The official orginal version is available at http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6338255

A Divide-and-Conquer Approach to Quad Remeshing

Muyang Zhang, Jin Huang, Xinguo Liu, and Hujun Bao

Abstract—Many natural and man-made objects consist of simple primitives, similar components, and various symmetry structures. This paper presents a divide-and-conquer quadrangulation approach that exploits such global structural information. Given a model represented in triangular mesh, we first segment it into a set of submeshes, and compare them with some predefined quad mesh templates. For the submeshes that are similar to a predefined template, we remesh them as the template up to a number of subdivisions. For the others, we adopt the wave-based quadrangulation technique to remesh them with extensions to preserve symmetric structure and generate compatible quad mesh boundary. To ensure that the individually remeshed submeshes can be seamlessly stitched together, we formulate a mixed-integer optimization problem and design a heuristic solver to optimize the subdivision numbers and the size fields on the submesh boundaries. With this divider-and-conquer quadrangulation framework, we are able to process very large models that are very difficult for the previous techniques. Since the submeshes can be remeshed individually in any order, the remeshing procedure can run in parallel. Experimental results showed that the proposed method can preserve the high-level structures, and process large complex surfaces robustly and efficiently.

Index Terms—Quad remeshing, divide-and-conquer, segmentation, mixed-integer optimization

1 Introduction

Quad mesh is very useful for its bilinear nature and superior performance in many graphics and scientific computing applications. However, it is very challenging to remesh a surface into a provably high-quality quad mesh, because there are very diverse requirements on the shape, the orientation and the size of the quads, the alignment of features, and the regularity of the mesh in term of singularity points. Some of the recently developed methods [1], [2] take into account these low-level requirements via global optimization techniques; however, the optimization cost may increase rapidly, and even become unacceptable, when processing large complex models.

On the other hand, many natural and man-made objects consist of simple primitives, similar components, and various symmetric structures. Taking advantages of the high-level structural information would be helpful for developing a better quadrangulation method. However, the previous methods are not aware of such high-level structural information. They conduct the same kind of computation on all components, which wastes a lot of computation resources. Moreover, the global structure might not be preserved because different results might be generated for those similar parts.

In this paper, we propose a top-down quadrangulation method that exploits the high-level structural information. Our primary goal is to develop a practical solution to robust and efficient remesh large complex surfaces with many components and regular structures. The basic idea to achieve this goal is "divide and conquer," i.e., segmenting the input surface into small, easy-to-remesh submeshes, so as to reduce the complexity of the problem and reuse the quadrangulation results on the other similar parts. For the submeshes, we developed two remeshing techniques. One is template based, which remeshes a submesh by subdividing a predefined quad mesh template. The benefits of templates include: 1) saving on computation cost, 2) control of the quad mesh structure, and 3) ability to generate the same quad mesh for similar and repeated components. The other is an extension of the wave-based quadrangulation method [2], which can preserve symmetric structures and generate compatible boundaries.

A key issue in developing a divide-and-conquer quadrangulation method is how to ensure that the individually remeshed submeshes can be stitched together seamlessly. As a technical contribution, we formulate a mixed-integer optimization problem by a set of novel stitching constraints, and design a heuristic solver to optimize the subdivision numbers and the size fields on the boundaries of the submeshes, such that a seamless stitching can be guaranteed. The proposed stitching technique allows us to convert a large quad remeshing task into many small remeshing tasks, and is substantially different from the previous stitching techniques [1], [2], [3], which find some proper transition functions across the cutting boundary to obtain global parameterizations.

The proposed method runs in a top-down manner, thus is suitable for processing large complex models, particularly CAD models. Besides, it has several useful features:

1. It can generate the same quadrilateral structures for similar components.
2. It can preserve the symmetry structures in the model.

• *The authors are with the State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310058, China.*
E-mail: {zhangmuyang, hj, xgliu, bao}@cad.zju.edu.cn.

Manuscript received 29 Nov. 2011; revised 17 June 2012; accepted 24 Sept. 2012; published online 22 Oct. 2012.
Recommended for acceptance by B. Levy.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2011-11-0299.
Digital Object Identifier no. 10.1109/TVCG.2012.301.

3. Its quadrangulation ability can be extended by predefining richer quad mesh templates.
4. It can run in parallel to remesh the submeshes.

2 Previous Work

Surface quadrangulation has received a lot of attention in geometry related fields. A direct approach to quadrangulation is by parameterization and regular grid sampling [4]. To deal with models of complex topology, Marinov and Kobbelt [5] partition the input mesh into nearly developable regions via variational shape approximation. Lai et al. [6] developed a local updating technique to incrementally transform triangles into quads. Though conceptually simple, it cannot transform all triangles. Recently, Remacle et al. [7] find the set of triangle pairs that forms the best possible quadrilaterals with the constraint of not leaving any remaining triangle in the mesh based on the Blossom algorithm [8].

To generate high-quality quad meshes, many methods directly trace a network of curves on the surface. For example, Alliez et al. [9] trace the minimum and maximum curvature lines. Marinov and Kobbelt [10] improved this algorithm with an efficient proximity query method for tracing. Ray et al. [11] and Kälberer et al. [3] proposed two globally smooth parameterization methods for arbitrary surfaces, which can be naturally applied in tracing the curve network. Since these methods construct the quad incrementally, pure quad mesh is not guaranteed.

Recently, some global chart creation techniques were proposed for pure quad remeshing. Dong et al. [12] developed the first automatic algorithm to partition the input mesh into a set of quadrilateral charts using the Morse-Smale complex (MSC) of Laplacian eigenfunctions. Huang et al. [13] extended this method to enable the control over the chart size. Their results showed that the MSC produces high-quality quadrilateral charts. Creating a global quadrilateral charting is equivalent to determining the singularity points. Tong et al. [14] introduced a user-assisted method to interactively identify the singularity points, while Bommes et al. [1] developed a mixed-integer solver to automatically optimize the singularity points. Myles et al. [15] used quadrilateral T-mesh to partition the input surface into smaller number of charts and achieved better feature alignment result. Tierny et al. [16] transfer the global structure (singularity and patch layout) from a quadrilateral mesh to another mesh to be remeshed as the domain of cross parameterization.

The ability to control the orientation and the size of the quads is important for a quadrangulation method to align the quads with the geometric features and sample the input mesh adaptively. Most of existing methods can orient the quads using a vector field over the surface. Such vector field can be obtained by computing and optionally smoothing the principal curvature directions [17], [18], [19]. The sharp feature lines on the input mesh are then taken into account during the smoothing step. Dong et al. [20] suggested to use the gradient vectors of a harmonic scalar field to avoid the noisy singularity points in the principal directions. To control the size of the quads, Ray et al. [11] augmented the surface with a scalar field to determine the spacing between the adjacent iso-parameter lines. Zhang et al. [2] extended this method to enable anisotropic control over the size. They also proposed a wave construction method to exactly align the quad mesh with the feature lines. Bommes et al. [1] achieved feature alignment by using a mixed-integer solver to determine the corresponding integer parameters. Kovacs et al. [21] proposed a simple method to control the shape of the quads by curvature to decrease the remeshing error.

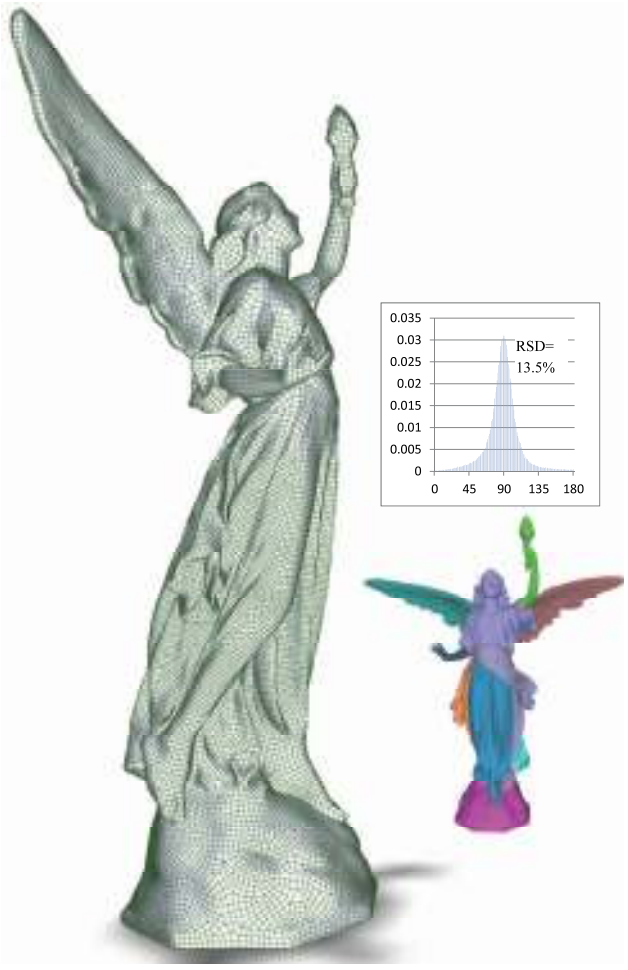


Fig. 1. The Lucy model remeshed by our divide-and-conquer method. The input model has $1,413K$ vertices and are segmented into 10 submeshes. It takes about 35 minutes to remesh. The result consists of 37,794 quads, and 4,686 singularity points. Hausdorff distance is $d_H = 0.0028$.

A number of recent algorithms aim at simplifying and optimizing a quad mesh [22], [23], [24]. Similar to triangle mesh simplification, quad mesh simplification techniques incrementally decimate some quads by applying some decimation operators, e.g., polychord collapse, quadrilateral collapse, and doublet collapse. These operators are carefully designed to maintain the quadrilateral structure of quad mesh. Bommes et al. [24] optimize the global quadrilateral structure by detecting the topological helices and applying a novel grid preserving simplification operator to remove them. Li et al. [25] optimize the shape of faces by adjusting the number and spacings of grid lines in each quad patch extracted from the input quad mesh. However, these methods do not optimize the global distribution of the singularity points.

1. Segment the input model into sub-meshes, and classify them as: matched, repeated, or misc sub-meshes
2. Initialize vector fields and size fields on the model
3. Set up and solve stitching constraints on the boundaries between the adjacent sub-meshes
4. Remesh the matched sub-meshes by subdividing the templates, and remesh the others using an improved wave-based method
5. Merge the remeshed sub-meshes and optimize the result

Fig. 2. The outline of the quadrangulation method.

The latest work in above shows that significant progresses in quad remeshing have been made. Most of them are bottom-up remeshing approaches. The two-step method by Marinov and Kobbelt [5] can be considered as a top-down approach; however, it is restricted to planar patch approximation, and is not flexible to control. Lizier et al. [26] decompose the surface into a set of triangular or quadrilateral patches according to the features as the base domain of global parameterization. As a major difference with the previous work, we propose a flexible top-down quadrangulation approach.

3 Overview

We begin with an overview to present our quadrangulation method. As many previous methods did, we assume that the input model is represented as a triangulated manifold surface $\mathcal{M}$. As shown in Fig. 2, our method consists of five main steps.

3.1 Segmentation

First we segment the input model into a set of submeshes: $\mathcal{M} = \cup\{M_i\}$, and analyze their shape and structures. We define a set of quad mesh templates to match with the submeshes, e.g., the rectangle, disc, and cylinder templates shown in Fig. 7 for the CAD models in our experiments.

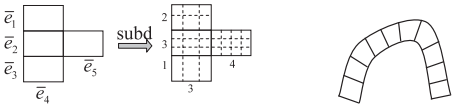
If a submesh M is similar to a predefined quad mesh template Q, and can be remeshed as Q up to a number of subdivisions, then M is identified as a *matched submesh*. The others that cannot be matched with any template are called *unmatched submeshes*.

If a group of submeshes have very similar geometry and can be remeshed in the same way up to small deformation, then they are called *repeated submeshes*. For each group of repeated submeshes, we randomly choose one as the representative submesh of the group.

For convenience, the submeshes other than matched and repeated are called *misc submeshes*. As segmentation and submesh analysis is not the focus of this work, we postpone the details to Section 6.

3.2 Initialize Vector Fields and Size Fields

After segmentation, we compute and smooth the principal curvature directions [11], [17] to generate a pair of vector fields $(\vec{X}, \vec{Y})$ for orienting the quads. To align the quads with boundaries of the submeshes, we replace the corresponding principal directions with the edge direction of the boundaries before smoothing. Then, we adopt the curl minimization-based method [2] to define a pair of size fields (μ, η) for anisotropically controlling the size of the quads.



(a) subdivision example (b) a quad strip

Fig. 3. (a) A simple example of quad mesh subdivision. The original quad mesh have five subdivision DoFs, namely $\kappa[\bar{e}_1], \kappa[\bar{e}_1], \ldots, \kappa[\bar{e}_5]$. Subdivision result is generated with $\kappa[\bar{e}_1] = 2$, $\kappa[\bar{e}_2] = 3$, $\kappa[\bar{e}_3] = 1$, $\kappa[\bar{e}_4] = 3$, and $\kappa[\bar{e}_5] = 4$. (b) A quad strip. By our subdivision rule, the edges perpendicular to the strip direction have the same subdivision number.

3.3 Set Up and Solve Stitching Constraints

Before remeshing, it is necessary to properly constrain the boundaries of adjacent submeshes to have the same number of quads, such that they can be seamlessly stitched together.

We propose four kinds of stitching constraints. The first is defined between a matched submesh and an unmatched submesh, which constrains the subdivision numbers of the corresponding quad mesh template to be compatible with the size fields of the unmatched submesh at the joint boundary (see Section 4.1).

The second is defined for unmatched submeshes, which constrain the size field on the boundary to be compatible with quad mesh (see Section 4.2).

The third is defined between two adjacent matched submeshes, which constrains the subdivision numbers for the corresponding quad mesh to produce the same number of quads at their joint boundaries (see Section 4.3).

The last is defined between two repeated submeshes, which constrains their size fields to be compatible at the corresponding boundaries, such that the quad meshing result can be reused on both of them (see Section 4.4).

Combining these constraints with the wave-based quadrangulation method [2], we obtain a mixed-integer optimization problem. Then, we design a heuristic solver to the optimization problem (see Section 4.6).

3.4 Remesh Submeshes

With the optimized subdivision numbers and size fields, we then remesh the submeshes individually.

The matched submeshes are remeshed by subdividing their templates. Since the template quad mesh can be regarded as the parameterization space of the matched submesh, we first subdivide the template, and then map the new vertices and the new edges of the template onto the matched submesh. To avoid introducing new singularity points, we keep a simple rule to subdivide the quad mesh templates: any two opposite edges of any quad must be simultaneously subdivided into the same number of subedges. Fig. 3 shows an example subdivided by this rule, where $\kappa[\bar{e}]$, called the *subdivision number* of edge $\bar{e}$, denotes the number of the subedges that $\bar{e}$ is subdivided into. Under this rule, one can see that subdivision number $\kappa[\bar{e}]$ has a transitive property along quad strips, as shown in Fig. 3b. Therefore, the subdivision DoF of a quad mesh equals the number of quad strips. In addition, for any open quad mesh, their boundary edges can be grouped into pairs, such that the two edges in every pair must have the same subdivision number.

The repeated submeshes and the misc submeshes are remeshed using an improved wave-based quadrangulation

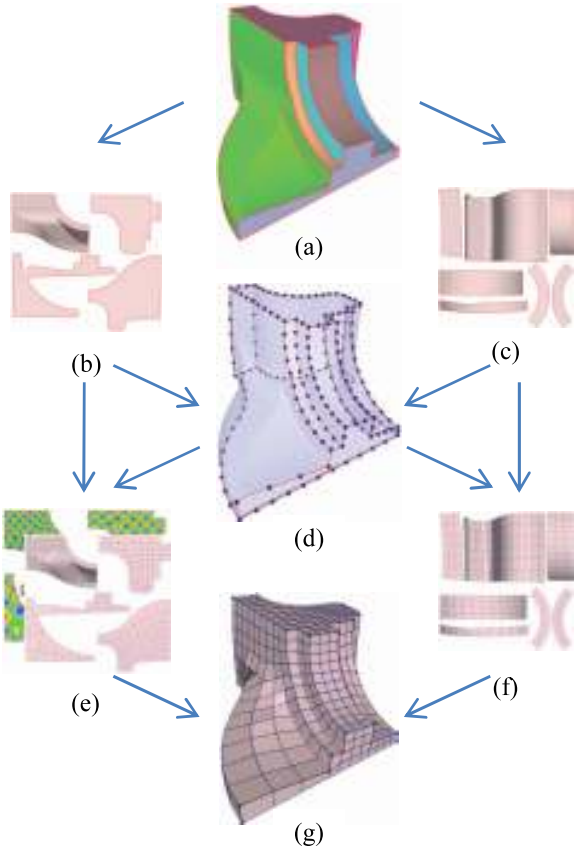


Fig. 4. Remeshing data flow. (a) Segment the original model into submeshes (shown in different color); (b) unmatched submeshes; (c) matched submeshes; (d) a solution to seamless stitching; (e) remeshing the unmatched submeshes; (f) remeshing the matched submeshes; (g) the final result.

method, which can generate prescribed number of quads along the boundary for seamless stitching, and preserve symmetric structures (see Section 5).

3.5 Optimize the Quad Mesh

Finally, we merge the remeshed submeshes together, and adopt the iterative relaxation method [2] to optimize the quadrangulation result, which is basically a Laplacian smoothing.

Fig. 4 shows the workflow of remeshing a Fandisk model. In the above steps, wave construction is divided to the submeshes, while field smoothing, curl minimization, and constraint system solving are performed globally.

4 Stitching Problem

In this section, we consider the stitching problem between two adjacent submeshes M and M'. Let $C = e_1 e_2 \dots e_k \subseteq \partial M \cap \partial M'$ be a *boundary curve* between M and M', where ∂ denotes the boundary of a mesh, and e_1, e_2, ..., and e_k are the consecutive edges of both ∂M and $\partial M'$.

Remember that we have constrained the vector fields to be aligned with the direction of the boundary edges during the initialization step (step 2 in Section 3). For convenience, we assume that $\vec{X}$ is aligned with the boundary edge direction. Then, the size field in the direction of an edge e_i on C is $\mu[e_i]$.

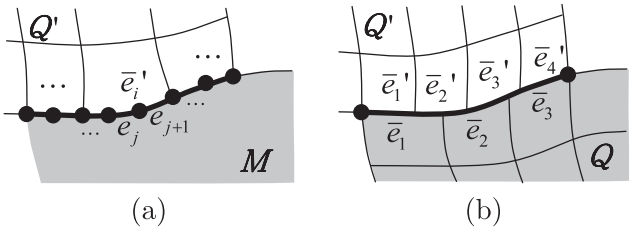


Fig. 5. Illustration of joint boundary of two submeshes. (a) A matched submesh and a unmatched submesh; (b) two matched submeshes.

To design the stitching method, we distinguish the following three cases for M and M':

1. Case 1. M is *not* a matched submesh, but M' is.
2. Case 2. M and M' are not matched submeshes.
3. Case 3. M and M' are matched submeshes.

To ensure seamless stitching, we must generate the same number of quads on curve C for M and M'. For a matched submesh, the number of quads on curve C is related to the subdivision numbers. For an unmatched submesh, it is related to the size field that we used to construct the wave function. We will setup the relationships between these variables in the following sections:

4.1 Constraints between Matched Submesh and Unmatched Submeshes

In Case 1, M' is a matched submesh, but M is not. As illustrated in Fig. 5a, Q' is the quad mesh template for M'. For each boundary quad edge $\bar{e}'_i$ of Q', we can identify all of triangle edges that have overlaps with it (Throughout this paper, e without a bar denotes the edge of the triangle mesh, while $\bar{e}$ with a bar denotes the edge of the template quad mesh).

Since the size field is an indication of the quad size, the quad number on $\bar{e}'_i$ can be approximated by $\sum_{e_j \subset \bar{e}'_i} \frac{|e_j|}{\mu[e_j]}$, where $|e_j|$ denotes the length of edge e_j. Therefore, we have the following stitching constraint:

$$\kappa[\bar{e}'_i] = \sum_{e_j \subset \bar{e}'_i} \frac{|e_j|}{\mu[e_j]}. \tag{1}$$

4.2 Constraints between Unmatched Submeshes

In Case 2, both M and M' are not matched submeshes. Similarly to Case 1, the quad number on C can be also approximated by: $\sum_{j=1}^{k} |e_j|/\mu[e_j]$. This number should be a positive integer. Denote it by $\kappa[C]$, then we have the following constraint:

$$\kappa[C] = \sum_{j=1}^{k} |e_j|/\mu[e_j], \kappa[C] \in \mathbb{Z}^+, \tag{2}$$

where $|e_j|$ denotes the length of edge e_j.

Process all constraints in (1) and (2), and assemble them into a matrix form, then we have

$$\kappa = \mathbf{C}_1 \mu^{-1} + \mathbf{C}_2 \eta^{-1} \equiv \mathbf{C} e^{-x}, \tag{3}$$

where $\mathbf{C}_1$ and $\mathbf{C}_2$ are the coefficient matrices, $\mathbf{C} = (\mathbf{C}_1 \mathbf{C}_2)$, and x is the column vector composed of $\tilde{\mu}$ and $\tilde{\eta}$, the logarithms of the size fields that Zhang et al. [2] defined to prescribe the size of the quad mesh.

We assume that the boundary edge e_j either completely lie in a quad edge or completely outside of it without loss of generality. If there is an edge crossing two quad edges, we can presplit it such that this assumption holds. Therefore, each column in matrix $\mathbf{C}$ has at most one nonzero value.

In addition, as pointed out in some previous work [22], [27], it is impossible to quadrangulate a surface with an odd number of vertices on the boundary. This can be shown by considering an opened quad mesh with N_q quads and N_i internal edges. Then, the number of the boundary edges is $4N_q - 2N_i$, which is obviously an even number.

Therefore, for each unmatched submesh M, we set up the following integer constraint:

$$\sum_{C \subset \partial M} \kappa[C] = 2\kappa[M], \kappa[M] \in \mathbb{Z}^+, \tag{4}$$

where $\kappa[M]$ denotes the integer variable associated with submesh M, which equals a half of the boundary quad edge number of M.

4.3 Constraints between Matched Submeshes

In Case 3, both M and M' are matched submeshes. As illustrated in Fig. 5b, Q and Q' denote the quad mesh templates of M and M'.

According to the 1-1 correspondence between Q and M, we can find all of the boundary quad edges that overlap with curve C. Denote the edges by $\bar{e}_1$, $\bar{e}_2$, ..., and $\bar{e}_m$ in consecutive order. Without loss of generality, we assume that these edges are all in curve C, i.e., $\bar{e}_1 \cup \cdots \cup \bar{e}_m = C$, as illustrated in Fig. 5b. Otherwise, we can presubdivide quad mesh template Q by subdividing $\bar{e}_1$ and/or $\bar{e}_m$ at the end point of C to clip the edges, such that this assumption holds. When Q is subdivided, the number of the subquads connected with curve C equals the summed subdivision numbers of the quad edges, i.e., $\sum_{i=1}^{m} \kappa[\bar{e}_i]$.

Similarly, the number of the subquads connected with curve C equals $\sum_{i=1}^{m'} \kappa[\bar{e}'_i]$, where $\bar{e}'_i \in \partial Q'$ and $\bar{e}'_1 \cup \cdots \cup \bar{e}'_{m'} = C$.

To ensure that Q and Q' can be seamlessly stitched together after subdivisions, the subdivided quad meshes must have the same number of edges on the boundary curve C. Therefore, we set up the following subdivision constraint:

$$\sum_{i=1}^{m} \kappa[\bar{e}_i] = \sum_{j=1}^{m'} \kappa[\bar{e}'_j]. \tag{5}$$

It is obvious that the above constraint is homogeneous and the possible solution space for these variables is large. Meanwhile, the subdivision numbers are better to meet the size requirements on the corresponding edges. Therefore, we find the quad edges whose subdivision number have not been related to the size fields, and apply the same type of constraints in (1) on these edges:

$$\kappa[\bar{e}'_i] = \sum_{e_j \subset \bar{e}'_i} \frac{|e_j|}{\mu[e_j]}, \quad \kappa[\bar{e}_i] = \sum_{e_j \subset \bar{e}_i} \frac{|e_j|}{\mu[e_j]}. \tag{6}$$

When apply the constraints in above for all of the quad edges on the interface curve of Q and Q', we have $\sum_{i=1}^{m} \kappa[\bar{e}_i] = \sum_{e_j \in C} \frac{|e_j|}{\mu[e_j]}$ and $\sum_{i=1}^{m'} \kappa[\bar{e}'_i] = \sum_{e_j \in C} \frac{|e_j|}{\mu[e_j]}$. Then, we have $\sum_{i=1}^{m} \kappa[\bar{e}_i] = \sum_{i=1}^{m'} \kappa[\bar{e}'_i]$, which implies the subdivision constraint in (5) becomes redundant. However, the subdivision constraint cannot be replaced, because the alternatives in (6) may not be satisfied accurately enough for producing the same number of subdivided edges along the interface curve, due to the possible numerical error of the solver (see Section 4.6).

Process all constraints in (4) and (5), and assemble them into a matrix form, then we have:

$$\mathbf{A}\ \kappa = 0, \tag{7}$$

where A is the coefficient matrix, κ is the vector composed of the subdivision variable of the boundary edges and the joint curves (see (2)), and the integer variable associated with the unmatched submeshes (see (4)).

By (4) and (5), we know that the coefficients in $\mathbf{A}$ can take only four possible values: $\{-2, -1, 0, 1\}$. Since any boundary quad edge belongs to only one boundary curve, each column of $\mathbf{A}$ has only one nonzero coefficient. In addition, the subdivision numbers are not all independent variables, since the subdivision rule (Section 3, step 4) indicates that the boundary edges of the same quad strip must have equal subdivision numbers. Therefore, we can condense $\mathbf{A}$ and κ, and the condensed matrix $\mathbf{A}$ satisfies the following: 1) All the coefficients are integers in $[-2, 2]$; 2) each column has at most two nonzero values. In the following, we still denote the condensed matrix and vector by $\mathbf{A}$ and κ without confusion.

$\mathbf{A}\kappa = 0$ is a set of hard constraint on the integer variables, which must be satisfied first of all. However, as the number of matched sum-meshes increases, the neighboring relationship may become more and more complex, such that it is possible that $\mathbf{A}\kappa = 0$ cannot be satisfied. To help the user to find this situation and avoid it, we must check whether there is a positive solution to fulfill $\mathbf{A}\kappa = 0$. Note that $\mathbf{A}\kappa = 0$ is a linear system with integer coefficients, which is also called linear Diophantine equations. We first conduct a series of unimodular row operations and column swaps to reduce the matrix into an upper triangular form. Then, we can divide the κ variables into two sets, the dependent variables κ' and the free variables κ'', and rewrite the equations as the following equivalent form:

$$(\mathbf{A}' - \mathbf{A}'') \begin{pmatrix} \kappa' \\ \kappa'' \end{pmatrix} = 0,$$

where both $\mathbf{A}'$ and $\mathbf{A}''$ are integer matrices, and $\mathbf{A}'$ is full rank and upper triangular.

If the free variable set κ'' is empty, then $\mathbf{A}\kappa = 0$ cannot be satisfied. In this case, we find the matched submesh that corresponds to the last variable in κ', remove it from the set of matched submeshes and treat it as an unmatched submesh. Then, we update the linear Diophantine equations accordingly.

Otherwise, κ'' is not empty. Then, we have $\kappa' = (\mathbf{A}')^{-1}\mathbf{A}''\kappa''$. Since the entries in $(\mathbf{A}')^{-1}$ are rational numbers, let g be the smallest positive integer g such that $g(\mathbf{A}')^{-1}$ is an integer matrix. Then, we have $g\kappa' = (g(\mathbf{A}')^{-1})\mathbf{A}''\kappa''$. It is easy to see that $g\kappa' = (g(\mathbf{A}')^{-1})\mathbf{A}''\kappa''$ and $\kappa' = (g(\mathbf{A}')^{-1})\mathbf{A}''\kappa''$ have the same number of solutions, because they are homogenous systems. Therefore, solving the original Diophantine equations is equivalent to finding positive integer κ'' that satisfy $g(\mathbf{A}')^{-1}\mathbf{A}''\kappa'' > 0$. Note that these inequations define some hyperplanes in the space of κ'', and

the solution is a hyper polyhedron. Then, we can find if the solution space is empty or not by linear programming. If the solution space does not contain any positive integer point, we find the matched submesh that corresponds to the first variable in κ' that cannot be satisfied (i.e., the first hyperplane that reduces the solution space to zero), remove it from the set of matched submeshes, and treat it as an unmatched submesh. Then, we update the linear Diophantine equations accordingly.

We repeat these examinations and updates until the integer hard constraints are all compatible.

4.4 Constraints between Repeated Submeshes

At last, we consider the corresponding boundaries of two repeated submeshes M' and M, and describe the size field constraints between them. To generate the same quad mesh for the repeated submeshes, it is natural to require their corresponding edges to have the same size fields. To reduce computation cost, we remesh only one submesh and transfer the result to the others. Therefore, we constrain their size fields only on their boundaries instead of the whole submeshes.

For each mesh edge $e \in \partial M$, we can find an edge $e' \in \partial M'$ by the correspondence between ∂M and $\partial M'$. Then, we constrain their size fields to be as close as possible using the following least squares energy:

$$E_{M,M'}(x) = \sum_{e \in \partial M} (\tilde{\mu}[e] - \tilde{\mu}[e'])^2 + \sum_{e \in \partial M} (\tilde{\eta}[e] - \tilde{\eta}[e'])^2, \quad (8)$$

where we use the logarithm of the size fields to have a quadratic function in (10). By summing the above energy terms over all pairs of repeated submeshes, we define the following energy function:

$$E_r(x) = \sum_{M,M'} E_{M,M'}(x). \quad (9)$$

4.5 Mixed-Integer Problem

Now, we describe a constrained mixed-integer problem for optimizing the subdivision numbers and the size fields. First, we add the energy term in (8) into the size field optimization energy function $E_{total}(\tilde{\mu}, \tilde{\eta})$ (see [2, (14)]), defining a new objective function as follows:

$$E(x) = E_{total}(x) + E_r(x), \quad (10)$$

where the terms are all quadratic functions of x. Then, take into account all of the constraints in (3) and (7), we formulate a minimization problem as follows:

$$arg \min_{x,\kappa} E(x) \quad \text{s.t.} \begin{cases} \mathbf{A}\kappa = 0 \\ \kappa = \mathbf{C}e^{-x} \\ \kappa_i \in \mathbb{Z}^+, i = 1, 2, \ldots \end{cases} \quad (11)$$

This is a large optimization problem, where the number of variables is the order of the edge number of the input triangle mesh. Though the objective function $E(x)$ is quadratic, the constraints consist of exponential functions and positive integer variables, which make it very challenging to solve. We note that Bommes et al. [1] developed an effective greedy solver for mixed-integer optimization problem for their quadrangulation technique, but their solver does not deal with the positive integer constraints that we have in (11). In the following section, we will describe a heuristic solver to approximate the solution based on Gurobi optimizer [28], a commercial mixed-integer quadratic optimizer.

4.6 A Heuristic Solver

The basic idea of the heuristic solver is to first solve the continuous relaxation problem, then solve for the integer variable to satisfy the hard constraints. It has three main steps:

1. *Initialization.* First solve the following continuous relaxation problem without the constraints:

$$arg \min_{x} E(x) \quad (12)$$

to obtain an initial solution x^0 for x.

2. *Rounding by MIQP*. Then, we initialize κ as $\kappa^0 = \mathbf{C}e^{-x^0}$. Note that κ^0 is not integer. To round κ^0 into integers, we would like to minimize its impact to the objective function $E(x)$. By the hard constraint of $\kappa = \mathbf{C}e^{-x}$, we have the following approximation:

$$\Delta\kappa \approx \mathbf{CE}\Delta x = \mathbf{D}\Delta x,$$

where $\Delta\kappa = \kappa - \kappa^0$, $\Delta x = x - x^0$, $\mathbf{E}$ is the diagonal matrix composed of $-e^{-x^0}$, and $\mathbf{D} = \mathbf{CE}$. Let $\mathbf{G} = \mathbf{D}^t(\mathbf{D}\mathbf{D}^t)^{-1}$ be a pseudoinverse of $\mathbf{D}$. Then, the changes in x due to the rounding error $\Delta\kappa$ can be computed as follows:

$$\Delta x = \mathbf{G}\Delta\kappa. \quad (13)$$

Remember that $E(x)$ is quadratic, its change by Δx can be expressed using the Hessian matrix $\mathbf{H}$ of $E(x)$ as: $\Delta E = \frac{1}{2}\Delta x^t \mathbf{H} \Delta x$. Then, we have

$$\Delta E = \frac{1}{2}(\kappa - \kappa^0)^t \mathbf{G}^t \mathbf{H} \mathbf{G} (\kappa - \kappa^0).$$

Therefore, we formulate the following constrained mixed-integer minimization problem to obtain κ:

$$arg \min_{\kappa} \Delta E \quad \text{s.t.} \begin{cases} \mathbf{A}\kappa = 0 \\ \kappa_i \in \mathbb{Z}^+, i = 1, 2, \ldots \end{cases} \quad (14)$$

We used the MIQP Optimizer in Gurobi software [28] to solve the above minimization problem.

3. *Update the size fields*. After getting the solution to κ, we update x at the boundary by $x \leftarrow x^0 + \mathbf{G}\Delta\kappa$. Since (13) is a first order approximation, this update may not satisfy the hard constraints in $\kappa = \mathbf{C}e^x$. To address this issue, we examine every constraint in $\kappa = \mathbf{C}e^x$, and perform a uniform scaling on the size fields of the corresponding boundary edges, such that the constraint is precisely satisfied. At last, we fix the values on the boundary edges, and propagate the errors over the whole surface by minimizing the objective energy function $E(x)$.

5 Quadrangulation of Submeshes

After solving the stitching problem, the quad number on each joint curve (the κ variables) and the corresponding size field are determined. Therefore, we can remesh the

	M — the input sub-mesh
1.	For all boundary curve C
2.	Setup up the wave value constraints by Equation 15
3.	For all symmetric structure
4.	Setup the symmetry constraints by Equation 17
5.	Construct the wave function on M
6.	Generate the quad mesh Q for M by extracting the quasi-dual MSC of the wave function
7.	If M is a repeated sub-mesh, Then
8.	Transfer Q onto the other repeated sub-meshes.

Fig. 6. Outline of the extended wave-based quadrangulation algorithm.

submeshes independently as long as we generate the same number of quads along the joint boundaries as prescribed in κ. This allows to remesh complex models in parallel fully taking advantage of the computing power of multicore processors.

As introduced in the overview section, the matched submeshes are remeshed by subdividing the corresponding quad mesh templates and projecting the new vertices to the submeshes.

For the unmatched submeshes, we adopt the wave-based method [2] to remesh them, and extend it for generating prescribed number of quads on the boundaries and preserving the symmetry structure. When a repeated quad mesh is remeshed, the generated quad mesh is then transferred to the other instances. The transfer process is the same as remeshing a matched submesh, except that it does not need subdivision.

Fig. 6 shows the outline of the extended wave-based quadrangulation method. In the following, we will present the details in setting up the wave value constraint and the symmetry preservation constraint.

5.1 Wave Value Constraint

An important step of the wave-based quadrangulation method is to construct a wave function. As in previous sections, let M denote a submesh to remesh, $C = e_1 \cup e_2 \cdots \cup e_m$ be boundary curve, and $e_i = (p_{i-1}, p_i)$, $i = 1, 2, \ldots m$. Since we have determined $\kappa[C]$ (the quad number along curve C) during the stitching step (step 3 in Section 3), we need to ensure that the wave function on C will generate $\kappa[C]$ quads. Otherwise, seamless stitching is not promised. We achieve this goal by imposing extra value constraints for the wave function.

Recall that the vector fields on M are aligned with the direction of the boundary edges during the initialization step. For convenience, we assume that $\vec{X}$ equals the edge direction of the boundary C. Then, $\vec{X}[e_i] = (p_i - p_{i-1})/|e_i|$. Let (θ_i, ϕ_i) be the phase angles of point p_i. Then, by the wave construction method in [2], we have

$$\theta_i - \theta_{i-1} = \omega_x \vec{X}[e_i] \cdot (p_i - p_{i-1}) = \frac{\pi}{\mu[e_i]} |e_i| = \frac{|e_i|}{\mu[e_i]} \pi,$$

$$\phi_i - \phi_{i-1} = \omega_y \vec{Y}[e_i] \cdot (p_i - p_{i-1}) = 0.$$

Then, $\theta_i = \theta_0 + \delta_i$ and $\phi_i = \phi_0$, where $\delta_i = \sum_{j=1}^{i} \frac{|e_i|}{\mu[e_i]} \pi$. Then, the wave function values at p_i can be expressed as

$$\mathbf{f}(p_i) = \begin{pmatrix} \cos(\theta_i)\cos(\phi_i) \\ \cos(\theta_i)\sin(\phi_i) \\ \sin(\theta_i)\cos(\phi_i) \\ \sin(\theta_i)\sin(\phi_i) \end{pmatrix} = \mathbf{T}_i \, \mathbf{f}(p_0) \tag{15}$$

by trigonometry, where $\mathbf{T}_i$ is a constant matrix

$$\mathbf{T}_i = \begin{pmatrix} c_i & 0 & -s_i & 0 \\ 0 & c_i & 0 & -s_i \\ s_i & 0 & c_i & 0 \\ 0 & s_i & 0 & c_i \end{pmatrix}, \begin{cases} c_i = \cos(\delta_i) \\ s_i = \sin(\delta_i) \end{cases}. \tag{16}$$

Therefore, the wave function on curve C is completely determined by its value at point p_0. Then, we substitute all $\mathbf{f}(p_i)$ with (15) in the wave construction energy function E_{wave} (see [2, (19)]) and construct the wave function.

Since $\theta_k - \theta_0 = \delta_k = \sum_{i=1}^{k} \frac{|e_i|}{\mu[e_i]} \pi = \kappa[C]\pi$, the wave function has $\kappa[C]$ half periods on curve C. By the MSC extraction method, $\kappa[C]$ quads will be generated on C, which exactly satisfies the stitching requirement mentioned above.

5.2 Symmetry Preservation

Many models contain symmetric or nearly symmetric structures. In this section, we describe a method to preserve the symmetric structures. Since the vector/size fields are defined on the mesh edges and it is too strict to require two edges to be symmetric, we set up the symmetric correspondence by the mid points of the edges: For each point, we compute the symmetric position, and then find the nearest midpoint to the symmetric position.

Given a symmetric region Ω and the symmetry plane P, we first prepare symmetric vector fields and symmetric size fields in Ω. During the vector field smoothing step, we require the symmetric edges to have symmetric vector fields. During the size fields optimization step, we add an extra energy term to minimize the squared differences between the size fields of the symmetric edges.

Then, we add a symmetry preservation energy term into the wave construction energy E_{wave} to constrain the wave function. Since both even and odd wave functions can produce symmetric quad meshes, we define the symmetry preservation energy term as follows:

$$E_\Omega(\mathbf{f}_1, \ldots, \mathbf{f}_n) = \sum_{(p,p') \in V_\Omega} \left(\mathbf{f}(p)^2 - \mathbf{f}(p')^2 \right)^2, \tag{17}$$

where V_Ω denotes the set consisting of all symmetric vertex pairs in Ω.

6 Implementation and Results

We have implemented the presented quadrangulation method, and tested it on various models. Before going to the experimental results, we first introduce how we segment and analyze the submeshes for quadrangulation.

Segmentation and analysis. To automatically segment the input model, we adopted the SDF-based consistent mesh partitioning tool [29] (public downloadable from http://www.cs.tau.ac.il/liors/research/projects/sdf/). With this tool, we can segment a mesh in seconds. For large complex models, we use a simplified proxy to perform the segmentation task, then map the result back to the original mesh and smooth the cutting boundaries. After segmentation, we

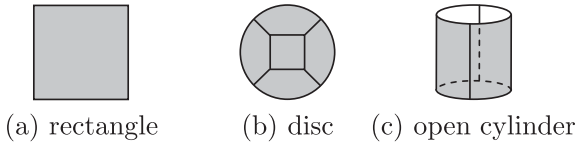


Fig. 7. Primitives for CAD model analysis.

perform shape comparison on the submeshes to find repeated components and match them with predefined templates. Given two meshes, we first compare the following simple shape characteristics between them: The number of the boundary curves, the relative length of the boundary curves, and the aspect ratio $(\lambda_2/\lambda_1, \lambda_3/\lambda_1)$ (where $\lambda_1 \geq \lambda_2 \geq \lambda_3$ are the eigen values obtained by applying the PCA method on the mesh vertices). When these simple comparisons pass, we try to build a 1-1 mapping based on heat kernel signature [30] and spectral embedding [31]. If any step in above fails, the two submeshes are regarded as different in shape. This lightweight analysis method can efficiently identify similar shapes up to small deformation. We leave it to a future work to handle large nonrigid deformation.

Since CAD models usually consists of simpler patches that we cannot segment out using the SDF-based method, we developed another lightweight segmentation method. Given a CAD model, we first find the sharp feature edges where the adjacent triangles' normal angle is greater than a threshold of 45 degree, and connect them to create some curves for segmentation. Then, we obtain a submesh by randomly seeding a triangle and growing the region as far as to the segmentation curves or the mesh boundaries. This seeding-and-growing procedure is repeated until there is no triangle left. At last, the user can optionally pick and merge some submeshes to eliminate over segmentation. To facilitate analyzing the CAD model submeshes, we define three simple primitives for matching, namely disc, rectangle, and open cylinder. Fig. 7 shows the quad mesh templates of these primitives.

Experimental results. Now, we show some quadrangulation results. In the following figures, RSD denotes the relative standard deviation of the quad angles, N_{sp} denotes the number of singularity points of the quad mesh, and d_H denotes the Hausdorff distance between the input model and the remeshed model and is normalized by the diagonal length of the input model's bounding box. All of the quad meshes shown in the result section are available in the supplementary materials, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2012.301. Fig. 8 plots the distribution of the scaled Jacobian measurement to show the mesh quality of our quadrangulation results.

First, we demonstrate the ability to remesh large complex objects using the Lucy model shown in Fig. 1. This model consists of $1,413K$ vertices, which the previous wave-based method cannot handle due to the huge memory requirement. In Figs. 9, 10, and 11, we test the divide-and-conquer quad remeshing techniques on larger, more complex models, which contain a great number of featured structures. The results showed that our method can successfully deal with them.

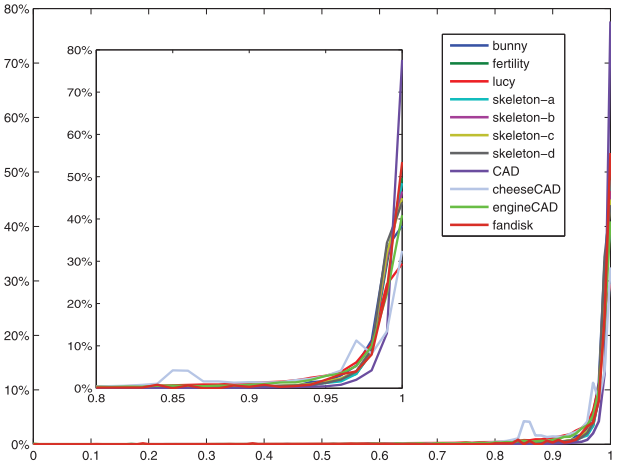


Fig. 8. Distribution of the scaled Jacobian measurement of the quadrangulation results. The range $[0, 1]$ of the scaled Jacobian is divided into 100 bins, and the number of the quads in each bin is normalized by the total number of quads.

Table 1 lists more timing results on a number of models, where N_v is vertex number, N_s is submesh number, N_t and N_r are the number of matched and repeated submeshes, t_m is the computation time in mixed-integer optimization, $|\kappa|$ is the number of integer variables, t_q is the computation time for remeshing all submeshes, and N_q is the quad number of the final result. The timing is carried out on a PC with a

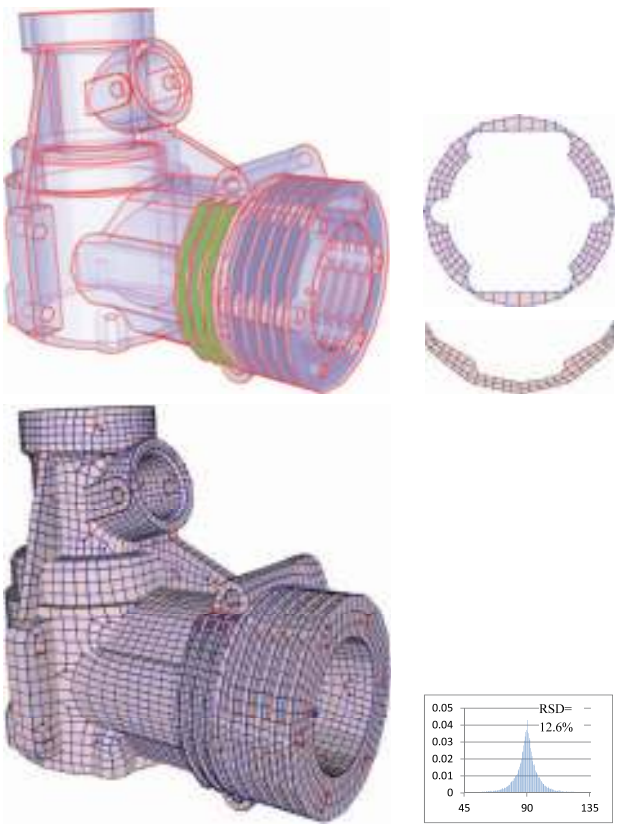


Fig. 9. The engine model. The input model is drawn with transparency to show the internal structures. This model is segmented into 185 parts. The right column show two repeated parts. $N_{sp} = 758$, $RSD = 12.65\%$. Hausdorff distance $d_H = 0.0079$.

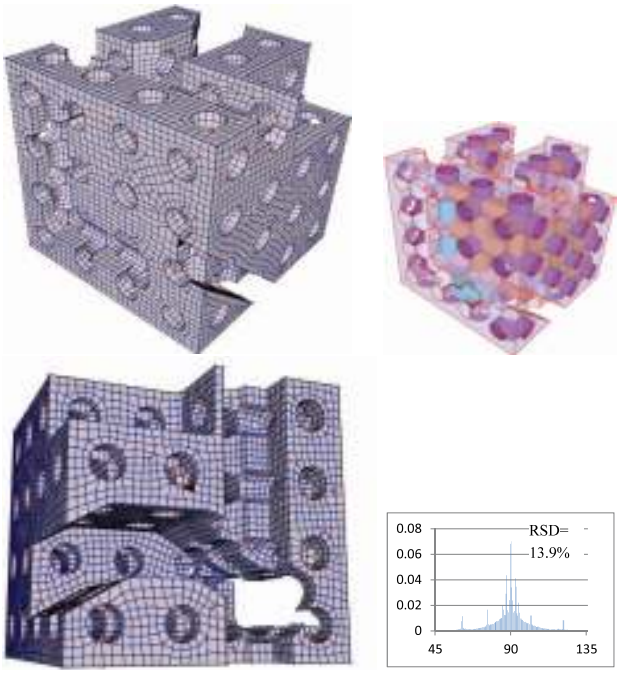


Fig. 10. An example of a complex CAD model. Left: two views of the quadrangulation result; Right: the input model and the angle distribution. The input model is drawn with transparency to show the internal structures. There are 279 submeshes. $N_{sp} = 1303$, $RSD = 13.9\%$. Hausdorff distance $d_H = 0.0038$.

2.8-GHz dual core CPU of 2-GB RAM, but our program is single threaded.

Now, we compare the computation time of this work with that of the wave-based method (Zhang et al. [2, Table 2]). For the Fandisk model, our method took 23.1 seconds of computation time when segmenting the model into 12 submeshes, which is 17.7 seconds faster. For the fertility model, our methods took 93.8 seconds to remesh with symmetry preservation, which approximately equals to the original wave-based method, showing that the symmetry constraint does not incur significant computation cost. In addition, the original waved-based method took about 586 seconds of computation to remesh a CAD model of 136 K vertices, while our method took about 140 seconds of computation to remesh a CAD model of 267 K vertices, which indicates an acceleration of eight times. In summary, our method can efficiently deal with large models by simply segmenting them into smaller ones, thanks to its divide-and-conquer nature. The last row in Table 1 shows that it takes about 10 minutes to remesh a large model with 500 K vertices.

As shown in Table 1, the major computation cost of our method is taken by the wave-based procedure. Compared with the surface meshing techniques based on advancing front, Delaunay and quadtrees [27], our method is much slower, while it can simultaneously optimize the size and the orientation of the final quads.

In Figs. 13 and 14, we demonstrate effects of symmetry preservation. In these examples, the symmetry regions were manually painted, while the symmetry plane is automatically computed. As shown in the results, we successfully obtain symmetric quad mesh in the identified regions. And the comparison in Fig. 14 shows that preserving the symmetry structure significantly improves the quality of the quad mesh.

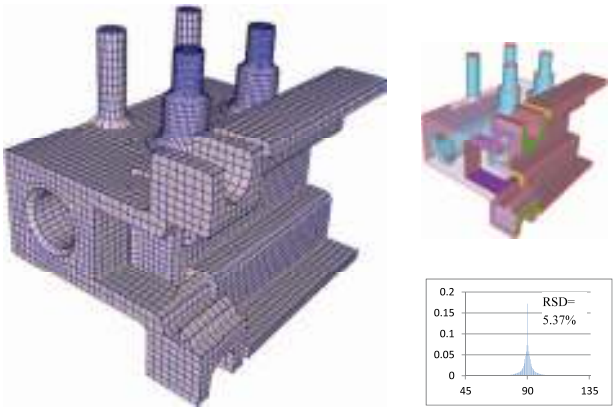


Fig. 11. Result of a complex CAD model. $N_{sp} = 148$, $RSD = 5.37\%$. Hausdorff distance $d_H = 0.0031$.

In Fig. 15, we test how much the granularity of the segmentation affects the final result. As shown in this example, while different segmentation will lead to different quadrangulation, the overall quality are similar to each other. The result in Fig. 15d shows that by identifying some as matched submeshes with cylinder, we can significantly reduce the number of the singularity points and thus improve the result.

At last, we compare our method with CUBIT [32], an industrial commercial software generating FEM meshes. CUBIT requires to partition the input surface into a cover of some *quad patches*, then uses the interval assignment technique [33], [34] to determine the number of subdivided mesh edges for each boundary curve. But it is a difficult and manual task to produce a reasonable quad patch decomposition. As mentioned in [35], among 10 steps of a typical the model generation and analysis process, geometry decomposition is the most time consuming one (about 32 percent). In our method, we allow for arbitrary segmentation, alleviating the constraints for geometry decomposition and making it possible to take advantage of the mesh segmentation techniques to automate the decomposition step.

Moreover, CUBIT does not take into account the distortion in the patches during the interval optimization, while our method is aware of the relationship between edge number and the quality related terms, such as curl-free energy, orientation, mesh size, and so on. As shown in Fig. 12b, the sizes of the subdivided edges (equivalent to the numbers of the intervals) affects the curl energy very much. Therefore, our method is likely to produce better results than CUBIT. As shown in Figs. 12c and 12d, both quadrangulation results satisfy the interval requirements, but the slight difference in the intervals leads to great difference on the quality.

TABLE 1
Model Size and Computation Time in Seconds

Mesh	N_v	N_s / N_t / N_r	t_m / $\lvert s \rvert$	t_q	N_q
Fandisk (Fig. 4)	25K	12/ 7/ 0	1.8 / 39	21.3	604
Bunny (Fig. 13)	46K	3/ 0/ 2	–	63.8	3492
Fertility (Fig. 14)	55K	1/ 0/ 0	–	93.8	2448
Skeleton (Fig. 15 a)	152K	–	–	349.3	19244
Skeleton (Fig. 15 b)	152K	6/ 0/ 0	20.7 / 10	222.4	17780
Skeleton (Fig. 15 c)	152K	29/ 14/ 0	10.6 / 53	119.3	17788
Skeleton (Fig. 15 d)	152K	29/ 0/ 0	20.7 / 66	250.1	18520
CAD (Fig. 11)	267K	86/ 68/ 10	19.8 / 144	119.2	16055
CheeseCAD (Fig. 10)	494K	279/ 31/155	155.9/1378	581.8	16955
EngineCAD (Fig. 9)	500K	185/144/ 15	89.9 / 309	503.7	10519

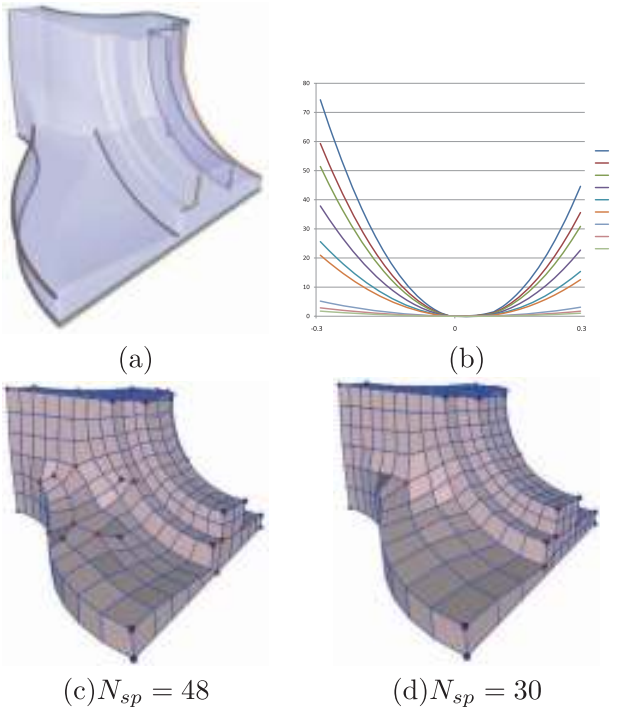


Fig. 12. Comparison with interval assignment technique. The decomposition of the input model is visualized in Fig. 4a. (a) Nine curves on the model. (b) The relationship between the curl energy and the size (in logarithm) of the subedges on the curve. Each plot corresponds to a feature curve on the model with the same color. (c) Quadrangulation results by interval assignment technique. (d) Results by our method, Hausdorff distance $d_H = 0.0130$.

7 Conclusion and Discussion

We have presented a top-down quadrangulation method, and successfully tested it on many models. The experimental results show that this method can generate high-quality results and preserve high-level mesh structures. Compared with the original wave-based method, the proposed method not only improves the performance very significantly, but also extends its ability and robustness in dealing with huge and complex models, thanks to its divide-and-conquer nature that avoids solving large nonlinear systems for wave functions. When structure information, e.g., symmetry and repetition, is provided, our method can not only be faster, but also generate better results. However, if there is no such structure information, it may generate worse result than the original wave-based method, because it is optimized under a specific segmentation, while the original method is globally optimized. This explains why there are more singularity points on the Lucy model in Fig. 1 than we expected.

Although not the focus of this work, shape segmentation and analysis play an important role in our divide-and-conquer quad remeshing framework. We developed a lightweight approach to automatic mesh segmentation and analysis based upon existing techniques. It works for our testing models, but is neither general nor robust enough for arbitrary mesh models, because it has very limited ability in matching submeshes with nonrigid deformation. In addition, the segmentation results do not match with the semantic meaning as the user perceived. In the future, we plan to exploit more shape segmentation and analysis techniques [36], [37], [38] to improve the robustness of our quadrangulation method, particularly to develop a better approach to dealing with more shape variations and automatically identifying the symmetry structures [39],

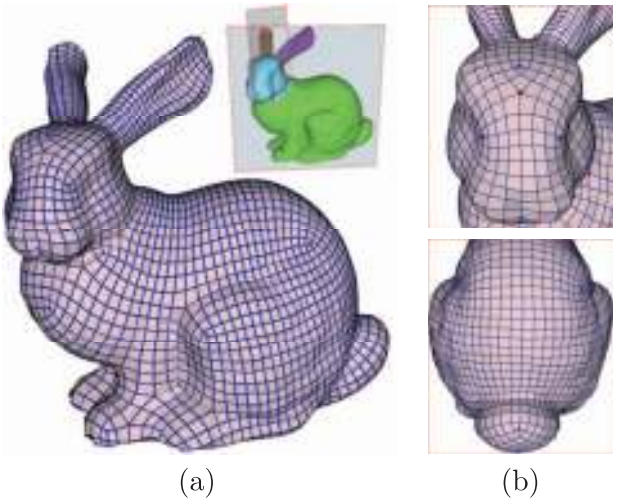


Fig. 13. Quadrangulation with symmetry preservation. (a) The results and the original model overlapped with the symmetry planes and segmentation shown in the top-right corner. (b) Two zoomed views showing the preservation result. $N_{sp} = 67$, $RSD = 7.9\%$, Hausdorff distance $d_H = 0.0118$.

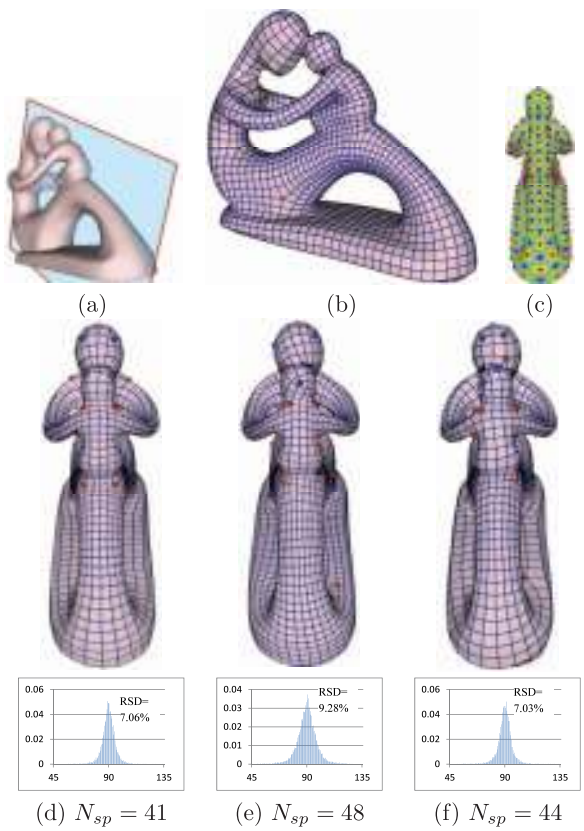
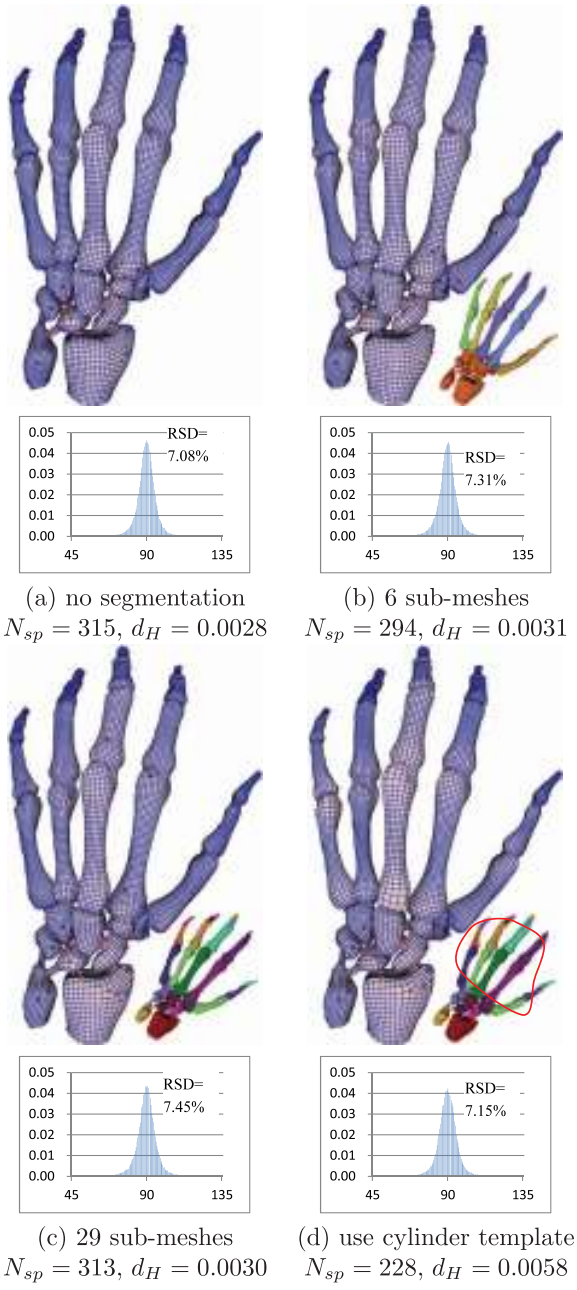


Fig. 14. Quadrangulation with symmetry preservation. (a) The input model overlapped with the symmetry plane. (b) and (d) The quadrangulation result in two views, Hausdorff distance $d_H = 0.0090$. (c) The corresponding wave function and MSC charts. (e) and (f) The result of Bommes et al. [1] and Zhang et al. [2] for comparison.



(a) no segmentation $N_{sp} = 315$, $d_H = 0.0028$

(b) 6 sub-meshes $N_{sp} = 294$, $d_H = 0.0031$

(c) 29 sub-meshes $N_{sp} = 313$, $d_H = 0.0030$

(d) use cylinder template $N_{sp} = 228$, $d_H = 0.0058$

Fig. 15. Result comparison of using different levels of segmentation. The segmentation is shown in the bottom right corner. (c) and (d) have the same segmentation, but in (d) we use cylinder template to remesh 14 submeshes (shown in the red curves).

[40]. Our current approach requires the user input to identify the region with symmetry structure. In addition, we plan to create a rich data set of quad mesh templates to improve the quality.

Acknowledgments

The authors thank the anonymous reviewers for their valuable comments about the experiments, presentation, the cyclic configurations, to name a few. This work was partially supported by China 973 Program (No. 2009CB320801), NSFC (No. 60970074), Fundamental Research Funds for the Central Universities, and Fok Ying Tung Education Foundation.

References

[1] D. Bommes, H. Zimmer, and L. Kobbelt, "Mixed-Integer Quadrangulation," *ACM Trans. Graphics*, vol. 28, no. 3, pp. 1-10, 2009.

[2] M. Zhang, J. Huang, X. Liu, and H. Bao, "A Wave-Based Anisotropic Quadrangulation Method," *ACM Trans. Graphics*, vol. 29, pp. 118:1-118:8, July 2010.

[3] F. Kälberer, M. Nieser, and K. Polthier, "Quadcover—Surface Parameterization Using Branched Coverings," *Computer Graphics Forum*, vol. 26, no. 3, pp. 375-384, Sept. 2007.

[4] K. Hormann and G. Greiner, "Quadrilateral Remeshing," *Proc. Vision, Modeling, and Visualization*, B. Girod, G. Greiner, H. Niemann, and H.-P. Seidel, eds., pp. 153-162, Nov. 2000.

[5] M. Marinov and L. Kobbelt, "A Robust Two-Step Procedure for Quad-Dominant Remeshing," *Computer Graphics Forum*, vol. 25, no. 3, pp. 537-546, Sept. 2006.

[6] Y.-K. Lai, L. Kobbelt, and S.-M. Hu, "An Incremental Approach to Feature Aligned Quad Dominant Remeshing," *Proc. ACM Symp. Solid and Physical Modeling*, pp. 137-145, 2008.

[7] J.-F. Remacle, J. Lambrechts, B. Seny, E. Marchandise, A. Johnen, and C. Geuzaine, "Blossom-Quad: A Non-Uniform Quadrilateral Mesh Generator Using a Minimum Cost Perfect Matching Algorithm," *Int'l J. Numerical Methods in Eng.*, vol. 89, no. 9, pp. 1102-1119, 2012.

[8] W. Cook and A. Rohe, "Computing Minimum-Weight Perfect Matchings," *INFORMS J. Computing*, vol. 11, no. 2, pp. 138-148, Feb. 1999.

[9] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Lévy, and M. Desbrun, "Anisotropic Polygonal Remeshing," *ACM Trans. Graphics*, vol. 22, no. 3, pp. 485-493, 2003.

[10] M. Marinov and L. Kobbelt, "Direct Anisotropic Quad-Dominant Remeshing," *Proc. Pacific Conf. Computer Graphics and Applications*, pp. 207-216, 2004.

[11] N. Ray, W.C. Li, B. Lévy, A. Sheffer, and P. Alliez, "Periodic Global Parameterization," *ACM Trans. Graphics*, vol. 25, no. 4, pp. 1460-1485, 2006.

[12] S. Dong, P.-T. Bremer, M. Garland, V. Pascucci, and J.C. Hart, "Spectral Surface Quadrangulation," *ACM Trans. Graphics*, vol. 25, no. 3, pp. 1057-1066, 2006.

[13] J. Huang, M. Zhang, J. Ma, X. Liu, L. Kobbelt, and H. Bao, "Spectral Quadrangulation with Orientation and Alignment Control," *ACM Trans. Graphics*, vol. 27, no. 5, pp. 1-9, 2008.

[14] Y. Tong, P. Alliez, D. Cohen-Steiner, and M. Desbrun, "Designing Quadrangulations with Discrete Harmonic Forms," *Proc. Eurographics Symp. Geometry Processing*, pp. 201-210, 2006.

[15] A. Myles, N. Pietroni, D. Kovacs, and D. Zorin, "Feature-Aligned T-Meshes," *ACM Trans. Graphics*, vol. 29, pp. 117:1-117:11, July 2010.

[16] J. Tierny, J. Daniels II, L.G. Nonato, V. Pascucci, and C.T. Silva, "Inspired Quadrangulation," *Computer Aided Design*, vol. 43, no. 11, pp. 1516-1526, 2011.

[17] D. Cohen-Steiner and J.-M. Morvan, "Restricted Delaunay Triangulations and Normal Cycle," *Proc. Symp. Computational Geometry*, pp. 312-321, June 2003.

[18] J. Palacios and E. Zhang, "Rotational Symmetry Field Design on Surfaces," *ACM Trans. Graphics*, vol. 26, no. 3, article 55, 2007.

[19] N. Ray, B. Vallet, W.C. Li, and B. Lévy, "N-Symmetry Direction Field Design," *ACM Trans. Graphics*, vol. 27, no. 2, pp. 1-13, 2008.

[20] S. Dong, S. Kircher, and M. Garland, "Harmonic Functions for Quadrilateral Remeshing of Arbitrary Manifolds," *Computer Aided Geometric Design*, vol. 22, no. 5, pp. 392-423, 2005.

[21] D. Kovacs, A. Myles, and D. Zorin, "Anisotropic Quadrangulation," *Computer Aided Geometric Design*, vol. 28, no. 8, pp. 449-462, 2011.

[22] J. Daniels, C.T. Silva, J. Shepherd, and E. Cohen, "Quadrilateral Mesh Simplification," *ACM Trans. Graphics*, vol. 27, pp. 148:1-148:9, Dec. 2008.

[23] M. Tarini, N. Pietroni, P. Cignoni, D. Panozzo, and E. Puppo, "Practical Quad Mesh Simplification," *Computer Graphics Forum*, vol. 29, no. 2, pp. 407-418, 2010.

[24] D. Bommes, T. Lempfer, and L. Kobbelt, "Global Structure Optimization of Quadrilateral Meshes," *Computer Graphics Forum*, vol. 30, pp. 375-384, 2011.

[25] Y. Li, W. Wang, R. Ling, and C. Tu, "Shape Optimization of Quad Mesh Elements," *Computers & Graphics*, vol. 35, no. 3, pp. 444-451, 2011.

[26] J. Daniels II, M. Lizier, M. Siqueira, C. Silva, and L. Nonato, "Template-Based Quadrilateral Meshing," *Computers & Graphics,* vol. 35, no. 3, pp. 471-482, 2011.
[27] S.J. Owen, "A Survey of Unstructured Mesh Generation Technology," *Proc. Int'l Conf. Meshing Roundtable,* pp. 239-267, 1998.
[28] *Gurobi, Gurobi Optimizer 4.5,* http://www.gurobi.com/, 2013.
[29] L. Shapira, A. Shamir, and D. Cohen-Or, "Consistent Mesh Partitioning and Skeletonisation Using the Shape Diameter Function," *Visual Computer,* vol. 24, no. 4, pp. 249-259, 2008.
[30] J. Sun, M. Ovsjanikov, and L.J. Guibas, "A Concise and Provably Informative Multi-Scale Signature Based on Heat Diffusion." *Computer Graphics Forum,* vol. 28, no. 5, pp. 1383-1392, 2009.
[31] V. Jain and H. Zhang, "Robust 3D Shape Correspondence in the Spectral Domain," *Proc. Int'l Conf. Shape Modeling and Applications,* pp. 118-129, 2006.
[32] C.D. Team, S. Benzley, C.E. Department, R. Kerr, S.R. Jankovich, and D.B. Mcrae, "Cubit Mesh Generation Environment Volume 1: Users Manual," technical report, 1994.
[33] S.A. Mitchell, "High Fidelity Interval Assignment," *Proc. Int'l Conf. Meshing Roundtable,* pp. 33-44, 1997.
[34] J. Shepherd, S. Benzley, and S. Mitchell, "Interval Assignment for Volumes with Holes," *Int'l J. Numerical Methods in Eng.,* pp. 49-277, 2000.
[35] Y. Bazilevs, V. Calo, J. Cottrell, J. Evans, T. Hughes, S. Lipton, M. Scott, and T. Sederberg, "Isogeometric Analysis Using T-Splines," *Computer Methods in Applied Mechanics and Eng.,* vol. 199, nos. 5-8, pp. 229-263, 2010.
[36] S. Katz and A. Tal, "Hierarchical Mesh Decomposition Using Fuzzy Clustering and Cuts," *ACM Trans. Graphics,* vol. 22, pp. 954-961, July 2003.
[37] X. Chen, A. Golovinskiy, and T. Funkhouser, "A Benchmark for 3D Mesh Segmentation," *ACM Trans. Graphics,* vol. 28, pp. 73:1-73:12, July 2009.
[38] E. Kalogerakis, A. Hertzmann, and K. Singh, "Learning 3D Mesh Segmentation and Labeling," *ACM Trans. Graphics,* vol. 29, pp. 102:1-102:12, July 2010.
[39] N.J. Mitra, L.J. Guibas, and M. Pauly, "Partial and Approximate Symmetry Detection for 3D Geometry," *ACM Trans. Graphics,* vol. 25, pp. 560-568, July 2006.
[40] J. Podolak, P. Shilane, A. Golovinskiy, S. Rusinkiewicz, and T. Funkhouser, "A Planar-Reflective Symmetry Transform for 3D Shapes," *ACM Trans. Graphics,* vol. 25, pp. 549-559, July 2006.

Muyang Zhang received the BS and PhD degrees in computer science from Zhejiang University in 2005 and 2011, respectively. He is currently with the Search Technology Center of Microsoft. His research interests are mainly in geometry processing and remeshing, especially for quadrangulation.

Jin Huang received the PhD degree from the Computer Science Department, Zhejiang University in 2007 with Excellent Doctoral Dissertation Award of China Computer Federation. He is an associated professor in the State Key Lab of CAD&CG of Zhejiang University, P.R. China. His research interests include geometry processing and physically based simulation. He has served as a reviewer for ACM SIGGRAPH, Euro-Graphics, Pacific Graphics, *TVCG*, and so on.

Xinguo Liu received the BS and PhD degrees in applied mathematics from Zhejiang University in 1995 and 2001, respectively. He is a professor at the School of Computer Science and Technology, Zhejiang University. He was with Microsoft Research Asia in Beijing during 2001-2006, and then joined in Zhejiang University. His main research interests are in graphics and vision, particularly geometry processing, realistic and image-based rendering, and 3D reconstruction.

Hujun Bao received the BS and PhD degrees in applied mathematics from Zhejiang University in 1987 and 1993, respectively. He is a distinguish professor at the School of Computer Science and Technology, Zhejiang University and the director of State Key Laboratory of CAD & CG. His main research interest is computer graphics and computer vision, including real-time rendering technique, geometry computing, virtual reality, and 3D reconstruction.