The Visual Computer

Optimized triangle mesh reconstruction from unstructured points

Yong-Jin Liu,
Matthew Ming-Fai Yuen

Department of Mechanical Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong, P.R. China
E-mail: liuyj@ust.hk, meymf@ust.hk

Published online: 28 January 2003

A variety of approaches have been proposed for polygon mesh reconstruction from a set of unstructured sample points. Suffering from severe aliases at sharp features and having a large number of unnecessary faces, most resulting meshes need to be optimized using input sample points in a postprocess. In this paper, we propose a fast algorithm to reconstruct high-quality meshes from sample data. The core of our proposed algorithm is a new mesh evaluation criterion which takes full advantage of the relation between the sample points and the reconstructed mesh. Based on our proposed evaluation criterion, we develop necessary operations to efficiently incorporate the functions of data preprocessing, isosurface polygonization, mesh optimization and mesh simplification into one simple algorithm, which can generate high-quality meshes from unstructured point clouds with time and space efficiency.

Key words: Geometric modeling – Three-dimensional shape recovery – Mesh optimization – Mesh simplification

Correspondence to: Y.-J. Liu

1 Introduction

State-of-the-art visualization techniques help to interpret experimental data and construct relations between data elements. In diverse application domains, including reverse engineering, scientific visualization, telemetry and geography, visualization of sample points plays an important role in data analysis and model construction. Concerning data organization, the point sets can be classified as either unstructured or structured. In this work, to deal with a general surface reconstruction problem, we consider unstructured point data.

Unstructured point-data modeling techniques have been widely discussed in related literature (Hoppe et al. 1992; Amenta et al. 1998; Bardinet et al. 1998; Schroeder et al. 1998; Gopi et al. 2000). Most proposed approaches construct polygon meshes that well approximate or interpolate the input point data. Due to the difficulties stemming from the lack of topology information within the data, however, three typical problems still exist:

- Strict sampling criteria are required in most proposed algorithms.
- A large amount of unnecessary faces are observed in the reconstructed meshes, especially if the input data is over-abundant.
- Severe aliases at sharp edges and corners frequently appear in the reconstructed meshes.

As a result, most reconstructed meshes need to be optimized using input point data in a postprocess. Note that although mesh simplification methods can clean up the mesh as well as mesh optimization, they cannot accurately reconstruct sharp features on the sampled physical surfaces.

One of the cardinal issues in mesh optimization using point data is the evaluation how well a mesh fits a given point set. In previous work, the criterion of the sum of squared distances from data points to the mesh is widely used (Hoppe et al. 1993; Kobbelt et al. 1998). However, for an iteratively renewed mesh, it is computationally expensive to calculate such squared distances sums in each step; thus, the mesh fitness evaluation becomes the bottleneck in the mesh optimization process.

We can achieve high-quality meshes from point clouds by applying a mesh reconstruction algorithm and a mesh optimization algorithm in turn, such as in the excellent work done in (Hoppe et al. 1992; Hoppe et al. 1993). However, since the mesh reconstruction algorithms and the mesh optimization and simplification algorithms are usually developed separately,

The Visual Computer (2003) 19:23–37
Digital Object Identifier (DOI) 10.1007/s00371-002-0162-2

the whole process is not as efficient as it could be. Addressing these problems in this work, we offer a solution.

1.1 Contributions

In this paper, we propose a fast algorithm to reconstruct optimized meshes of arbitrary topological type from unstructured point clouds. Specifically, we make the following contributions:

- We propose a new mesh evaluation criterion with which the mesh optimization process is speeded up. The quadric error metric has been demonstrated to be a powerful tool in mesh simplification (Garland and Heckbert 1997). In this work, we introduce the quadric error metric into our evaluation criterion to evaluate quickly and accurately how well an iteratively renewed mesh fits a set of given points.
- Based on the proposed evaluation criterion, we develop necessary operations to concisely integrate the functions of point data preprocessing (Hoppe et al. 1992), the marching cubes (MC) algorithm (Lorenzen and Cline 1987), mesh optimization, and mesh simplification (Garland and Heckbert 1997) into one simple algorithm.

1.2 Related work

Our proposed algorithm relates to three types of algorithms in mesh modeling: mesh reconstruction algorithms, mesh simplification algorithms and mesh optimization algorithms.

Mesh reconstruction algorithms can be classified according to whether the reconstructed mesh interpolates the input point data or not. The interpolating algorithms, such as the Delaunay triangulation algorithms (Amenta et al. 1998; Gopi et al. 2000), require strict sampling criteria (cf. Table 2) and do not work well at sharp edges and corners, either in theory or in practice. The approximating algorithms, such as the implicit methods (Hoppe et al. 1992; Curless and Levoy 1996) and the parametric methods (Qin and Terzopoulos 1996; Bardinet et al. 1998), can work with loose sampling criteria, but they are still problematic at sharp edges and corners. Moreover, the parametric methods are primarily developed to reconstruct surfaces of a prescribed topological type, such as disk-like, sphere-like or torus-like, and user intervention is needed to set up a patch network for surfaces of an arbitrary topological type; one exception is presented in (Eck and Hoppe 1996).

Mesh simplification is a well-studied problem. The promising algorithms are those that iteratively make local changes to the geometry. In particular, some publicly available algorithms in this class are the progressive meshes (Hoppe 1996), simplification envelopes (Cohen et al. 1996), JADE (Ciampalini et al. 1997) and Qslim (Garland and Heckbert 1997). Comparisons among different mesh simplification algorithms are sometimes difficult, since they strongly depend on the representation of the original mesh model. Besides the geometric simplification algorithms, the topological simplification algorithms are also considered (He et al. 1996; Guskov and Wood 2001). For a detailed survey on this topic, the reader is referred to (Garland 1999) and the references therein.

A typical work for mesh optimization using a set of given points is the energy minimization method proposed in (Hoppe et al. 1993). Their energy function consists of three terms. The distance energy E_{dist} is the sum of squared distances from data points S to the mesh, and the representation energy E_{rep} penalizes meshes with a large number of vertices. To calculate E_{dist}, $\forall \boldsymbol{s}_i \in S$, face f_j in the mesh closest to $\boldsymbol{s}_i$ needs to be found and the barycentric coordinate vector $\boldsymbol{b}_i$ of projection of $\boldsymbol{s}_i$ onto f_j is calculated. To make a numerical solution possible, it is assumed that the set of $\boldsymbol{b}_i$ associated with $\boldsymbol{s}_i$ is not changed in the process of mesh optimization with fixed topology. Since the minimization of $E_{\text{dist}} + E_{\text{rep}}$ may not converge to a stable solution, the spring energy E_{spring} is added for stability. In the energy minimization process, two nested loops are performed: an inner loop for minimization over vertex positions with fixed topology and an outer loop for minimization over topology. Although the results are satisfactory, the complete algorithm is difficult to implement and the computational complexity is expensive.

2 Overview of our approach

In this paper, we present a new algorithm to quickly generate optimized meshes from discrete data sets. In Sects. 3 and 4, we outline Hoppe et al.'s implicit method (1992) to build a signed distance field from input point data and then polygonize the isosurface using an MC algorithm (Schroeder et al. 1998) to

generate an initial mesh. The core of our approach is based on the observation that during the process of data implicitization and polygonization, besides the initial mesh, we can access other useful information with which the subsequent mesh optimization process can be simplified and accelerated. To achieve these benefits, the crucial point is that, as presented in Sect. 5, we show how to use the isosurface and the initial mesh to establish a new mesh evaluation criterion. Then based on our proposed criterion, in Sects. 6 and 7, we develop the necessary operations to fuse the MC algorithm and the mesh simplification algorithm (Garland and Heckbert 1997) into a simple, efficient optimization algorithm. Finally, comparisons with other related algorithms and our concluding remarks are presented in Sects. 8 and 9 respectively.

3 Point data preprocessing

Given a set of unstructured sample points $S = \{\boldsymbol{s}_1, \boldsymbol{s}_2, \cdots\}$, we use the implicit method from (Hoppe et al. 1992) to build a signed distance field function f, such that within a local region around S, $f(\boldsymbol{s}_i) = 0, \forall \boldsymbol{s}_i \in S$; thus, the underlying surface represented by S can be determined by the isosurface $f = 0$. Our variation of Hoppe et al.'s implicit approach is summarized in the rest of this section.

3.1 Distance field representation

For each sample point $\boldsymbol{s}_i \in S$, k neighboring points nearest to $\boldsymbol{s}_i$, denoted by $Nbhd(\boldsymbol{s}_i)$, are identified. Usually k is set to be 4∼8. From $Nbhd(\boldsymbol{s}_i)$ a tangent plane $Tp(\boldsymbol{s}_i)$ is indicated by solving a least squares fitting problem. $Tp(\boldsymbol{s}_i)$ is represented by the sample point $\boldsymbol{s}_i$ together with a unit normal vector $\boldsymbol{n}(\boldsymbol{s}_i)$; that is, $Tp(\boldsymbol{s}_i) = (\boldsymbol{s}_i, \boldsymbol{n}(\boldsymbol{s}_i))$. It is necessary to orient the tangent plane at each sample point. To achieve consistent orientation, $\forall \boldsymbol{s}_i \in S$, every neighboring point $\boldsymbol{s}_j \in Nbhd(\boldsymbol{s}_i)$ is connected to $\boldsymbol{s}_i$; this leads to a directed graph $G = (V, E)$. Each edge $(\boldsymbol{s}_i, \boldsymbol{s}_j) \in E$ in the graph G is weighted by assigning a cost $1 - \left|\boldsymbol{n}(\boldsymbol{s}_i)^T \cdot \boldsymbol{n}(\boldsymbol{s}_j)\right|$. Starting with an arbitrary vertex, the minimum-spanning tree (MST) is extracted from the weighted graph G. Subsequently, the MST is traversed in a depth-first search to propagate the prescribed tangent plane orientation. Then the following steps are taken to determine a signed distance field function f which assigns to any point $\boldsymbol{p} \in \boldsymbol{R}^3$ a value $f(\boldsymbol{p})$:

1. Find the sample point $\boldsymbol{s}_k \in S$ nearest to point $\boldsymbol{p}$.
2. If point $\boldsymbol{p}$ lies in a local region around $\boldsymbol{s}_k$, i.e. $\|\boldsymbol{p} - \boldsymbol{s}_k\| \leq r$, where r is an influence radius, then $f(\boldsymbol{p}) = (\boldsymbol{p} - \boldsymbol{s}_k)^T \cdot \boldsymbol{n}(\boldsymbol{s}_k)$.
3. Else $f(\boldsymbol{p}) = \infty$.

The prime advantage of this distance field representation is that it puts no restrictions on the object's topological type (cf. Fig. 1).

3.2 Hole filling

Usually the sample points do not completely cover the object, e.g. some portions of the physical surface are inaccessible to the sensor; thus, holes or gaps may appear in the reconstructed mesh. Any promising algorithm should possess the property of hole filling.

Note that in signed distance field generation, the influence radius r controls the extent of the tangent plane at each sample point. Hoppe et al. (1992) use a fixed influence radius r; therefore, to truly reconstruct the surface, their sampling criterion requires the data to be uniformly sampled (cf. Table 2). To cope with non-uniform and incompleted data, we use an adaptive influence radius r. On the hole boundary, we can choose one sample point whose tangent plane covers the hole correctly; then appropriately setting the value of the chosen point's r will make the isosurface $f = 0$ pass the hole continuously. In our implementation, the influence radius r can be set either automatically or interactively:

Automatic setting. When we search $Nbhd(\boldsymbol{s}_i)$ for each $\boldsymbol{s}_i \in S$, we set r at $\boldsymbol{s}_i$ being the minimum value of radius with which a sphere centered at $\boldsymbol{s}_i$ contains $Nbhd(\boldsymbol{s}_i)$.

Interactive setting. Users can interactively control the output mesh shape (with the aid of polygonization) by assigning different values of influence radii to different regions. Note that user intervention is inevitable in the ambiguous situation that either there is indeed a hole or that this hole should be closed.

Provided with the adaptive influence radius, our sampling criterion needs and only needs, on the physical surface, the tangent plane at each sample point $\boldsymbol{s}_i \in S$ to be indicated correctly by $Nbhd(\boldsymbol{s}_i)$, and the two-sided Hausdorff distance between the physical surface and the combination (controlled by influence

radii) of all the tangent planes to be under a predefined tolerance.
Examples of mesh shape control (hole filling) using adaptive influence radius r are illustrated in Fig. 1 (automatic), Fig. 3 (interactive) and Fig. 4 (interactive for the underneath section of the bunny data).

3.3 Acceleration

There are two expensive operations in this data preprocessing procedure: (1) $\forall \boldsymbol{s}_i \in S$, find $Nbhd(\boldsymbol{s}_i)$ in S; (2) extract the MST from a weighted graph G. In our implementation, to speed up the spatial k-nearest-neighbour searching process, we use a hierarchical spatial partitioning scheme, a range tree (Berg et al. 1997): a layered 3D range tree with n elements uses $O(n \log^2 n)$ storage, and combined with a fractional cascading technique, it can report k nearest neighbors for an arbitrary inquiry point $\boldsymbol{p} \in \boldsymbol{R}^3$ in $O(\log^2 n + k)$ time. Note that the $O(k)$ empirical time complexity of the searching algorithm of Hoppe et al. (1992), based on the assumption that the data is uniformly sampled, is no longer true for nonuniform data. For MST extraction from a weighted graph $G = (V, E)$, we apply the Prim's algorithm with a Fibonacci heap as its priority queue (Cormen et al. 1997) to improve the running time to $O(e + n \lg n)$. Noting that the edge number e in G is at most nk, the time complexity is actually $O(n \lg n)$.

4 Initial mesh generation

Notice that given a signed distance field function f, the underlying surface represented by S is determined by the isosurface $f = 0$. We use an MC algorithm (Schroeder et al. 1998) to polygonize the isosurface $f = 0$ to obtain an initial mesh that well approximates S.
The MC algorithm can be performed either with a fixed cube size or, more complicatedly, with an adaptive cube size that is locally proportional to surface detail (Bloomenthal et al. 1997). Adaptive methods depend upon a recursive subdivision scheme like octree, which leads to the potential discontinuity (crack) problem in the vicinity of boundaries across different levels of resolution. In our implementation, we use a fixed-resolution scheme for its simplicity and robustness (Montani et al. 1994). To capture the full details of the isosurface $f = 0$, the prescribed cube size should be less than or equal to the finest detail size in the point data S, i.e., $\min\{\|\boldsymbol{s}_i - \boldsymbol{s}_j\| | \forall \boldsymbol{s}_i, \boldsymbol{s}_j \in S, i \neq j\}$. Due to the potential discontinuity of the signed distance field function itself, the initial mesh generation is somewhat sensitive to the prescribed cube size; holes or gaps may appear in the mesh. In this case, the topological simplification methods (He et al. 1996; Guskov and Wood 2001) have to be applied.
Note that, as illustrated in Figs. 1, 3 and 4, there are two principal defects in the resulting initial meshes:

(1) These meshes are not optimal in areas of low curvature variation.
(2) Severe aliases are observed at sharp edges and corners.

The first defect is due to the algorithmic structure of the MC algorithm: each active cube that intersects the isosurface is processed separately, and a corresponding polygonal patch is generated locally. The second defect results from the fact that in each active cube a (trilinear) interpolation function is used to find a piecewise linear approximation to the original surface. It is worth noting that decreasing the cube size can only reduce the size of aliases but cannot remove them; meanwhile, the face number will increase dramatically.

5 A new mesh evaluation criterion

Aimed at obtaining a precise and compact representation, the initial mesh must be optimized based on the original sample points S. In previous work, the criterion of the sum of squared distances from S to the mesh is widely used, e.g. the distance energy E_{dist} in Hoppe et al. (1993). Although the calculation of E_{dist} for an iteratively renewed mesh can be localized using spatial partition methods (Hoppe et al. 1993; Kobbelt et al. 1998), some assumptions have to be made, and thus inaccuracy is introduced into their methods.
In this work, we propose a new evaluation criterion that is accurate and is easy to calculate. Noting that polygonization is a sampling process, we treat the set of vertices in the initial mesh as a set of resampling points S' over the true physical surface. Since the initial mesh is generated in full detail using the MC algorithm with a fixed cube size, the resample S' is somewhat uniform and a little denser than the original sample S. Furthermore, since all the vertices in the initial mesh lie on the edges

of active cubes, the resample S' is not accurate and is just another approximate sample of the true surface.

It is important to note that the underlying true surface is represented by the set of sample points S equipped with associated tangent planes and influence radii. Given this interpretation, we now have two sets of sample points: one is S that represents the true surface; the other is S' that is organized into a structured form, the initial mesh.

For mesh optimization, we take full advantage of the true surface information related to S and the structure information related to S'. We find that calculating the sum of squared distances from S to the initial mesh is one of possible ways to evaluate how well S and S' fit with each other; great efficiency can be achieved if we evaluate the fitness in a reverse way, calculating the sum of squared distances from S' to the tangent planes that are associated with S.

To offer an accurate evaluation, the tangent planes associated with S must be bounded in a local region $D \subset \boldsymbol{R}^3$ around S. This localization property is explicitly guaranteed by the adaptive influence radius of each sample point $\boldsymbol{s}_i \in S$, and the following steps offer an efficient computational method to use our proposed evaluation criterion, provided that S' is a good and dense approximation of S.

First, we relate points in S to points in S' in the following way:

1. $\forall \boldsymbol{s}'_i \in S'$,attach an empty point set $Attach(\boldsymbol{s}'_i)$ to $\boldsymbol{s}'_i$.
2. $\forall \boldsymbol{s}_i \in S$,find the point $\boldsymbol{s}'_j \in S'$ nearest to $\boldsymbol{s}_i$ and add $\boldsymbol{s}_i$ into $Attach(\boldsymbol{s}'_j)$.

The above operation partitions the set S into subsets $Attach(\boldsymbol{s}'_i)$, $\forall \boldsymbol{s}'_i \in S'$, which satisfy the following properties:

1. Exhaustiveness, i.e. $\bigcup_i Attach(\boldsymbol{s}'_i) = S$.
2. Mutual exclusion, i.e. $Attach(\boldsymbol{s}'_i) \cap Attach(\boldsymbol{s}'_j) = \emptyset$, for any $i \neq j$.

Based on these two properties, the first note is that if $Attach(\boldsymbol{s}'_i) = \emptyset$, $\boldsymbol{s}'_i \in S'$, then $\boldsymbol{s}'_i$ is an auxiliary vertex in the initial mesh, and we can remove it from the mesh without any influence on the mesh quality. This is derived from the fact that the true surface is the local combination of tangent planes associated with S, but not the initial mesh. The second point to note is that the sum $Sum(S')$ of squared distances from S' to the tangent planes $\bigcup_i Tp(\boldsymbol{s}_i)$ can be decomposed and thus achieved by atomic calculations, i.e.

$$Sum(S') = \sum_i Sum(\boldsymbol{s}'_i),\ \boldsymbol{s}'_i \in S'.$$

The atomic calculation $Sum(\boldsymbol{s}'_i)$ is the sum of squared distances from point $\boldsymbol{s}'_i$ to its attached tangent planes in $Attach(\boldsymbol{s}'_i)$, i.e.

$$Sum(\boldsymbol{s}'_i) = \sum_{\boldsymbol{s}_j \in Attach(\boldsymbol{s}'_i)} SD_j(\boldsymbol{s}'_i),$$

where $SD_j(\boldsymbol{s}'_i)$ is the squared distance from $\boldsymbol{s}'_i$ to the tangent plane $Tp(\boldsymbol{s}_j) = (\boldsymbol{s}_j, \boldsymbol{n}(\boldsymbol{s}_j))$. $SD_j(\boldsymbol{s}'_i)$ is calculated by

$$\begin{aligned} SD_j(\boldsymbol{s}'_i) &= \left(\boldsymbol{n}(\boldsymbol{s}_j)^T \cdot (\boldsymbol{s}'_i - \boldsymbol{s}_j)\right)^2 \\ &= \boldsymbol{s}'_i \otimes Q(\boldsymbol{A}_j, \boldsymbol{b}_j, c_j) \quad , \end{aligned} \tag{1}$$

where $\boldsymbol{A}_j$ is a 3×3 matrix determined by vector direct product $\boldsymbol{n}(\boldsymbol{s}_j) \cdot \boldsymbol{n}(\boldsymbol{s}_j)^T$, $\boldsymbol{b}_j$ is a 3-vector determined by $\boldsymbol{A}_j \cdot \boldsymbol{s}_j$, and c_j is a scalar determined by $(\boldsymbol{n}(\boldsymbol{s}_j)^T \cdot \boldsymbol{s}_j)^2$. We define the operator $\otimes$ as $\boldsymbol{s}'_i \otimes Q(\boldsymbol{A}_j, \boldsymbol{b}_j, c_j) = \boldsymbol{s}_i'^T \cdot \boldsymbol{A}_j \cdot \boldsymbol{s}'_i - 2 \cdot (\boldsymbol{b}_j)^T \cdot \boldsymbol{s}'_i + c_j$. The triple $Q(\boldsymbol{A}_i, \boldsymbol{b}_i, c_i)$ is in fact the quadric error metric defined by Garland and Heckbert (1997). Thus, $Sum(\boldsymbol{s}'_i)$ can be calculated as

$$\begin{aligned} Sum(\boldsymbol{s}'_i) &= \sum_{\boldsymbol{s}_j \in Attach(\boldsymbol{s}'_i)} SD_j(\boldsymbol{s}'_i) \\ &= \sum_{\boldsymbol{s}_j \in Attach(\boldsymbol{s}'_i)} \left(\boldsymbol{s}'_i \otimes Q(\boldsymbol{A}_j, \boldsymbol{b}_j, c_j)\right) \\ &= \boldsymbol{s}'_i \otimes Q\Big(\sum_{\boldsymbol{s}_j \in Attach(\boldsymbol{s}'_i)} \boldsymbol{A}_j, \sum_{\boldsymbol{s}_j \in Attach(\boldsymbol{s}'_i)} \boldsymbol{b}_j, \sum_{\boldsymbol{s}_j \in Attach(\boldsymbol{s}'_i)} c_j\Big) \\ &= \boldsymbol{s}'_i \otimes Q(\boldsymbol{A}'_i, \boldsymbol{b}'_i, c'_i). \end{aligned} \tag{2}$$

The value $Sum(S') = \sum_i Sum(\boldsymbol{s}'_i)$ offers us a quantitative attribute to evaluate the mesh quality. Since

(1) the calculation of the $Sum(S')$ can be exactly decomposed into a set of atomic calculations associated with each mesh vertex s', and

(2) most mesh optimization and simplification operators, e.g. the vertex removal and edge collapse operators used in our approach, have an effect only on the local area incident to the related vertex or edge,

the fitness between point clouds and a locally changed mesh can be evaluated accurately by few atomic calculations restricted to the local affected area. Since the atomic calculation is based on quadric Q, the evaluation is also fast.

6 Mesh optimization

Based on our proposed evaluation criterion, we develop necessary operations to concisely integrate different functional modules into a simple algorithm. To optimize the initial mesh, $\forall \boldsymbol{s}'_i \in S'$, we fill in $Attach(\boldsymbol{s}'_i)$ and transform the set $Attach(\boldsymbol{s}'_i)$ to a triple $Q(\boldsymbol{A}'_i, \boldsymbol{b}'_i, c'_i)$ using Eqs. (1,2). Based on the simple calculation with Q, we optimize the initial mesh using vertex removal and edge collapse operators (cf. Fig. 1). The mesh quality is further improved by taking the triangle aspect ratio into account, as presented in Sect. 6.5.

6.1 Optimization prerequisite

Our proposed evaluation criterion is based on the calculation of $Sum(S')$. One potential problem is that before starting evaluation, $\forall \boldsymbol{s}'_i \in S'$, we must find $Attach(\boldsymbol{s}'_i)$ in S. This search operation may be computationally expensive. Fortunately, we find that this operation can be efficiently accomplished in linear time during the initial mesh generation process, as presented below.

We use an MC algorithm to polygonalize the isosurface $f = 0$ to obtain the initial mesh. In this process, all the active cubes that intersect the isosurface are traced out and the values of the eight vertices of each active cube are calculated for cell polygonization. In our case, to calculate the signed distance value at the position of any cube vertex $\boldsymbol{v}$, the sample point $\boldsymbol{s}_i \in S$ closest to $\boldsymbol{v}$ is found and is associated with $\boldsymbol{v}$ by searching a range tree in $O(\log^2 n + k)$ time. Provided that every initial mesh vertex $\boldsymbol{s}' \in S'$ lies on an edge of the active cubes and the cube size is set to be a little smaller than the size of finest detail in S, the vertex $\boldsymbol{s}'_j \in S'$ closest to the point $\boldsymbol{s}_i \in S$ must be among the local set of initial mesh vertices that lie on the edges of the active cubes incident to $\boldsymbol{v}$ associated with $\boldsymbol{s}_i$.

To this end, the vertex set S' is partitioned into two subsets: the set of auxiliary vertices

$$Aux(S') = \left\{ \boldsymbol{s}'_i \middle| Attach(\boldsymbol{s}'_i) = \emptyset, \boldsymbol{s}'_i \in S' \right\},$$

and the set of active vertices

$$Act(S') = S' \backslash Aux(S').$$

6.2 Position optimization of active mesh vertices

As pointed out in Sect. 5, the resample S' is not accurate. Therefore, in our mesh optimization process, we first find optimized positions for $Act(S')$; this operation minimizes the $Sum(S')$ with fixed mesh topology. Since

$$Sum(S') = \sum_i Sum(\boldsymbol{s}'_i),$$

this is equivalent to calculating an optimized position for each $\boldsymbol{s}'_i \in Act(S')$ by minimizing

$$Sum(\boldsymbol{s}'_i) = \boldsymbol{s}'_i \otimes Q(\boldsymbol{A}'_i, \boldsymbol{b}'_i, c'_i).$$

Since the error metric Q is quadric, finding its minimum is a linear problem: by equating its derivative to zero,

$$\frac{\partial Sum(\boldsymbol{s}'_i)}{\partial \boldsymbol{s}'_i} = \boldsymbol{A}'_i \cdot \boldsymbol{s}'_i - \boldsymbol{b}'_i = 0,$$

the optimized position for $\boldsymbol{s}'_i$ is simply $(\boldsymbol{A}'_i)^{-1} \cdot \boldsymbol{b}'_i$. However, in our case, the rank of matrix $\boldsymbol{A}'_i$ is usually deficient and thus $\boldsymbol{A}'_i$ is not invertible. This problem can be solved by adding geometric constraints that force the problem to be overconstrained and lead to a real symmetric positive definite matrix $\boldsymbol{A}'_i$ (Lindstrom and Turk 1999); one constraint from the requirement that the triangles in the mesh are well shaped is presented in Sect. 6.5. Lindstrom (2000) proposes a slightly different approach that performs a singular value decomposition (SVD) of $\boldsymbol{A}'_i$, which leads to the pseudoinverse when $\boldsymbol{A}'_i$ is not invertible. Provided that S' is a good approximation of S, in our method, we use the following heuristic to find the optimized position for $\boldsymbol{s}'_i$: we calculate $Sum(\boldsymbol{s}'_i)$ at each position of $\boldsymbol{s}_j \in Attach(\boldsymbol{s}'_i)$, and take the optimized position as $\boldsymbol{s}'_i = \boldsymbol{s}_k$, where

$$k = \arg\min_j \left\{ Sum(\boldsymbol{s}_j) \middle| \ \forall \boldsymbol{s}_j \in Attach(\boldsymbol{s}'_i) \right\}.$$

Compared with analytic minimization using SVD, this scheme is fast and stable. In addition, this

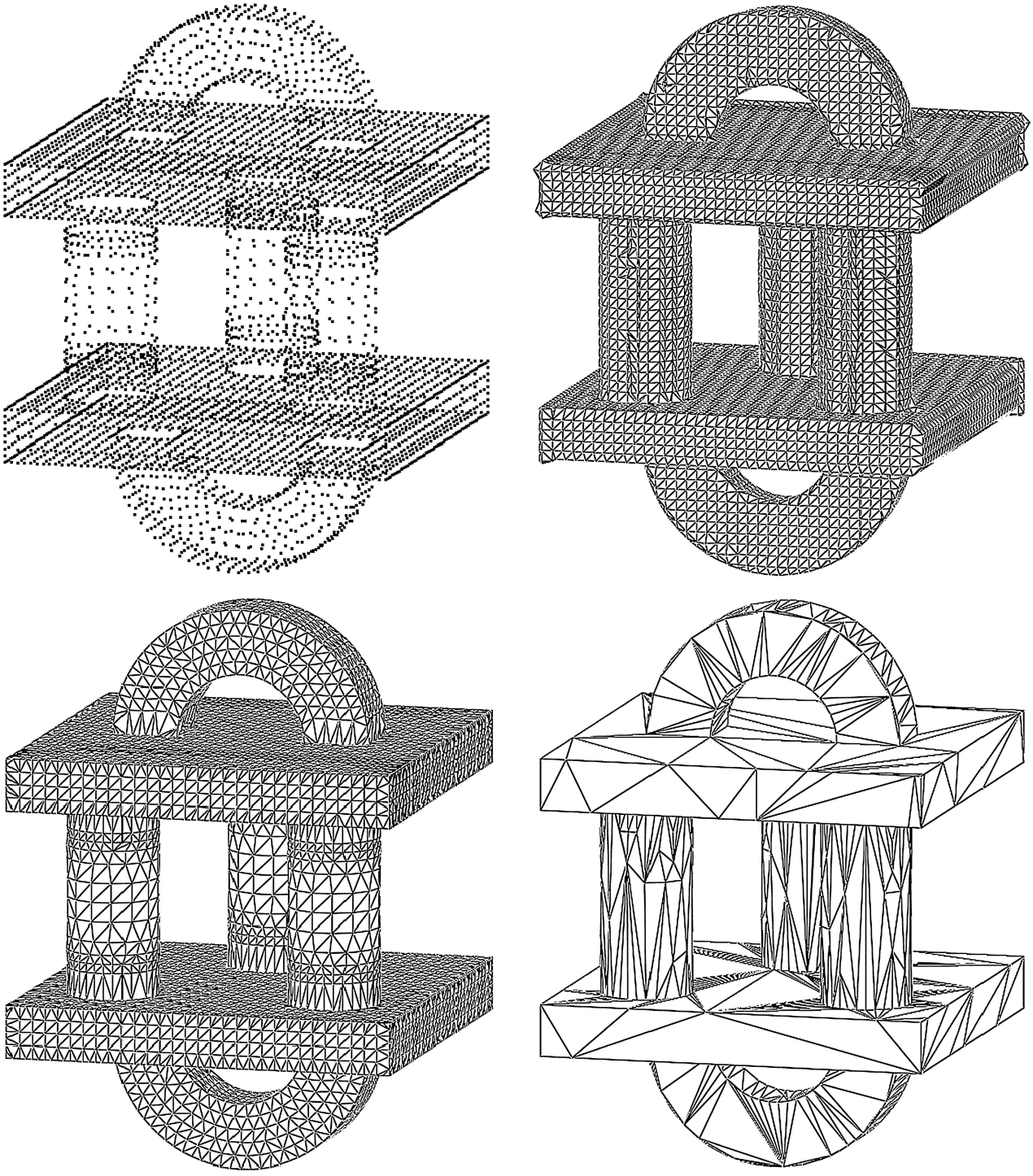


Fig. 1. The optimized mesh reconstruction process of a mechanical part. *Top left*: a non-uniform sample S (6528 points). *Top right*: initial mesh generation (55 268 faces). *Bottom left*: mesh optimization (13 255 faces) by optimizing the positions of $Act(S')$ and removing the set of $Aux(S')$ from the mesh. *Bottom right*: mesh optimization (1158 faces) using the edge collapse operator

scheme makes the set $Act(S')$ be a subset of sample S.

6.3 Auxiliary mesh vertices removal

Recall that the true surface is represented by the local combination of tangent planes associated with sample S, and the initial mesh only offers structure information for mesh optimization. Therefore, removing all the auxiliary vertices from the mesh has no influence on $Sum(S')$ and optimizes the mesh in the topological sense. However, removing $Aux(S')$ in different ways will result in different triangulations of the point set $Act(S')$; any careless removal will lead to serious distortion of the mesh's geometric shape.

In our method, we use the retiling method (Turk 1992) to remove $Aux(S')$ from the mesh. Turk's retiling method is originally developed for mesh simplification. First, a set of new vertices is distributed over the polygonal surface using point repulsion. Then a mutual tessellation incorporating old and new vertices is formed. Finally, by carefully taking topological consistency and triangle shape into consideration, a simplified mesh with good quality is achieved by removing old vertices from mutual tessellation. In our case, the initial mesh, auxiliary vertices and active vertices exactly correspond to the mutual tessellation, old vertices and new vertices respectively.

To remove an old vertex $\boldsymbol{v}$, the local region formed by triangles incident to $\boldsymbol{v}$ needs to be retessellated. Given that the topological consistency is checked, Turk's retiling method uses the triangle shape as the quality measure to optimize the tessellation. That is why we first optimize the positions for $Act(S')$ and subsequently remove $Aux(S')$ from the mesh.

Recall that after position optimization for $Act(S')$, we have $Act(S') \subset S$. In our implementation of Turk's method, if the neighboring vertices of an auxiliary vertex $\boldsymbol{v}$ contain $k \geq 1$ points $\boldsymbol{s}_i \subset S$, $i = 1, \ldots, k$, prior to the triangle shape, we use the normal information $\boldsymbol{n}(\boldsymbol{s}_i)$ as the quality measure to optimize the retessellation centered at $\boldsymbol{v}$. This operation is important, especially at the places where sharp features are presented.

6.4 Mesh optimization using edge collapse

After the removal of auxiliary mesh vertices, we further optimize the mesh using the edge collapse operator. For each edge $(\boldsymbol{s}'_i, \boldsymbol{s}'_j)$ in the mesh, we calculate an optimal position $\bar{\boldsymbol{s}}'$ for the potential collapse $(\boldsymbol{s}'_i, \boldsymbol{s}'_j) \rightarrow \bar{\boldsymbol{s}}'$ by minimizing the edge cost function

$$\Delta Sum(S') = \bar{\boldsymbol{s}}' \otimes Q(\boldsymbol{A}'_i + \boldsymbol{A}'_j, \boldsymbol{b}'_i + \boldsymbol{b}'_j, c'_i + c'_j) - \boldsymbol{s}'_i \otimes Q(\boldsymbol{A}'_i, \boldsymbol{b}'_i, c'_i) - \boldsymbol{s}'_j \otimes Q(\boldsymbol{A}'_j, \boldsymbol{b}'_j, c'_j).$$

We then put all the edges into a priority queue keyed on edge cost with the minimum cost edge at the top. For mesh optimization, we iteratively extract the edge $(\boldsymbol{s}'_i, \boldsymbol{s}'_j)$ from the top of the queue, perform $(\boldsymbol{s}'_i, \boldsymbol{s}'_j) \rightarrow \bar{\boldsymbol{s}}'$, and locally update the collapse information for all the edges involving $\boldsymbol{s}'_i$ and $\boldsymbol{s}'_j$. For each collapse $(\boldsymbol{s}'_i, \boldsymbol{s}'_j) \rightarrow \bar{\boldsymbol{s}}'$, we set the triple $Q(\bar{\boldsymbol{A}}', \bar{\boldsymbol{b}}', \bar{c}')$ associated with $\bar{\boldsymbol{s}}'$ to be $Q(\boldsymbol{A}'_i + \boldsymbol{A}'_j, \boldsymbol{b}'_i + \boldsymbol{b}'_j, c'_i + c'_j)$. Given the properties of exhaustiveness and mutual exclusion of the set $Attach(\boldsymbol{s}'_i)$, $\boldsymbol{s}'_i \in S'$, the sum of Q is accurate for evaluation using $\Delta Sum(S')$. Note that we do not need to explicitly calculate $Sum(S')$; instead, we only need the change $\Delta Sum(S')$.

The iteration process is terminated when the top edge in the queue has a cost $\Delta Sum(S') > \varepsilon$. Ideally, the threshold ε should be set to zero. In our implementation, due to the precision of computer's floating-point operation and the precision of calculating normal vectors in point data preprocessing, we normalize all the models in a unit cube and set $\varepsilon = 1.0 \times 10^{-6}$.

The performance of our optimization algorithm depends upon how we implement the priority queue. In our implementation, we use the Fibonacci heap again as the priority queue. The running times for operations that use a Fibonacci heap consist of the following: creating a new Fibonacci heap with n elements takes $\Theta(n)$ amortized time; extracting an element with minimum cost takes $\Theta(1)$ time; and renewing an element in the heap with a changed cost takes $\Theta(1)$ time.

Compared with the mesh optimization approach in Hoppe et al. (1993), our approach has the following advantages:

(1) We use a much faster fitting evaluation criterion based on quadric Q.
(2) In Hoppe et al. (1993), the mesh topology optimization and the mesh vertex position optimization are performed separately in two nested loops; in our approach, we use a single edge-collapse operation to simultaneously optimize

the mesh topology and the mesh vertex positions.

6.5 Triangle shape optimization

In some applications, long, thin "sliver" triangles are undesirable. To optimize the triangle shape, $\forall \boldsymbol{s}'_i \in S'$, we consider the following shape cost function:

$$Sp(\boldsymbol{s}'_i) = \sum_{j \in N_1(i)} \left\| \boldsymbol{s}'_i - \boldsymbol{s}'_j \right\|^2,$$

where $N_1(i)$ are 1-ring neighbors of $\boldsymbol{s}'_i$. Minimizing this shape cost function is related to maximizing the area to perimeter ratios of the resulting triangles (Lindstrom and Turk 1999). This shape cost function can be reformulated in the quadric form:

$$\begin{aligned} Sp(\boldsymbol{s}'_i) &= \sum_{j \in N_1(i)} \left\| \boldsymbol{s}'_i - \boldsymbol{s}'_j \right\|^2 \\ &= \sum_{j \in N_1(i)} (\boldsymbol{s}'_i - \boldsymbol{s}'_j)^T \cdot (\boldsymbol{s}'_i - \boldsymbol{s}'_j) \\ &= \sum_{j \in N_1(i)} (\boldsymbol{s}_i'^T \cdot \boldsymbol{s}'_i - 2\boldsymbol{s}_j'^T \cdot \boldsymbol{s}'_i + \boldsymbol{s}_j'^T \cdot \boldsymbol{s}'_j) \\ &= \sum_{j \in N_1(i)} (\boldsymbol{s}_i'^T \cdot \boldsymbol{I} \cdot \boldsymbol{s}'_i - 2\boldsymbol{s}_j'^T \cdot \boldsymbol{s}'_i + \boldsymbol{s}_j'^T \cdot \boldsymbol{s}'_j) \\ &= \sum_{j \in N_1(i)} \boldsymbol{s}'_i \otimes Q(\boldsymbol{I}, \boldsymbol{s}'_j, \boldsymbol{s}_j'^T \cdot \boldsymbol{s}'_j) \\ &= \boldsymbol{s}'_i \otimes Q\Big(\sum_{j \in N_1(i)} \boldsymbol{I}, \sum_{j \in N_1(i)} \boldsymbol{s}'_j, \sum_{j \in N_1(i)} \boldsymbol{s}_j'^T \cdot \boldsymbol{s}'_j \Big). \end{aligned}$$

One interesting note is that minimizing the shape function $Sp(\boldsymbol{s}'_i)$ leads to

$$\begin{aligned} \bar{\boldsymbol{s}}'_i &= \left(\sum_{j \in N_1(i)} \boldsymbol{I} \right)^{-1} \cdot \sum_{j \in N_1(i)} \boldsymbol{s}'_j \\ &= \frac{1}{n} \sum_{j \in N_1(i)} \boldsymbol{s}'_j = \boldsymbol{s}'_i + U(\boldsymbol{s}'_i), \end{aligned}$$

where $n = \#N_1(i)$ and $U(\boldsymbol{s}'_i) = \frac{1}{n} \sum_{j \in N_1(i)} \boldsymbol{s}'_j - \boldsymbol{s}'_i$. The $U(\boldsymbol{s}'_i)$ turns out to be the umbrella operator defined by Kobbelt et al. (1998): the umbrella operator is the discrete Laplacian of the 1-ring surrounding surface that is parameterized over a symmetrical configuration. Applying this operator minimizes the membrane energy and makes the mesh edge length uniform, so it is widely used as a mesh relaxation operator (Kobbelt et al. 1999; Kobbelt et al. 2000).

In our implementation, we use a weighted shape-cost function

$$\begin{aligned} WSp(\boldsymbol{s}'_i) &= \frac{\sqrt{3}}{4Area(\boldsymbol{s}'_i)} \sum_{j \in N_1(i)} \left\| \boldsymbol{s}'_i - \boldsymbol{s}'_j \right\|^2 \\ &= \frac{\sqrt{3}}{4Area(\boldsymbol{s}'_i)} \boldsymbol{s}'_i \\ &\quad \otimes Q\Big(\sum_{j \in N_1(i)} \boldsymbol{I}, \sum_{j \in N_1(i)} \boldsymbol{s}'_j, \sum_{j \in N_1(i)} \boldsymbol{s}_j'^T \cdot \boldsymbol{s}'_j \Big) \\ &= \boldsymbol{s}'_i \otimes Q(\boldsymbol{D}'_i, \boldsymbol{e}'_i, f'_i) \,, \end{aligned}$$

where $Area(\boldsymbol{s}'_i)$ is the surrounding area incident to the vertex $\boldsymbol{s}'_i$. We use the term $Area(\boldsymbol{s}'_i)$ to nondimensionalize the cost function and to normalize the shape cost in the area sense, such that the resulting shape cost is scale-invariant; the coefficient $\sqrt{3}/4$ is chosen to assign a cost 1 to a surrounding area formed by equilateral triangles.

Then in the position optimization of $Act(S')$, we find the optimized position $\bar{\boldsymbol{s}}'_i$ for $\boldsymbol{s}'_i$ by minimizing the objective function

$$\begin{aligned} F(\boldsymbol{s}'_i) &= \lambda Sum(\boldsymbol{s}'_i) + (1-\lambda) WSp(\boldsymbol{s}'_i) \\ &= \boldsymbol{s}'_i \otimes Q(\lambda \boldsymbol{A}'_i + (1-\lambda)\boldsymbol{D}'_i, \lambda \boldsymbol{b}'_i \\ &\quad + (1-\lambda)\boldsymbol{e}'_i, \lambda c'_i + (1-\lambda) f'_i) \\ &= \boldsymbol{s}'_i \otimes Q(\boldsymbol{L}'_i, \boldsymbol{m}'_i, n'_i) \,, \end{aligned}$$

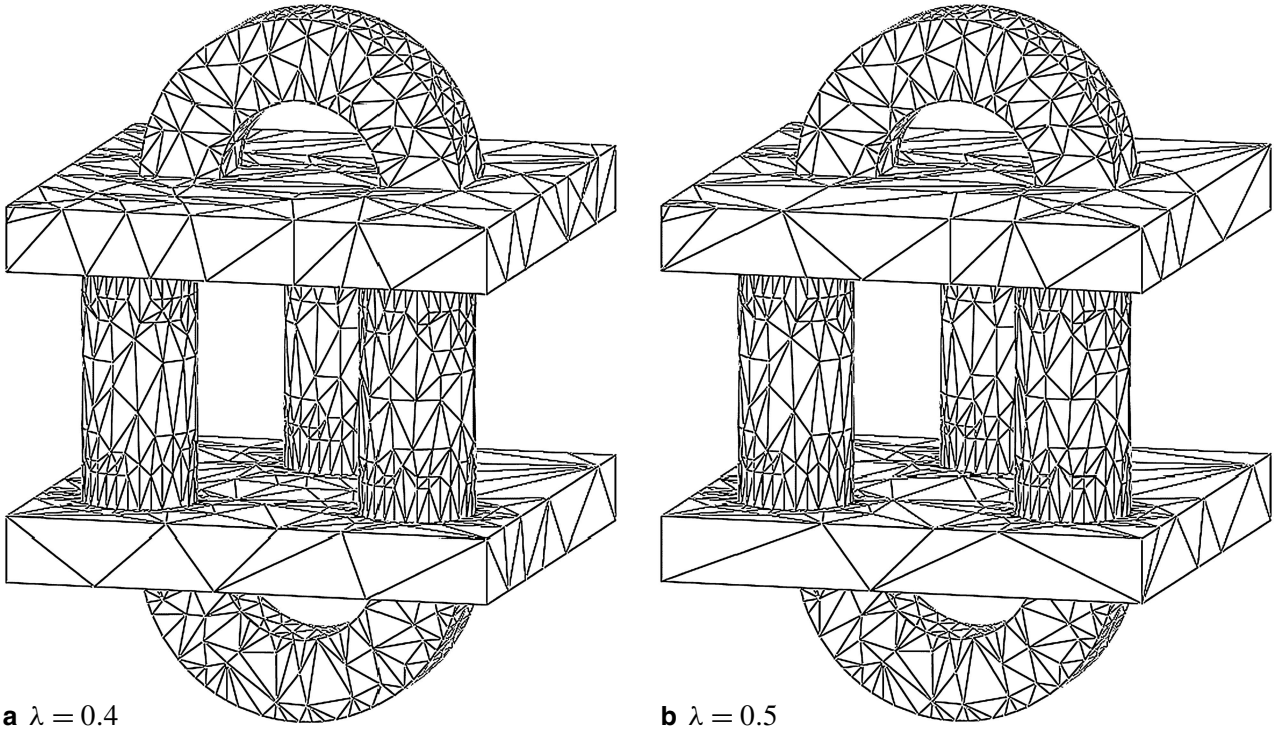
i.e. $\bar{\boldsymbol{s}}'_i = (\boldsymbol{L}'_i)^{-1} \cdot \boldsymbol{m}'_i$. The weight coefficient λ offers the users an intuitive parameter to balance the optimization of geometric fitting and the optimization of triangle shape.

Similarly, in the subsequent optimization using edge collapse, we calculate the optimal position $\bar{\boldsymbol{s}}'_i$ for any potential edge collapse $(\boldsymbol{s}'_i, \boldsymbol{s}'_j) \to \bar{\boldsymbol{s}}'$ by minimizing the objective function

$$F(\boldsymbol{s}'_i) = \lambda \Delta Sum(S') + (1-\lambda) WSp(\boldsymbol{s}'_i).$$

Then we assign to the edge $(\boldsymbol{s}'_i, \boldsymbol{s}'_j) \to \bar{\boldsymbol{s}}'$ a cost

$$\begin{aligned} \Delta Sum(S') = \bar{\boldsymbol{s}}' &\otimes Q(\boldsymbol{A}'_i + \boldsymbol{A}'_j, \boldsymbol{b}'_i + \boldsymbol{b}'_j, c'_i + c'_j) \\ &- \boldsymbol{s}'_i \otimes Q(\boldsymbol{A}'_i, \boldsymbol{b}'_i, c'_i) \\ &- \boldsymbol{s}'_j \otimes Q(\boldsymbol{A}'_j, \boldsymbol{b}'_j, c'_j). \end{aligned}$$



a $\lambda = 0.4$ **b** $\lambda = 0.5$

Fig. 2. Mesh optimization with triangle shape control using parameter λ

We illustrate one example of triangle shape optimization in Fig. 2.

7 Mesh simplification

Recently, with the complexity of object models increasing, large data sets and large face counts for the resulting optimized meshes become commonplace. For example, the point data in Fig. 3 consists of 151 321 sample points. The recovered optimized mesh consists of 252 042 triangles. To manipulate these highly detailed objects in real time, mesh simplification becomes important.

In our case, after mesh optimization (the collapsed edges have the cost $\Delta Sum(S') \leq \varepsilon$), we can simply keep extracting edges from the queue top and collapsing them to achieve mesh simplification.

Our mesh simplification is similar to the one in Garland and Heckbert (1997): we use the same quadric error metric and the same edge collapse operator. However, three major differences between our simplification and the Garland's simplification are as follows:

(1) Garland's method uses the initial mesh information to simplify the models, but our method uses the original point data information.
(2) Our method uses a different edge cost function to reflect the geometric error introduced into the approximation models.
(3) Although using the same quadric error metric, the summation of quadric matrices introduces inaccuracy into the Garland's method, since each face in the initial mesh contributes to all the quadric matrices of its three vertices; but in our method, equipped with the properties of exhaustiveness and mutual exclusion, the summation of Q is accurate.

Fig. 3 offers a visual interpretation of these differences.

8 Discussion and comparison with other methods

The complete algorithm including all the functions mentioned above has been fully implemented in C++ code on a PC platform with 256 Mb of memory

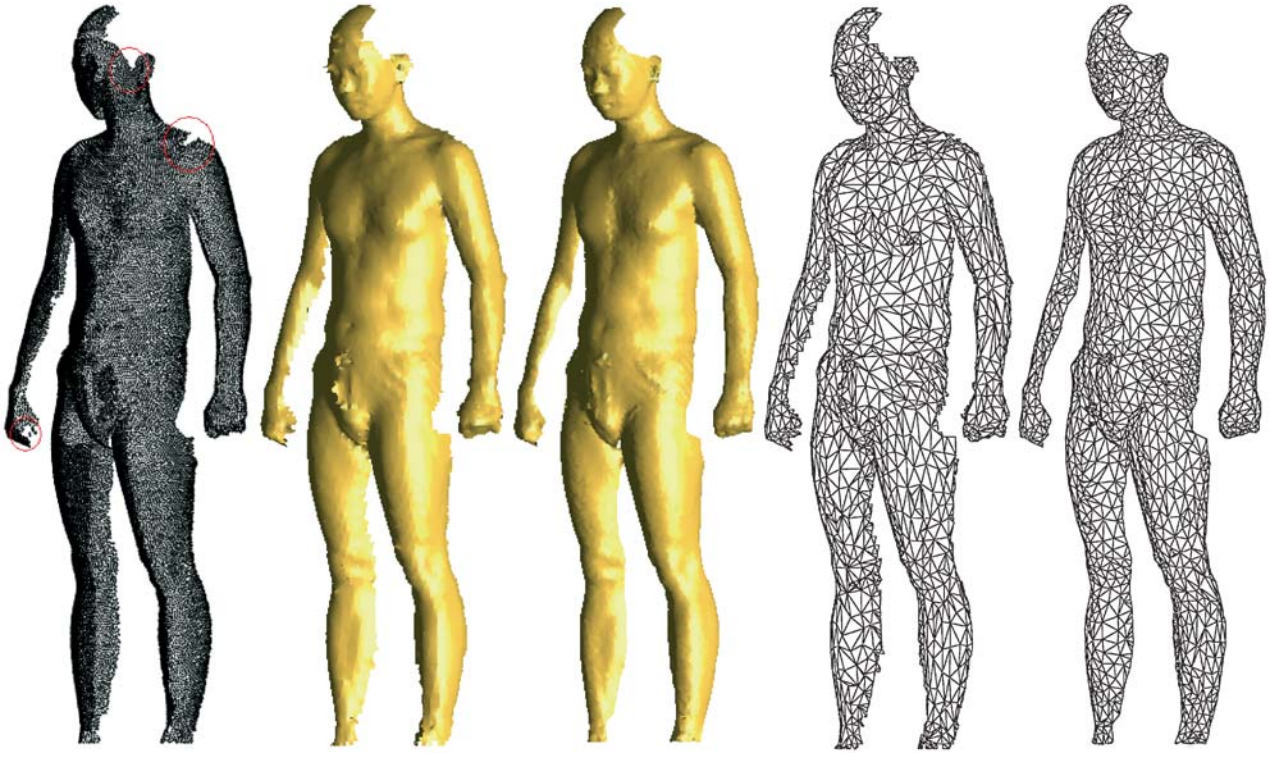


Fig. 3. A human body model. *Far left*: a set of unstructured sample points S (151 321 points). *Center left*: the Gouraud shading of the initial mesh (360 547 faces). *Center*: the Gouraud shading of the optimized mesh (252 042 faces). *Center right*: Garland's simplification using initial mesh information (4000 faces). *Far right*: our simplification using input point data information (4000 faces). Note that the wedge gaps in the circled regions in the data image are filled by an elongated influence radius of the sample point located at each wedge vertex

and a Pentium III processor running at 500 MHz. To make an easy comparison with other methods, we test our algorithm on some widely used point data obtained from the Internet, and the results are shown in Fig. 4. Table 1 summarizes the running time of all the models presented in this paper, using our proposed algorithm.

Our approach generates optimized meshes from unstructured sample points in a single algorithm, which relates to three types of algorithms: the mesh reconstruction algorithms, the mesh simplification algorithms and the mesh optimization algorithms.

We first compare the performance of our mesh reconstruction function with three typical mesh reconstruction algorithms: an approximating algorithm in Hoppe et al. (1992) and two interpolating algorithms in Amenta et al. (1998) and Gopi et al. (2000). Our reconstruction function is nearly the same as the algorithm in Hoppe et al. (1992), except that a 3D range tree and adaptive influence radii are used in our function to deal with non-uniform data. The algorithm complexities in Hoppe et al. (1992), Gopi et al. (2000) and ours are all $O(n \log n)$, since the MST needs to be traversed for tangent plane orientation; the algorithm complexity in Amenta et al. (1998) is $O(n^2)$, since the asymptotic complexity of 3D crust algorithm is $O(n^2)$. For an intuitive comparison, we collect the published information (sampling criteria, implementation platform and execution times) on the oil pump and bunny models, as presented in Table 2.

We then compare the performance of our mesh optimization and simplification functions with various simplification algorithms. Note that the effectiveness of different simplification algorithms strongly depends on the representation of the original mesh. We have presented the advantages of our simplification function over the simplification algorithm in Garland and Heckbert (1997) in Sect. 7 (cf. Fig. 3). Since both Garland's algorithm and ours use the same quadric error metric and the same edge collapse op-

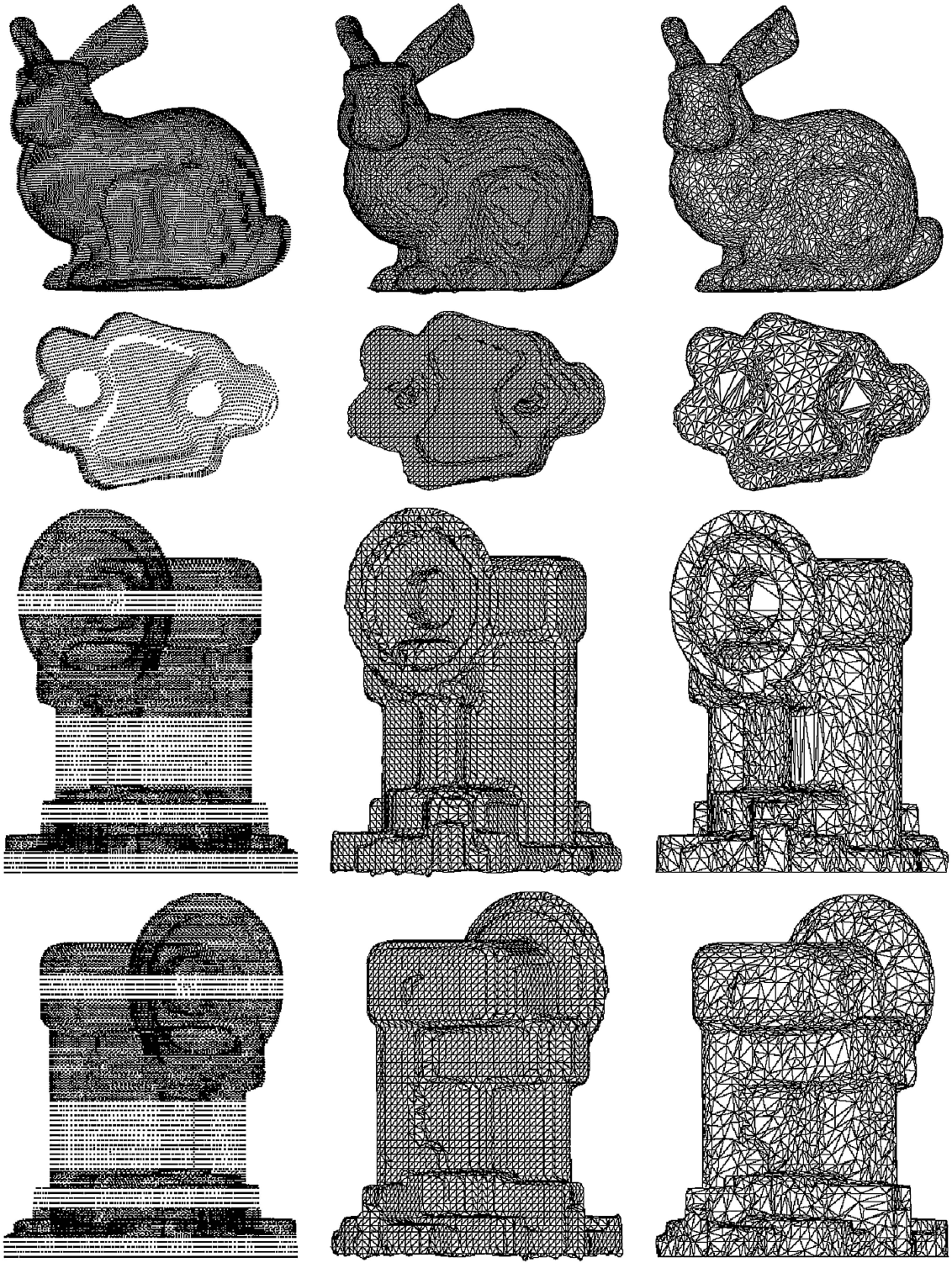


Fig. 4. More results by applying our algorithm on some publicly available data. The *first column* is the set of sample points. The *second column* is the set of reconstructed initial meshes. The *last column* is the set of optimized meshes. The Stanford bunny model is represented in the *first two rows*; the *second row* is the bottom view of the *first row*. The oil pump model is represented in the *last two rows*; the *last row* is the back view of the *third row*

Table 1. Running time of all the models presented in this paper

Model	Point number	Running time (s) Point data preprocessing	Initial mesh generation	Optimized mesh generation
Mech. part	6528	6.0	4.0	2.5
Human body	151 321	662.0	145.0	46.0
Stanford bunny	35 947	43.0	19.0	3.2
Oil Pump	30 937	38.0	18.0	4.0

Table 2. Comparisons of performance of various mesh-reconstruction algorithms

Algorithms	Scheme	Sampling criteria	Complexity	Implementation platform	Running time (s) Bunny	Oil pump
Amenta et al. (1998)	Interpolating	The sampling density is inversely proportional to the distance to the medial axis.	$O(n^2)$	SGI Onyx with 512 Mb of memory	1380	–
Gopi et al. (2000)	Interpolating	The sampling density at a point $s \in S$ along a particular direction in the tangent plane is inversely proportional to the directional curvature at s.	$O(n \log n)^{\dagger}$	SGI Onyx with a R1000 processor running at 194 MHz	18.64	20.99
Hoppe et al. (1994)	Approximating	The sample S is uniform (ρ-dense), i.e. any sphere with radius ρ and centered on sampled surface M contains at least one sample point s in S, and the value of ρ is the size of finest surface feature that can be recovered.	$O(n \log n)^{\dagger}$	SGI Indigo workstation	–	104
Ours*	Approximating	The tangent plane at each point s on the physical surface M is indicated correctly by $Nbhd(s) \in S$, and the two-sided Hausdroff distance between the physical surface and the local combination of all the tangent planes is under a predefined tolerance.	$O(n \log n)^{\dagger}$	PC with 256 Mb of memory and a Pentium III processor running at 500 MHz	62.0	56.0

* Including functions of data preprocessing, initial mesh generation and optimization prerequisite.
† Excluding the k-nearest-neighbor searching process; in our case it is $O(n \log^2 n)$ for non-uniform data.

erator, these two algorithms must run at the same rate. Given two fixed original meshes – one simple and the other complex – the performance of six published simplification algorithms, i.e., mesh optimization (Hoppe et al. 1992), progressive meshes (Hoppe 1996), simplification envelopes (Cohen et al. 1996), JADE (Ciampalini et al. 1997), Qslim (Garland and Heckbert 1997) and memoryless simplification (Lindstrom and Turk 1998), are evaluated in detail in Lindstrom and Turk (1999) using the Metro tool (Cignoni et al. 1998).

Our proposed algorithm can quickly generate high-quality meshes of arbitrary topological type from unstructured point data. Besides the application to complex geometric model acquisition, our proposed optimization technique can also find application in the area of volume graphics (Kaufman et al. 1993). Volume representation of geometric objects is developed as a powerful graphics primitive (Frisken et al. 2000): it is independent of the object topology, and many operations based on this representation, e.g. Boolean operations, are easy to implement. To

convert the volume representation of an object to an explicit surface representation, in which the intrinsic surface properties like curvatures are accessible and the surface-based operations like smoothing can be performed, two basic problems have to be solved: (1) how to find an adequate set of surface samples; and (2) how to organize them into a structured form and optimize the results. To develop an efficient conversion algorithm, the surface normal information can be incorporated into the distance field representation; this evolves the traditional signed distance field into the directed distance field (Kobbelt et al. 2001). Then using a standard marching-cubes algorithm (Lorensen and Cline 1987; Schroeder et al. 1998), followed by our proposed optimization technique, high-quality meshes can be obtained.

9 Conclusion

In this paper, we propose a fast algorithm to reconstruct high-quality meshes from unstructured sample points. The key contribution of our proposed algorithm is that we introduce a new mesh evaluation criterion into unstructured point-data modelling techniques that unifies the functions of data preprocessing, isosurface polygonization, mesh optimization and mesh simplification into one simple algorithm. The results show that our proposed algorithm is time and space efficient and, thus, can process large data sets on a low-cost PC platform.

Acknowledgements. The authors would like to thank Dr. Shiang-Fong Chen for her constructive advice at an early stage of this research work. The authors would also like to thank Susan Peverelle for her careful proofreading of the manuscript. In Fig. 4, the Stanford bunny data is obtained from http://graphics.stanford.edu/ and the oil pump data is obtained from http://research.microsoft.com/∼hoppe/.

References

1. Amenta N, Bern M, Kamvysselis M (1998) A new Voronoi-based surface reconstruction algorithm. In: Proceedings of SIGGRAPH '98. ACM SIGGRAPH, pp 415–421
2. Bardinet E, Cohen L, Ayache N (1998) A parametric deformable model to fit unstructured 3D data. Comput Vision Image Understanding 71(1):39–54
3. Berg M, Kreveld M, Overmars M, Schwarzkopf O (1997) Computational geometry: algorithm and applications. Springer, Berlin Heidelberg New York
4. Bloomenthal J, Bajaj C, Blinn J, Cani-Gascuel M, Rockwood A, Wyvill B, Wyvill G (1997) Introduction to implicit surfaces. Morgan Kaufmann, San Francisco
5. Ciampalini A, Cignoni P, Montani C, Scopigno R (1997) Multi-resolution decimation based on global error. Vis Comput 13(5):228–246
6. Cignoni P, Rocchini C, Scopigno (1998) Metro: measuring error on simplified surfaces. Comput Graph Forum 17(2):167–174
7. Cohen J, Varshney A, Manocha D, Turk G, Weber H, Agarwal P, Brooks F, Wright W (1996) Simplification envelopes. In: Proceedings of SIGGRAPH '96. ACM SIGGRAPH, pp 119–128
8. Cormen T, Leiserson C, Rivest R (1997) Introduction to algorithms. MIT Press,
9. Curless B, Levoy M (1996) A volumetric method for building complex models from range images. In: Proceedings of SIGGRAPH '96. ACM SIGGRAPH, pp 303–312
10. Eck M, Hoppe H (1996) Automatic reconstruction of B-spline surfaces of arbitrary topological type. In: Proceedings of SIGGRAPH' 96. ACM SIGGRAPH, pp 325–334
11. Frisken S, Perry R, Rockwood A, Jones T (2000) Adaptively sampled distance fields: a general representation of shape for computer graphics. In: Proceedings of SIGGRAPH '00. ACM SIGGRAPH, pp 249–254
12. Garland M (1999) Multiresolution modeling: survey and future opportunities. Eurographics '99, State of the art reports, pp 111–131
13. Garland M, Heckbert P (1997) Surface simplification using quadric error metrics. Proceedings of SIGGRAPH '97. ACM SIGGRAPH, pp 209–216
14. Gopi M, Krishnan S, Silva CT (2000) Surface reconstruction based on lower dimensional localized Delaunay triangulation. Comput Graph Forum (Eurographics '00 Proceedings) 19(3):467–478
15. Guskov I, Wood Z (2001) Topological noise removal. In: Proceedings Graphics Interface '01, pp 19–26
16. He T, Hong L, Varshney, Wang S (1996) Controlled topology simplification. IEEE Trans Vis Comput Graph 2(2):171–184
17. Hoppe H, DeRose T, Duchamp T, McDonald J, Stuetzle W (1992) Surface reconstruction from unorganized points. In: Proceedings of SIGGRAPH '92. ACM SIGGRAPH, pp 71–78
18. Hoppe H, DeRose T, Duchamp T, McDonald J, Stuetzle W (1993) Mesh optimisation. In: Proceedings of SIGGRAPH '93. ACM SIGGRAPH, pp 19–26
19. Hoppe H (1994) Surface reconstruction from unorganized points. PhD thesis, Department of Computer Science and Engineering, University of Washington
20. Hoppe H (1996) Progressive meshes. In: Proceedings of SIGGRAPH '96. ACM SIGGRAPH, pp 99–108
21. Kaufman A, Cohen D, Yagel R (1993) Volume graphics. IEEE Computer 26(7):51–64
22. Kobbelt L, Bareuther T, Seidel H-P (2000) Multiresolution shape deformations for meshes with dynamic vertex connectivity. Comput Graph Forum (Eurographics '00 Proceedings) 19(3):249–259
23. Kobbelt L, Campagna S, Seidel H-P (1998) A general framework for mesh decimation. In: Proceedings Graphics Interface '98, pp 43–50
24. Kobbelt L, Campagna S, Vorsatz J, Seidel H-P (1998) Interactive multi-resolution modeling on arbitrary meshes.

In: Proceedings of SIGGRAPH '98. ACM SIGGRAPH, pp 105–114
25. Kobbelt L, Botsch M, Schwanecke U, Seidel H-P (2001) Feature sensitive surface extraction from volume data. In: Proceedings of SIGGRAPH '01, ACM SIGGRAPH, pp 57–66
26. Kobbelt L, Vorsatz J, Labsik U, Seidel H-P (1999) A shrink wrapping approach to remeshing polygonal surfaces. Comput Graph Forum (Eurographics '99 Proceedings), 18(3):119–129
27. Lindstrom P, Turk G (1998) Fast and memory efficient polygonal simplification. In: Proceedings of Visualization '98, pp 279–286
28. Lindstrom P, Turk G (1999) Evaluation of memoryless simplification. IEEE Trans Vis Comput Graph 5(2):98–115
29. Lindstrom P (2000) Out-of-core simplification of large polygonal models. In: Proceedings of SIGGRAPH '00. ACM SIGGRAPH, pp 259–262
30. Lorensen W, Cline H (1987) Marching cubes: a high resolution 3D surface construction algorithm. Comput Graph (SIGGRAPH '87 Proceedings) 21(4):163–169
31. Montani C, Scateni R, Scopigno R (1994) A modified look-up table for implicit disambiguation of marching cubes. Vis Comput 10(6):353–355
32. Qin H, Terzopoulos D (1996) D-NURBS: A physics-based framework for geometric design. IEEE Trans Vis Comput Graph 2(1):85–96
33. Schroeder W, Martin H, Lorensen B (1998) The visualization toolkit, 2nd edn. Prentice Hall, Upper Saddle River, N.J.
34. Turk G. (1992) Re-tiling polygonal surfaces. Comput Graph (SIGGRAPH '92 Proceedings) 26(2):55–64

YONG-JIN LIU is currently a PhD student at Hong Kong Univeristy of Science and Technology (HKUST). He received his BEng (1998) in mechano-electronic engineering, Tianjin University, PRC, and his MPhil (1999) in mechanical engineering at HKUST. His research interests include geometric modeling, computer graphics and assembly modeling.

DR MATTHEW MF YUEN is the director of the Technology Transfer Center at HKUST. He received his PhD in mechanical engineering from the University of Bristol, UK, in 1977. He has extensive research experience in design and manufacturing automation and received the 1987 Edwin Walker Prize from the Institution of Mechanical Engineers, UK. His research interests include intelligent CAD/CAM systems, soft-object modeling, electronic packaging and polymer processing.