EUROGRAPHICS 2010 / T. Akenine-Möller and M. Zwicker
(Guest Editors)

Volume 29 (2010), Number 2

Practical quad mesh simplification

Marco Tarini[1,2], Nico Pietroni[1], Paolo Cignoni[1], Daniele Panozzo[3], and Enrico Puppo[3]

[1]Visual Computing Group - ISTI-CNR, Pisa, Italy
[2]DICOM - Dipartimento Informatica e Comunicazione, Università dell'Isubria, Varese, Italy
[3]DISI - Dipartimento di Informatica e Scienze dell'Informazione - Università di Genova, Italy

Abstract

In this paper we present an innovative approach to incremental quad mesh simplification, i.e. the task of producing a low complexity quad mesh starting from a high complexity one. The process is based on a novel set of strictly local operations which preserve quad structure. We show how good tessellation quality (e.g. in terms of vertex valencies) can be achieved by pursuing uniform length and canonical proportions of edges and diagonals. The decimation process is interleaved with smoothing in tangent space. The latter strongly contributes to identify a suitable sequence of local modification operations. The method is naturally extended to manage preservation of feature lines (e.g. creases) and varying (e.g. adaptive) tessellation densities. We also present an original Triangle-to-Quad conversion algorithm that behaves well in terms of geometrical complexity and tessellation quality, which we use to obtain the initial quad mesh from a given triangle mesh.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Surface representations.s Additional keywords: Quad-meshes; Mesh-simplification.

1. Introduction

Quad meshes, i.e. meshes composed entirely of quadrilaterals, are important data structures in computer graphics. Several applications in modeling, simulation, rendering, etc. are better suited for quad meshes than for triangle meshes. In spite of this, much literature in geometry processing in the past has addressed more the case of triangle meshes, while similar problems for quad meshes are either relatively unexplored, or they have been addressed only very recently. This is the case of mesh simplification, i.e., the task of producing a low complexity mesh M' out of a high complexity one M.

Compared to the case of triangle meshes, simplification of quad meshes poses extra challenges, because quads are less adaptive and more delicate structures than triangles. The main goal here is to obtain a mesh with good quality, i.e., having almost flat and square faces, and most vertices with regular valence four. Quality of approximation and adaptiveness are usually addressed only indirectly.

In this paper we present a novel approach to the problem of quad mesh simplification, striving to use practical local operations, while maintaining the same goal to maximize tessellation quality. We aim to progressively generate a mesh made of convex, right-angled, flat, equally sided quads, with a uniform distribution of vertices (or, depending on the application, a controlled/adaptive sample density) having regular valency wherever appropriate. The orthogonal objective of maximizing shape similarity between input and output surfaces can be achieved indirectly by enforcing line features and adaptive sampling.

The presented novel approach to quad mesh simplification is incremental, greedy, and based on local operations only. It includes a novel set of local operators preserving the quad structure, prioritized by a simple yet effective criteria, and interleaved with vertex smoothing in tangent space. We show that this approach is effective to solve the otherwise difficult problem of producing quad meshes with a good quality.

The system lends itself well to efficient implementation, and it is easily extended to reconstruct feature lines, or to progressively produce variable tessellation densities.

As a minor contribution we offer a short analysis of co-

herence preservation for local operations in quad meshes. We also discuss in which context configurations traditionally regarded as degenerate (doublets) are useful and can be kept.

The approach is completed with an original Triangle-to-Quad conversion algorithm that behaves well in terms of tessellation quality and, given a closed mesh with n triangles, always generates $n/2$ quads.

2. Comparison with previous work

Triangle mesh simplification. Simplification of triangle meshes has been studied in depth during the Nineties and can now be considered a mature technology. Good algorithms for simplifying triangle meshes are available in common modeling packages like Maya, Blender or MeshLab [CCR08].

Most triangle mesh simplification algorithms focus on adaptive meshing, with the primary goal of obtaining a good approximation of the original shape with a small number of triangles [CMS97,LRC*02]. Many such algorithms work by means of local modifications, which are iterated until the required LOD is obtained. This approach lends itself naturally to the construction of Continuous LOD (CLOD) models. Local operators can also be useful in a variety of contexts (e.g., mesh editing). Hoppe et al. [HDD*93] introduced the use of additional local operators, such as edge flips, which improve the quality of tessellation rather then reducing mesh complexity. They also introduced the idea to drive the choice of local operations aimed at minimizing of an objective function. We reformulate these ideas for the case of quad meshes.

Quad mesh simplification. Simplification algorithms targeting quad meshes have been developed only recently, and they pose extra difficulties. Collapse of a quad diagonal (a.k.a. quad-close [Kin97], quad-collapse [DSSC08], quad-vertex merge [DSC09]) is recognized as a valid local operation that preserves quad structure, but a simplification algorithm cannot be based just on it. The standard approach is to use also operations affecting larger areas, so that the quad structure and the overall quality of the mesh are preserved.

Following this direction, the poly-chord collapse has been adopted in [DSSC08], adapting it from the ring collapse introduced in [BBS02]. In a poly-chord collapse, an entire line of side-to-side quads is removed, so that quad regularity is maintained. Poly-chords are alternated to diagonal collapses, striving to maximize the number of regular vertices. Global operations are inherently unpractical: not only they make the change in resolution less continuous, but also their all-or-nothing nature makes them a clumsy tool to maximize any sought objective: an operation with a large footprint (like poly-chord collapse) is likely to have opposite effects on quality in different subparts of the affected area, and still it must be either performed or discarded as a whole. Moreover, the lack of locality makes it difficult, for example, to selectively decimate only an area of the mesh, leaving the rest unaffected, or to be used in a quad-only region of a mixed mesh. Finally, local operations lends themselves better to out-of-core adaptations of the simplification algorithm, being possible to be performed in a small region even if the mesh is constrained elsewhere.

To alleviate the problem linked to global operations, in [SDW*09], rings to be collapsed are "steered" to constrain the affected area inside a user defined subregion. In [DSC09] poly-chord collapses are split into smaller independent substeps, resulting in the first local-only framework for quad meshes. Our scheme is also local-only, but it uses finer grade operations, see Sec. 3.2. Exclusive use of local operations tends to produce lower quality meshes, though. To improve them, tangent space smoothing is applied to the final result in [DSC09]. This however has no effect on connectivity. In our proposal, tangent space smoothing is interleaved to local operations at each iteration, and it helps selecting the next operation to be performed.

Local operations have been proposed for improving the quality of 2D quad meshes in [Kin97] and they have been used also to produce quad meshes from 2D triangle meshes in [OSCS99]. However, the problem of optimizing meshes in 3D is quite different from the 2D case.

In the context of an unrelated application, the problem of obtaining a triangle mesh with edges of constant length, similarly to what we propose, was addressed in [IGG01].

Quad-remeshing. A related yet different topic is remeshing. The aim of remeshing is to obtain a completely new mesh, not necessarily at lower complexity, which represents the input shape well and has a superior quality. Again, the focus here is on quality of the mesh, but remeshing is inherently not progressive: the output is built from scratch, using the input mesh just as the reference shape.

A few algorithms for remeshing of quad meshes have been proposed in the literature. Methods proposed in [ACSD*03,LKH08] use alignment to principal curvatures to drive remeshing, while those proposed in [DBG*06,DKG05, HZM*08, TACSD06] resort to mesh parametrization and uniform sampling in parameter space. Either strategy imposes to solve difficult problems. The methods proposed in [BZK09, RLL*06] belong to both groups, because they use the principal curvatures within a parametrization approach.

Since the objectives of the two tasks (simplification and remeshing) are similar, it is worth to underline when one should be preferred to the other. Most considerations that makes, in some context, local-operation based simplification more attractive than global-operation based one, discussed above, are stronger when applied to remeshing, which is inherently a global operation performed on the entire mesh (for example, it does not lend itself either to the construction of CLOD models, or to local editing). Another issue

relates to robustness: in general, remeshing requires solving more complex sub-problems compared to mesh simplification (like parametrization, or identification of principal curvature direction) which are difficult to tackle robustly; remeshing is often less robust to noise, or requires clear, well distanced cone singularities. Also it is hard to produce extremely low resolutions meshes (which can also serve as base domains or control meshes). On the other hand, remeshing can benefit from a global, analyze-then-process, top-down approach, and thus produces output meshes with superior quality. For example, it is possible identify cone singularities [BZK09] and to explicitly enforce a correspondence between vertex valencies of output mesh and gaussian curvature of original mesh [LKH08], which is ideal with man-made objects and CAD models. Notwithstanding that, the proposed local, greedy simplification approach performs almost comparably with [BZK09] even in these cases (Tab. 1).

3. Overview of the method

Our proposal is to approach the problem of maximizing the quality of a quad mesh during simplification from a new, straightforward perspective. Consider an ideal two-manifold quad mesh composed entirely of flat, equally sided, regular squares. The surface of a regular poly-cube constitutes an example of surface that allows for this ideal tessellation (see Fig. 1). Note that this ideal condition can be enforced just by measuring lengths, i.e.: all edges of the mesh have the exact same length l, and all diagonals of faces have exactly length $l\sqrt{2}$. We call this condition on edges and diagonals *homeometry*.

Homeometry indirectly implies that vertex valencies depend on local surface shape: vertices in regions of zero Gaussian curvature have valence 4; vertices in regions of (high) positive Gaussian curvature have valence < 4; and vertices in regions of (high) negative Gaussian curvature (i.e., saddles) have valence > 4. This relation between valence and Gaussian curvature may be brought to an extreme by considering as profitable to have valence 2 in regions of extremely high positive curvature (Sec. 3.3). Also, a homeometric quad mesh is optimal in the sense that all the angles are right, all the faces flat, and the distribution of vertices is uniform.

Clearly, it is hardly possible that a general surface allows for a fully homeometric quad tessellation. However, homeometry gives us an easier objective to pursue, i.e., one that is only length based, works at all scales, and substitutes well, in practice, more complex criteria like the ones involving Gaussian curvature or vertex valencies.

We measure how far a given mesh M is from being homeometric by means of the variance of lengths of edges and diagonals:

$$\sum_{e \in M^E} (|e| - \mu)^2 + \sum_{d \in M^D} (|d| - \sqrt{2}\mu)^2 \qquad (1)$$

where e and d span over the sets of edges M^E and diagonals M^D of M, respectively, and μ represents the ideal edge length, computed as the side of an ideal square quad of M:

$$\mu = \sqrt{Area(M)/|M|}, \qquad (2)$$

where $|M|$ denotes the number of faces of M.

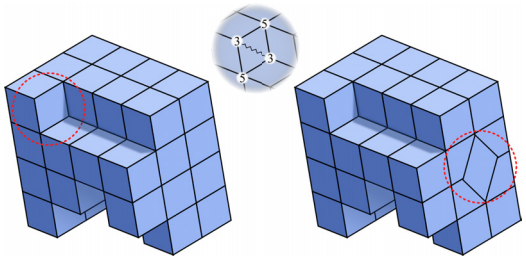


Figure 1: *Left: an example of surface allowing for a fully homeometric quad meshing (a polycube surface). The diag-collapse in the inset affects 4 irregular vertices and would make then all regular. However (right) such diag-collapse is beneficial in this case only.*

Our simplification method modifies the input mesh to reduce its complexity, while trying to minimize the objective function (1). Homeometry-driven simplification blends naturally the need for regular vertices (or, rather, valence matching curvature) with other desiderata, such as uniform vertex spacing. In fact, we believe this approach to be superior than trying to impose regular valence at all vertices, as, e.g., in [DSSC08, DSC09]. As a clarifying, intuitive example, consider the situation in Fig. 1, left. A criterion trying to maximize regular vertices would see as beneficial the collapse of the marked quad diagonal. On the other hand, the initial situation containing irregular vertices is obviously optimal, due to the local discrete curvature, and such a collapse should be considered harmful. Note that this situation can happen at any scale and with unbounded frequency. On the right side of Fig. 1, the same connectivity configuration is present on an underlying geometry with zero Gaussian curvature, where valence four is always appropriate: here, the same diagonal collapse would be beneficial. This collapse would be favored in our method because that diagonal is shorter than the prescribed one, against the homeometry criterion, and not by identifying the curvature (which is a relatively complex task involving computation of discrete curvature at varying scale during the simplification process).

3.1. Conceptual algorithm

An input mesh M_0 is progressively coarsened by a sequence of either complexity-reducing or local optimizing operations, thus producing a sequence of meshes M_i until a user defined criterion is met (e.g. on the number of quads). One strength of our approach is the use of local operations only, which preserve the quad structure at all steps.

The proposed method is based on three mutually interacting ingredients: a novel set of local operations for quad-only

meshes, which constitute the atomic steps of the framework; a heuristic, practical criterion to select the most promising (or less threatening) operation, which tends to maximize homeometry; a tangent smoothing operator, which displaces vertices over the surface of the mesh, without leaving it. The method can be summarized as follows:

0. [Convert input triangle mesh into quad mesh M_0] (Sec. 6)
1. Initial global smoothing of mesh M_0 (Sec. 3.4)
2. Iteratively process mesh M_i to produce mesh M_{i+1} until user-defined criterion is met. In each loop:
   a. for a fixed number of times:
      i. perform any profitable local optimizing-operation, until none is available (Secs. 3.2.1 and 3.3)
      ii. select and perform a local coarsening-operation (Secs. 3.2.2 and 3.3)
   b. local smoothing (Sec. 3.4)
3. Final global smoothing of mesh M_n (Sec. 3.4)

Step 0 is applied only in case the input comes in the form of a triangle mesh. Only collapse operations applied during Step ii simplify the mesh, while the other operations are aimed on one hand at improving mesh quality in terms of both connectivity (Step i) and sample distribution (Step b), and on the other hand, at driving the selection of best coarsening operations to be performed next. In particular, during the final Step 3 the main purpose of smoothing is to improve the quality of the mesh, while elsewhere (in Step 1 and Step 2) it has a more crucial role: by modifying lengths of linear elements over the mesh, it effectively drives the selection of local operations to be performed at the next cycle. This is the key why very good results are obtained, even if operations in Steps i and ii are very local and the criterion to select them is very simple.

This schema lends itself well to an efficient implementation. The resulting method is fully automatic and it depends only on a small number of parameters that are used mainly to control the tradeoff between accuracy and speed and do not need to be adjusted depending on specific input.

3.2. Local Operations

We define three kinds of local operations (see Fig. 2): *coarsening operations,* which reduce complexity; *optimizing operations,* which change local connectivity without affecting the number of elements; and *cleaning operations,* which resolve local configurations considered degenerate.

3.2.1. Optimizing operations (or rotations)

Edge rotate: consider a non-border edge, shared by two quads, and dissolve it, leaving a hexagonal face. There are two other ways to split that face into a pair of quads. We substitute the deleted edge with either one of the two possibilities, calling the two alternatives a clockwise and a counterclockwise edge rotation, respectively. Thus, for each non-border edge in the current mesh, there are two potential edge-rotate operations.

Vertex rotate: consider a non-border vertex v. Each of the k quads sharing v can be split in 2 triangles along the diagonal emanating from v. The $2k$ triangles can be merged next along the former quad edges. Diagonals used to split the quads thus become the new edges. We call this operation a rotation because it can be seen as a rotation of edges around the vertex, in either direction (like sails in a windmill). For each non-border vertex of the current mesh, there is one potential vertex-rotate operation.

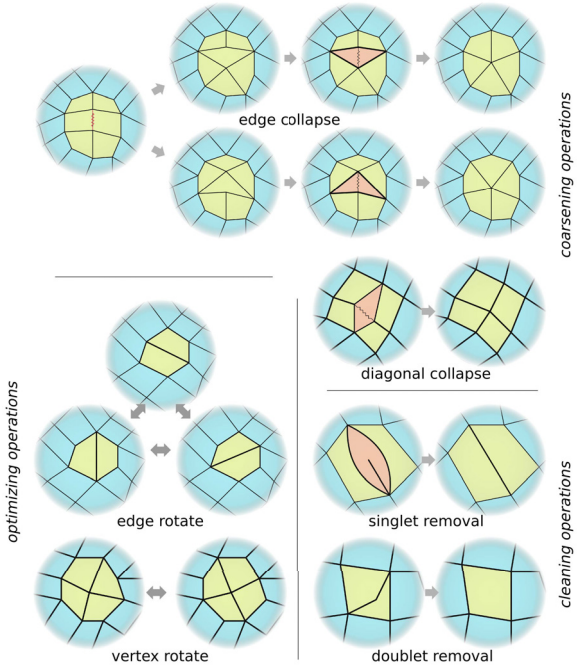


Figure 2: *The set of local operations.*

3.2.2. Coarsening operators (or collapses)

Diagonal collapse: a quad q can be collapsed on either diagonal, removing q from the mesh, and merging the two vertices at the end of the collapsing diagonal into one new vertex. The structure of the quad mesh is preserved, and its complexity is reduced by one quad, two edges and one vertex. For each quad in the current mesh, there are two potential diagonal collapse operations, one along each diagonal. This is the most widely used operation for quad-meshes. The position of the new vertex resulting from collapse is set so that the objective function (1) is minimized in its star.

Edge collapse: given a non-border edge e, we can perform a vertex rotation around either endpoint, turning e into a quad

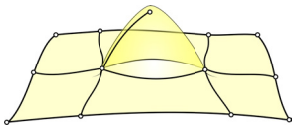


Figure 3: *The "singlet" degenerate configuration. Edges are shown as curved lines for illustration purposes.*

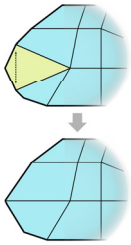
diagonal, and then collapse it. There are two alternatives, corresponding to which endpoint is rotated, producing two distinct potential edge-collapse operations for e (non border edges connecting two border vertices cannot be collapsed).

3.2.3. Cleaning operations (or removals)

Doublet removal: a doublet is a well-known configuration where two adjacent quads share two consecutive edges. The situation can also be described, and it is best detected, as having a valency 2 non-border vertex. A doublet can be eliminated by dissolving the two shared edges, removing the vertex in the middle, merging the two quads into a single one.

Singlet removal: a singlet is a degenerate configuration where a quad is folded such that two consecutive edges become coincident (see Fig. 3). Singlets arise, for example, if two quads initially share three edges and then either one of the two consequential doublets is removed. A singlet is healed by removing the degenerate quad and substituting it with an edge. The valence 1 vertex is also removed, but everything else is kept unmodified.

3.2.4. Discussion on the operation set



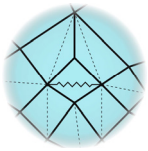
At first, it may seem that border edges cannot be subject to any decimation operation. In fact, a border edge cannot be removed through edge-collapse, otherwise a single triangle would be produced and there would be no local operation to bring a pure quad configuration back. A mesh decimation process that keeps all the original border edges untouched would be clearly unusable. This problem is not a real one, because eventually quads will be generated having two consecutive edges on the border. Collapsing the corresponding diagonal of one such quad removes both border edges and the dangling vertex as a side effect.

Redundancy: It is easy to check that vertex-rotations and edge-rotations are independent operations, meaning that neither one can be replaced by a sequence of the other. Moreover an edge-collapse could be seen as a combination of a vertex-rotation and a diag-collapse. However, it is convenient to consider it as an atomic operation, because it is often the case that its effect on mesh quality (homeometry) is very different from the effect of the first sub-operation alone.

Note that the edge operation described as a "qeMerge" (quad-edge merge) in [DSC09] can be considered as the combination of two vertex rotations and four diagonal collapses. Using the latter represents a finer granularity (in this case, we can think of no apparent advantage in considering the sequence of all six operations to be atomic).

Doublet-removal can be seen as a special case of diagonal collapse, the only difference being how they affect geometry: in doublet removal the position of the new vertex is set as one of the two extremes of the collapsing diagonal.

Consistency: All the above operations preserve topology. There are only two potential inconsistencies arising from their application. Any operation creating a quad edge connecting a vertex to itself must be prevented. The only other problem is that of singlets (and possibly doublets, if they are to be considered degenerate: see discussion later). Detecting and removing them right after creation suffices to ensure consistency.



This approach to consistency preservation is an advancement over the practice to reduce the problem to the triangular case, i.e., splitting quads into triangles and then checking the consistency of the resulting triangle mesh, using [DEGN98], as for example in [DSSC08]. By explicitly considering the problem in terms of quads, one allows for legal operations that would be barred by using triangle mesh criteria. For instance, the potential diagonal collapse in the inset would be forbidden since it produces degenerate configurations in the triangle mesh including dotted edges, while it is legal in the quad mesh.

3.3. Prioritizing operations

Consider a typical closed mesh with n quads and, thus, with about n vertices and $2n$ edges. There is a total of $11n$ potential operations ($2n$ diagonal-collapses, $4n$ edge-collapses, and $4n$ edge-rotations, and n vertex-rotations), plus doublet and singlet removals that can be performed on such a mesh. Clearly, many operations would invalidate other potential operations and create the preconditions for other operations yet. For practical purposes, it is important that the choice of which operation to perform at every iteration is taken very efficiently.

We have seen how we reduced the problem of mesh quality in terms of homeometry. However, equation (1) is still a complex objective function, with multiple local minima, awkward to minimize explicitly (using the above or any other set of discrete operations). Finding the global optimum solution for a target number of quads is not practical. Instead, length based heuristics can be adopted that reach a good solution in a much shorter time. Good performance of this approach has been empirically demonstrated, measuring the objective function (1) (Sec. 7).

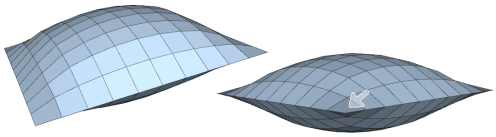


Figure 4: *The doublet in each corner of the "pillow" dataset (two views shown) can be considered a good meshing solution, because of local extreme Gaussian curvature.*

Prioritizing collapses: The proposed solution is to collapse the shortest element of the mesh. Since a collapse typically causes neighbor elements to expand, systematical removal of the shortest element (either edge, or diagonal) causes the population of survivors to have a similar length. We expect diagonals to be $\sqrt{2}$ times as long as the edges, so we divide their measured length by $\sqrt{2}$ for the purpose of identifying the shortest element.

This evaluation process is simple yet effective, depending only on lengths (which change locally), not on the value of μ (which changes globally) in (1); this allows for a practical and efficient implementation (Sec. 5).

Prioritizing optimizations: Contrary to collapses, rotations do not reduce complexity and are performed just to improve tessellation quality. Their role in our framework is similar to the one edge flips play in triangle mesh optimization [HDD*93]. It can be seen as dual to that of collapses: rotations shorten linear elements that are too long, whereas collapses remove elements that are too short, both contributing to achieve length uniformity.

Each potential rotation is assessed by its *profitability*, a value which is, in first approximation, correlated with the related change of function (1). We perform only rotations with a positive effect, starting from the most profitable. This criterion is stated only in terms of length changes, and again it does not depend on the value of μ.

A vertex rotation around v turns edges emanating from v into diagonals, and viceversa. We consider a vertex rotation to be profitable if, in the current star of v, the sum of the edge lengths overcomes the sum of the diagonals. The difference between the two quantities is the corresponding amount of profitability.

The purpose of edge rotations is to eliminate overlong elements (while short ones are collapsed). An edge rotation affects only one edge and two diagonals, each in a different quad (the other diagonal is unaffected). We consider an edge rotation to be profitable if it shortens the rotated edge and both such diagonals. Profitability is the amount of shortening.

Prioritizing cleaning operations: Cleaning operations are not scheduled, but they are performed during both steps i and ii of the simplification algorithm as soon as any degenerate configuration is detected.

In the literature, doublets are considered degenerate configurations. In fact, in a fully homeometric mesh, a doublet necessarily corresponds to a pair of geometrically coincident faces with opposite orientations. However, a doublet can represent an optimal configuration in regions with extremely high positive Gaussian curvature, as depicted in Fig 4.

If application dependent considerations dictate that doublets are to be considered degenerate, then they are removed. This is also the best route in case the original mesh does not present regions with extreme positive Gaussian curvature. However, we have also the alternative of keeping "good" doublets, by inhibiting this cleaning operation. "Bad" doublets are neither detected explicitly (e.g., by measuring curvature), nor treated as a special cases: they are just removed with a diagonal collapse when their geometric shape makes that collapse to become the next operation.

Singlets instead are always degenerated configurations and, as such, they are removed as soon as they appear.

3.4. Tangent space smoothing

This operation consists in moving vertices so that they never leave the surface of the mesh and, at the same time, the overall homeometry (1) is increased. For a better match between the simplified model and the original mesh M_0, vertices are kept on M_0, rather than on current mesh M_i. Factor μ of equation (1) is computed for the current mesh M_i, to account for the minor area deformations occurring during coarsening.

Smoothing is performed through a relaxation process and it has two main purposes: first, by maximizing homeometry, it directly improves mesh quality; second, and more importantly, it helps selecting the best candidate operation to perform next. The rationale is that the elements that cannot be made homeometric by smoothing are good candidates for the next collapse/rotate operation. For example, when the number of quads incident at a vertex is too high with respect to what is required by the local Gaussian curvature, then, *even after smoothing*, one diagonal of each such quad will be shorter than the prescribed one ($\sqrt{2}\mu$). As such, that quad may be selected for collapse.

Depending on the initial data, mesh M_0 can be very far from being homeometric. Thus, global tangent smoothing is applied to mesh M_0 during Step 1 of the algorithm until convergence. Global tangent smoothing is also applied to improve the final mesh M_n during step 3, similarly to [DSC09, DSSC08, SDW*09].

Conversely, smoothing operations performed during Step b are localized to a small area, just around the regions of influence of the local operations preceding it. Vertices affected by local operations during Steps i and ii are initially scheduled for smoothing during Step b. In case any such vertex is

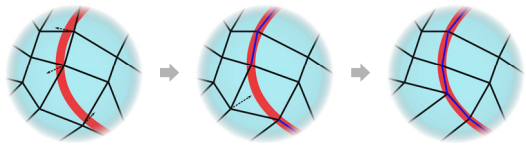


Figure 5: *Preservation of a feature line (red line). Vertices are moved on the feature line (left, middle). This causes some quads to fall across the line: moving a third vertex on the line solves the issue (right).*

moved during smoothing for an amount larger than a given threshold, then also its neighbors are scheduled for smoothing.

4. Extending the method

The method described in the previous section lends naturally to two useful extensions: creating meshes with varying, customizable tessellation densities (Sec. 4.1); and creating meshes that preserve feature lines (Sec. 4.2).

4.1. Customizable tessellation at variable density

Since both the smoothing phase and the selection phase are length based, it is easy to make the system produce tessellation densities that vary over the surface, according to a given importance function taken in input. This function $\lambda(p)$ is defined over the surface M_0 and determines the required tessellation density around each point p of the surface.

Since the average edge length is already implicitly defined by the number of elements $|M_i|$ and the total area of M_i, we make $\lambda(p)$ define the prescribed *ratio* between the edge lengths around p over average edge length (and similarly for the diagonals). This function is normalized so that the area-weighted average of $1/\lambda^2$ over all M_0 is 1.

During the smoothing phase, the objective function (1) is minimized using $\mu\lambda(p)$ instead of μ, p being the center of the given element. During the selection phase, length of edges and diagonals are divided by $\lambda(p)$ for the purpose of identifying the next element to be collapsed.

Depending on the application, importance functions can be either defined by the user (for example, in order to devote more vertices to regions of interest of a model), or pre-computed by geometry processing performed over M_0 and/or its attributes, so to achieve an adaptive simplification. Since the resulting mesh has no T-junctions, spatial changes in the resolution cannot be sudden, so λ is supposed to vary smoothly over the mesh.

4.2. Preservation of feature lines

In many contexts it can be useful to preserve feature lines in the quad mesh. Feature lines can be either prescribed by the user, or automatically identified, e.g., as: attribute discontinuity lines; creases extracted from the original geometry in CAD models; or high curvature lines extracted by analyzing the geometry of range scanned models. In open meshes, borders should also be preserved.

Our objective is to make feature lines emerge as collections of edges in most levels of detail produced by the simplification process. This is achieved by acting just in the tangent smoothing phase, adding a term in the cost function, which penalizes vertices that are close but not over such lines. In other words, feature lines are made to attract nearby vertices toward them. The radius R at which this happens is computed as a fixed fraction k of the average edge length in that zone. In our experiments we used $k = 3/4$.

For each vertex v_i we find the closest point f_i on a feature line with the help of a spatial index, and we compute an attraction factor a_i, which is equal to 1 if v_i is on f_i and decreases linearly to vanish at distance R.

Uncontrolled snapping of vertices to nearby features may cause some quads to fall across them, i.e., to have their diagonal aligned with the feature line. This happens when two opposite vertices v_a and v_c of the diagonal of a quad (v_a, v_b, v_c, v_d) are attracted to the same line, while vertices v_b and v_d are not. This situation is healed by making one of v_c and v_d also move toward the same line, regardless of their distance from it (see Fig. 5).

Use of a scaling factor $k < 1$ for the radius of influence R ensures that both endpoints of an edge are attracted to a feature line only if such an edge is sufficiently close to the feature itself. In this way, we try to avoid compressing quads around features, which would be against our objective of homeometry. Even when such undesirable situation occurs, compressed quads soon disappear because their elements (either diagonals or edges) are selected for collapse.

Figure 6 shows a how this method performs on a simple example containing a circular feature: even in that challenging case (for quad meshing) both the feature and homeometry are maintained to some degree during the simplification.

An advantage of this approach is that feature lines are enforced, rather than just preserved, meaning that they need not be present as edges in the input mesh. For example, CAD models often contain explicit sharp crease lines that are prescribed during design, while scanned models rarely contain sharp creases.

5. Implementation details

Prioritizing operations. Finding the top priority operation can hinder performance in a naïve implementation, because of the linear search it involves. As in standard iterative simplification approaches [LRC*02], a good solution is to build a heap of potential operations sorted by prerecorded priorities. Every time a local operation is performed, only the

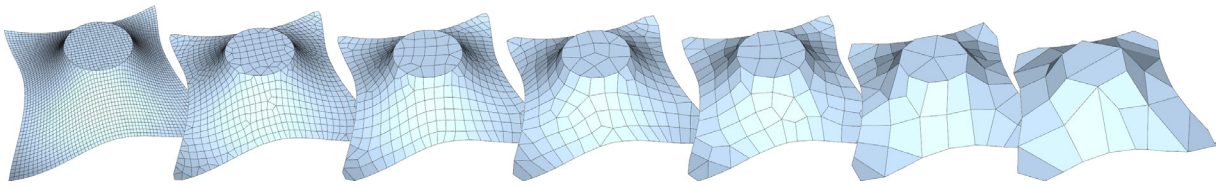


Figure 6: *Feature lines preserved though the simplification process (TableCloth dataset, simplified from 3.25K to 40 quads).*

potential operations related to neighboring elements are affected and must be updated in the heap.

This is possible because the priority criterion does not depend on the average edge length μ as defined in Eq. (2), i.e., priorities do not change when the global number of faces changes (otherwise, the entire heap would be invalidated at each simplification step). This is one reason to use a heuristic criterion (Sec. 3.3), rather than trying to explicitly find the operation that minimizes (1).

The situation is different for tangent-space smoothing. Global smoothing would change the geometry of the entire mesh and, thus, it would invalidate the precomputed priority of all potential operations. Instead, we resort the following incremental scheme.

Tangent space smoothing. There are several possible ways to perform smoothing. One would be to minimize objective function (1) iteratively in 3D space and, after each iteration, re-project every vertex to the surface of M_0 via some spatial indexing structure [THM*03]. Otherwise, if an almost isometric parametrization of M_0 is available, smoothing can be solved in parametric space. In this case, gradient descending vectors are computed in 3D, then converted to 2D vectors via the mapping defined by parametrization. In our experiments, we adopted the latter approach, making use of a parametrization produced by the method described in [PTC09], which provides a reasonably isometric, seamless, globally smooth parametrization of M_0. Any other technique for smoothing in tangent space could be used instead.

We perform tangent space smoothing with an iterative, explicit solver that lets the mesh relax as in a mass and spring system [MHHR07], where the rest position of each spring coincides with the ideal length of its associated edge or diagonal, i.e., either μ or $\sqrt{2}\mu$, respectively.

In Step b of the algorithm, smoothing is stopped relatively early, i.e., after a fixed number of iterations (we used 20), because a coarse minimization of function (1) is sufficient in practice to guide the selection of the next operation. Smoothing is performed until convergence in Steps 1 and 3.

Feature line extraction. Any feature line, whether it is present or not in the original mesh as a collection of edges, can be enforced during the simplification process. Thresholding dihedral angles is sufficient to extract sharp creases from CAD models, such as the fandisk or the tableCloth datasets. For range scanned models, such as the Mohai dataset, creases are extracted with a modification of the method proposed in [HPW05] (see Fig. 7).

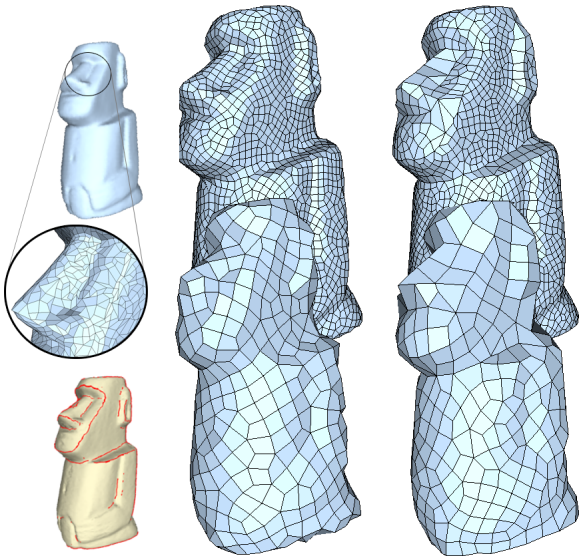


Figure 7: *Left: input of the simplification algorithm (top: input mesh M_0; bottom: feature lines extracted from M_0). Center: models simplified with basic algorithm. Right: models simplified with feature preservation. See also tab. 1 and video.*

6. Tri-to-Quad mesh conversion

Many datasets, e.g., those from range scanning, come originally as triangle meshes. Such datasets must be converted to quad meshes prior to applying our simplification method. Note that any other polygonal meshes can be trivially reduced to triangle meshes first.

Some authors obtain the initial quad mesh by performing one step of Catmull-Clark subdivision [DSSC08, DSC09]. The obtained quad mesh has the desirable property of preserving all original edges, but it has also several drawbacks: in terms of complexity, is has three times more quads than triangles of the initial mesh; in terms of quality, no more than 50% of its vertices are regular.

In [VZ01], a hybrid tri-quad mesh is built first, similarly to what we do in Step 1 below, which is converted next into

a purely quad mesh. Each face of the tri-quad mesh is split into triangles by barycentric subdivision; next, pairs of such triangles are merged to form quads by deleting all edges that existed prior to subdivision. This is better than doing a Catmull-Clark on the input triangle mesh, both in terms of quality and in terms of complexity, but it still produces a non-negligible increase in the total number of quads.

Since our simplification method is able to enforce feature lines even if they are not present as edges in the input mesh, we find it more convenient to develop another method, which does not preserve original edges, but produces a smaller number of faces and a mesh of better quality.

6.1. No complexity increase scheme

Our scheme always produces a quad quad mesh featuring half as many quads as triangles in the starting mesh. It requires, as input, a (connected) mesh with an even number of triangles. If the mesh is closed (and two-manifold), this condition is guaranteed; otherwise, any border edge can be split in half, increasing the number of triangles by one unit.

Step 1: making a quad-dominant mesh. First, most triangles are merged pairwise into quads, dissolving their shared edge. Edges of the triangle mesh can be flagged as dissolved with any heuristics, with the constraint that no triangle is allowed to have more than one flagged edge. The objectives are to maximize the number of flagged edges and to prioritize creation of quads with (nearly) right angles. We adopted a simple, linear time approach. First, for each triangle we select its best candidate edge to be dissolved, scoring each edge by the "squareness" of the corresponding quad, measured as the sum of pairwise dot products of the four normalized edges. Next, any selected edge is flagged only if this does not invalidate the selection of edges with a better score.

Step 2: making a pure quad-mesh. A few triangles (still an even number) remain after Step 1. These triangles are made to "crawl" over the mesh toward each other until they can be merged into quads. Iteratively, a triangle t_i is selected and quads in a region around it are marked, with a breadth first visit, with cross-edge distance from t_i until another triangle t_j is reached. The triangle t_j is moved toward t_i by means of a sequence of edge flip operations: specifically an edge between t_j and the neighboring quad on the path to t_i is dissolved, forming a pentagonal face, which is split back into a quad and a triangle t'_j (there are four other possible ways to do this), thus making the triangle "crawl" over the surface. The split that moves t_j faster toward t_i is selected, breaking ties in favor of the alternative maximizing squaredness.

6.2. Comparison to other conversion strategies

In terms of complexity of the resulting quad mesh, this method improves by a factor of 6 over direct use of Catmull-Clark subdivision. This is a significant gain, considering that quad simplification is still a relatively time consuming process. Quality is also drastically improved, as more regular vertices are produced (see Fig. 8).

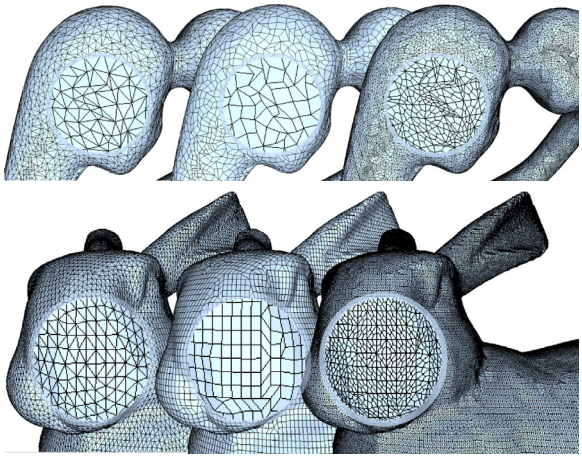


Figure 8: *Comparison of Tri-to-Quad conversion methods. Left: input triangle mesh. Middle: quad mesh built with no complexity increase scheme. Right: quad mesh built with one step of Catmull-Clark subdivision.*

The most striking gain in term of quality occurs when the tri-mesh contains regions tessellated with square isosceles triangles (Fig. 8, bottom). Tri-meshes falling in this category are not uncommon in practice. They occur, for example, with models obtained by Marching Cubes-like algorithms (in presence of flat regions with almost any orientation), with models obtained by zippering together re-triangulated depth scans or height fields, with procedural meshes, with CAD models, etc. For example, at least two among the most common tri-meshes used as benchmarks, Stanford bunny and Fandisk, feature this kind of structure.

7. Results and discussion

We tested our quad simplification method on quad meshes converted from triangle meshes (Sec. 6) coming from either range scanning, or CAD. Examples of results are illustrated in Fig. 6, 7, 9, 10, 11, 12, and 13. A numerical assessment is given in Table 1.

Discussion: In spite of using only local connectivity operators, the simplified models show a very good quality, throughout all steps of simplification. Regular results are obtained independently from regularity of the input. Tangent space smoothing has proven to be very effective not only to improve the final result, but also – and more importantly – to drive the selection of the operator to be performed at the next cycle. This concept makes the process robust and general, so quad mesh simplification can be easily addressed in spite of the intrinsically harder challenges it poses with

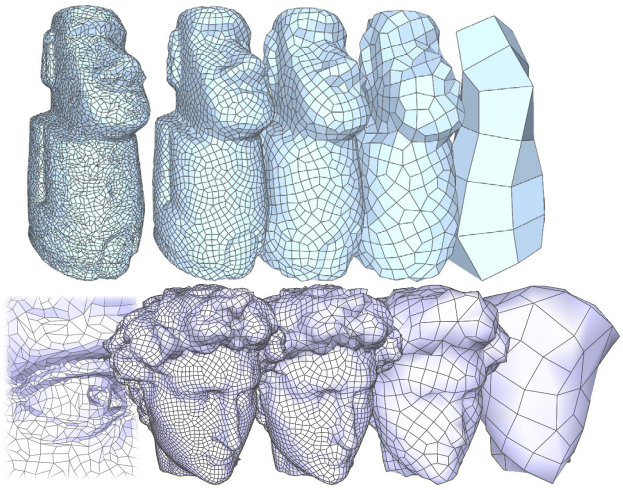


Figure 9: *Examples of models simplified at decreasing number of quads (including extremely low ones, useful for base mesh or as a base for a refinement). Above, Muai dataset: 8.2K (original), 3.3K, 1.4K, 600, and 40 quads. Below, David dataset: 100K (original, detail), 10K, 5K, 1.2K and 150 quads.*

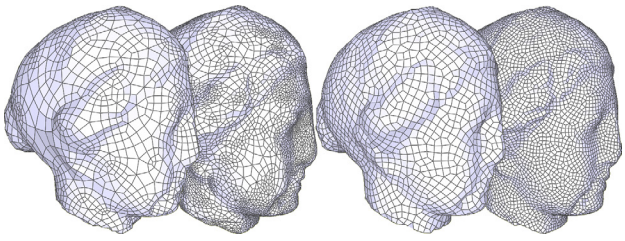


Figure 10: *Simplified mesh pairs (5K and 3K) obtained from Igea dataset with [DSC09] (left) and our approach (right). For a quantitative assessment, refer to table 1.*

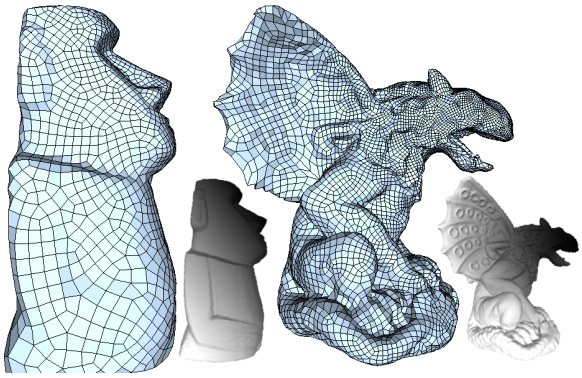


Figure 11: *Examples of models simplified following a user defined importance map (shown in small: darker regions correspond to higher density).*

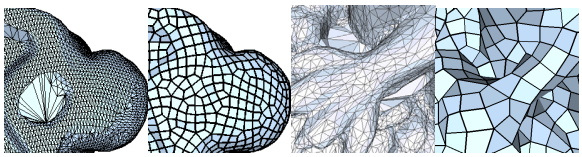


Figure 12: *Examples showing robustness to unevenly sized or poorly shaped initial triangles. Left: original. Right: output (close-ups of Bunny and Gargoyle datasets).*

respect to the case of triangle meshes. Our length based approach is also efficient, scalable, robust with the initial tessellation quality (see Fig. 12) and easily extended to allow for feature line preservation and varying controlled tessellation densities. The system is capable of reducing an initial quad mesh to extremely low number of faces (see Fig. 9, right). Semi-regular meshes can be obtained from such a simplified model, by a regular subdivision and reprojection onto the original surface. Examples of results obtained this way are shown in Fig. 13 (top-right).

We compared our approach, which strives to maximize homeometry, with [DSC09], which strives to maximize regularity (and, to a lower extent, geometrical faithfulness). Models maximizing the sought objective are always obtained (Table 1). In terms of Hausdorff distance, however, the presented strategy is around twofold better, suggesting that our objective is more suited for that. This is probably also due to the better sampling distributions that is implied by homeometry. The same approach presented here can probably be applied to triangle mesh simplification in contexts where quality of meshing is a concern as much as geometric similarity. Our method naturally lends itself to the construction of progressive and CLOD models; however, further investigation is needed to efficiently incorporate the changes occurring through simplification in the data structure.

Our current method has some limitations. Geometric fidelity is not measured and it is addressed only indirectly; the general objective to place extraordinary vertices at points that concentrate strong Gaussian curvature is achieved, but only partially; vertex snapping, used to enforce features, can produce nearly triangular elements that would be harmful for, e.g., numerical analysis; alignment of edges to curvature directions has not been addressed. These issues might be addressed by suitable modifications of the objective function in the context of the same framework.

Acknowledgements. The research leading to these results has received funding from the EG 7FP IP "3D-COFORM" project (2008-2012, n. 231809) and from the Regione Toscana initiative "START". Authors wish to thank Claudio T. Silva for kindly sharing datasets.

Project page: `http://vcg.isti.cnr.it/quadSemplif`

	$\|M\|$	val max	reg (%)	Homeometry min	Homeometry max	Homeometry var	Dist 10^{-3}
ideal values:		—	—	1	1	0	0
Moai	*8.2K*	9	44	0.18	3.8	0.49	0
(11sec)	3.3K	6	64	0.60	1.82	0.16	0.5
	0.6K	6	62	0.70	1.73	0.17	1.7
Pensatore	*15K*	8	48	0.12	3.4	0.44	0
(28sec)	10K	7	61	0.58	2.01	0.19	0.3
	5K	7	57	0.68	2.17	0.19	0.7
	2K	6	68	0.59	1.76	0.18	1.4
	1K	7	64	0.6	2.01	0.19	2.0
Gargoyle	*25K*	8	46	0.15	12.22	0.53	0
(63sec)	11K	7	61	0.2	2.8	0.17	0.5
	4K	7	57	0.5	2.28	0.19	1.2
	2K	7	54	0.58	2.31	0.20	2.0
Igea	*25K*	8	48	0.12	4.09	0.48	0
(66sec)	12K	7	65	0.64	2.05	0.16	0.24
	3K	6	67	0.69	2.11	0.16	0.7
	3K [DSC09]	6	80	0.22	3.74	0.41	1.7
Bunny	*35K*	12	88	0.1	14.53	0.34	0
(85sec)	11K	7	69	0.66	3.3	0.19	0.3
	5K	7	68	0.65	2.8	0.18	0.5
	5K [DSC09]	6	93	0.24	4.87	0.41	1.1
	3K	6	61	0.47	2.45	0.17	0.7
Fertility	*40K*	8	47	0.09	9.56	0.48	0
(115sec)	22K	6	63	0.59	3.95	0.18	0.12
	5K	6	67	0.65	2.94	0.17	0.4
	5K [DSC09]	7	65	0.23	6.4	0.48	0.8
	2K	6	67	0.61	2.22	0.19	1
	2K [DSC09]	7	81	0.23	4.20	0.46	2.5
	3.3K	7	71	0.60	1.81	0.15	0.56
	3.3K [BZK09]	5	98	0.59	3.20	0.22	0.61
Rampart	*50K*	9	48	0.08	7.58	0.50	0
(174sec)	20K	7	75	0.29	3.19	0.21	0.2
	10K	7	62	0.34	2.66	0.21	0.35

Table 1: *Simplification results on various datasets. The computation times required for simplifying the initial dataset into the coarsest model (on a Intel Core2 2.4Ghz 2.00 GB). For each quad-mesh (input and simplified), we report: vertex valency (max and % of regular vertices); homeometry (min, max and variance of edge or diagonal length, all normalized with ideal length μ); and Hausdorff distance (computed with [CCR08]), with respect to bounding box diagonal. When possible, results from [DSC09] and [BZK09] are reported too (the latter is a quad-remeshing approach).*

References

[ACSD*03] ALLIEZ P., COHEN-STEINER D., DEVILLERS O., LÉVY B., DESBRUN M.: Anisotropic polygonal remeshing. *ACM Trans. Graph. 22*, 3 (July 2003), 485–493.

[BBS02] BORDEN M., BENZLEY S., SHEPHERD J.: Hexahedral sheet extraction. In *Proc. 11th Int. Meshing Roundt.* (2002), pp. 147–152.

[BZK09] BOMMES D., ZIMMER H., KOBBELT L.: Mixed-integer quadrangulation. *ACM Trans. Graph. 28*, 3 (2009), 1–10.

[CCR08] CIGNONI P., CORSINI M., RANZUGLIA G.: Meshlab: an open-source 3d mesh processing system. *ERCIM News (73)* - http://meshlab.sourceforge.net/ (2008), 45–46.

[CMS97] CIGNONI P., MONTANI C., SCOPIGNO R.: A comparison of mesh simplification algorithms. *Computers and Graphics 22* (1997), 37–54.

[DBG*06] DONG S., BREMER P.-T., GARLAND M., PASCUCCI V., HART J.: Spectral surface quadrangulation. *ACM Trans. Graph. 25*, 3 (2006), 1057–1066.

[DEGN98] DEY T., EDELSBRUNNER H., GUHA S., NEKHAYEV D.: Topology preserving edge contraction. *Publ. Inst. Math. (Beograd) (N.S 66* (1998), 23–45.

[DKG05] DONG S., KIRCHER S., GARLAND M.: Harmonic functions for quadrilateral remeshing of arbitrary manifolds. *Comput. Aided Geom. Des. 22*, 5 (2005), 392–423.

[DSC09] DANIELS J., SILVA C., COHEN E.: Localized quadrilateral coarsening. *Comput. Graph. Forum 28*, 5 (2009), 1437–1444.

[DSSC08] DANIELS J., SILVA C., SHEPHERD J., COHEN E.: Quadrilateral mesh simplification. *ACM Trans. Graph. 27*, 5 (2008), 1–9.

[HDD*93] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Mesh optimization. In *Proc. SIGGRAPH '93* (New York, NY, USA, 1993), ACM, pp. 19–26.

[HPW05] HILDEBRANDT K., POLTHIER K., WARDETZKY M.: Smooth feature lines on surface meshes. In *Proc. 3rd Eurographics Symp. on Geom. Proc.* (2005), p. 85.

[HZM*08] HUANG J., ZHANG M., MA J., LIU X., KOBBELT L., BAO H.: Spectral quadrangulation with orientation and alignment control. *ACM Trans. Graph. 27*, 5 (2008), 1–9.

[IGG01] ISENBURG M., GUMHOLD S., GOTSMAN C.: Connectivity shapes. In *VIS '01: Proceedings of the conference on Visualization '01* (Washington, DC, USA, 2001), IEEE Computer Society, pp. 135–142.

[Kin97] KINNEY P.: Cleanup: Improving quadrilateral finite element meshes. In *6th Int. Meshing Roundt.* (1997), pp. 449–461.

[LKH08] LAI Y.-K., KOBBELT L., HU S.-M.: An incremental approach to feature aligned quad dominant remeshing. In *Proc. 2008 ACM Symp. on Sol. and Phys. Mod.* (New York, NY, USA, 2008), ACM, pp. 137–145.

[LRC*02] LÜBKE D., REDDY M., COHEN J., VARSHNEY A., WATSON B., HÜBNER R.: *Level Of Detail for 3D Graphics*. Morgan Kaufmann, 2002.

[MHHR07] MÜLLER M., HEIDELBERGER B., HENNIX M., RATCLIFF J.: Position based dynamics. *J. Vis. Comun. Image Represent. 18*, 2 (2007), 109–118.

[OSCS99] OWEN S., STATEN M., CANANN S., SAIGAL S.: Q-morph: An indirect approach to advancing front quad meshing. *International Journal for Numerical Methods in Engineering 44*, 9 (March 1999), 1317–1340.

[PTC09] PIETRONI N., TARINI M., CIGNONI P.: Almost isometric mesh parameterization through abstract domains. *IEEE Trans. on Vis. and Comp. Graph.* (2009).

[RLL*06] RAY N., LI W.-C., LÉVY B., ALLIEZ P., SHEFFER A.: Periodic global parameterization. *ACM Trans. Graph.* (2006).

[SDW*09] SHEPHERD J., DEWEY M., WOODBURY A., BENZLEY S., STATEN M., OWEN S.: Adaptive mesh coarsening for quadrilateral and hexahedral meshes. *Finite Elements in Analysis and Design 46*, 1-2 (2009), 17 – 32.

[TACSD06] TONG Y., ALLIEZ P., COHEN-STEINER D., DESBRUN M.: Designing quadrangulations with discrete harmonic forms. In *Proc. 4th Eurographics Symp. on Geom. Proc.* (2006), pp. 201–210.

[THM*03] TESCHNER M., HEIDELBERGER B., MÜLLER M., POMERANTES D., GROSS M.: Optimized spatial hashing for collision detection of deformable objects. In *VMV* (2003), Ertl T., (Ed.), Aka GmbH, pp. 47–54.

[VZ01] VELHO L., ZORIN D.: 4-8 subdivision. *Computer-Aided Geometric Design 18* (2001), 397–427.

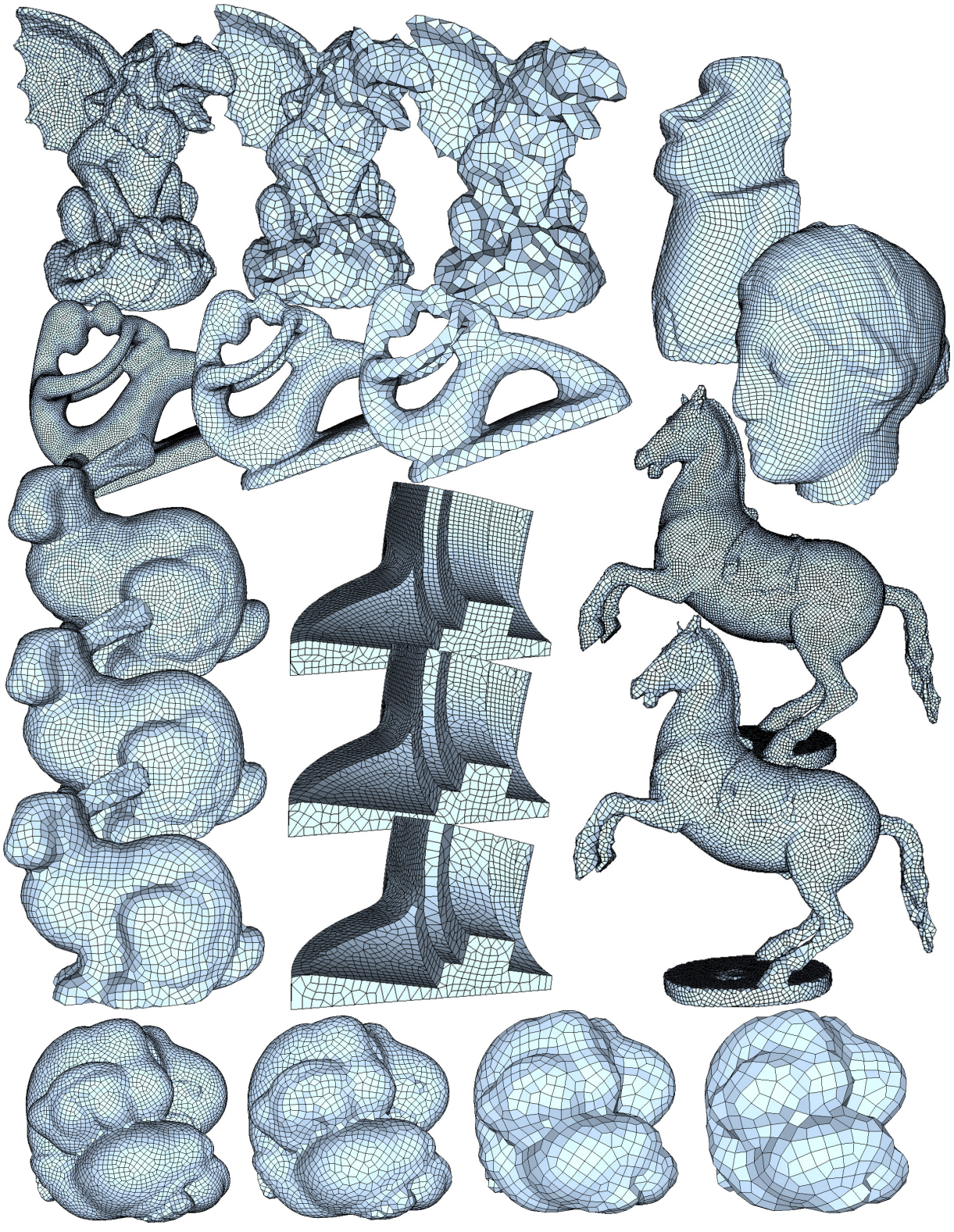


Figure 13: *Different datasets shown at various simplification steps. Refer to table 1 for numerical data. Original models are not shown (see also attached video). Top right, in blue: two semi-regular models obtained by regularly subdividing extremely simplified models (see text).*