Automatic Construction of Quad-Based Subdivision Surfaces using Fitmaps

Daniele Panozzo, Enrico Puppo, Marco Tarini, Nico Pietroni, and Paolo Cignoni

Abstract—We present an automatic method to produce a Catmull-Clark subdivision surface that fits a given input mesh. Its control mesh is coarse and adaptive, and it is obtained by simplifying an initial mesh at high resolution. Simplification occurs progressively via local operators and addresses both quality of surface and faithfulness to the input shape throughout the whole process. The method is robust and performs well on rather complex shapes. Displacement mapping or normal mapping can be applied to approximate the input shape arbitrarily well.

Index Terms—Quadrilateral meshes, Subdivision surfaces, Displacement mapping, Mesh compression

1 Introduction

Subdivision surfaces have become of central interest during the last few years, because of their high potential in shape design and rendering. A major breakthrough in this context comes from methods for fast evaluation of subdivision surfaces [1] that, combined with recent advances in GPUs [2], may support rendering with unprecedented quality and speed [3]. Fine detail can be added via displacement mapping [4]. With these methods, high resolution meshes are produced directly in the graphics card, bypassing the limit of throughput from main memory to GPU. Expectations are that subdivision-based modeling will soon replace polygonal modeling even for real-time applications such as videogames.

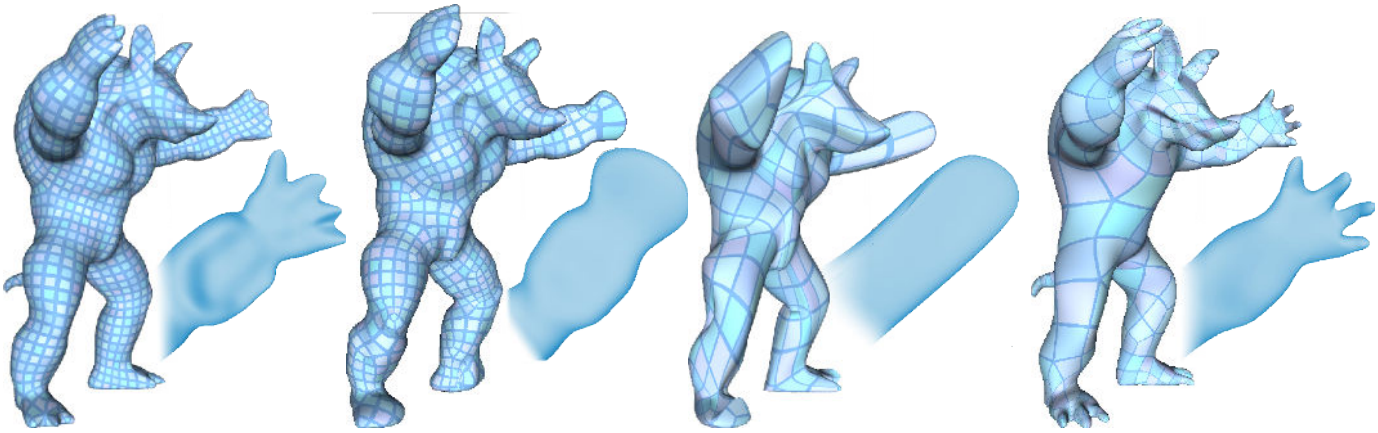
This approach is relatively straightforward as long as shapes are created directly as subdivision surfaces [4]. However, many shapes come as polygonal meshes at high resolution, e.g., built by range-scanning real-world objects, or as iso-value surfaces, or by tessellating implicit surfaces, etc. Range scanning is customary in cultural heritage applications; and during production of animation movies and videogames, plaster mock-ups are created by artists prior to being modeled through CAD.

Hence, the problem of converting an input mesh at high resolution into a subdivision surface with a coarse control mesh, possibly enriched with a displacement map or normal map. This problem involves issues from two other sub-problems, which have been studied independently in the literature: creating a coarse mesh to approximate a given shape; and fitting a surface to a shape, for a given connectivity of its control mesh. Displacement or normal mapping also require that the output surface can be projected to the input shape. In this work we show that tackling the original problem as a whole leads to better results than combining techniques aimed at resolving each of the sub-problems.

- *Daniele Panozzo and Enrico Puppo are with the Department of Computer and Information Sciences, University of Genoa, Genoa, Italy. E-mail: lastname@disi.unige.it*
- *Marco Tarini is with the Department of Computer Science and Communication, University of Insubria, Varese, Italy E-mail: tarini@isti.cnr.it*
- *Nico Pietroni and Paolo Cignoni are with the Visual Computing Lab, ISTI-CNR, Pisa, Italy. E-mail: firstname.lastname@isti.cnr.it*

1.1 Adaptivity vs regularity

The primary application we address here is fast GPU rendering of displaced subdivision surfaces. To this aim, it is important that the control grid of the subdivision surface is made of as few patches as possible, while preserving the overall shape, and that displacement mapping is used just to add fine detail. For natural objects with details at different scales, such as the Armadillo shown in Figure 1, the contrasting objectives of having a good fit and a coarse control mesh can be achieved only if the mesh is adaptive. While techniques for adaptive triangle-based meshing can be considered a consolidated subject, producing an adaptive quad-only mesh for a given shape is still a challenging task: not only the mesh (in fact, its limit surface in our case) should fit the input shape, but also its connectivity, as well as the shape of its patches, should be as regular as possible. Adaptivity and regularity are highly contrasting objectives: transition from coarse to fine patches requires introducing some irregular vertices, or warping the shape of some quads, or both. Several works have been proposed in the literature recently that address the problem of producing a quad mesh that approximates a given input shape well while being made of regular faces, which are possibly aligned either with lineal features, or with a cross field defined on the surface [5], [8], [9], [6]. Such methods achieve very good results in terms of regularity of the mesh (see, e.g., the first two examples on the left in Figure 1), but they are intrinsically not adaptive, thus requiring many more faces to achieve the same quality



Mixed Integer [5] 2184 patches | PGP [6] 1600 patches | VSA [7] 534 patches | OUR approach 476 patches

Fig. 1. Subdivision surfaces approximating the Armadillo (original mesh 345k triangles) obtained from control meshes produced with different methods. Our approach gives a much better approximation by using fewer patches, thanks to adaptivity.

of approximation. This is especially true for objects with intricate geometries that are difficult to represent via displacement mapping (see, e.g., hair curls in the David head in Figures 5, 7): lack of adaptivity leads either to artifacts, or to meshes with many tiny faces even on smooth regular areas. For these reason, in this work we trade some regularity for adaptiveness. Our method combines tools for progressive mesh simplification based on local operations, together with tangential smoothing to keep a regular shape of patches, and subdivision surface fitting to keep the limit surface close to the input shape. Such ingredients are not used independently in separate stages, but they are combined in the context of an integrated framework, in order to take into account all aspects of the problem throughout the whole process.

1.2 Objectives

Given a triangular input mesh M, we strive to build a quad-based control mesh K with a limit surface S_K through the Catmull-Clark (CC) subdivision scheme [10], such that surface S_K approximates surface M, with the following requirements:

(I) *Conciseness*: the number of faces of K is small;
(II) *Regularity*: most vertices have valence four, and patches have nearly right angles;
(III) *Accuracy*: the difference between S_K and M is small;
(IV) *High projectability*: The great majority of points of M can be reached from S_K by projection along the surface normal.

Conciseness relates to rendering efficiency, ease of editing, minimization of memory consumption, etc. *Regularity* relates to the quality of the meshing that can be obtained with a regular sampling of S_K (e.g., performed at rendering times). *Accuracy* relates to the quality of generated geometrical approximation: it implies not only that positions of vertices of K are carefully chosen, but also that K is tessellated adaptively. Finally *projectability* is very important for several techniques in computer graphics: in practice it means that any attribute of M (e.g., per vertex color or normal) can be stored as texture maps associated to faces of K. Moreover it also means that M can be faithfully reproduced from S_K via displacement mapping in the normal direction, which requires storing just a texture of scalar offsets for each patch. Ideally, one wish to have perfect projectability, i.e., the projection along surface normal defining a bijection between S_K and M. However, since the input can be affected by noise and/or contain detail at arbitrarily high frequency, such a strict requirement may prevent building coarse control meshes. For this reason, we allow for some loss of projectability, which can be controlled by the unique parameter used in our method, as a trade-off for improving adaptivity and coarseness of the output mesh.

1.3 Contribution

Our contribution can be summarized as follows:

- We present an integrated approach that incorporates both the automatic construction of a pure-quad, concise and semi-regular control mesh, and the optimization of its related CC subdivision surface, in terms of both projectability and accuracy with respect to the input shape. To the best of our knowledge, this is the first method to produce a pure-quad control mesh while addressing quality of subdivision fit during mesh construction. The only other attempts in this direction have been proposed in [11] for triangle meshes and Loop surfaces and in [12] for T-meshes and T-splines.
- Our method is fully automatic and "one-click": it takes in input a geometric mesh and it builds a

coarse CC subdivision surface through progressive simplification of its control mesh.
- Mesh simplification is driven from an innovative and effective heuristic, which is based on an static analysis of the input made during preprocessing: a pair of scalar fields, called a *Fitmap*, are computed on the input mesh, which roughly estimate for each vertex of the surface, how well the mesh can be locally modeled with patches, in terms of both geometric error and projectability. This allows us to avoid performing cumbersome geometric tests during the simplification process. This approach is fairly general and probably it can be adapted to any other form of parametric surfaces, as well as to the simpler case of polygonal mesh simplification.
- We experiment our method on several meshes, representing objects with various topologies and details at different scales. We compare our results to CC subdivision surfaces obtained by first building a quad mesh with other state-of-the-art methods, and then using such meshes for subdivision surface fitting and displacement mapping. Our results clearly outperform those obtained with the other methods.
- Our control meshes contain about two orders of magnitude less faces than the input meshes, thus our method works also as a shape compressor. In fact, displaced subdivision surfaces can be efficiently encoded as a control mesh plus single channel, highly compressible, displacement textures.

2 Related work

The literature on surface fitting is immense and its review is beyond the scope of this paper. Dozens of algorithms tackle exclusively the task of geometric fitting a subdivision surface starting with a control mesh with known connectivity. See, e.g., literature reviews in [13], [14], [15]. We adopt a rather standard approach for solving this problem [16], which is orthogonal to our contribution. In this review we rather focus on methods that address the problem of automatically building a suitable control mesh.

2.1 Simplification and fitting

The problem of finding a coarse subdivision surface that fits an input mesh has been studied in the past for the case of triangle-based subdivision surfaces. Hoppe et al. [17] first simplify a triangle mesh, then they build a control mesh for Loop subdivision by optimizing vertex positions; further simplification is performed by alternating local operators for mesh simplification and geometric optimization. Later on, Lee et al. [11] combine this approach with displacement mapping. To this aim, they address approximated projectability during simplification.

Similar simplify-then-fit approaches have been proposed in [13], [18], [15]. Beside simplification, some of these methods consider refining the control mesh via local operations, such as edge split and edge swap, to improve portions of surface affected by large error. Conversely, Suzuki et al. [19] adopt a semi-automatic approach based on refinement of a given, manually built, coarse mesh. Kanai [20] modifies the QEM method [21] to perform mesh simplification by taking into account the position of points sampled from the limit surface of a Loop subdivision.

It is not clear how such algorithms could be extended to work on quad-based subdivision schemes. However, following the same simplify-then-fit approach, other known techniques could be adopted to produce a coarse control mesh of quads.

Boier-Martin et al. [22] and Cohen-Steiner et al. [7] propose clustering algorithms that generate coarse and adaptive polygonal meshes. Such algorithms take into account projectability to some extent, as clustering is driven from alignment of face normals. Resulting meshes can be quite coarse, but also irregular, containing faces with concave boundaries and many edges. Such faces can be decomposed further to obtain semi-regular quad meshes, but this process usually increases their number for about one order of magnitude (see also Section 4).

Similarly Marinov and Kobbelt [23] use a face-merge method to compute a coarse polygonal mesh, by also taking into account normal projectability to the input. Again, they diagonalize each polygonal face at the end of the process, to obtain a quad-dominant control mesh for Catmull-Clark subdivision, and they compute a normal displacement map from it. Lavoué and Dupont [14] use VSA [7] to build a polygonal control mesh of hybrid tri-quad subdivision surfaces for mechanical objects.

2.2 Polygonal remeshing and simplification

Strictly related with the above topics there are various simplification and remeshing algorithms designed to target quad (or quad-dominant) meshes.

Remeshing algorithms, such as those proposed in [5], [6], [8], [9], replace an input mesh with a semi-regular quad (or quad-dominant) mesh, which can represent the same shape with arbitrarily good accuracy. Since all these methods are aimed at achieving face uniformity, they are inherently not adaptive, thus it is hard to apply them with the purpose of drastic simplification. For instance, in the Mixed Integer method [5] grid step is set by the smallest distance between cone singularities of a smooth guidance field computed on the input. Such a distance is likely to be quite small on complex natural shapes. In these cases, drastic simplification can be achieved only if cone singularities are placed manually (see also Section 4). Myles et al. [12] use T-meshes to obtain a small number of adaptive patches while preserving regularity and alignment, but the representation scheme (T-splines) is much more complex.

Algorithms for progressive simplification of quad meshes [24], [25], [26] are based on sequences of small operations combined with tangent space smoothing.

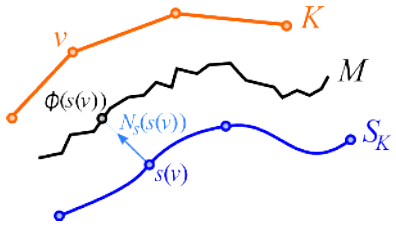


Fig. 2. The reference mesh M, the control mesh K and the subdivision surface S_K; a vertex v of K has its limit position at $s(v)$ and N_S is the normal at the limit point. Projection ϕ of S_K to M is a normal displacement.

Such algorithms usually aim at obtaining a mesh with good quality, in terms of shape of faces and vertex valencies, while quality of approximation and adaptiveness are usually addressed only indirectly.

We would like to remark that all the above methods are designed to produce general coarse polygonal meshes, rather than control meshes meant to be subdivided. As such, they provide direct control neither on the quality of the subdivision surface, nor on normal projectability to the input data.

We adopt a framework similar, in concept, to [26], interleaving local simplification with a smoothing phase. Our criteria to drive both local and global operations are rather different, though: we address both accuracy and adaptivity by introducing Fit-Maps, a guidance field derived from a pre-analysis of the mesh designed to drive the selection of operations (see Sec. 3.1); we push on adaptivity, because the coarse mesh we produce is intended to be the base of an higher order surface; we reduce the set of local operations down to just one simplification operator (*diagonal collapse*), one cleaning operator (*doublet removal*) and one optimization operator (*edge flip*)—also, we make a different use of the latter (we employ it to explicitly increase regularity); finally, in our case the smoothing phase is completely different, as it combines tangent space smoothing with local fitting, so to obtain a better approximation.

3 The Algorithm

We take in input a mesh M and we want to build a CC subdivision surface S_K, with control mesh K, that fits M according to requirements stated in Section 1.2.

We start with a subdivision surface interpolating M at all its vertices, and we progressively simplify K through local operations. Let v be a vertex of K: $s(v)$ denotes its limit position on S_K; $N_S(v)$ denotes the surface normal at $s(v)$; $\phi(s(v))$ denotes the *projection* of $s(v)$ to M along direction $N_S(v)$. Symbols are summarized in Figure 2.

The algorithm has the following outline:

1) Analyze input mesh M and compute its *Fitmap*, which is made of a pair of scalar fields that will be used to drive simplification during Step 3 (Sec. 3.1);
2) Compute initial control mesh K, having the same connectivity of M, and such that S_K interpolates the vertices of M (Sec. 3.2);
3) Iteratively process control mesh K. At each step:
   a) Perform a diagonal collapse, followed by edge swaps and/or cleaning operations, if appropriate (Sec. 3.3);
   b) Fit and smooth positions of vertices in the area affected by the previous operation(s) (Sec. 3.4);
4) Globally fit S_K to M (Sec. 3.5).

We describe our method to work on a watertight model, its extension to models with boundary being just straightforward. We assume M to be a discrete representation of a smooth manifold. Smoothness is intended in a relaxed sense here: meshes with sharp edges can also be taken in input, but they will be treated as discrete approximations of smooth manifolds containing zones with very high curvature. Since we model our output with smooth CC surfaces in output, some smoothing of sharp creases is unavoidable, and this represents a limit of our current method. See Section 5 for a possible extension to explicitly represent sharp creases.

Throughout the algorithm, we adopt the method proposed by Loop and Schaefer [1] to evaluate the limit point $s(p)$ and its surface normal $N_S(p)$ for an arbitrary point p sampled on control mesh K. A spatial index on M is used to support ray-casting to evaluate function ϕ.

3.1 Mesh analysis

During Step 3 of the algorithm, we need a mechanism to select local operations, as well as a halting criterion. The general idea is that conciseness is achieved through simplification, which should proceed as long as the surface maintains acceptable regularity, accuracy and projectability. Therefore, we wish to select, at each iteration, the local operation that best preserves such criteria. Regularity is somehow easy to test on a local basis, and it is addressed explicitly by the edge swap operations performed in Step 3a and by smoothing performed in Step 3b. On the contrary, variations in accuracy and projectability are very expensive to test, involving surface fitting and normal projection of the portion of mesh affected by an operation. For this reason, we rely on an analysis of the input mesh, both to drive the selection of collapse operations and to halt simplification.

The general idea is to estimate, for each point p of the input mesh, how well neighborhoods of p can be modeled, in terms of accuracy and projectability, by using a single patch. This analysis provides a rough estimate of how patches generated during simplification should behave.

3.1.1 Fitmaps

A Fitmap consists of a pair of values for each point p of a surface M: the *S-fitmap* ("Scale" fitmap) estimates how the RMS error of fitting a patch P to a neighborhood B of p increases with radius of B; the *M-fitmap* ("Maximal radius" fitmap) estimates how much a patch P can extend around p before correct projection of P to M through normal displacement becomes impossible. The

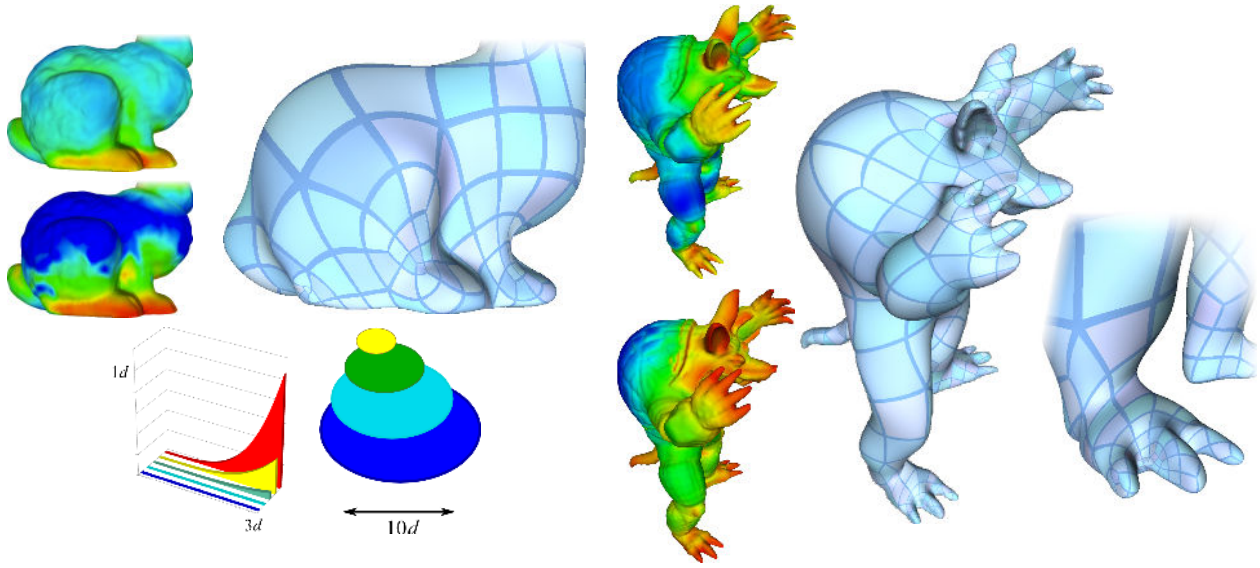


Fig. 3. The two channels of the Fitmap (S-fitmap upper; M-fitmap lower), together with subdivision surfaces built by our method, with patches outlined: adaptive distribution of patches follows the Fitmap. Color coding is depicted at the bottom left (d is 1% of the bounding box diagonal): color of the S-fitmap (left) represents the coefficient a of the model of RMS residual; color of the M-fitmap (right) corresponds to size limit for the patch.

Fitmap is computed at vertices of M and extended by linear interpolation to its faces (See Figure 3).

The Fitmap of mesh M can be interpreted as a prescription on the patches of an ideal approximation S:

- The local radius of each patch of S should be inversely proportional to the value of the *S-fitmap* computed at its central point;
- No patch of S should have a radius larger than the value of the *M-fitmap* computed at its central point.

The first condition aims at distributing error evenly, while the second condition aims at preserving projectability.

3.1.2 Building the S-fitmap

In order to estimate the S-fitmap $\mathcal{F}_S$, we proceed as follows. For each vertex p of M, we consider neighborhoods $B_{p,i}$ of p of increasing radii r_i, for $i = 0 \dots h$. For each i, we collect all vertices of the input mesh in $B_{p,i}$: vertices are gathered with a breadth-first traversal of M starting at p; for simplicity, we use Euclidean distance instead of geodesic distance to stop the search. In all our experiments, we set $h = 8$, r_0 equal to the average length of edges of M, r_h equal to $1/4$ the length of the diagonal of the bounding box, and we distribute the other radii on an exponential scale.

Next, we express $B_{p,i}$ as the graph of a bivariate function, on a local frame defined with tangent plane at p, and we fit a cubic polynomial to its vertices. Cubic polynomials serve as an easy and conservative surrogate to the more general bicubic patches that constitute our output surface S, as they are easy to fit and they are independent of the orientation of the local frame. Roughly speaking, we are assuming that bicubic patches of S will do at least as well as we can do with cubic polynomials. Vertices of $B_{p,i}$ are expressed in the tangent frame $(\mathbf{u}, \mathbf{v}, \mathbf{n})$ at p, $\mathbf{u}$ and $\mathbf{v}$ being mutually orthogonal tangent directions at p and $\mathbf{n}$ its normal. Surface $B_{p,i}$ is then expressed as the graph of function $g_{p,i} : \mathbb{R}^2 \to \mathbb{R}$ in the tangent frame, and a linear least squares problem is resolved to fit a cubic polynomial $\tilde{g}_{p,i}(u, v)$ to vertices of $B_{p,i}$. We measure the RMS residual as

$$E(r_i) = \frac{1}{n_{p,i}} \sqrt{\sum_{j=1}^{n_{p,i}} (\tilde{g}_{p,i}(u_j, v_j) - n_j)^2},$$

where summation is run over all vertices (u_j, v_j, n_j) of $B_{p,i}$, and $n_{p,i}$ is their number.

The sequence of $E(r_i)$ values for $i = 0 \dots h$ provides a sampling of how the RMS error grows in the neighborhood of p. Now, we need to compress information into a single scalar value. To this aim, we model error increase as a function of radius r. Without loss of generality, assume function g_p (i.e., the surface M expressed in the tangent frame of p) admits a Taylor expansion. Let T_3 be the cubic Taylor polynomial of g_p centered at p. Then the error of approximating g_p with T_3 at a single point at distance r from p belongs to $O(r^4)$. Now the RMS error E_{T_3} of T_3 is computed by integrating the square error over the circle of radius r, computing its square root and normalizing this by the area of the circle. It readily follows from integration and normalization that also E_{T_3} belongs to $O(r^4)$. Let $E(r)$ be the RMS error of the best fitting cubic polynomial $\tilde{g}_{p,r}$, then we will have $E(r) \leq E_{T_3}(r) \in O(r^4)$, i.e., $E(r) \leq ar^4$ for some $a > 0$. Thus, we model error with a simple function $E(r) = ar^4$. Having collected h measurements of errors at different radii r_i, we fit (in the least squares sense) such function to these values to estimate parameter a. We have also

validated this error model with empirical tests: we have fitted functions of the type ar^k for various values of k to sampled error measures; the best average fit to actual errors was consistently obtained for $k = 4$ on all datasets.

We set the value for the S-fitmap $\mathcal{F}_S(p)$ to $a^{1/4}$, so that we obtain a function that increases linearly with the radius. In this way, if two patches centered in p_0, p_1 have radii r_0 and r_1, respectively, they contribute approximatively the same error if the values of $r_0 \cdot \mathcal{F}_S(p_0)$ and $r_1 \cdot \mathcal{F}_S(p_1)$ are equal.

3.1.3 Building the M-fitmap

The M-fitmap $\mathcal{F}_M$ is built together with the S-fitmap. For a given neighborhood $B_{p,i}$ of radius r_i, we sample $\tilde{g}_{p,i}$ and we use a spatial index to cast a ray from each sample point along its surface normal (computed analytically from $\tilde{g}_{p,i}$) to M. Samples are distributed randomly on the neighborhood and their number is $4n_M \frac{|B_{p,i}|}{|M|}$, where n_M is the number of vertices of M, $|M|$ is its total area and $|B_{p,i}|$ is the area of the neighborhood. In this way, the density of samples is roughly twice the density of vertices of M. We flag each triangle hit by a ray and we count the percentage of triangles in the spanned neighborhood that are not hit by any ray, which are (conservatively) classified as flipped faces. The value of $\mathcal{F}_M(p)$ is set to the largest tested radius at which the portion of neighborhood covered by flipping faces is smaller than a "tolerance" threshold τ. Parameter τ can be user-defined, depending on the amount of high frequency noise expected in the input mesh, or on the amount of 3D high frequency detail that could be ignored, to avoid an excessive fragmentation of patches. Parameter τ is the **only** parameter used by our simplification method, and it can be used to trade-off between projectability and size of the simplified surface.

3.2 Building the initial control mesh

We first apply a tri-to-quad conversion algorithm [26] to transform the reference mesh M into a quad mesh M_q having the same set of vertices. We set initial control mesh K to the same connectivity of M_q, while placing its vertices to have the limit surface S_K interpolate the vertices of M. This is done by resolving a sparse linear system

$$LV_K = V_M, \tag{1}$$

where V_M is the vector of positions of vertices of M, V_K is the vector of (unknown) positions of vertices of K, and L is the limit subdivision matrix, which is determined by the connectivity of K and by the subdivision masks. Because of its sparsity, this system can be solved easily and efficiently with any sparse solver, even for meshes of large size [27]. We rather adopted a simple natural fixed point iterative method, which displaces each control point towards its limit position until convergence. This is perhaps not the fastest available method, but it is very simple as it does not require to manipulate L directly, and it is sufficiently fast for our purposes. In fact, this phase takes just about one second for the largest mesh we have processed. Besides, it has the advantage to be also applicable locally: we will exploit this feature during vertex optimization (see Section 3.4).

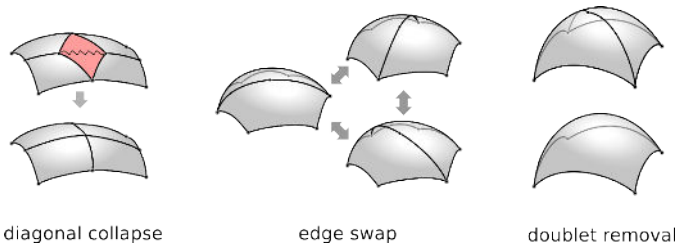


Fig. 4. The set of local operators. See details in Sec. 3.3.

Next, we define a mapping ϕ from the subdivision surface S_K to M for every point $p \in S_K$ by taking the closest point of M intersected by a ray cast from p along its normal $N_S(p)$.

3.3 Local operations

We use a subset of local operators that have been introduced in [26] to modify control mesh K throughout simplification (see Fig. 4). We have found this subset to be sufficient and effective for our purposes, the quality of results being not different from those obtained with the complete set of operators, while implementation is simpler and more robust. The criteria for applying operators during simplification, which are rather different from those in [26], are discussed in the following.

3.3.1 Diagonal collapse

This is the main operator used to reduce mesh complexity. It eliminates a quad q, two edges and a vertex, by collapsing one diagonal of q.

We maintain a heap of potential collapses, which is kept up-to-date throughout the simplification process. Cost of collapsing a diagonal d is given by $|d| \cdot \mathcal{F}_S(\phi(c))$, where $|d|$ denotes the length of d, c is the center of the patch containing d, and $\phi(c)$ is its projection to M. Collapses are prioritized according to least cost.

Collapsing an element causes its neighbors to expand. Systematically executing the least expensive collapse causes survivors to have a size proportional to the inverse of the S-fitmap, thus producing patches of variable sizes, yet yielding a roughly uniform distribution of the approximation error, as depicted in Figure 3. At the same time, where S-fitmap is nearly uniform, equality of diagonal lengths tend to favor rectangular patches. Note that the contrasting objectives of tessellation adaptivity and of patch regularity are sought together, with a natural trade-off, by operating a single criterion on lengths.

The M-fitmap is used to try avoiding collapses that hinder projectability. Given a potential collapse, we evaluate the M-fitmap at the center of surrounding patches that the collapse would extend. We perform the collapse only if, at each such patch, the M-fitmap is smaller than

its diameter, measured as the maximal distance between its center and one of its corners. Simplification is halted when no feasible collapses remain.

The vertex generated from a collapsed diagonal is set to its midpoint, prior to optimization (see Section 3.4).

3.3.2 Edge swap

This operator is used to improve the quality of the mesh. It substitutes an existing edge e with one of the other two diagonals of the hexagon formed by the two quads incident at e. Edge swaps have also the side effect of modifying lengths of diagonals, effectively driving the selection of local operations to be performed next.

After performing a diagonal collapse, we consider all faces in the 1-ring of the collapsed element and we test all their edges for potential swap. A swap operation is performed if it improves the valencies of vertices (note that this criterion is rather different from the one adopted in [26]). Given an edge e, let $v_1, \ldots, v_6$ be the vertices bounding the pair of faces incident at e. We measure the valence $D(v_i)$ of each such vertex and we set an energy $\sum_{i=1}^{6} |D(v_i) - 4|$. We swap e if and only if such a swap decreases this energy. In this way, we tend to increase the number of regular vertices of K.

3.3.3 Doublet removal

Collapse and swap operations may generate *doublets*, i.e., configurations where two adjacent quads share two consecutive edges, which join at a vertex with valence two. Doublet-removal is applied to eliminate a doublet as soon as it appears, by simply merging the two quads.

3.4 Local optimization of vertex positions

We wish to maintain surface S_K close to M throughout the simplification process. We do not need to warrant accurate fit, though, since this is done just on the final mesh, during Step 4 of the algorithm (see Section 3.5).

After each diagonal-collapse (and edge-swap and doublet-removal operations potentially triggered by it), positions of all affected vertices of K must be updated to re-fit S_K to M. Let $W \subseteq K$ be the portion of mesh directly affected by the last local operation(s), plus all the mesh spanned by its 1-ring. We resolve system (1) with only the vertices in W as unknowns, while freezing the remaining vertices of K. As W usually contains few dozens vertices, this operation is very fast. This is just an approximation: limit points of vertices in W interpolate M, but vertices in the 2-ring of W may be perturbed and leave M. However such perturbation is generally irrelevant to subsequent processing.

A better approximation to fit could be computed by using the least squares fitting algorithm described in the following subsection, by sampling just faces of W and using just vertices of W as unknowns. However, this solution is more time-consuming and we did not notice any relevant benefit in terms of final result.

In order to obtain more regularly shaped patches, we interleave local fit with Laplacian smoothing: each vertex $w \in W$ is moved midway between its current position and the average of centroids of its incident faces, before being displaced to interpolate M. We empirically found that alternating smoothing and fitting twice is sufficient.

The combined effect of Laplacian smoothing and local fit is equivalent to smooth the vertices of S_K in tangent space, without leaving the surface of M (similarly to [26], but without the need for a parametrization). This greatly improves the shape of patches.

3.5 Final fitting

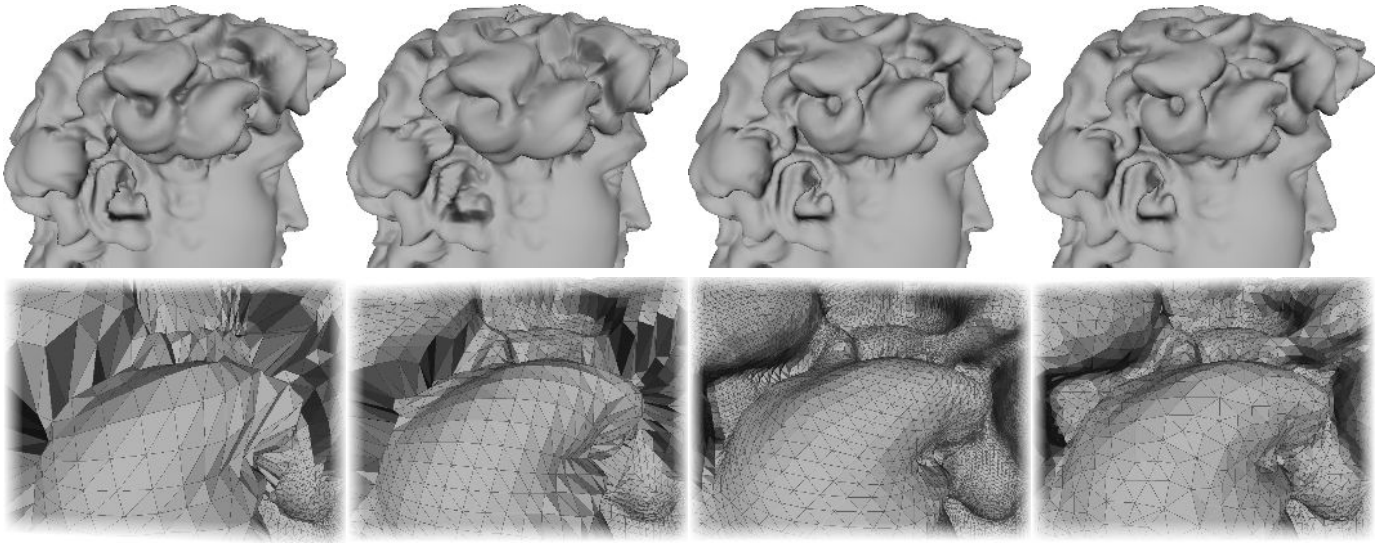
After the connectivity of control grid K has been obtained through simplification, a more accurate and global fitting process is run over the entire K. In this phase quality is crucial, therefore, following [16](2.3.3), we add extra equations to the system (1) to enforce that not only vertices of S_K but also points sampled inside its patches are close to M. Each patch of the limit surface is sampled with a number of points proportional to its area. Let $|P|$ be the area of patch P, $|M|$ be the total area of M and n be the number of its vertices. We sample P on a $k \times k$ regular grid in its parametric domain, where $k = \lfloor \sqrt{n \frac{|P|}{|M|}} \rfloor$. The limit surface corresponding of each face f of K is evaluated as a Bézier patch, whose coefficients are a linear combination of vertices in the 1-ring of f [1]. Let p be a point sampled on f, with parametric coordinates (u, v). The position of $s(p)$ on the limit surface is defined in terms of (u, v) and of the coefficient of the Bézier patch $s(f)$. This allows us filling the corresponding row of matrix L in system (1). Projection $\phi(s(p))$ of $s(p)$ on M is used as a target position for $s(p)$, which is plugged into the corresponding position of column V_M in system (1). The resulting overdetermined sparse linear system is solved in the least squares sense using a sparse solver from the Eigen library [27].

A better fit could be probably obtained by using a more advanced technique, e.g., based on the *square distance minimization (SDM)* introduced in [28]. However, control mesh simplification is independent on the technique used for final fitting, thus we did not explore this possibility so far.

4 Experimental results

The proposed method was tested on several datasets coming from range scanning. Some results are shown in Figure 8. In Table 1, we provide statistics in terms of the four requirements outlined in Section 1.2:

1) *Conciseness* is reported as the number of patches of surface S_K (faces of K) with respect to the number of faces of M;
2) *Accuracy* is reported as the RMS error of approximating M with either S_K, or its normal displaced surface;



PGP [6] 616 patches | MI (w/ manual cones) [5] 542 patches | OUR approach 774 patches | Original mesh 98K triangles

Fig. 5. Comparison of displaced subdivision surfaces of the David's hair. Because of higher projectability, we produce a more accurate approximation of the original mesh using a similar number of patches. In this example patches are sampled uniformly 10×10.

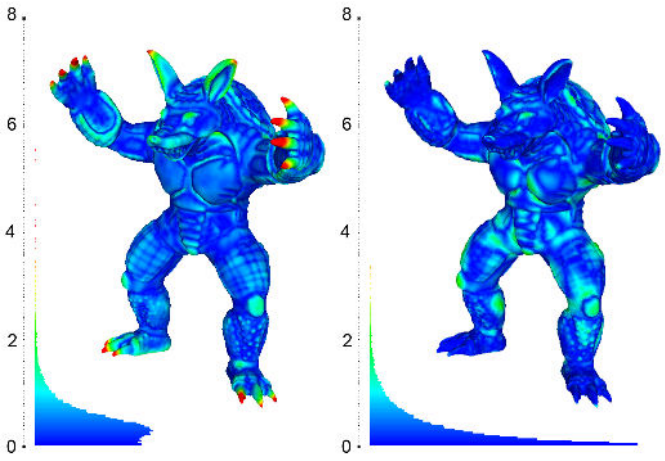


Fig. 6. Comparison of error distribution of the Limit surface for the meshes built with MI and our approach as reported in table 1 with a and b.

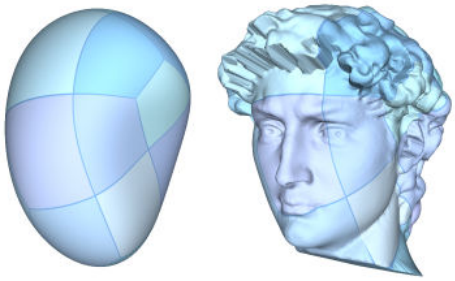


Fig. 7. A subdivision surface made of just 20 patches obtained from the original mesh of 50,446 triangles by deactivating tests on the M-fitmap. The overall shape is still preserved but relevant artifacts appear due do severe loss of projectability. Surface without (left) and with displacement mapping (right) - see artifacts on hair and ear.

3) *Regularity* is reported as the number of irregular vertices in the control mesh K;
4) *Projectability* is reported as the percentage of surface M that is not reached correctly by normal projection from S_K (0.0% means perfect projectability).

Parameter τ of the M-fitmap, as defined in 3.1.3, is used to obtain more or less simplified meshes, depending of the tolerance on the loss of projectability: higher tolerances allow for more drastic simplifications. The set values turned out to be very conservative in practice. As shown in table 1, we used tolerances of 1% and 10%, but the projectability of results is always larger than 99%, except for the David dataset which shows 4% of non-projectable surface if the tolerance is set at 10%. Overall, our meshes achieve better projectability than those produced with other methods, for a comparable, and sometimes even much smaller, number of patches.

Subdivision surfaces with a relatively large number of patches, which can be obtained by setting a small tolerance for the M-fitmap, or by stopping simplification before its completion, approximate the input shape well even without displacement. In such cases, normal mapping is sufficient to achieve a reasonably good quality of rendering. Conversely, even drastically simplified surfaces made of few patches can achieve almost perfect quality through displacement mapping.

In order to test the effectiveness of the M-fitmap to prevent losing projectability, we have run some experiments by inhibiting tests on the M-fitmap, thus allowing

Mesh (△)	Alg.	□K	Irreg. vert.	Unp. (%)	RMS Error L	RMS Error D
Garg. (49k)	PGP	742	127	0.3	4.05	0.23
	MI	2904	83	**0.0**	1.00	0.02
	VSA	2946	816	0.1	1.46	0.06
	1%	1096	345	**0.0**	2.36	0.07
	10%	**493**	182	0.4	4.50	0.15
	*	2897	821	**0.0**	1.32	0.03
Moai (52k)	PGP	830	61	**0.0**	1.30	0.20
	VSA	240	61	**0.0**	2.62	0.13
	1%	92	30	**0.0**	5.00	0.12
	10%	**41**	16	**0.0**	8.22	0.18
Bunny (69k)	PGP	1042	114	**0.0**	1.98	0.13
	VSA	450	104	**0.0**	3.03	0.11
	1%	506	216	**0.0**	3.41	0.08
	10%	**218**	72	**0.0**	8.48	0.14
Fert. (80k)	PGP	3784	176	**0.0**	0.34	0.04
	MI	3357	48	**0.0**	0.28	0.10
	VSA	2772	739	0.2	2.16	0.03
	1%	489	194	**0.0**	2.35	0.04
	10%	**323**	130	0.6	5.10	0.40
	*	3352	956	**0.0**	**0.13**	**0.005**
David (98k)	PGP	616	108	4.9	5.70	1.08
	MI	39K	261	**0.0**	0.24	0.08
	(MI)	542	8	3.9	4.96	0.45
	VSA	1348	349	0.8	3.80	0.16
	1%	1603	496	0.2	3.29	0.07
	10%	**375**	145	3.0	9.09	0.43
Arm. (345k)	PGP	1600	231	13.2	11.03	5.86
	MI	2184	74	0.8	[a] 3.32	0.19
	VSA	534	143	2.3	8.24	1.29
	1%	1402	466	**0.1**	2.46	0.07
	10%	**476**	258	0.3	4.48	0.12
	*	2170	773	**0.1**	[b] **2.35**	**0.08**

TABLE 1
From the left: name of input mesh (number of triangles); method used to produce the control mesh (percentages refer to our method with different values of parameter τ of the M-fitmap; (MI) refers to the MI method with manual placement of cone singularities); number of patches □K in the final control mesh; total number of irregular vertices; portion of the input mesh that is not reached correctly by normal projection from the limit surface; RMS error (in 1000th of the diagonal of the bounding box) by using either the limit (L), or the displaced (D) subdivision surface. The line denoted with the (*) refers to a model built with our approach and with a number of patches similar to the MI output.

all collapses to be performed. The process is stopped manually after very drastic simplification. As shown in Figure 7, an extremely simple subdivision surface made of just 20 patches is still able to give a reasonable reconstruction through displacement mapping, but relevant artifacts appear.

The proposed method works with real world objects of medium resolution. In case a subdivision surface, with a reasonably small number of patches, is to be extracted from a large high resolution mesh M, it is possible to compute the Fitmap (and the spatial index necessary for ray casting) on the original mesh M, while starting simplification from a simplified mesh M' with a smaller size, which can be computed with any standard program for triangle mesh simplification. The control mesh K is initialized on the basis of M', but all computation, including fitting and mapping, is referred to the original mesh M. We followed this approach for running experiments on the Armadillo dataset, starting with a mesh M' with about 100K triangles.

The technique is somehow time consuming, partly because the code was not optimized for fast prototyping. However, timings reported in Table 2 are reasonable considering that this is a pre-processing computation. See Section 5 about possible optimizations.

	Dataset name and size					
Times (secs.):	Gargo 40K	Moai 53K	Bunny 70K	Fert. 80K	David 99K	Arm. 98K
Fitmap	69	149	293	332	454	442
simplif./fit	486	592	780	722	1320	1360

TABLE 2
Running times in seconds for computing Fitmaps, and for simplification and subdivision surface fitting.

4.1 Comparison with literature

We compare our results against the ones obtained by first computing a control mesh with alternative methods, then fitting a Catmull-Clark surface with the algorithm described in Section 3.5. We tested two remeshing algorithms, *Mixed Integer (MI)* [5] and *Periodic Global Parametrization (PGP)* [6], and one clustering method, *Variational Shape Approximation (VSA)* [7].

PGP meshes have been computed with the publicly available *Graphite* software, extracting the coarsest possible mesh with standard parameters. Since produced meshes containts triangles, one step of Catmull-Clark subdivision was needed to remove them.

MI meshes (provided by the authors of [5]) are available only for some datasets. In the David dataset MI places a large number of cone singularities in areas containing small features (hair curls), so that model cannot be coarsened below 39K quads. A much coarser version has been produced by *manually* placing just 8 cone singularities.

VSA meshes were produced with a software provided by the authors of [6]. The process is semi-automatic: the user chooses the number of seeds and when to stop optimization. We have tried to roughly match the coarseness of meshes extracted with our method, or the coarsest which could be obtained before results differed too much from the input mesh for the fitting algorithm to work. Again, one step of Catmull-Clark was necessary to get rid of triangles in the in resulting models.

Results show that our method is consistently able to obtain much coarser meshes than competitors. In most

cases the number of quads is lower by about one order of magnitude with respect to other methods, yet perfect or almost perfect projectability is maintained.

To assess accuracy, in table 1, the RMS error of limit and displaced surface is measured in 1000th's of the bounding-box diagonal of the object (in each experiment, the displacement maps are obtained by sampling each patch with a fixed resolution chosen so that the total number of samples roughly matches the number of vertices of the original mesh). For sake of comparison, in a few experiments we forced our Fitmap based approach to produce base meshes with a face-count similar to the best competitors (rows marked with a $*$ in tab. 1). The results show a comparable, often better, RMS error obtained with our method. Moreover, visual comparisons (e.g. Fig. 1) and error distributions (Fig. 6) reveal that our method better preserves local small scale features, something that the RMS error fails to clearly detect, being averaged over the entire surface. Again, we stress that one strength of our method consists in the ability to reduce the face-count of base mesh drastically more than other methods can, while preserving good accuracy and projectability.

In terms of regularity, comparison yields mixed results. Note that, with our method, presence of a few extra irregular vertices is implied by the adaptiveness of the tessellation, as irregular vertices are unavoidable in zones of transition among different levels of detail (e.g. from the back to the arm to the fingers in the Armadillo, or from the cheek area to the hair area of the David). The MI gives fewest irregular vertices with some datasets (Gargoyle, Fertility and Armadillo), and a comparable number with the David (unless, predictably, singularities are placed manually). Our method performed comparably with PGP and VSA, resulting in more regular or less regular results depending on the dataset and the used parameters. *Highly* irregular vertices (valency > 5) are very rare with our method (unlike, for example, with VSA). VSA and MI in some cases also generated doublets (valency 2 vertices), whereas our method is bound to never output them.

5 Concluding remarks

We have presented a fully automatic method for building a coarse control mesh for approximating an input shape with a Catmull Clark surface. Our method produces surfaces with low complexity, good quality and accuracy, and almost perfect projectability. In contrast with standard quad-simplification and quad-remeshing approaches, our method explicitly works on a control mesh for subdivision, and the limit surface is targeted throughout the process. To our knowledge, this is the first method to address the problem in this integrated way, in the context of quad-based subdivision surfaces.

A key issue here is adaptiveness of patch density to geometrical complexity. To this aim, we have introduced Fitmaps, which provide an *a-priori* estimation for the ideal localized patch densities. Fitmaps demonstrate to be very effective at driving the local simplification process to build better control meshes that adapt locally and concisely to fine details. The concept of Fitmaps can probably be adapted to any other form of parametric surfaces, as well as to the simpler case of adaptive mesh simplification.

Surfaces produced with our method are suitable for GPU-assisted rendering via displacement mapping. The method may also support reverse engineering, by providing initial control grids that may be adjusted and refined with modeling tools.

Our current method has some limitations, though. Compared to global remeshing methods (like [5]), it produces meshes containing more irregular vertices. Some such vertices are necessary to warrant transitions through different levels of resolution inherent to adaptive tessellation. However, we believe that irregular vertices could be further reduced by more careful tessellation. The definition of a good balance between adaptivity and regularity demands further investigation.

Another problem is the lack of alignment of patch boundaries to either curvature directions, or other line features. Alignment to features could be enforced with a snapping mechanism, similarly to [26]. The same mechanism can be also used to model surfaces with sharp creases, by combining it with the extension of subdivision surface evaluation presented in [29]. Also this issue requires further investigation.

The time performance of the method can be certainly improved, possibly of orders of magnitude. The main bottlenecks come from subdivision surface evaluation, which is performed many times at many samples during simplification, and by the extraction of large neighborhoods during the construction of fitmaps. Surface evaluation could be easily delegated to the GPU. The extraction of large neighborhoods could be made faster by using an ad-hoc data structure, such as a hierarchical spatial index supporting range queries according to geodesic distance.

References

[1] C. Loop and S. Schaefer, "Approximating catmull-clark subdivision surfaces with bicubic patches," *ACM Trans. Graph.*, vol. 27, no. 1, pp. 1–11, 2008.

[2] K. Gee, "Direct3d 11 tessellation," GameFest - Microsoft game technology conference, 2008.

[3] C. Eisenacher, Q. Meyer, and C. Loop, "Real-time view-dependent rendering of parametric surfaces," in *I3D '09: Symposium on Interactive 3D Graphics and Games*. New York, NY, USA: ACM, 2009, pp. 137–143.

[4] I. Castaño, "Next-generation hardware rendering of displaced subdivision surfaces," SIGGR2008 - Exhibitor Tech Ses., 2008.

[5] D. Bommes, H. Zimmer, and L. Kobbelt, "Mixed-integer quadrangulation," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 1–10, 2009.

[6] N. Ray, W. C. Li, B. Lévy, A. Sheffer, and P. Alliez, "Periodic global parameterization," *ACM Trans. Graph.*, vol. 25, no. 4, pp. 1460–1485, 2006.

[7] D. Cohen-Steiner, P. Alliez, and M. Desbrun, "Variational shape approximation," in *ACM Trans. Graph. (SIGGRAPH)*. New York, NY, USA: ACM, 2004, pp. 905–914.

[8] S. Dong, P.-T. Bremer, M. Garland, V. Pascucci, and J. C. Hart, "Spectral surface quadrangulation," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 1057–1066, 2006.

[9] F. Kälberer, M. Nieser, and K. Polthier, "Quadcover surface parameterization using branched coverings." *Computer Graphics Forum*, vol. 26, no. 3, pp. 375–384, 2007.

[10] E. Catmull and J. Clark, "Recursively generated b-spline surfaces on arbitrary topological meshes," *Computer Aided Design*, vol. 10, pp. 350–355, 1978.

[11] A. Lee, H. Moreton, and H. Hoppe, "Displaced subdivision surfaces," in *ACM Trans. Graph. (SIGGRAPH)*. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 2000, pp. 85–94.

[12] A. Myles, N. Pietroni, D. Kovacs, and D. Zorin, "Feature-aligned t-meshes," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 1–11, 2010.

[13] K.-S. D. Cheng, W. Wang, H. Qin, K.-Y. K. Wong, H. Yang, and Y. Liu, "Fitting subdivision surfaces to unorganized point data using sdm," in *PG '04: Proc. of the Computer Graphics and Applications, 12th Pacific Conference*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 16–24.

[14] G. Lavoué and F. Dupont, "Technical section: Semi-sharp subdivision surface fitting based on feature lines approximation," *Comput. Graph.*, vol. 33, no. 2, pp. 151–161, 2009.

[15] M. Marinov and L. Kobbelt, "Optimization techniques for approximation with subdivision surfaces," in *SM '04: Proceedings of the ninth ACM symposium on Solid modeling and applications*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2004, pp. 113–122.

[16] J. Hoschek and D. Lasser, *Fundamentals of computer aided geometric design*. Natick, MA, USA: A. K. Peters, Ltd., 1993, translator-Schumaker, Larry L.

[17] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle, "Piecewise smooth surface reconstruction," in *ACM Trans. Graph. (SIGGRAPH)*. New York, NY, USA: ACM, 1994, pp. 295–302.

[18] W. Ma, X. Ma, S.-K. Tso, and Z. Pan, "A direct approach for subdivision surface fitting from a dense triangle mesh," *Computer Aided Geometric Design*, vol. 36, no. 16, pp. 525–536, 2004.

[19] H. Suzuki, S. Takeuchi, F. Kimura, and T. Kanai, "Subdivision surface fitting to a range of points," in *PG '99: Proceedings of the 7th Pacific Conference on Computer Graphics and Applications*. Washington, DC, USA: IEEE Computer Society, 1999, p. 158.

[20] T. Kanai, "Meshtoss: Converting subdivision surfaces from dense meshes," in *Proc. of the Vision Modeling and Visualization Conference*. Aka GmbH, 2001, pp. 325–332.

[21] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *ACM Trans. Graph. (SIGGRAPH)*. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1997, pp. 209–216.

[22] I. Boier-Martin, H. Rushmeier, and J. Jin, "Parametrization of triangle meshes over quadrilateral domains," in *Proceedings Symposium on Geometry Processing*, 2004.

[23] M. Marinov and L. Kobbelt, "Automatic generation of structure-preserving multi-resolution models," *CG Forum*, vol. 24, no. 3, pp. 479–486, 2005.

[24] J. Daniels, C. Silva, and E. Cohen, "Localized quadrilateral coarsening," *Comput. Graph. Forum*, vol. 28, no. 5, pp. 1437–1444, 2009.

[25] J. Daniels, II, C. T. Silva, and E. Cohen, "Semi-regular quadrilateral-only remeshing from simplified base domains," in *SGP '09: Symposium on Geometry Processing*. EG Association, 2009, pp. 1427–1435.

[26] M. Tarini, N. Pietroni, P. Cignoni, D. Panozzo, and P. E., "Practical quad mesh simplification," *CG Forum (Eurographics 2010)*, vol. 29, no. 2, pp. 407–418, 2010.

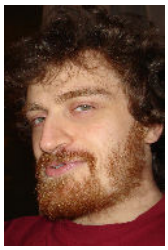
[27] "Eigen library - sparse direct solvers." [Online]. Available: http://eigen.tuxfamily.org/dox-devel/TutorialSparse.html#TutorialSparseDirectSolvers

[28] H. Pottmann, S. Leopoldseder, and M. Hofer, "Approximation with active b-spline curves and surfaces," in *PG '02: Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*. Washington, DC, USA: IEEE Computer Society, 2002, p. 8.

[29] D. Kovacs, J. Mitchell, S. Drone, and D. Zorin, "Real-time creased approximate subdivision surfaces," in *I3D '09: Proceedings of the 2009 symposium on Interactive 3D graphics and games*. New York, NY, USA: ACM, 2009, pp. 155–160.

Daniele Panozzo received his undergraduate degree in 2007 and his master degree in 2008, both in Computer Science, from the University of Genova. He is currently enrolled as PhD student in Computer Science at the University of Genova, and temporarily a visiting graduate student at the Courant Institute of Mathematical Sciences of the New York University. During 2008, he has been a visiting student at the Department of Computer Science of the University of Maryland, where he worked on his master thesis. He has written eight scientific publications.

Enrico Puppo is professor of computer science at the Department of Computer and Information Sciences (DISI) of the University of Genova. He received a Laurea in Mathematics from the University of Genova in March 1986. From April 1986 to October 1998 he has been research scientist at the Institute for Applied Mathematics of the National Research Council of Italy. He is with DISI since November 1998. He has written over one hundred scientific publications on the subjects of algorithms and data structures for spatial data handling, geometric modeling, and geometry processing.



Marco Tarini (Ph.D. 2003, Univ. of Pisa) is an Assistant Professor at the Univ. of Insubria (Varese, Italy) and an Associate Researcher with CNR-ISTI. He teaches and researches in Computer Graphics, his main published contributions being in 3D surface acquisition, modelling, parametrization, real-time rendering, and scientific visualization. Marie Curie Mobility Fellow in 2001 (spent in MPI-Saarbrüken). He received "Best Young Researcher" award by the Eurographics association in 2006.

Nico Pietroni is a researcher at the Istituto di the Scienza e Tecnologie dell'Informazione (ISTI) of the National Research Council (CNR) in Pisa, Italy. His research interests include mesh parametrization, texture synthesis and deformable object modeling. He received in 2003 an advanced degree in Computer Science (Laurea) from the University of Pisa and in 2009 a Ph.D. Degree in Computer Science at the University of Genova.

Paolo Cignoni is a Senior Research Scientist with CNR-ISTI. He received a Ph.D. Degree in Computer Science at the University of Pisa in 1998. He has been awarded "Best Young Researcher" by the Eurographics association in 2004. His research interests cover Computer Graphics fields ranging from visualization and processing of huge 3D datasets, to 3D scanning in the cultural heritage field and to Scientific Visualization. He has published more than one hundred papers in international refereed journals and conferences.

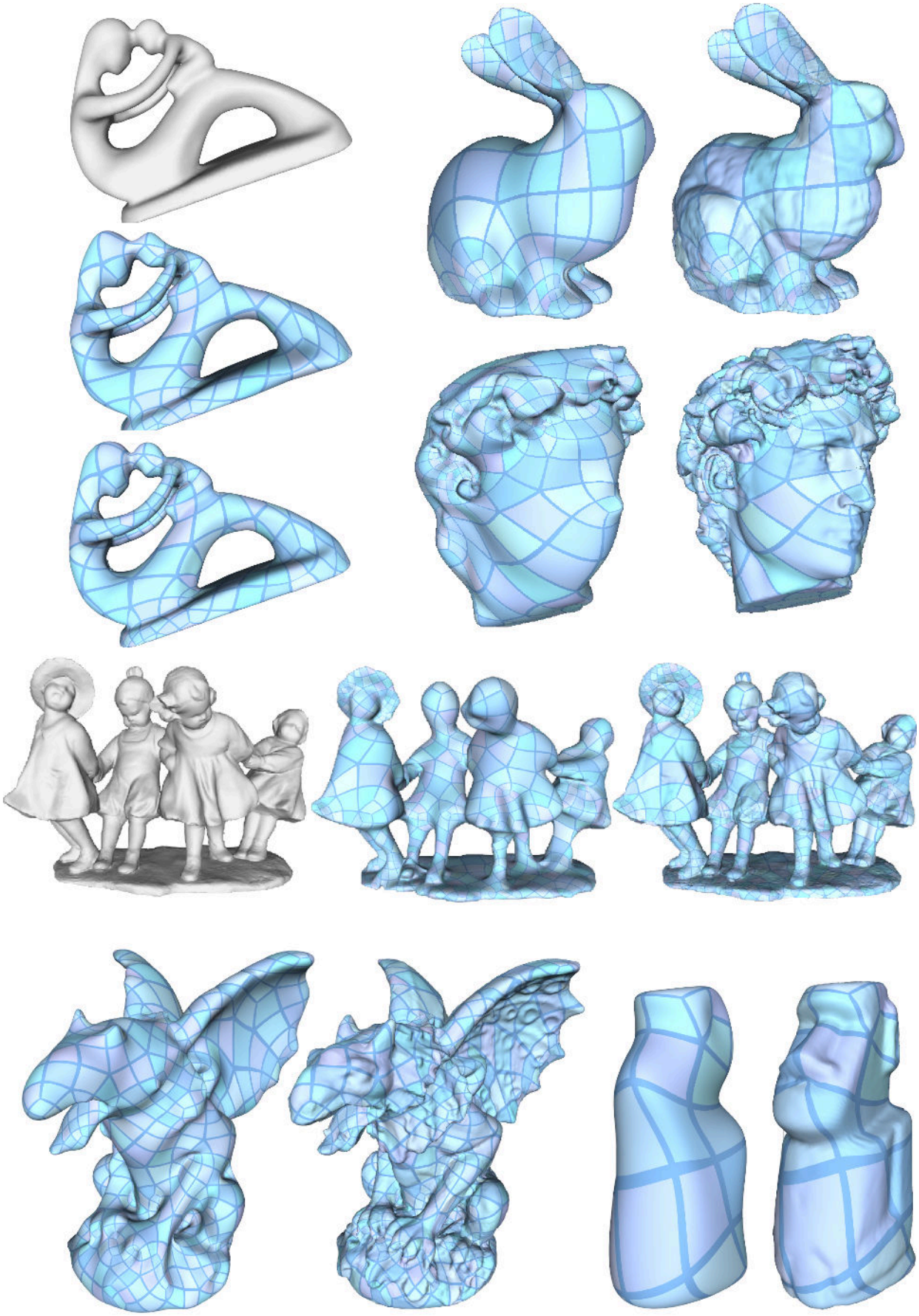


Fig. 8. Examples of meshes computed with our system. From the left: original mesh M in white; subdivision surface S_K; and displaced subdivision surface.