Computer-Aided Design ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Feature aligned quad dominant remeshing using iterative local updates

Yu-Kun Lai [a], Leif Kobbelt [b], Shi-Min Hu [a,*]

[a] Tsinghua University, Beijing, China
[b] RWTH Aachen University, Aachen, Germany

ARTICLE INFO

Article history:
Received 27 June 2008
Accepted 20 February 2009

Keywords:
Quad dominant remeshing
Feature alignment
Local updates
Iteration

ABSTRACT

In this paper we present a new algorithm which turns an unstructured triangle mesh into a quad dominant mesh with edges well aligned to the principal directions of the underlying surface. Instead of computing a globally smooth parameterization or integrating curvature lines along a tangent vector field, we simply apply an iterative relaxation scheme which incrementally aligns the mesh edges to the principal directions. We further obtain the quad dominant mesh by dropping the not-aligned diagonal edges from the triangle mesh. A post-processing stage is introduced to further improve the results. The major advantage of our algorithm is its conceptual simplicity since it is merely based on elementary mesh operations such as edge collapse, flip, and split. Various results are presented in the paper; they show a good alignment to surface features and rather uniform distribution of mesh vertices. This makes them well suited, e.g., as Catmull–Clark Subdivision control meshes.

1. Introduction

After the technology for 3D geometry acquisition has become both powerful and simple to use in recent years, the generation of faithful digital 3D models of complex objects is now being used in more and more application fields ranging from Computer Graphics and Computer Aided Design to Rapid Prototyping and Computer Aided Manufacturing. However, while earlier algorithms for 3D reconstruction have focused on the generation of highly detailed but unstructured triangle meshes, a recent shift can be observed towards the generation of structured meshes where the vertex sample pattern takes the underlying surface geometry into account.

Since the local geometry of a surface can be characterized by two principal curvatures and the corresponding two principal directions, quad dominant meshes are usually preferred over triangle meshes because they can represent this structure in a more natural way. Moreover there are many classical results from differential geometry [1] which imply that well-aligned quad meshes provide a better surface approximation [2] and minimize normal noise [3].

Among the many different algorithms for quad mesh generation, we can identify several classes of algorithms which are designed such that certain mesh properties are guaranteed. For example we can distinguish quad dominant vs. pure quad meshing schemes and conforming schemes vs. schemes that produce T-vertices. Besides this, the fundamental quality criteria are mostly identical:

- The mesh structure should be as regular as possible. (no "unstructured quad meshes").
- Individual faces should be as rectangular as possible.
- Faces should be aligned to the principal directions in general and to sharp features in particular.
- The size of the faces should adapt to the local curvature.

Especially the last requirement is, however, in conflict to the other ones since adaptive mesh resolution is difficult to achieve in conforming quad meshes. This is due to the fact that regularity and rectangularity imply non-local consistency conditions (see Fig. 1).

In order to relax these consistency conditions, we propose in this paper a *quad dominant* meshing scheme (not pure quad) which generates *T-vertices*. These relaxed conditions allow us to compute quad meshes with high rectangularity and good adaptivity of the mesh resolution. In fact, T-vertices seem to be the appropriate means to produce quad meshes with adaptive resolution as can be seen in many hand-made CAD models (see Fig. 2).

In Computer Graphics, T-vertices seem to have a "bad reputation" because in a polygonal mesh, obtuse inner angles close to π usually lead to shading artifacts due to badly estimated normal vectors and interpolation singularities. While this is true for polygonal meshes, it turns out to be less relevant if we consider quad meshes as a structured geometry representation that is being used for sophisticated downstream applications like shape modeling, segmentation into constructive elements, or tool

* Corresponding address: Tsinghua University, Department of Computer Science & Technology, 100084 Beijing, China. Tel.: +86 10 62782052; fax: +86 10 62771138.
E-mail addresses: laiyk03@mails.tsinghua.edu.cn (Y.-K. Lai), kobbelt@informatik.rwth-aachen.de (L. Kobbelt), shimin@tsinghua.edu.cn (S.-M. Hu).

0010-4485/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2009.02.017

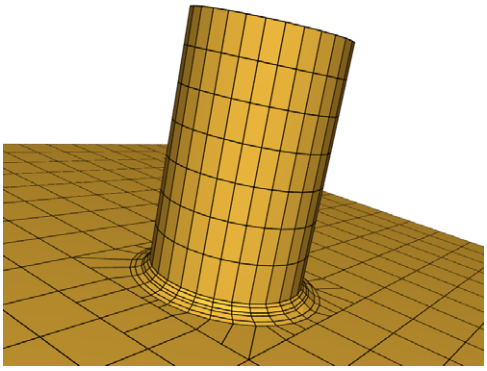

Fig. 1. Global regularity and conformity for quad meshes lead to non-local consistency conditions and may cause strongly distorted quads. T-vertices provide the flexibility to adjust the quad mesh resolution and to avoid the accumulation of distortion.

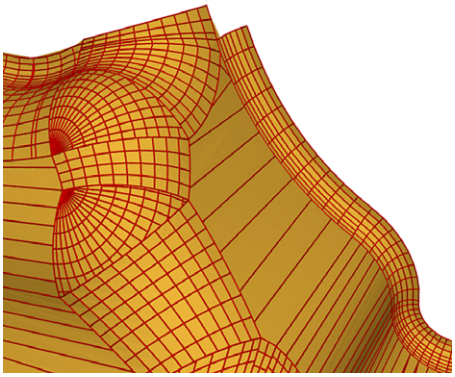

Fig. 2. Quad meshes are a well-established representation in shape design and modeling applications. In order to flexibly change the resolution while not compromising the local regularity T-vertices are introduced.

path generation. Higher order geometry representations such as T-splines [4] and Catmull–Clark subdivision surfaces can handle T-vertices in a natural manner and no surface singularities are caused by them.

Another design goal for our algorithm is simplicity. Instead of doing complex calculations to analyze the geometric and topological structure of the input shape, our algorithm is based on a small number of mesh operations which are available in every polygonal mesh library. The major processing stage of our algorithm is a simple mesh relaxation scheme that moves mesh vertices in tangent direction in order to promote edge alignment to the principal directions. The vertex sliding is interleaved with local connectivity updates (edge collapses, splits, and flips) in order to prevent the mesh connectivity from degenerating. The overall process is illustrated in Figs. 3 and 4.

This paper is an extended version of [5]. In particular, more discussions and experimental results with respect to different resolutions and feature sensitivity are presented. Section 2 briefly reviews related work. The overall algorithm is given in Section 3 and the key component of incremental alignment in Section 4. Experimental results are given in Section 5, with conclusions and discussions in Section 6.

2. Related work

Remeshing has been an active research topic for years and a thorough survey is well beyond the scope of this paper. The reader is rather referred to [6] for an excellent and comprehensive overview. Remeshing methods can be classified based on the output mesh structure into triangle remeshing and quad remeshing.

2.1. Triangle remeshing

Early work focuses on semi-regular triangle remeshing which produces mostly regular vertices (of valence 6) except for a few isolated extraordinary vertices in the output mesh. A coarse triangle mesh is constructed from the dual of a quasi-Voronoi diagram [7] or from mesh simplification [8] and uniform subdivision is applied.

Isotropic remeshing [9–12] produces approximately equilateral triangles of the same size without caring about topological regularity. [9] uses a global conformal parameterization which restricts the allowed topology of the input meshes. [10] generalizes this by using local parameterizations. The methods in [11,12] achieve high quality isotropic remeshing by a series of local mesh modifications, which is similar in spirit to our approach in this paper.

Isotropic remeshing can be adapted to vary the sizes of triangles according to the local curvature. [9] uses a density map to control the mesh resolution. Lai et al. [13] propose to use a curvature sensitive distance metric to anisotropically remesh models so that triangles are elongated along sharp and semi-sharp features. Alignment of edges with such features was specifically studied in [3]. Other approaches (e.g. [14]) try to recover artifacts of features introduced by scanning or remeshing by a post-filtering operation, which may be used after feature unaware remeshing methods to improve the quality of output meshes.

2.2. Quad remeshing

Alliez et al. [15] propose a method to integrate principal curvature lines in the parameter domain of the input mesh and generate a quad dominant output mesh by intersecting these lines. Marinov and Kobbelt [16] extend this work by directly integrating curves on the input model. Streamline integration methods add integration curves in a greedy fashion, thus they cannot guarantee a globally uniform distribution. Principal directions usually suffer from singularities, even after smoothing. Dong et al. [17] hence suggest to use the gradient of a smooth harmonic scalar field instead of principal directions. More regular results are obtained at the cost of necessary user intervention and the loss of feature alignment.

Global parameterization has also proven to be a powerful tool for quad remeshing. Geometry images [18] remesh arbitrary input meshes into all-quad meshes, by cutting the input mesh into a topological disk (a fundamental domain), parameterizing and regularly sampling it over a square domain. Multi-chart geometry images [19] extend this idea and reduce the mapping distortion significantly. The input model is segmented into pieces and they are parameterized one by one. A zippering operation is performed to keep the remeshed model water-tight. Periodic global parameterization is proposed in [20] to parameterize the input model so that principal directions are aligned with coordinate axes in the parameter domain. Non-linear optimization is required to achieve this. Remeshing can then be performed by regular sampling in the parameter domain. Kälberer et al. [21] improve this by converting a given frame field into a single vector field on a branched covering of the 2-manifold and producing quadrilateral meshes with fewer singularities.

When using local parameterizations for quad meshing, the input mesh has to be decomposed into patches and compatibility conditions have to be satisfied along patch boundaries. Dong et al. [22] propose a quad remeshing method which connects extrema of Laplacian eigenfunctions via gradient flow to form a quadrangular base mesh. High quality remeshing with few extraordinary vertices is achieved this way, although features are usually not well captured. In [23] the patch layout is

Please cite this article in press as: Lai Y-K, et al. Feature aligned quad dominant remeshing using iterative local updates. Computer-Aided Design (2009), doi:10.1016/j.cad.2009.02.017

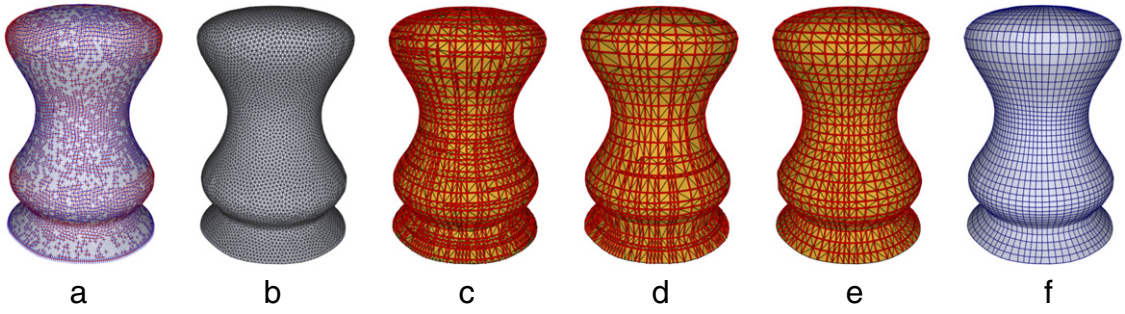

Fig. 3. Intermediate results produced by the individual steps of our algorithm. (a) The input triangle mesh with smoothed principal direction field. (b) The triangle mesh produced by curvature adaptive isotropic remeshing. (c) and (d) The triangle mesh during iterative optimization when α reduced to 50% and 0% of the initial value; well-aligned, aggregated edges are highlighted. (e) The triangle mesh after the incremental optimization, with well-aligned edges highlighted. (f) Output quad dominant mesh after post-processing.

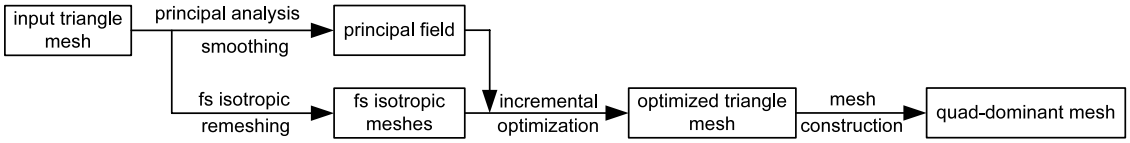

Fig. 4. Mesh processing pipeline for incremental quad dominant meshing.

prescribed manually and more general compatibility conditions are considered allowing for a swap of the principal directions. The work by Boier-Martin et al. [24] produces quad dominant meshes by first constructing a coarse domain using clustering techniques. A two-stage remeshing algorithm is proposed in [25] that first segments the input mesh into patches using a variant of the variational shape approximation algorithm [2] and then applies a combinatorial optimization procedure to build the output mesh from a set of smooth curves. This method is specifically targeting at the generation of coarse output meshes. A contouring based approach was proposed in [26] that can run at interactive rates, however sacrificing regularity and feature alignment in the output mesh.

Unlike most other approaches, Liu et al. [27] considers the problem of producing quad dominant meshes with each quad being planar. This is accomplished by a post-optimization of the vertex positions of a quad dominant mesh, keeping the connectivity unchanged. This post-optimization could be applied to our quad meshes if this property is desired. Tchon and Cameredo [28] propose to use iterative, specialized local operators to produce quad dominant meshes, which bears some similarity with our method, however, their method applies only to 2D meshes.

In [29] an algorithm for the generation of high quality T-spline control meshes is presented which is based on the periodic global parameterization but allows for more flexible user intervention. Our output meshes can also be used as T-spline control meshes.

3. Overview of the algorithm

Given an unstructured triangle mesh $\mathcal{M}$ as input, we first have to compute a tangent direction field to which we want to align the quad faces. Usually we will use the principal directions of the underlying surface estimated at the vertices of $\mathcal{M}$, but other choices would be possible as well. Next we perform an isotropic remeshing $\mathcal{M} \rightarrow \mathcal{M}'$ which is feature sensitive in the sense that the local vertex density is adapted to the maximum curvature [13]. This pre-process is necessary to provide a sufficient number of degrees of freedom for the incremental alignment procedure.

The basic idea of the incremental alignment is to let the vertices of $\mathcal{M}'$ slide over the input mesh $\mathcal{M}$ such that edges of $\mathcal{M}'$ become aligned to the principal directions of $\mathcal{M}$. The sliding is controlled by various forces which promote alignment as well as uniform distribution. Since at each surface point there are four principal directions (minimum and maximum curvatures in two opposite directions each), the alignment force can only take up to four adjacent edges into account. The other edges do not imply any forces and will end up as diagonals. Eventually these diagonals are removed from the mesh $\mathcal{M}'$ such that a quad dominant output mesh is generated. In order to avoid degenerate triangles, we adjust the mesh connectivity during vertex sliding. By using such techniques, mesh edges tend to align principal directions (or other given orthogonal fields) well, which is a very important property for quad meshes to be useful.

3.1. Principal direction estimation

Our algorithm aligns mesh edges to an arbitrary pair of tangent direction fields. The geometrically most relevant directions on a surface are the principal directions which indicate at each point on a surface the (mutually orthogonal) tangent directions of minimum and maximum curvatures [1].

We use the methods proposed in [30,31] to estimate the principal directions at the vertices of the input mesh $\mathcal{M}$. Since this estimate is sensitive to noise and since it cannot guarantee a smooth direction field in nearly umbilic regions of the surface, we have to apply a smoothing operator. The number of singularities in the resulting direction fields can be reduced considerably by treating the local direction information as a 4-symmetric vector field [32]. We apply the method proposed in [33] to smooth the principal frames by using a simple non-linear optimization scheme which tends to converge very robustly and fast.

The principal direction estimation and smoothing provides an orthogonal coordinate frame $F_i = (X_i, Y_i, Z_i)$ for each vertex in the input mesh $\mathcal{M}$ where X_i is the minimum curvature direction, Y_i is the maximum curvature direction, and Z_i is the normal vector. We interpolate this direction information linearly across triangles by using barycentric coordinates.

For the normals, linear interpolation is straightforward. For the tangent directions, however, we first have to find the proper permutations of the directions for interpolation (due to 4-symmetry). Let F_i, F_j, and F_k be the local frames at the three vertices

Please cite this article in press as: Lai Y-K, et al. Feature aligned quad dominant remeshing using iterative local updates. Computer-Aided Design (2009), doi:10.1016/j.cad.2009.02.017

of a triangle. For each triplet (D_i, D_j, D_k) we check the consistency defined by

$$\text{consistency}(D_i, D_j, D_k) := D_i^{\mathrm{T}} D_j + D_j^{\mathrm{T}} D_k + D_k^{\mathrm{T}} D_i$$

where the direction D_i is taken from the set $\{X_i, -X_i, Y_i, -Y_i\}$ and the directions D_j and D_k are chosen analogously (yielding a total of $4^3 = 64$ combinations). From the triplet $(\hat{D}_i, \hat{D}_j, \hat{D}_k)$ with maximum consistency we compute the first interpolated tangent vector X by barycentric combination. The second tangent vector is then defined as $Y = Z \times X$ where Z is the interpolated normal vector.

Notice that only the barycentric interpolation has to be computed during the incremental mesh optimization. Estimating the principal directions at the vertices of $\mathcal{M}$ and finding the most consistent triplet per triangle is done in a pre-process.

3.2. Initial mesh generation

The starting configuration is crucial for incremental optimization. Hence we make sure that the initial mesh $\mathcal{M}'$ locally has sufficiently many degrees of freedom, i.e. that the vertex density of $\mathcal{M}'$ is higher in curved regions and lower in flat areas. We use the method proposed in [13] which produces isotropic meshes with the vertex density adapted to the local curvature. This is achieved by computing a uniform vertex density with respect to the augmented metric

$$\text{dist}(V_i, N_i, V_j, N_j) = \sqrt{\|V_i - V_j\|^2 + \omega^2 \|N_i - N_j\|^2}, \quad (1)$$

where V_i and N_i are position and unit normal vectors of vertex v_i; correspondingly V_j and N_j are those of vertex v_j. This metric takes both Euclidian distance $\|V_i - V_j\|^2$ as well as unit normal rotation $\|N_i - N_j\|^2$ into account. The non-negative coefficient ω is used to control the sensitivity of the density adaption. The algorithm first distributes unconnected vertices on the surface by using particle repulsion [34] and then recovers the mesh connectivity by constrained Delaunay triangulation in a local parameterization [13].

3.3. Incremental optimization

Sliding the vertices of $\mathcal{M}'$ over $\mathcal{M}$ such that the (some) edges of $\mathcal{M}'$ become aligned to the tangent direction fields of $\mathcal{M}$ is the most important step of our quad dominant remeshing scheme. Since at every surface point we can identify four principal directions and the average valence of a vertex in a triangle mesh is six, it is obvious that not all edges can be aligned. In fact, some edges will turn out to become diagonally oriented with respect to the principal direction fields. The various forces that act on the vertices in this stage and the overall incremental optimization procedure is described in detail in Section 4.

Technically, the vertex sliding is implemented by computing a local parameterization for the 1-ring of each vertex in $\mathcal{M}'$ and then performing the position update in the parameter domain. After the update, the 3D position is recovered by simply evaluating this parameterization. Even if this already guarantees a good preservation of the geometry, it can still happen that the vertices of $\mathcal{M}'$ drift away from the input mesh $\mathcal{M}$. Hence we project the vertices of $\mathcal{M}'$ back to the nearest point of $\mathcal{M}$ after every iteration. The vertex re-location is concluded by re-evaluating the (interpolated) principal direction field in order to update the local coordinate frame associated with a vertex.

When the vertices of $\mathcal{M}'$ are freely sliding across the input mesh $\mathcal{M}$, the mesh structure can degenerate. Hence, in order to preserve a good mesh quality, we have to apply a local remeshing procedure. Since the step width of the vertex motion is rather small in our incremental procedure, simple local connectivity updates turn out to be sufficient. We use a variant of the connectivity update procedure described in [12]:

After re-location and back-projection of the vertices, we

- collapse edges that have become shorter than some threshold $\Theta_{\min}$.
- split edges that have become longer than some threshold $\Theta_{\max}$.
- flip edges if the maximum inner angle of one of the adjacent triangles is above some threshold $\Phi_{\max}$.

The choice of the three thresholds $\Theta_{\min}$, $\Theta_{\max}$, and $\Phi_{\max}$ does not require too much tuning. $\Theta_{\min}$ and $\Theta_{\max}$ control the mesh resolution and the aspect ratio of the resulting quad faces. $\Phi_{\max}$ has always been set to $\pi - \varepsilon$ in our experiments.

3.4. Quad mesh generation and post-processing

After a few iterations of the incremental optimization, we obtain a triangle mesh $\mathcal{M}'$ which exhibits a good local edge alignment to the principal directions of $\mathcal{M}$. Removing the badly aligned edges (the diagonals) from the mesh already leaves a quad dominant mesh with mostly rectangular faces.

3.4.1. Post-processing for improving polygon quality

Some simple post-processing can further improve the quality of the output mesh by removing badly shaped faces through merging and splitting.

In the merging phase, adjacent faces are merged by removing their common edge from the mesh. An edge is removed if the quality of the resulting face is superior to the quality of the two faces before the merge. Here, the quality of a face is measured by its rectangularity, i.e. by the number of quasi-perpendicular corners (should be four) and by the number of non-convex corners (should be zero). A quasi-perpendicular corner is one with an inner angle close to $\frac{\pi}{2}$, a non-convex corner is one with an inner angle well above π. The inner angle of a corner is measured in the tangent plane defined by the local normal vector.

Notice that we do not penalize T-vertices, i.e. vertices with an inner angle of approximately π. Please refer to our discussion of the role of T-vertices in quad meshing in the introduction.

The merging phase is followed by a splitting phase where we remove the remaining non-convex corners by splitting the corresponding non-convex faces into convex ones. Here a simple heuristic finds the most rectangular decomposition of a non-convex face.

3.4.2. Post-processing for T-vertices

Due to the local minimum nature of our method, it may produce some unnecessary T-vertices in certain cases. Although as discussed before, T-vertices are not an impediment for various geometry processing applications, reducing the number of T-vertices, or placing them at more reasonable places is still of benefit. For this purpose, a simple post-processing stage may be introduced. For T-vertices located within regular regions, the consecutive edge sequences ending at the T-vertex can be extended by appropriately splitting a sequence of quads. An extended edge sequence terminates when another T-vertex is met or when the edge sequence is sufficiently close to nearby edges. The extension of edge sequences is always restricted to regular quad regions so that regularity is guaranteed after splitting. After this phase, edge sequences that are too short or too close to neighboring sequences will be removed.

Besides proper alignment, another goal of the iterative vertex re-location detailed in Section 4 is to promote a globally uniform size of the quad faces. However, after the elimination of unnecessary T-vertices, the line density, i.e. the distribution of (parallel) edge sequences can become non-uniform because new

Please cite this article in press as: Lai Y-K, et al. Feature aligned quad dominant remeshing using iterative local updates. Computer-Aided Design (2009), doi:10.1016/j.cad.2009.02.017

edges have been inserted and quad faces have been split. We re-establish uniformity by iterating a simple smoothing operator. Here we can exploit the fact that the mesh structure is highly regular at this stage of the algorithm.

The smoothing operator performs a simple relaxation with back-projection to the original surface. For the relaxation we distinguish between regular vertices (valence four with all adjacent faces being quads), T-vertices (between two small and one larger quad), and irregular vertices (all the rest).

The positions of the irregular vertices are not changed by the smoothing operator. T-vertices are shifted towards the mid-point of the edge on which they lie and regular vertices are shifted towards the average of their four adjacent vertices. Fig. 8 in the experimental results section shows an example mesh before and after the final relaxation step.

4. Incremental alignment

After we have explained the overall algorithm, we now focus on the most crucial stage of the process which is the incremental motion of vertices in order to promote principal direction alignment. For every vertex **v** of $\mathcal{M}'$, we evaluate the interpolated principal direction field and associate a local coordinate frame F with it. We do the same for the midpoint of every edge.

4.1. Local parameterization

To simplify the vertex motion, we map the 1-ring neighborhood of a vertex **v** to a 2D domain. In order to minimize distortion, we use the exponential map [35,8] which preserves the length of the edges adjacent to **v** and scales the adjacent inner angles such that they sum to 2π (planar configuration). The vertex **v** is mapped to the origin and the configuration is rotated such that the principal directions X and Y at **v** coincide with the x- and y-axis of the 2D domain.

For the re-location of the center vertex, we have to identify the most appropriate set of (up to four) adjacent edges which qualify as candidates to be aligned to the minimum or maximum principal directions (x- and y-axis respectively). We do this by selecting those edges which have the least angle to the (positive or negative) x- or y-axis. Selected edges are rejected if their angle with respect to the corresponding axis is more than $\frac{\pi}{3}$. This leads to situations where less than four edges may remain as alignment candidates. Only these candidates will affect the re-location of the center vertex, the others are not taken into account.

4.2. Curl compensation

Since the edges of $\mathcal{M}'$ have a finite length, the orientation of the direction field may change along an edge (*curl*). For symmetry reasons we want to align a candidate edge **e** to the direction field evaluated at its midpoint. However, since in general the corresponding local frame $F_e = (X_e, Y_e, Z_e)$ is different from the local frame $F_v = (X_v, Y_v, Z_v)$ at the center vertex **v** we have to compensate the curl by rotating **e** in the parameter domain. To simplify the following explanation we assume that the tangent direction vectors X_e and Y_e have already been chosen such that maximum consistency with the vectors X_v and Y_v is established.

Let E_e be the 3D (geometric) embedding of the (topological) edge **e** then $(x_e, y_e) = (E_e^{\mathrm{T}} X_e, E_e^{\mathrm{T}} Y_e)$ are the coordinates of **e** in the tangent plane with respect to the local coordinate frame F_e.

The curl compensation consists in a rotation of the frame F_e into the frame F_v. In the parameter domain where F_v coincides with the x- and y-axis, we are only interested in the tangential component of this rotation which corresponds to replacing the edge **e** obtained by the exponential map parameterization (in the last section) with the edge spanned by (x_e, y_e) and the origin (= location of **v** in the parameter domain).

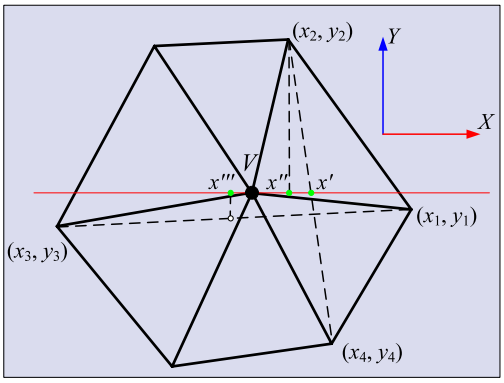

Fig. 5. Illustration of vertex re-location. The center vertex is pulled towards a weighted average of three target positions x', x'', and x'''.

Notice that even if we will use the curl compensated edges in the parameter domain to compute the update vector for the center vertex (in the next section), we still stick to the candidate selection based on the orientation *before* the compensation. The reason for this is to maintain the compatibility of the minimum and maximum principal directions at the center vertex.

4.3. Vertex re-location

Let $(x_1, y_1), \ldots (x_4, y_4)$ be the curl compensated endpoints of the four adjacent edges being candidates for alignment to the minimum (X), maximum (Y), negative minimum ($-X$), and negative maximum ($-Y$) principal directions respectively (see Fig. 5). We will define three different forces that attract the center vertex **v** towards an optimized position in the parameter domain. For brevity we will only explain how to compute the x-coordinates. The y-coordinates are obtained analogously.

The first force ("collinearity force") is trying to make the points (x_2, y_2), **v**, and (x_4, y_4) collinear:

$$x' = \frac{|y_4|}{|y_2| + |y_4|}\, x_2 + \frac{|y_2|}{|y_2| + |y_4|}\, x_4.$$

The corresponding force for the y-coordinates is making (x_1, y_1), **v**, and (x_3, y_3) collinear.

The second force ("snapping force") is promoting edge alignment by enforcing vertical edges. This is achieved by snapping to the closer x-coordinate of the corresponding neighbors:

$$x'' = \begin{cases} x_2 & (|x_2| \le |x_4|) \\ x_4 & (|x_2| > |x_4|). \end{cases}$$

For the y-coordinates horizontal edges are enforced.

The third force ("relaxation force") is promoting uniform vertex distribution *along* the principal directions:

$$x''' = \frac{x_1 + x_3}{2}.$$

Eventually, the updated x-coordinate for the center vertex is defined as a weighted average:

$$(1 - \lambda)\, x + \lambda \left(\alpha\, x' + (1 - \alpha - \beta)\, x'' + \beta\, x'''\right).$$

During the incremental optimization we change the weight coefficients α and β according to the following schedule. We begin with α being relatively large (e.g. $\alpha = 0.3$) and $\beta = 0$. With each iteration, we let α decrease towards zero (with a constant decrease of amount after each iteration). Once α vanishes, we start to increase β with each iteration up to some moderate value (e.g. $\beta = 0.2$).

The rationale behind this schedule is that initially the mesh should be straightened by making vertices locally collinear.

Please cite this article in press as: Lai Y-K, et al. Feature aligned quad dominant remeshing using iterative local updates. Computer-Aided Design (2009), doi:10.1016/j.cad.2009.02.017

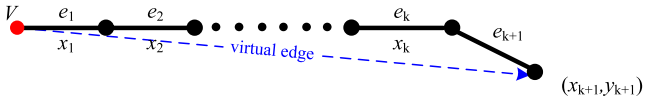

Fig. 6. Well-aligned edges are aggregated and treated as rigid components. In the vertex re-location, this is taken into account by constructing virtual edges (as delineated in dash line).

Then the actual alignment to the principal directions should be promoted. After some iterations, the alignment is established and we fade in a regularization force to make the vertex distribution more uniform. Although the weights for collinearity force and relaxation force are relatively small, they are important to improve the results. The collinearity force is introduced to improve the alignment in the initial iterations. Without the relaxation force, aligned edge sequences will not tend to distribute uniformly. These forces take effect together with the snapping force that promotes the alignment with the principal directions. The suggested values are obtained from experiments, and we have found that the same set of values is suitable for all the examples presented in the paper.

This schedule together with a damping factor of $\lambda = \frac{1}{2}$ usually guarantees stable convergence of the scheme to a good local minimum. Experiments also show that the same set of parameters (and parameter variation after iteration) is suitable for a wide range of models.

For vertices with less than four alignment candidates, we apply just the snapping force for a direction where only one candidate is present. For the other coordinate, collinearity and snapping is applied but no relaxation. If no candidate is present for one direction then there is no update of the respective coordinate at all.

4.4. Aggregation

Updating the position of each vertex in $\mathcal{M}'$ individually may lead to a very slow convergence and has a high risk of getting stuck in a sub-optimal local minimum. Hence we have to stabilize the convergence by making sure that we are not destroying a good local configuration in future relaxation steps.

We achieve this by *aggregation*, i.e. by combining adjacent well-aligned edges into a single rigid component which can only be updated simultaneously. Aggregation speeds up convergence significantly since position and orientation information is propagated much faster through the mesh.

Initially each edge of $\mathcal{M}'$ represents an individual rigid component. After each iteration, we classify edges as *well aligned*, if their orientation deviates from one of the two principal directions by less than some small threshold. If two adjacent edges are both well aligned with respect to the same principal direction they are treated as one rigid component in the next iteration. Contiguous sequences of well-aligned edges can be aggregated into chains in the same way. In practical implementation, we do not exert extra limitations to the length of an aggregated chain. Edges are re-classified and chains are re-aggregated in each iteration to avoid bad local minima.

The update procedure for vertices $\mathbf{v}$ that belong to an aggregated chain has to be modified. For simplicity we, again, assume that all the local coordinate frames have been permuted correctly to establish maximum consistency between adjacent vertices and edges. Without loss of generality we can further assume that the aggregated edges are well aligned to the maximum principal direction (X). Y-alignment is handled analogously.

As illustrated by Fig. 6, we still start by computing a parameterization of the 1-ring neighborhood of $\mathbf{v}$, however, instead of applying the curl compensation to an already well-aligned edge $\mathbf{e}_1$, we follow the aggregated chain $\mathbf{e}_2, \ldots, \mathbf{e}_k$ to its end vertex where we find a not yet well-aligned edge $\mathbf{e}_{k+1}$ which is a candidate for alignment. Since all the intermediate aggregated edges are already well aligned (to the X direction), their local coordinates in the respective tangent planes are $(x_i, y_i) = (x_i, 0)$. Propagating the curl compensation along the aggregated chain leads to the *virtual* edge:

$$\left(\sum_{i=1}^{k+1} x_i, y_{k+1}\right)$$

This definition makes sure that the collinearity and the snapping forces for the y-coordinates are compatible for all vertices that belong to the same aggregation chain in X direction.

The relaxation force for the y-coordinate has to be treated differently since it uses adjacent edges in Y direction which are different for each vertex in an X chain. In order to make sure that the well-alignment of the whole chain is not affected, we compute the y-coordinate relaxation force for each vertex in an X chain individually and then compute an average update which we apply to all the vertices. By this we allow the X chain to move parallel in Y direction in order to make the overall vertex distribution more uniform.

Intermediate results of iterative update with aggregation are presented in Fig. 3 (c–e). The detected aggregations are highlighted. We may consider the iterative local updates process as two stages. In the first stage (from (b) to (d)), the purpose of iteration is to make the candidate edges well align with principal fields, and this is achieved by a combination of collinear force and snapping force. In the second stage (from (d) to (e)), due to the introduction of relaxation force, edge sequences tend to become more uniformly distributed.

5. Experimental results

We tested our algorithm with various geometric models of different complexity. The results were generally quite good with mesh edges mostly well aligned to the principal directions and quads distributed evenly.

Fig. 7 shows various graphical models remeshed with our method. Smoothed principal directions are used in these examples to guide the orientation of the edges. Since we do not rely on a global parameterization, the method can deal with high-genus models in the same way, as illustrated by the genus 4 model of 'fertility'.

Fig. 8 is the remeshed rockerarm model, a typical mechanical part. Thanks to the non-linear 4-symmetric vector field smoothing, even critical configurations like the non-convex flat region can be remeshed quite reasonably with mostly quads. The left and right figures show the model before and after relaxation, respectively; uniformity is re-established after relaxation.

In Fig. 9 we used a user specified direction field instead of the principal directions to improve the mesh quality in noisy regions. Remeshing with the principal directions tends to produce irregular meshes on the palm due to the existence of creases. The tangent vector field shown on the left is computed by interpolating 5 key vectors, one on each finger, using radial basis functions.

Although it is generally difficult to precisely control the resolution of the output mesh, the number of vertices in the initial phase of feature sensitive isotropic remeshing does have certain influence. Fig. 10 shows the remeshing results obtained with initial isotropic remeshing of 10,000 and 20,000 vertices, respectively. The obtained remeshing results contain 1786 and 2842 faces, corresponding to coarser and finer initial remeshing. Besides using different resolutions of initial triangle meshes, the number of faces in the output mesh is also affected by the post-processing stage. Merging and splitting of polygons will slightly decrease or increase the number of faces. Extending edge sequences terminating at T-vertices will increase the number of faces due to the splitting

Please cite this article in press as: Lai Y-K, et al. Feature aligned quad dominant remeshing using iterative local updates. Computer-Aided Design (2009), doi:10.1016/j.cad.2009.02.017

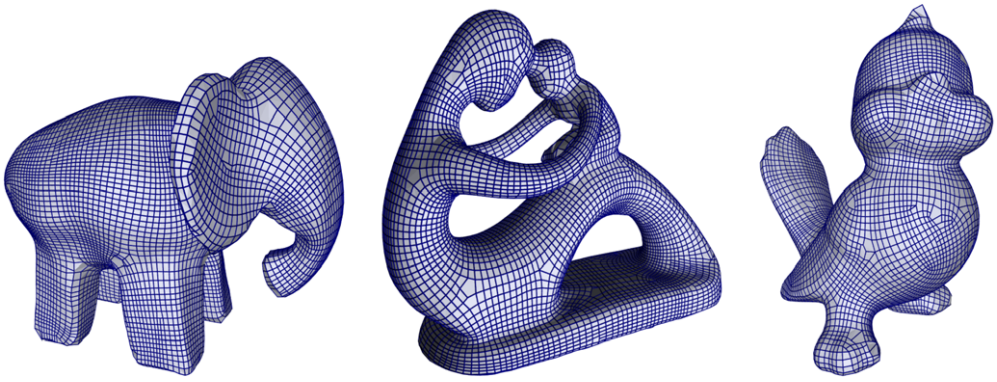

Fig. 7. Quad dominant remeshing of various graphical models. From left to right: 'elephant', 'fertility' and 'tweety'.

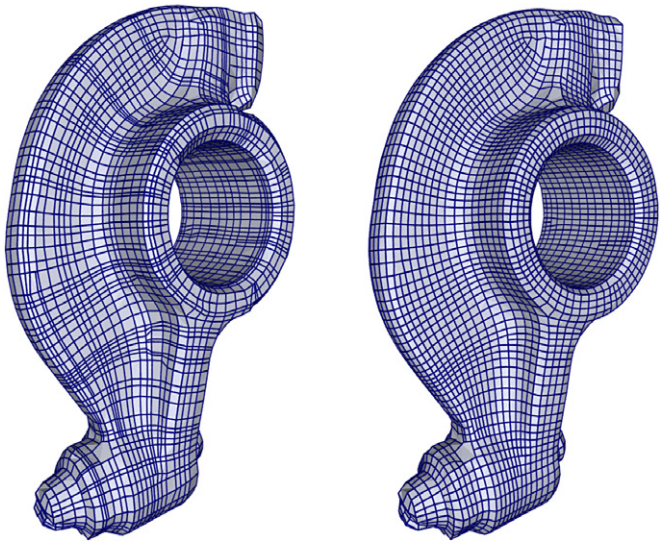

Fig. 8. Quad dominant remeshing of the rockerarm model. Left: before relaxation; right: after relaxation.

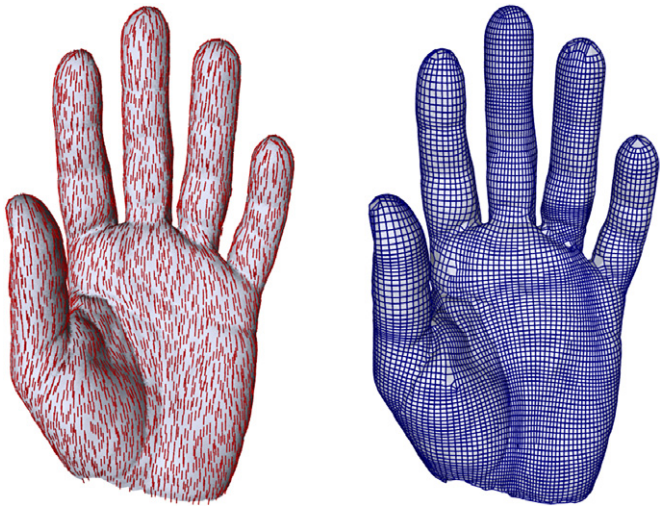

Fig. 9. Hand model remeshed with the specified vector field.

of regular faces; thus enabling this post-processing will lead to an output mesh with more faces.

Note that there still exist some polygons (other than quads) on the produced meshes, thus the output results are considered as quad dominant remeshing (not pure quads). This is partially due to the singularities of principal field (even after 4-symmetry smoothing) and partially due to the local minima nature of the iterative update scheme. Such polygons may be further improved by some heuristic rules that try to split or merge them appropriately to produce better results. It can be noticed that remeshing results usually have edges follow principal directions and features.

Feature sensitive metric [13] may be used not only as a way to control initial triangulation, but may also be applied to the whole computation pipeline. The metric is described by Eq. (1), and is used whenever the distance between two adjacent vertices is computed. By using this technique, not only mesh edges are aligned with principal directions or features, the density of faces are also adapted to the local features: feature regions tend to contain more faces than flat regions. Fig. 11 gives an example of producing quad dominant remeshing of 'bahkauv' model without and with feature sensitivity (ω is set to 0.1). Since certain steps of the algorithm use more global connectivity consistency around the model (e.g. aggregation based local remeshing and post-processing), the result obtained with feature sensitivity may not differ much from that without feature sensitivity. However, it can still be noticed that with feature sensitivity, feature regions (e.g. the legs and the head) contain increased number of faces, while other regions have the same or even less faces as the result without feature sensitivity. This property is useful for better representing feature regions with a limited number of faces.

Our remeshing results can be used as control meshes of T-splines or Catmull–Clark subdivision surfaces. Fig. 12 shows such an example.

Catmull–Clark subdivision can be generalized to meshes with T-vertices in a straightforward manner. The original Catmull–Clark scheme computes a new vertex for every edge and every face of the input mesh. Then each n-sided face is split into n quads by connecting the edge-vertices to the face vertices. If an edge of the input mesh has a T-vertex then it can be used instead of computing a new vertex for this edge.

In order for this modified Catmull–Clark subdivision to work, we have to make sure that there is no edge in the mesh that has more than one T-vertex. We achieve this by applying some more face splits to our quad meshes. Usually only very few edges have multiple T-vertices.

All our experiments are carried out on a commodity PC with Intel Core2Duo 1.86 GHz processor and 1 GB RAM. Most models are initially remeshed to 10,000 vertices. The principal direction computation takes 1–2 s, the non-linear vector field smoothing (not optimized) 15–20 s, the incremental optimization 15–20 s, mesh reconstruction less than a second and finally post-processing 2–3 s. The hand model is initially remeshed to 20,000 vertices and the timings for each step are 3.21 s, 122.31 s, 31.73 s, 0.72 s and 3.37 s respectively.

Please cite this article in press as: Lai Y-K, et al. Feature aligned quad dominant remeshing using iterative local updates. Computer-Aided Design (2009), doi:10.1016/j.cad.2009.02.017

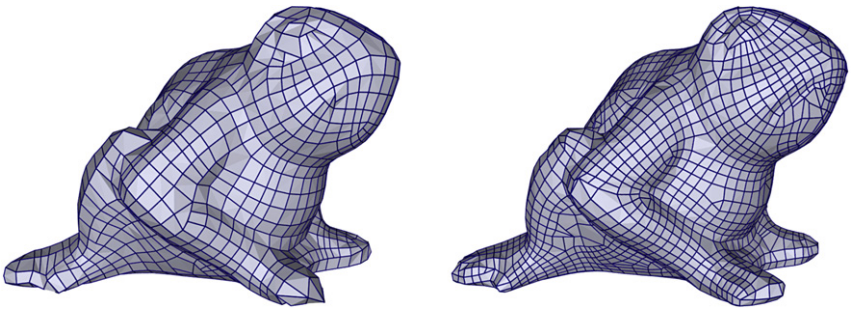

Fig. 10. Remeshing of 'frog' model with two different resolutions.

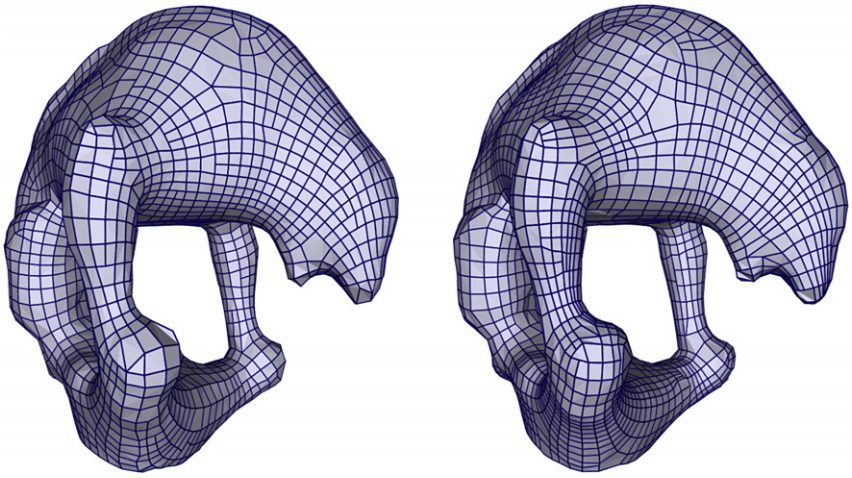

Fig. 11. Remeshing of 'bahkauv' model without and with feature sensitivity, $\omega = 0.10$.

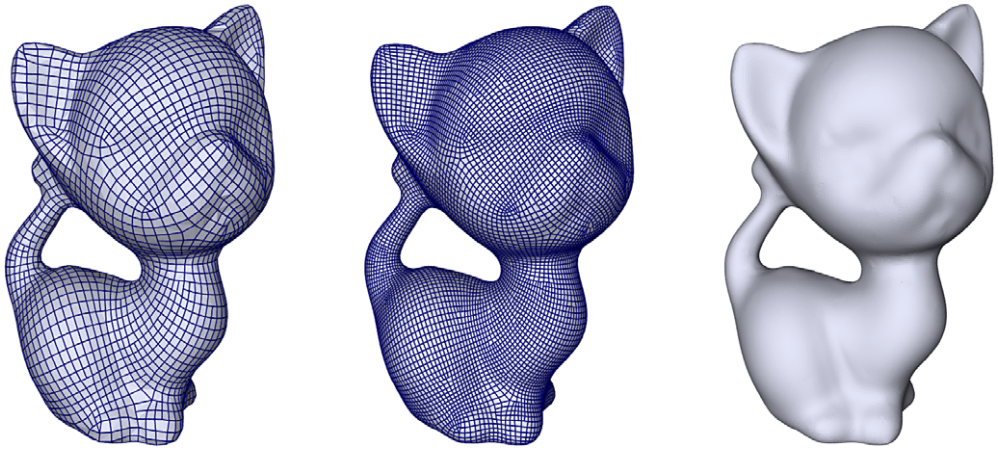

Fig. 12. The kitten model. Left: quad dominant remeshing; middle and right: the remeshed model undergone one and two times of Catmull–Clark subdivision.

Statistics of the number of faces, the number of quads, the average deviation of inner angles of quads from $\pi/2$ (in degrees) and the planarity measure (in degrees, explained below) are given in Table 1. From the statistics, it is clear that our method produces meshes with mostly quads. Since the mesh edges are optimized to align with principal directions (or other orthogonal fields), the shape of quads is close to a rectangle, which is shown by the small averaged deviation angle of corners from $\frac{\pi}{2}$. Moreover, planarity measure is computed as the difference of averaged sum of inner angles from 2π over all the quads appear in the model. Experimental results given in the table show that this measure is very close to 0 for all the models (less than 1.0 degree for most examples). Most faces are actually close to planar quadrilaterals, thus the output of our method is suitable as input for generating PQ meshes (quad meshes with planar faces) using some variant of [27].

6. Conclusions and future work

In this paper, a quad dominant remeshing method based on incremental optimization has been proposed. The approach is conceptually simple, and produces quad dominant meshes that show a very good alignment to geometric features and a rather uniform distribution of mesh vertices. T-vertices are allowed in the output meshes, since they provide more flexibility in order to improve the alignment and the shape of the faces.

Some limitations still exist. One is that it is generally difficult to precisely control the resolution of the output mesh. The number of

Please cite this article in press as: Lai Y-K, et al. Feature aligned quad dominant remeshing using iterative local updates. Computer-Aided Design (2009), doi:10.1016/j.cad.2009.02.017

Table 1
Statistics of remeshing results.

Model	No. of faces	No. of quads	Avg. angle dev. (deg)	Planarity (deg)
Elephant	10,450	10,349 (99.03%)	2.98	0.266
Fertility	9,486	9,390 (98.99%)	3.83	0.262
Tweety	7,069	6,765 (98.53%)	3.84	0.414
Rockerarm	6,132	6,093 (99.36%)	2.56	0.288
Hand	14,763	14,675 (99.40%)	2.24	0.455
Kitten	3,916	3,829 (97.78%)	4.99	0.441
Frog (coarse)	1,692	1,603 (94.74%)	5.87	1.181
Frog (dense)	2,842	2,641 (92.93%)	7.39	0.913
Bahkauv ($\omega = 0$)	2,710	2,596 (95.79%)	5.57	0.501
bahkauv ($\omega = 0.1$)	3,224	3,073 (95.32%)	5.12	0.475

vertices in the initial phase of feature sensitive isotropic remeshing does have certain influence. But an accurate specification of the target number of vertices cannot be fulfilled. Another limitation is that there might remain some auxiliary irregular polygons due to singularities in the tangent vector field. The update scheme is local and may also produce irregular polygons in certain cases. This could be relieved by additional post-filtering driven by some suitable heuristics.

We plan to extend our approach to multi-resolution quad remeshing in the future. Coarse quad dominant remeshing will be performed first, and finer meshes can then be computed by Catmull–Clark refinement followed by further incremental optimization (see Fig. 12). We expect that multi-resolution approaches may result in improved convergence speed and robustness, too.

Acknowledgments

The models in this paper are courtesy of AIM@SHAPE Repository. This work was supported by the National Basic Research Project of China (Project Number 2006CB303106), the Natural Science Foundation of China (Project Number 60673004) and the National High Technology Research and Development Program of China (Project Number 2007AA01Z336).

References

[1] do Carmo MP. Differential geometry of curves and surfaces. Prentice-Hall; 1976.
[2] Cohen-Steiner D, Alliez P, Desbrun M. Variational shape approximation. ACM Transactions on Graphics 2004;23(3):905–14.
[3] Botsch M, Kobbelt L. Resampling feature and blend regions in polygonal meshes for surface anti-aliasing. Computer Graphics Forum 2001;20(3): 402–10.
[4] Sederberg TW, Zheng J, Bakenov A, Nasri A. T-splines and T-NURCCs. ACM Transactions on Graphics 2003;22(3):477–84.
[5] Lai Y-K, Kobbelt L, Hu S-M. An incremental approach to feature aligned quad-dominant remeshing. In: Proc. ACM symposium on solid and physical modeling. 2008. p. 137–45.
[6] Alliez P, Ucelli G, Gotsman C, Attene M. Recent advances in remeshing of surfaces. Technical report. AIM@SHAPE Network of Excellence; 2005.
[7] Eck M, DeRose TD, Duchamp T, Hoppe H, Lounsbery M, Stuetzle W. Multiresolution analysis of arbitrary meshes. In: Proc. ACM SIGGRAPH. 1995.
[8] Lee AWF, Sweldens W, Schroder P, Cowsar L, Dobkin D. MAPS: Multiresolution adaptive parameterization of surfaces. In: Proc. of ACM SIGGRAPH. 1998. p. 95–104.
[9] Alliez P, de Verdière EC, Devillers O, Isenburg M. Isotropic surface remeshing. In: Proc. shape modeling international. 2003. p. 49–58.
[10] Surazhsky V, Alliez P, Gotsman C. Isotropic remeshing of surfaces: A local parameterization approach. In: Proc. 12th int'l meshing roundtable. 2003.
[11] Surazhsky V, Gotsman C. Explicit surface remeshing. In: Proc. Eurographics symposium on geometry processing. 2003. p. 20–30.
[12] Botsch M, Kobbelt L. A remeshing approach to multiresolution modeling. In: Proc. Eurographics symposium on geometry processing. 2004. p. 189–96.
[13] Lai Y-K, Zhou Q-Y, Hu S-M, Wallner J, Pottmann H. Robust feature classification and editing. IEEE Transactions on Visualization and Computer Graphics 2007; 13(1):34–45.
[14] Attene M, Falcidieno B, Rossignac J, Spagnuolo M. Edge-sharpener: Recovering sharp features in triangulations of non-adaptively re-meshed surfaces. In: Proc. Eurographics symposium on geometry processing. 2003. p. 63–72.
[15] Alliez P, Cohen-Steiner D, Devillers O, Lévy B, Desbrun M. Anisotropic polygonal remeshing. ACM Transactions on Graphics 2003;22(3):485–93.
[16] Marinov M, Kobbelt L. Direct anisotropic quad-dominant remeshing. In: Proc. of Pacific graphics. 2004. p. 207–16.
[17] Dong S, Kircher S, Garland M. Harmonic functions for quadrilateral remeshing of arbitrary manifolds. Computer-Aided Geometric Design 2005;22(5): 392–423.
[18] Gu X, Gortler SJ, Hoppe H. Geometry images. ACM Transactions on Graphics 2002;21(3):355–61.
[19] Sander PV, Wood ZJ, Gortler SJ, Snyder J, Hoppe H. Multi-chart geometry images. In: Proc. Eurographics symposium on geometry processing. 2003. p. 146–55.
[20] Ray N, Li WC, Lévy B, Sheffer A, Alliez P. Periodic global parameterization. ACM Transactions on Graphics 2006;25(4):1460–85.
[21] Kälberer F, Nieser M, Polthier K. QuadCover — surface parameterization using branched coverings. Computer Graphics Forum 2007;27(3):375–84.
[22] Dong S, Bremer P-T, Garland M, Pascucci V, Hart JC. Spectral surface quadrangulation. ACM Transactions on Graphics 2006;24(3):1057–66.
[23] Tong Y, Alliez P, Cohen-Steiner D, Desbrun M. Designing quadrangulations with discrete harmonic forms. In: Proc. Eurographics symposium on geometry processing. 2006. p. 201–10.
[24] Boier-Martin I, Rushmeier H, Jin J. Parameterization of triangle meshes over quadrilateral domains. In: Proc. Eurographics symposium on geometry processing. 2004. p. 193–203.
[25] Marinov M, Kobbelt L. A robust two-step procedure for quad-dominant remeshing. Computer Graphics Forum 2006;25(3):537–46.
[26] Canas GD, Gortler SJ. Surface remeshing in arbitrary codimensions. The Visual Computer 2006;22(9):885–95.
[27] Liu Y, Pottmann H, Wallner J, Yang Y-L, Wang W. Geometric modeling with conical meshes and developable surfaces. ACM Transactions on Graphics 2006; 25(3):681–9.
[28] Tchon K-F, Camarero R. Quad-dominant mesh adaptation using specialized simplicial optimization. In: Proc. 15th international meshing roundtable. 2006. p. 21–38.
[29] Li W-C, Ray N, Lévy B. Automatic and interactive mesh to t-spline conversion. In: Proc. Eurographics symposium on geometry processing. 2006. p. 191–200.
[30] Cohen-Steiner D, Morvan JM. Restricted delaunay triangulations and normal cycle. In: Proc. ACM symposium on computational geometry. 2003. p. 312–21.
[31] Yang Y-L, Lai Y-K, Hu S-M, Pottmann H. Robust principal curvatures on multiple scales. In: Proc. Eurographics symposium on geometry processing. 2006. p. 223–6.
[32] Ray N, Vallet B, Li WC, Lévy B. N-symmetry direction direction field design. ACM Transactions on Graphics 2008;27(2):10:1–10:13.
[33] Hertzmann A, Zorin D. Illustrating smooth surfaces. ACM Transactions on Graphics 2000;19(3):517–26.
[34] Witkin A, Heckbert P. Using particles to sample and control implicit surfaces. In: Proc. ACM SIGGRAPH. 1994. p. 269–77.
[35] Welch W, Witkin A. Variational surface modeling. In: Proc. ACM SIGGRAPH. 1992. p. 157–66.