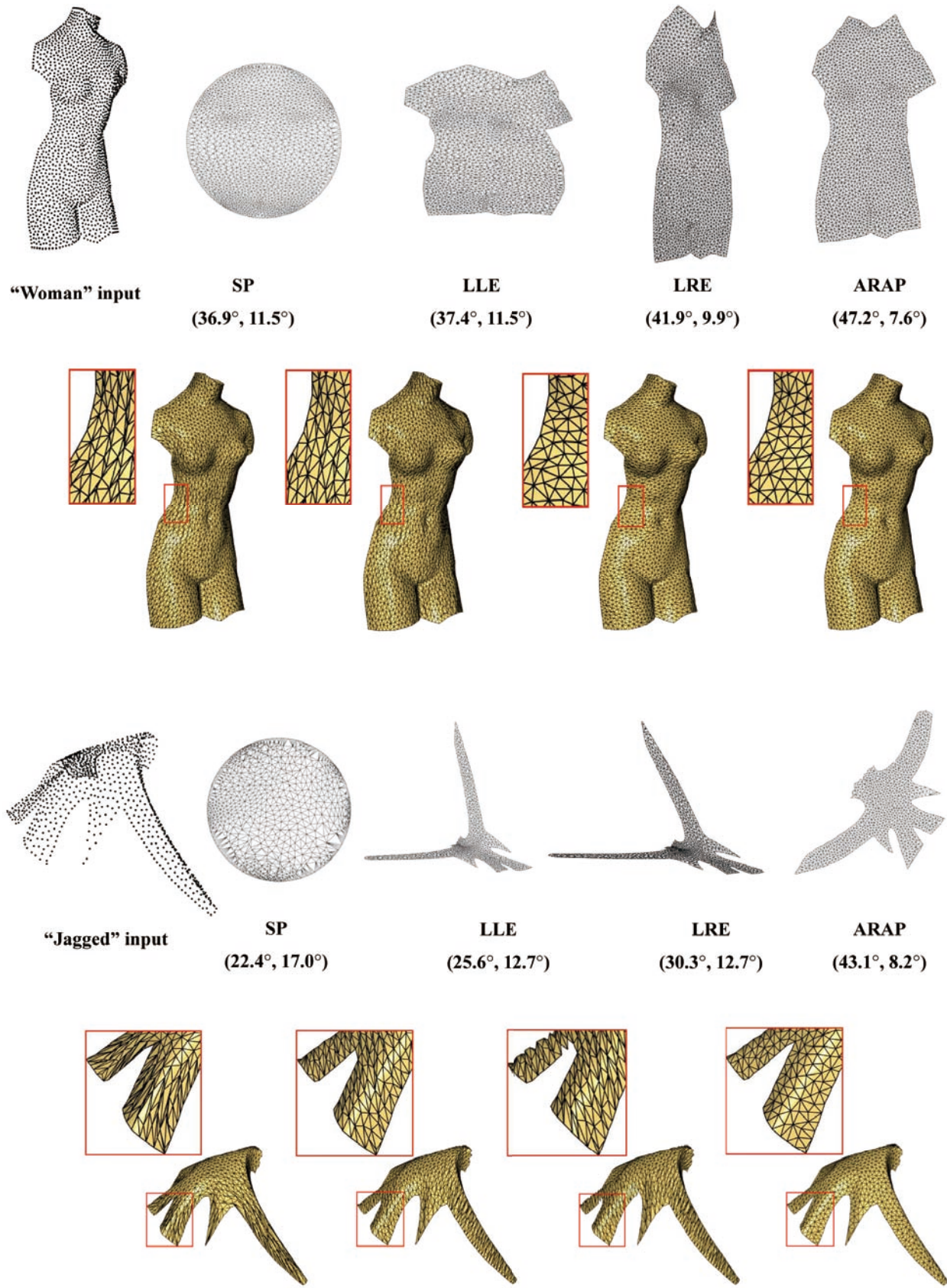


Fig. 9. Comparison of meshless 2D parameterizations and resulting 3D triangle meshes on a point cloud sampled from two surfaces having disk topology. The approaches for comparison include SP [13], LLE [33], LRE [35], and our proposed ARAP. Numbers in parentheses are the average and standard deviation of the minimal angle per triangle (taken over all mesh triangles).