Combinatorial Mesh Optimization

Vincent Vidal, Christian Wolf, Florent Dupont

Visual Computer manuscript No.
(will be inserted by the editor)

Combinatorial mesh optimization

Vincent Vidal · Christian Wolf · Florent Dupont

Received: date / Accepted: date

Abstract A new mesh optimization framework for 3D triangular surface meshes is presented, which formulates the task as an energy minimization problem in the same spirit as in Hoppe et al. [1]. The desired mesh properties are controlled through a global energy function including data attached terms measuring the fidelity to the original mesh, shape potentials favoring high quality triangles and connectivity as well as budget terms controlling the sampling density. The optimization algorithm modifies mesh connectivity as well as the vertex positions. Solutions for the vertex repositioning step are obtained by a discrete graph cut algorithm examining global combinations of local candidates.

Results on various 3D meshes compare favorably to recent state-of-the-art algorithms. Applications consist in optimizing triangular meshes and in simplifying meshes, while maintaining high mesh quality. Targeted areas are the improvement of the accuracy of numerical simulations, the convergence of numerical schemes, improvements of mesh rendering (normal field smoothness) or improvements of the geometric prediction in mesh compression techniques.

Keywords Triangular meshes · mesh optimization · discrete optimization · graph cuts

This work has been supported by the French National Research Agency (ANR) through the MDCO program (project MADRAS No. ANR-07-MDCO-015)

V. Vidal and C. Wolf
Université de Lyon, CNRS
INSA-Lyon, LIRIS, UMR5205, Villeurbanne F-69621, France
Tel.: +33-4-7243 E-mail: {vincent.vidal,christian.wolf}@liris.cnrs.fr

F. Dupont
Université de Lyon, CNRS
Université Lyon 1, LIRIS, UMR5205, Villeurbanne F-69622, France
E-mail: florent.dupont@liris.cnrs.fr

1 Introduction

Nowadays, 3D triangular meshes are commonly used in many fields, as the hundreds of thousands of existing 3D triangular models can attest. They are of interest in visual effects, video games, scientific visualization, 3D animation and medical surgery simulation based on finite element methods, to name a few. Most of the existing triangular meshes are of unsatisfying quality because of their inappropriate vertex sampling, which is responsible for inequilateral triangles and irregular connectivity. The origin of this irregular sampling may be due to the scanning device, to 3D interactive solid modeling software or to simplification algorithms.

This poor quality causes instability and divergence of various mesh processing applications [2]. In that context, several remeshing techniques have been introduced in the literature. They consist in improving some quality requirements under soft and/or hard constraint conditions. Targeted goals vary according to the application [3]. *Simplification techniques* tend to preserve the overall shape of the mesh while removing as many triangles as possible. *Mesh smoothing methods* consist in removing high frequency noise so as to fair the mesh. Finally, *Mesh optimization* aims at improving the mesh quality, i.e. the regularity of the sampling and of the connectivity.

In this paper, the mesh optimization problem is stated as an energy minimization problem. The proposed energy function (see equation 4) captures the fidelity to the initial mesh geometry, the mesh quality (expressed in terms of triangle shape and vertex valence) and the number of vertices (simplification versus remeshing). Hence, according to the initial mesh and desired goals of the method, it is possible to improve the compactness of the representation, the triangle shape,

the vertex valence, while controlling the geometric error introduced by such competing desires.

We solve the proposed energy minimization problem with an iterative two step minimization method. The first step consists of a greedy mesh connectivity energy minimization process. Edge flips and edge collapses/splits are done only if they decrease the energy function. The second step performs global vertex repositioning, which is formulated as a discrete max-flow/min-cut problem in a directed graph. More specifically, this problem is solved using a variant of QPBO (Quadratic Pseudo-Boolean Optimization), a recent graph cut technique which guarantees a non-increase of the energy function after the vertex relocation step. As a result, the energy minimization problem is solved with guarantees on its convergence. Figures 4 and 5 show some results of our regularization algorithm.

1.1 Related work

Two classes of approaches have been identified in the vast mesh optimization literature. The first group consists of methods which do not offer a high control over the geometric error and/or over the mesh connectivity when new points are sampled on the surface. *Variational partitioning frameworks* [4–6] are based on vertex/triangle clustering and are often used to produce a coarser mesh with a high approximation quality. This coarse mesh is then usually retiled or refined using local re-triangulation to improve the point sampling. *Parameterization methods* [7] optimize 2D patches instead of a 3D surface, but they generally suffer either from distortion produced by the global parameterization, or need patch boundary post-processing resulting from local parameterization. *Semi-regular remeshing* [8] uses an initial coarse mesh partition and treats each patch separately using subdivision rules. This produces a few irregular vertices (those of the coarse mesh), well-shaped triangles, and a small geometric error. However, it is sensitive to the patch structure and the resulting vertex sampling is difficult to control. *Geodesic front propagation techniques* [9,10] consider geodesic equidistant curves over the surface and allow to get well-shaped triangles (vertices can be distributed according to the local curvature). However, some post-processing steps are needed to avoid artifacts when the curve topology is complex and the geometric approximation error is located near sharp features. All these methods give an overall good triangle shape, but they suffer from a lack of geometric error control when new points are sampled on the surface, since they cannot locally control the geometric approximation of some high frequency features.

The second group of methods works directly on the initial mesh simplices (vertices, edges and resulting triangles), which allows a better control of the geometric fidelity to the original 3D surface. *Local approaches* to mesh optimization consist in using a set of local legal moves (e.g. towards barycenters or angle-bisectors) and connectivity modifications (topological operators) to decrease an energy function. Such an approach may lead to a local minima configuration, especially in combination with greedy optimization (e.g. gradient descent). Surazhsky and Gotsman [11] use local operations such as edge-collapse, edge-split and edge-flip to regularize the mesh connectivity. Surazhsky et al. [11, 12] apply area-based smoothing to control both triangle quality and vertex sampling over the mesh. To achieve a precise isotropic vertex placement, Surazhsky et al. [12] use a centroidal Voronoi tessellation (CVT). *Global approaches* to mesh optimization attempt to resolve the vertex repositioning problem in a global way, most of the time solving a sparse linear system [13,14] or using least squares approximation [15]. The main idea in the global approaches using a so-called *Laplacian global operator* [13–15], is to infinitely apply a Laplacian operator such that applying it one more time will not change the current vertex positions. That allows a direct formulation as a linear system. Then other constraints are added to take into account invariant vertex positions or to avoid the shrinking effect due to Laplacian smoothing. These global vertex repositioning techniques depend on the initial sampling and connectivity, and the triangle shape may therefore be difficult to improve in case of irregular configurations (e.g. irregular vertex degrees and/or low sampling). In most local approaches, connectivity and the vertex positions are optimized separately as in [1]. That is essentially due to the fact that the combinatorial complexity of mesh connectivity optimization does not allow to solve it neither globally nor jointly with the vertex repositioning problem.

Other works related to the surface sampling improvement (e.g. direct sampling) or to the approximation error control (e.g. volume computation between two meshes) may be of interest to the reader, but are beyond the scope of this paper.

The proposed method is characterized by the following advantages:

- Many objects present a noticeable amount of sharp features (sharp edges and corners), which are robustly detected and preserved during our optimization process (see section 2.3). The coherency of geometric features is exploited in the feature edge detection process to be more robust to the presence of noise.

- Geometric error is kept low, while triangle quality and compactness of the representation are improved. No high frequency noise is introduced in the smooth parts of the processed objects during all optimization stages.
- Vertex regularity is improved only when it does not penalize the triangle quality and the geometric error.
- The locality of our approach allows a high control on the proposed vertex new positions, which can be exploited to favor vertex positions alignment along a curve.

Our main contributions are two-fold:

- A discrete-continuous mesh position optimization algorithm for triangular meshes. New vertex positions are proposed locally and the decisions which keep either the new vertex position or the old one are globally taken based on the calculation of the minimum *st*-cut/maximum flow in a graph. This ensures a local control over the candidate positions while minimizing the energy globally and avoiding oscillation problems.
- A method for feature edge detection based on an improved Potts model, which favors contiguous lines of sharp edges with approximately the same dihedral angles and directions for adjacent sharp edges.

The paper is organized as follows: section 2 introduces the notations used throughout the paper, presents an outline of our regularization algorithm and explains how the method copes with sharp features. In section 3, we present our global energy minimization problem and in particular we define the objective function. Section 4 deals with the minimization of the objective function and more precisely the optimization of mesh vertices and connectivity. In section 5, we give experimental results and compare them with other recent algorithms from the state of the art. In section 6, we conclude and discuss some future work.

2 Outline of the proposed mesh optimization algorithm

2.1 Notations

The following notations are used throughout the paper:

- M (resp. M') denotes the initial mesh (resp. optimized/remeshed model).
- X stands for all the remeshed model vertex positions. Vertices are indexed and thus X_s means the current vertex position s. X^{new} (resp. X_s^{new}) embodies all new candidate positions (resp. the candidate position for vertex s) at a given vertex repositioning iteration. X_s, $X_s^{new} \in \mathbb{R}^3$.
- Y stands for all the initial model vertex positions, $Y_s \in \mathbb{R}^3$.
- N_s represents the set of vertex indices which are in the one-ring neighborhood of the vertex indexed by s (s $\notin N_s$). X_{N_s} accounts for the set of vertex positions which are in the one-ring neighborhood of X_s. The neighbor relationship is symmetric: $r \in N_s \Rightarrow s \in N_r$.
- $\mathcal{E}$ denotes the current remeshed model edges. $(s, r) \in \mathcal{E}$ iff $s \in N_r$.
- $\mathcal{F}$ stands for all the initial mesh edges.
- T represents the remeshed model triangles. $(s, r, q) \in T$ iff $s \in N_r$ and $s \in N_q$ and $r \in N_q$.

2.2 Algorithm description

The global scheme of our method for optimizing 2-manifold triangular meshes is illustrated in figure 1 and in algorithm 1. In the initialization phase, we compute principle curvatures and detect geometric features, which remain constant during the subsequent steps. Curvature extraction is based on normal cycles calculus [16]. The feature detection procedure is explained in section 2.3. Features are preserved during the mesh regularization process. After the initialization, the algorithm follows a loop in which a single objective energy function (defined in equation 4) is minimized by two different steps. Vertex positions are optimized at each iteration, while the mesh's connectivity is optimized less frequently (every five iterations in our experiments). The interest of optimizing the mesh connectivity less often resides in significantly decreasing the computation time (i.e. by an order of magnitude) while not degrading the output quality. The vertex repositioning and mesh connectivity improvement stages are, respectively, detailed in sections 4.1 and 4.2. The algorithm offers the possibility to use the gradient of the energy function during the vertex optimization stage, which is particularly useful in the final iterations when vertex positions are refined. The terminology [gradient]-guided (resp. non-guided) iterations will thus refer to vertex repositioning iterations which do (resp. do not) make use of the gradient direction.

2.3 Robust feature edge and corner detection

Automatic detection of geometric features such as feature lines and corners has already been studied before

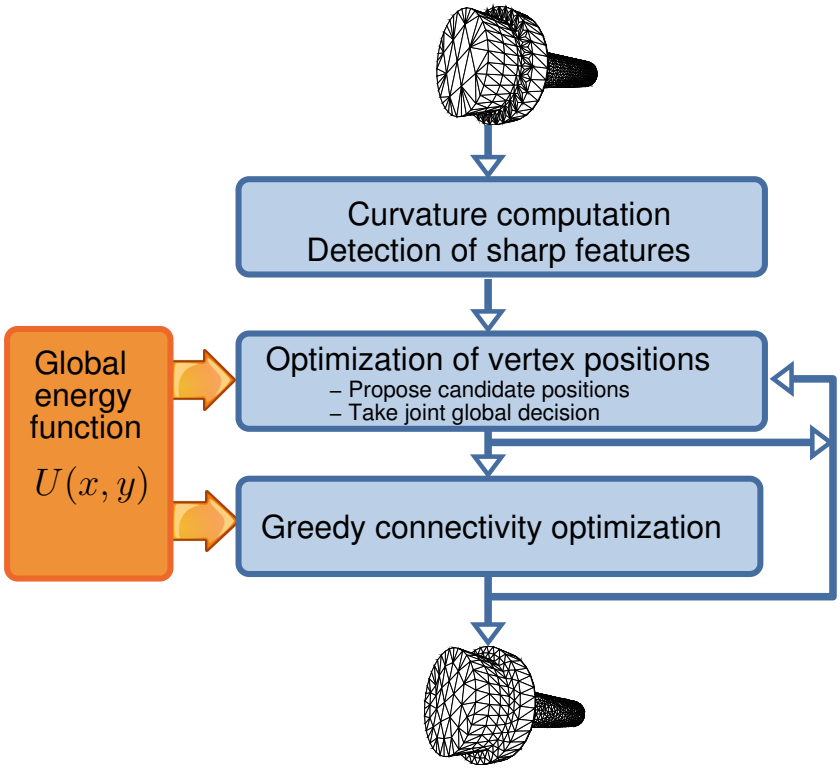


Fig. 1 The remeshing pipeline: at every iteration i, new vertex positions are chosen using a global energy minimum approximation; less frequently, the mesh connectivity is improved to favor better global vertex configurations.

(e.g. [17]). Geometric features are generally defined as discontinuities in the normal direction (i.e. first derivatives) or boundaries of the object.

In our method, feature detection allows to keep corner positions unchanged and to restrict vertices on feature lines to stick to these lines in the vertex optimization step. In addition, during the connectivity optimization stage, feature edges are prohibited from modifying their direction. For instance, feature edge flipping is forbidden. By doing so, the intrinsic geometric properties of the initial triangular mesh are preserved.

Intuitively, discontinuity edges or feature edges could be detected by thresholding the edge dihedral angles θ_i to classify them as either normal or sharp using a threshold θ_{Th}. However, due to the presence of noise, feature edges may be difficult to characterize with this technique. For instance, some geometric models do not have any dihedral angle threshold capable of distinguishing feature edges from others without connecting true features with geometric artifacts or vice versa.

In the same lines of our vertex repositioning optimization procedure described in the rest of the paper, the classification performance of the feature edge detection process can be significantly boosted by taking the decisions for all edges of the 3D model jointly, i.e. globally. This can be achieved through a model which complements the information residing in the dihedral angles of all edges with additional terms favoring consistent feature lines, i.e. coherent neighboring edge labels. We propose an improved Ising/Potts model, known from image restoration [18], which in our case minimizes a global energy function over all edge labels:

$$\begin{aligned} \hat{W} &= \arg\min_{W} E(W;\theta,\phi) \\ &= \arg\min_{W} \sum_{i} E_d(W_i;\theta_i) - \mu \sum_{i,j:i\in\mathcal{N}_j} S(\phi_{ij})\delta(W_i,W_j) \end{aligned} \tag{1}$$

where W is the set of all indexed edge variables W_i, θ is the set of all indexed dihedral angles θ_i, and ϕ is the set of all indexed turning angles ϕ_{ij} measured on pairs of edges. Each binary edge variable W_i can take a value/label in $\mathcal{L}=\{0,1\}$, which correspond to *normal edge* and *feature edge*. $\mathcal{N}_i$ is a set of edge indices which correspond to edge variables adjacent to the edge indexed by i.

The energy function is characterized by data attached terms E_d which push the solution into the direction determined by the dihedral angles, as well as regularizing pairwise terms which favor geometrically coherent feature lines. The weight μ sets the relative strength of the pairwise terms compared to the data attached terms.

$E_d(W_i,\theta_i)$ is a data attached term taking into account the dihedral angle θ_i of the edge and which has been designed as a softened thresholding with threshold

θ_{Th} and linear dependency of the energy on the angle:

$$E_d(W_i;\theta_i) = \begin{cases} \cos(\theta_i) & \text{if} \quad W_i = 1 \Leftrightarrow \text{feature} \\ 2\cos(\theta_{Th})\text{-}\cos(\theta_i) & \text{else } W_i = 0 \Leftrightarrow \text{normal} \end{cases} \quad (2)$$

δ is the Kronecker symbol equal to one whenever the condition $W_i = W_j$ is satisfied and zero otherwise. δ can be interpreted as a term favoring the same labeling for neighboring edges. $S(\phi_{ij})$ is a function which controls the strength of the regularizing term through a measure of geometric coherency, i.e. homogeneous labels are favored more if the neighboring edges are similarly oriented:

$$S(\phi_{ij}) = \exp\left\{-50.(1 - cos(\phi_{ij}))\right\}. \quad (3)$$

All pairwise terms in (1) are submodular (see also section 4.1.2 on submodularity), therefore the global minimum can be very efficiently computed using graph cuts and Kolmogorov et al.'s st-graph construction [19].

Once the feature edges are detected, vertices can be labeled given the decisions on the edges: a vertex is considered as a corner if it has at least three adjacent feature edges, or if it has two adjacent and non-aligned boundary edges.

Algorithm 1: The whole method, including the continuous-discrete solution to vertex relocation. $K^{(0)}, C$ and i_{max} are, respectively, start temperature, cooling speed and number of iterations. Remaining notations are defined in section 2.1. The graph cut algorithm takes the global decision minimizing the energy presented in section 3 for the whole set of vertices, considering for each vertex of the remeshed model the current position X_s and a new candidate position X_s^{new}.

Input: M(Y,$\mathcal{F}$), $K^{(0)}$, C, i_{max}
Output: M'(X,$\mathcal{E}$)

Compute curvatures and feature edges
$X \leftarrow Y$; $\mathcal{E} \leftarrow \mathcal{F}$; $K \leftarrow K^{(0)}$;
for $i \leftarrow 0$ **to** $i_{max} - 1$ **do**
  if $i \ mod \ 5 = 0$ **then**
    Regularize connectivity
  end
  for $X_s \in X$ **do**
    $\sigma_{freedom} \leftarrow \frac{0.5}{1+e^{-K}}.||X_s, X_{N_s}||_g.min(1, \rho_{max})$
    $X_s^{new} \leftarrow$ first with $\Delta_s U < 0$ and in sphere, out of the following: { angle based smooth. / Laplacian smooth. / guided uniform / random }
  end
  $X \leftarrow$ globally_optimize_QPBO-P (X, X^{new}, Y)
  $K \leftarrow K \cdot C$
end

3 The global energy minimization problem

In this work the initial mesh M is copied and kept as reference geometry during the remeshing process. The optimized mesh M' starts with the same vertices and connectivity as M. All vertex positions Y of M remain constant during the optimization process, but the set X may evolve (addition/removal of vertices) since the initial sampling may not be the one which minimizes the objective function (4) given below, which depends both on the vertex positions and on the mesh connectivity. To each vertex of the remeshed model we keep a link to the closest position of the original mesh, which allows us to keep track of the geometric distance between a vertex of the optimized mesh and the surface represented by the original mesh (cf. figure 2).

3.1 Definition of the objective function

The specific form of our objective energy function $U(X;Y)$ is defined as a scalar combination of sums of positive and unitless energy potentials. Each potential, which we also call *feature function*, locally measures a criteria (geometric error or a quality) and decreases (resp. increases) when the criteria is locally more (resp. less) respected. Unitless energy potentials have the advantage of being scale invariant.

$$\begin{aligned} U(X;Y) = \lambda_s & \sum_{(s,r,q)\in T} \psi_s(X_s, X_r, X_q) && \text{Shape} \\ + \lambda_d & \sum_{(s,r,q)\in T} \psi_d(X_s, X_r, X_q, Y) && \text{Fidelity} \\ + \lambda_v & \sum_{s} \psi_v(X_s) && \text{Valence} \\ + \lambda_p & \sum_{s} 1 && \text{Penalty} \end{aligned} \quad (4)$$

Here λ_s, λ_d, λ_v and λ_p are positive scalar weights. The respective label subscripts s, d, v, and p denote shape quality, data fidelity, valence quality, and vertex penalty, respectively. The other symbols have been defined in section 2.1.

$U(X;Y)$ evaluates the global configuration of the optimized mesh, and assigns a scalar energy value to each possible solution, i.e. each possible result mesh. To make the equation easier to read, we abusively simplified the dependency on the mesh connectivity in the notation; in particular, there are no variables associated to the connectivity information. It should nevertheless

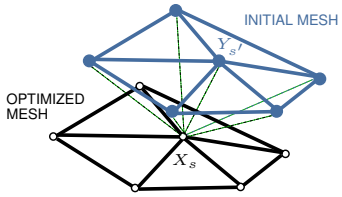


Fig. 2 Representation of the graph used for constructing the remeshed model: each vertex of the optimized mesh X_s is linked to its current closest initial mesh vertex $Y_{s'}$ and has access to the whole initial mesh if needed.

be clear that the *energy function depends on the vertex positions as well as the mesh connectivity.*

As a consequence, at each vertex relocation or at each mesh connectivity modification, $U(X;Y)$ may vary. By setting the parameters λ, a user can efficiently and easily create meshes with the desired properties.

3.2 The feature functions

Our model contains four different kinds of positive feature functions: a shape function measuring the quality of the mesh triangles, a data attached function encoding the approximation quality of the new surface, a valence potential function encoding the quality of the mesh connectivity, as well as a term penalizing a high vertex budget, which allows the user to control the number of vertices of the resulting mesh.

The role of the first type is to favor equilateral mesh triangles, therefore it is calculated on triangles:

$$\psi_s(X_s, X_r, X_q) = \frac{R(X_s, X_r, X_q)}{\min(||X_s - X_r||, ||X_s - X_q||, ||X_r - X_q||)} \tag{5}$$

where $R(X_s, X_r, X_q)$ denotes the circumradius associated with the triangle (X_s, X_r, X_q) and $||.||$ is the usual Euclidean norm. Note that this feature function does not depend on the initial vertices Y and that it is scale invariant. $\psi_s(X_s, X_r, X_q)$ can be extended to take the value $+\infty$ when the triangle (X_s,X_r,X_q) is degenerated (singularity of the function). The minimum of this feature function for one triangle is reached when the triangle is equilateral.

The data attached feature function $\psi_d(X_s, X_r, X_q, Y)$ measures the geometric error between a triangle (X_s, X_r, X_q) of the optimized model and the initial mesh. Ideally, ψ_d should be equal to the absolute volume error produced by a local operation (either vertex repositioning or topological operation on edges). However, exact volume error computations may add extra cost to the whole iterative energy minimization process, especially for the vertex repositioning step, where we calculate two different candidate positions for each vertex (see section 4), which gives eight possibilities to check for each triangle. We therefore calculate the exact direct volume error only during the optimization of mesh connectivity. In particular, in the latter case the tetrahedron volume error is calculated over the subsequent local edge operations (and not between the optimized and initial meshes for computation time purposes). More precisely, the volume error associated with an edge flip, an edge collapse or an edge split are, respectively, the tetrahedron volume induced by the 4 points of the 2 adjacent triangles, the sum of the tetrahedron volumes induced by each one-ring neighboring edge and its adjacent old and new center vertex position, and zero.

During the optimization of the vertex positions, we approximate the fidelity term ψ_d by a point-to-surface distance, whose main advantage is that it can be calculated vertexwise:

$$\psi_d(X_s, X_r, X_q, Y) = F(X_s, Y) + F(X_r, Y) + F(X_q, Y) \tag{6}$$

where $F(X_s, Y)$ is the square of the shortest distance between X_s and the initial mesh, i.e. the geometric distance. The distance F could be approximated up to second order with the Pottmann distance [20,21], which allows to directly compute the gradient of the energy. Since this is not necessary in our discrete framework, a direct computation of F as the orthogonal projection distance on the initial mesh was possible, which is slightly faster and more robust in the presence of sharp features.

The proposed $\psi_d(X_s, X_r, X_q, Y)$ approximation works very well provided that the weighting factor λ_d is high, which is the case for the targeted applications, namely regularization where geometric fidelity must be high.

The valence potential function is calculated on optimized mesh vertices:

$$\psi_v(X_s) = (|N_{opt}| - |N_s|)^2 \tag{7}$$

where $|N_s|$ is the number of edges adjacent to vertex s, i.e. the vertex degree, and $|N_{opt}|$ is the optimal (desired) vertex degree: 4 on borders and 6 otherwise.

The vertex budget energy cost is linear in the number of remeshed model vertices, and as can be seen from equation (4), each vertex costs λ_p. In general, λ_p controls the desired amount of vertices.

4 Minimization of the objective function

The next two subsections describe how the unique global energy function is minimized through two dif-

ferent steps, which update the vertex positions and the mesh connectivity.

4.1 Global vertex repositioning

Updating the vertex positions of the current mesh requires minimizing equation (4) over all variables $X_s \in \mathbb{R}^3$, a continuous optimization problem. The energy potentials associated with the vertex valence and with the vertex budget are constant at this stage and therefore omitted.

Unfortunately, the function $U(X;Y)$ is not convex and standard gradient descent methods will most likely return a sub-optimal solution. In addition, a least-square or linear system approach will return an over-smooth mesh, even if some constraints are added. This suggests a discrete approach for obtaining high fidelity meshes. Our work benefits from recent advances in optimization theory for discrete Markov Random Fields (MRFs) [19] by transforming the continuous problem into a discrete problem, similar to the technique proposed for optical flow by Lempitsky et al. [22]. However, instead of employing a global discrete optimizer to merge several solutions obtained by existing techniques applied with different parameters, in our case the global discrete optimizer takes decisions on candidates calculated at each step in an iterative process. At each iteration and for each vertex of the remeshed model, a new candidate position is proposed and the optimal decision for the whole set of vertices is calculated, i.e. the decision minimizing (4). In the following two subsections 4.1.1 and 4.1.2, we explain how we generate new candidates and how we globally decide to keep or not a new position.

4.1.1 Local candidate proposals

We force each valid candidate position to stay within a small freedom sphere to prevent too much geometric error and to avoid creating a geometric fold-over. A new vertex candidate position will be rejected if it is outside of the corresponding freedom sphere, leaving the current vertex position unchanged. The more the region around a vertex X_s is curved, the smaller its freedom radius $\sigma_{freedom}$ is forced to be, which avoids large moves around a point of high curvature and thus limits the introduced geometric error. Moreover, like the stepwidth in gradient descent, this radius decreases with time to avoid big moves at the end of the optimization process. Similarly to simulated annealing techniques [18], a temperature parameter K decreases at each iteration (introduced in the algorithm 1 and in section 5.1). The radius $\sigma_{freedom}$ is related to this temperature K through a sigmoidal function, as well as to the local geodesic radius of the X_s one-ring, and to the local maximal curvature radius measured on the initial mesh:

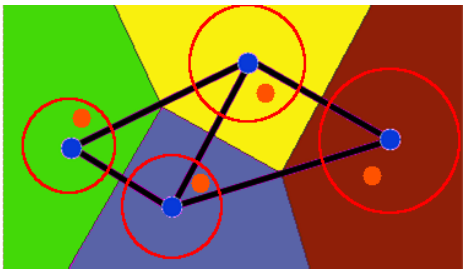


Fig. 3 One vertex relocation step: take the best global decision for each vertex between the current (blue) and the new candidate position (orange) located in the vertex freedom sphere.

$$\sigma_{freedom}(X_s, X_{N_s}, Y) = \frac{0.5}{1+e^{-K}}.||X_s, X_{N_s}||_g.min(1, \rho_{max}) \tag{8}$$

where ρ_{max} is the maximum absolute curvature radius at the closest initial mesh vertex $Y_{s'}$ from X_s ($\sigma_{freedom}$ thus depends on Y). $||X_s, X_{N_s}||_g$ is the maximum radius such that the sphere centered on X_s with that radius does not intersect the X_s one-ring neighborhood of edges. In other words, it is the local geodesic radius, computed as the minimum Euclidean distance from X_s to its one-ring neighborhood of edges:

$$||X_s, X_{N_s}||_g = \sup_{\rho \in \mathbb{R}^+} \left\{ \begin{array}{l} \rho : \rho < ||X_s, u|| \\ \forall u \text{ on } (r,q) \in \mathcal{E}, \\ \forall r \in N_s, \forall q \in N_s \end{array} \right\} \tag{9}$$

It is easy to see that a freedom radius strictly less than $0.5||X_s, X_{N_s}||_g$ for each optimized mesh vertex X_s prevents the creation of new fold-overs (see figure 3). For the same reason, the factor $min(1, \rho_{max})$ in equation (8) limits the radius to the value described above. Since the initial mesh is normalized (its coordinates are divided by its bounding box diagonal) before processing and unnormalized at the end, the diagonal of the mesh bounding box does not appear in the equation involving the curvatures radius.

A good candidate position X_s^{new} for replacing X_s must decrease the energy. The local energy variation $\Delta_s U$ due to the new vertex position X_s^{new} can be computed quickly from the vertex X_s and its one-ring neighborhood N_s. Because of the form of the energy function (4),

$$\Delta_s U = U\Big(\big(X \setminus \{X_s\}\big) \cup \{X_s^{new}\}; Y\Big) - U(X;Y) \tag{10}$$

can also be computed much quicker from only a small number of terms as

$$\Delta_s U = U\Big(\{X_s^{new}, X_{N_s}\}; Y\Big) - U\big(\{X_s, X_{N_s}\}; Y\big) \tag{11}$$

where the so-called *local evidence* $U(\{X_s, X_{N_s}\}; Y) = U(X;Y)_{|\{X_s,N_s\}}$ only contains the terms in $U(X;Y)$ which involve the modified mesh vertex X_s or the one ring X_{N_s}.

The global algorithm which jointly takes decisions on candidate pairs (i.e. keeping either X_s or X_s^{new}) for the whole mesh, is given below in section (4.1.2). The convergence of the algorithm depends on the quality of the new candidate positions, which leads us to choose one candidate for each vertex among several candidates calculated by different methods. They have been ranked empirically in experiments, and the chosen candidate is the first one satisfying the freedom sphere constraints given above, i.e. X_s^{new} must strictly be in the freedom sphere and $\Delta_s U < 0$, cf. (8), (10) and (11). If no such candidate is found, the old position is kept. The techniques are, in decreasing rate of empirically measured convergence, angle-based smoothing, Laplacian smoothing, uniformly taken and guided random candidates. They consist in computing a displacement vector $\overrightarrow{v}$ with different rules according to the method, from which the new candidate position X_s^{new} is set: $X_s^{new} = X_s + \overrightarrow{v}$.

For angle-based (resp. Laplacian) smoothing, $\overrightarrow{v} = \gamma\overrightarrow{v'}$ ($\gamma \in \mathbb{R}^+$), where $\overrightarrow{v'}$ is computed as the mean direction towards surrounding angle bisectors (resp. the umbrella-operator $U(X_s) = \frac{1}{\sum_{r\in N_s} z_r}\sum_{r\in N_s} z_r X_r - X_s$ where the weights z_r are set to the local area dispersion). For uniformly taken and random candidate positions, $\overrightarrow{v}$ has the global energy gradient direction for gradient-guided iterations and is a direction on the local tangent plane otherwise.

Special care is taken for vertex positions on feature edges, for which new candidates are forced to lie on the feature edge and which are kept only if the surrounding feature edge directions do not change with this new candidate. Vertices on corners remain unchanged.

4.1.2 Global candidate decisions

The global decision (i.e. keeping the current position or choosing the new candidate) on the whole set of vertices is taken by a graph cut technique. This involves constructing an *st*-graph representing the energy function (4) such that the minimum cut/maximum flow on this graph will give the solution which globally minimizes the energy. Although there are known graph cuts techniques able to optimize functions of some restricted classes where each variable may take values from sets larger than 2, the eventual gain in quality is far outweighed by the high computational complexity. We therefore concentrated on a single new candidate proposal for each vertex, which leads to binary valued functions, i.e. each variable associated to vertex may take values from a set of two values. For that purpose, it is necessary to associate a binary labeling to $\{X_s, X_s^{new}\}$, for instance 0 could denote the current vertex position X_s and 1 the proposed position X_s^{new}.

Unfortunately, the energy function (4) is not submodular, which makes its exact global minimization with graph cuts difficult [19]. Submodularity is an important concept in discrete optimization theory. Its meaning is the equivalent of the "convex" concept, restricted to discrete sets. This criterion requires that, taking one of the vertices of a given triangle (X_r, X_s, X_t) and fixing the decision to either 0 or 1, the projection onto the two other vertices satisfies the following constraint (without loss of generality we suppose that X_r has been fixed):

$$\begin{aligned}&\psi_s(X_r,0,0) + \psi_s(X_r,1,1) \leq \\ &\psi_s(X_r,0,1) + \psi_s(X_r,1,0)\end{aligned} \tag{12}$$

In the case of (4), this criterion is generally not satisfied for the shape term $\psi_s(X_s, X_r, X_q)$ of every triangle, although a subset of triangles may satisfy the criterion.

In the proposed method, (4) is approximatively minimized with a variant of QPBO (Quadratic Pseudo-Boolean Optimization), a graph cuts algorithm which can deal with non-submodular terms (see section 5.1). However, the output of QPBO is a partial labeling, i.e. for some binary variables, the algorithm does not know if it is better to choose zero or one. On the other hand, convergence is guaranteed since, starting from an initial configuration, QPBO guarantees that the new labeling does not increase the energy (robustness). For a detailed explanation of QPBO and its properties, see [23].

4.2 Iterative mesh connectivity optimization

Our goal is to optimize the remeshed model connectivity by minimizing the objective function (4). Unfortunately a global optimization of the connectivity is intractable — the optimal scheduling of local topological operations cannot be computed in a reasonable amount of time. Here, a greedy scheme has been adapted to allow control through the global energy function (4), i.e. a local topological operator will be applied only if it decreases the energy.

Three priority queues handle local topological operations, respectively for edge-flips, edge-splits and edge-collapses. Each priority queue only contains valid local operations, which do neither create a vertex fold-over, nor change the direction of a feature edge and which will decrease the energy if applied. The priority of each queue is directly related to the energy decrease.

Again, calculating the optimal order of operation types — and therefore of priority queues — is intractable. Experimental tests (and intuition) led to the choice of giving higher priorities to flips, then to splits, and last to collapses.

Each priority queue is mutable: after the application of a local operation, each neighboring edge that already was in the priority queue is updated, i.e. removed if the local operation is no more applicable or its priority is updated. To make sure that the connectivity optimization step will terminate in a finite number of local operations, we limit the maximum number of consecutive operations of the same topological operator.

Topological changes and their energy — edge collapses and splits change the number of triangles of the mesh and therefore also the number of terms of the energy function. As a consequence, a change of the energy function is not necessarily an image of a quality/fidelity change of the mesh, but related to differences in the amount of mesh simplices. The budget term controlled through weight λ_p has been designed to compensate for this. Its role is actually two-fold:

- To compensate for the lost terms in topological changes, leading to roughly unchanged vertex counts before and after optimization, if λ_p has been judiciously chosen.
- To control the vertex budget, if desired. A mesh simplification algorithm producing high quality meshes is obtained simply by increasing λ_p.

5 Experimental results

In order to demonstrate the efficiency of our method, we applied it to several mesh models with unadapted sampling and very irregular connectivity, and which contain both, smooth parts and sharp features. Some visual results are given in figures 4, 5, 6, 7 and 8. The presented results have been obtained on an Intel Core 2 Duo P8400 (2.26 GHz) with 4 GB RAM with running time between 31 s and 3 min 56s. About 50% (resp. 50%) of the total running time is used for the connectivity optimization (resp. vertex repositioning). Clearly, the complexity of the connectivity optimization is linear in the number of edges of the input mesh, while the complexity of vertex repositioning is directly related to the complexity of the chosen graph-cut technique. The selected graph-cut technique is based on a push-relabel version of the max-flow algorithm with polynomial complexity in the worst case. Fortunately, the empirical (or average) complexity is nearly-linear with respect to the size of the input [24]. The average complexity of our method is nearly-linear with respect to the size of the input mesh (e.g. number of vertices, edges and triangles), which has been confirmed by our experiments (cf. figure 11).

5.1 Implementation details and settings

The temperature and cooling parameters mentioned in section 4.1.1 and detailed in algorithm 1 allow to decrease the radius of the freedom sphere associated with a given vertex when the number of iterations/the elapsed time increases. Their names are borrowed from physics, more precisely from the solidification of a melting metal. For the cooling schedule we used the suggestions in [25] (page 356), setting the temperature K to $K^{(i)} = K^{(0)} \cdot C^i$ where i denotes the current iteration ($0 \leq i \leq i_{max} - 1$). The initial temperature $K^{(0)}$ has been set to 100 and the constant controlling the speed of the cooling process C has been set to 0.95. The chosen total number of iterations of our algorithm is $i_{max} = 170$: a higher number of iterations does not improve significantly the results while adding extra computation time on the presented models (cf. figure 12). If the energy gradient can be computed for the vertex optimization step, the last 20 iterations are gradient-guided which tends to fine-tune the final result.

Concerning the detection of robust features (cf. figure 13), we experimentally set $\theta_{Th} = 35^o$ and $\mu = 0.1$. We also experimentally fixed the quality/fidelity trade-off of our method by setting the scalar weighting factors of the objective function with the following values: $\lambda_s = 1$, $\lambda_d = 10^5$, $\lambda_v = 0.1$ and $\lambda_p = 1$. For candidate proposals in the vertex repositioning stage, γ is set to 0.1 for angle-based or Laplacian candidates.

In the connectivity optimization step, the maximum number of local topological change iterations is set to 10.

For the vertex repositioning step and its related energy minimization, we used the implementation of QPBO-P given in Rother et al. [26], which is available online[1]. We also performed experiments with other approximative graph cut techniques designed for non-submodular functions, namely classical QPBO and QPBO-I (both introduced in [26]) and energy truncation to create a submodular function [27]. The results were slightly lower with these alternative techniques. Belief Propagation (BP) [28] is also an alternative to graph cuts techniques like QPBO-P. While both types of methods give approximate results only on this type of energy functions, their strengths and weaknesses are different. BP can be applied to non-submodular functions on multiple labels, it's complexity being $O(|\mathcal{L}|^{|C|})$,

[1] http://www.cs.ucl.ac.uk/staff/V.Kolmogorov/software.html

where $\mathcal{L}$ is the set of labels and $|C|$ the maximum clique size (3 in our case). The exactness of BP depends on the absence of cycles in the graph, whereas the exactness of QPBO-P depends on the submodularity of the energy function. Our choice towards QPBO-P, rather than towards BP, is motivated by the fact that QPBO-P gives an exact solution if the function is submodular, and high quality solutions if the number of non-submodular terms is low, which is the case in our problem, whereas the number of cycles is extremely high. Furthermore, nodes for which QPBO-P is able to find a solution are optimal (part of the exact solution of the problem). In our problem, at each step more than 99% of the vertex repositioning decisions are globally optimal, which gives in practice excellent results, in addition to the complexity advantages of the method. In addition, QPBO-P guarantees that the global energy does not increase after each vertex repositioning step. This last point is no more guaranteed with the BP in the presence of cycles, even the convergence of the energy minimization is not guaranteed in that case [29].

Figure 12 shows the evolution of the global energy during 170 steps for a single CAD model.

5.2 Discussion

To illustrate the mesh quality, average triangle minimum and maximum angles are presented in table 1. Large angles cause discretization errors and large errors in interpolated derivatives while small angles are responsible for poor conditioning [2]. For high-quality meshes, the average minimum (resp. maximum) angle should be greater than 30^o (resp. less than 90^o). The closer to 60^o the mean minimum and maximum angles are, the better are the results. According to table 1 and figure 8 and regarding the mean minimum and maximum triangle angles, regularized meshes obtained by our method are high-quality meshes. Moreover, figures 4, 5, 6 and 8 confirm this. Note that the regularized mesh vertex sampling is similar to the original mesh vertex sampling. That is convenient for regularizing meshes for which the initial vertex sampling must be preserved. Table 1 also shows that the vertex valence is slightly improved by our method, i.e. when it does not penalize neither triangle shape nor geometric fidelity. The number of irregular vertices can be significantly reduced by increasing λ_v (e.g. to 0.6). However, the improvement in valence can degrade the fidelity to the original surface and may need additional global iterations to keep the mean triangle quality high.

To evaluate the surface fidelity of the remeshed models, the Hausdorff distance and the maximum of the two RMS (Root Mean Square) distances normalized to the bounding box diagonal are presented in table 1. These distances have been obtained using the Metro tool [30]. According to these distances, to table 1 and to figure 8, the geometric error introduced by our method is small.

Our method preserves high frequency mesh features (cf. figure 7), while considerably improving triangle quality. Let's note that the number of vertices does not need to be chosen. It adapts itself to the geometry while maintaining the same order of magnitude, given the proposed setting of the vertex budget weight λ_p.

We compared our method to those of Valette et al. [6], Surazhsky et al. [11,12], and Liu et al. [14]. As can be seen in table 1, our method gives better results in terms of triangle shape and surface fidelity when compared to Valette et al. and Liu et al.; Surazhsky et al.'s methods generate more regular triangles (better mean min and max angles), but our method better approximates the original surface. For instance, in reference [11] their proposed method smoothes the triceratops eye (cf. figure 7) resulting in a significant loss of details.

Our method can deal with high genus (i.e. > 1) models (cf. figure 10), and thus avoids the stitching problem that occurs in parameterization-based approaches (e.g. [7]).

5.3 Mesh simplification

The proposed energy minimization framework allows to simplify meshes by simply changing the setting of the weight parameters, such that edge collapses are favored. This is achieved by setting the vertex penalty weight λ_p to higher values ($\lambda_p = 25$ in our simplification examples). Some simplified meshes are presented in figure 9. These results show a high fidelity to the original triangular surface mesh, while the number of vertices is significantly decreased. However, by favoring the removal of triangles, the average triangle quality is degraded.

In the same manner, it is possible to refine a coarse mesh by changing the vertex penalty weight λ_p to negative values. To avoid varying vertex density over the refined mesh, refinement steps should alternate with regularization steps.

6 Conclusion and future work

In this paper we have presented a method for mesh optimization which includes robust feature detection with an improved Potts model and an original way of computing vertex positions using global combination of lo-

Model	#v	Irreg (%)	Amin (deg)	Amax (deg)	Er_{Haus} (10^{-3})	Er_{RMS} (10^{-3})	Time (sec)
Fandisk (init)	6495	20	43.4	86.1	-	-	-
Fandisk (Liu)	6495	20	44.7	82.0	3.3	0.8	n/a
Fandisk (our)	5905	**12**	**49.0**	**75.9**	**1.6**	**0.03**	232
Cow (init)	2904	53	30.2	93.7	-	-	-
Cow (Liu)	2904	53	35.1	88.2	**5.3**	0.9	n/a
Cow (our)	2695	**39**	**41.0**	**81.0**	5.5	**0.5**	59
Shark (init)	2560	32	20.8	97.4	-	-	-
Shark (Liu)	2560	32	26.2	107.5	**3.0**	**0.3**	n/a
Shark (Sur1)	2560	**31**	**50.6**	**71.1**	6.8	0.8	n/a
Shark (our)	1719	47	36.2	84.8	4.0	0.6	42
Hand (init)	7950	58	32.4	94.1	-	-	-
Hand (Liu)	7950	58	34.3	92.2	8.8	0.4	n/a
Hand (Val)	6802	45	46.1	77.5	2.6	**0.2**	9
Hand (our)	5847	**33**	**50.2**	**72.3**	**1.7**	**0.2**	193
Bimba (init)	8857	62	34.2	92.8	-	-	-
Bimba (Liu)	8857	62	38.1	87.0	4.9	0.5	n/a
Bimba (Sur1)	8857	**20**	**53.6**	**67.6**	6.0	0.5	n/a
Bimba (Val)	8143	48	45.2	78.1	6.0	0.4	10
Bimba (our)	7986	41	47.6	75.3	**3.0**	**0.2**	232
Egea (init)	8268	75	34.7	93.5	-	-	-
Egea (Liu)	8268	75	38.2	88.3	**2.6**	**0.2**	n/a
Egea (Sur2)	8705	**7**	**52.4**	**69.1**	2.7	**0.2**	15
Egea (our)	7783	43	48.8	74.1	**2.6**	**0.2**	236
Triceratops (init)	2832	59	29.6	95.5	-	-	-
Triceratops (Sur2)	2758	**13**	**42.2**	82.5	8.4	1.1	12
Triceratops (our)	2412	44	41.5	**81.0**	**3.6**	**0.5**	55

Table 1 Statistics on the remeshed models: number of vertices, percentage of irregular vertices, mean minimal angle, mean maximal angle, Hausdorff distance, maximum between the 2 RMS distances measured by Metro normalized to the bounding box diagonal, and running time. Liu, Val, Sur1, and Sur2 correspond respectively to [14], [6], [12], and [11]. Displayed times for Sur2 have been computed on a Pentium 4 PC (2.4 GHz) with 512 RAM [11], while others are from an Intel Core 2 Duo P8400 (2.26 GHz) with 4 GB RAM.

cally proposed candidates. Its main advantages are its feature sensitiveness and its ability to improve triangle shapes while preserving the original surface fidelity. The obtained results are better than other methods in terms of surface fidelity and surface fidelity/triangle quality trade-off. Our method is quite general, because by setting well-chosen weights in our objective function, a user can improve the vertex valence, improve the compactness of the representation or improve the quality of the triangles.

As future work, we will investigate quadrangular and anisotropic remeshing. We will also tackle combinatorial optimization (when computationally tractable) in the connectivity processing to improve mesh connectivity configuration. The robustness against local shape variations in the edges detection process will be improved by setting θ_{Th} automatically (using statistics on geometrical measures) in equation 2, rather than by asking the end-user.

Acknowledgments We wish to thank Vitaly Surazhsky and Ligang Liu for their kind help in applying their respective remeshing methods on our 3D models and for sharing their results with us.

References

1. Hoppe H., DeRose T., Duchamp T., McDonald J., Stuetzle W.: Mesh optimization. In SIGGRAPH'93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques (1993).
2. Shewchuk J. R.: What is a good linear element? Interpolation, conditioning, and quality measures. In 11th International Meshing Roundtable (2002), pp. 115–126.
3. Alliez P., Ucelli G., Gotsman C., Attene M.: Recent Advances in Remeshing of Surfaces. In Springer Berlin Heidelberg (2008).
4. Cohen-Steiner D., Alliez P., Desbrun M.: Variational shape approximation. In ACM Siggraph (2004), pp. 905–914.
5. Valette S., Chassery J.-M.: Approximated centroidal voronoi diagrams for uniform polygonal mesh coarsening. Computer Graphics Forum (Eurographics 2004 proceedings) 23, 3 (2004), pp. 381–389.
6. Valette S., Chassery J.-M., Prost R.: Generic remeshing of 3d triangular meshes with metric-dependent discrete voronoi diagrams. IEEE Trans Visu Comp Grap 14, 2 (2008), pp. 369–381.
7. Alliez P., Verdière E.C.D , Devillers O., Isenburg M.: Isotropic surface remeshing. In IEEE Shape Modeling International (2003), pp. 49–58.
8. Guskov I., Khodakovsky A., Schrder P., Sweldens W.: Hybrid meshes: multiresolution using regular and irregular refinement. In ACM SIGGRAPH Symposium on Computational geometry (2002), pp. 264–272.
9. Sifri O., Sheffer A., Gotsman C.: Geodesic-based surface remeshing. In 12th International Meshing Roundtable (2003), pp. 189–199.

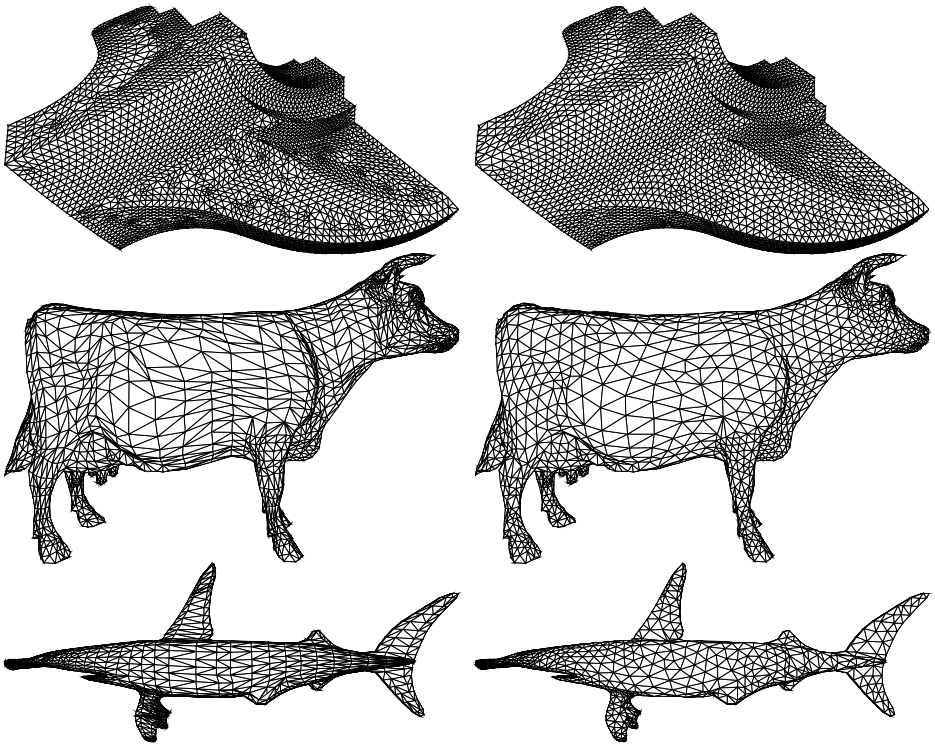


Fig. 4 Results obtained with our method. From top to bottom: fandisk, cow, shark. Left: input model, right: remeshed model.

10. Peyré G., Cohen L.: Geodesic remeshing using front propagation. International Journal of Computer Vision 69, 1 (2006), pp. 145–156.
11. Surazhsky V., Gotsman C.: Explicit surface remeshing. In Eurographics/ACM SIGGRAPH symposium on Geometry processing (2003), pp. 20–30.
12. Surazhsky V., Alliez P., Gotsman C.: Isotropic remeshing of surfaces: a local parameterization approach. In 12th International Meshing Roundtable (2003), pp. 215–224.
13. Winkler T., Hormann K., Gotsman C.: Mesh massage: A versatile mesh optimization framework. Visual Computer 24, 7 (2008), pp. 775–785.
14. Liu L., Tai C.-L, Ji Z., Wang G.: Non-iterative approach for global mesh optimization. Computer Aided Design 39, 9 (2007), pp. 772–782.
15. Nealen A., Igarashi T., Sorkine O., Alexa M.: Laplacian mesh optimization. In ACM Graphite (2006), pp. 381–389.
16. Cohen-Steiner D., Morvan J.: Restricted delaunay triangulations and normal cycle. In ACM SIGGRAPH Symposium on Computational geometry (2003), pp. 312–321.
17. Jiao X., Heath M. T.: Feature detection for surface meshes. In Proceedings of 8th International Conference on Numerical Grid Generation in Computational Field Simulations (2002), pp. 705–714.
18. Geman S., Geman D.: Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. IEEE Transactions on Pattern Analysis and Machine Intelligence 6, 6 (11 1984), pp. 721–741.
19. Kolmogorov V., Zabih R.: What energy functions can be minimized via graph cuts? IEEE Transactions on Pattern Analysis and Machine Intelligence 26, 2 (2004), pp. 147–159.
20. Pottmann H., Hofer M.: Geometry of the squared distance function to curves and surfaces. In Visualization and Mathematics III (2003), Hege H.-C., Polthier K., (Eds.), Springer, pp. 223–244.
21. Wang W., Pottmann H., Liu Y.: Fitting B-spline curves to point clouds by curvature-based squared distance minimization. ACM Transactions on Graphics 25, 2 (2006), pp. 214–238.
22. Lempitsky V., Roth S., Carsten R.: Fusionflow: Discrete-continuous optimization for optical flow estimation. IEEE Conference on Computer Vision and Pattern Recognition (2008), pp. 1–8.
23. Kolmogorov V.,Rother C.: Minimizing nonsubmodular functions with graph cuts-a review. IEEE Transactions on Pattern Analysis and Machine Intelligence 29, 7 (2007), pp. 1274–1279.
24. Boykov Y.,Kolmogorov V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. IEEE Transactions on Pattern Analysis and Machine Intelligence 26, 9 (2004), pp. 1124–1137.
25. Duda R., Hart P., Stork D.: Pattern Classification, 2nd Edition. Wiley, New York, NY, Nov. 2000.
26. Rother C.,Kolmogorov V.,Lempitsky V.,Szummer M.: Optimizing binary MFRs via extended roof duality. IEEE Conference on Computer Vision and Pattern Recognition (2007), pp. 1–8.
27. Rother C., Kumar S., Kolmogorov V., Blake A.: Digital tapestry. IEEE Conference on Computer Vision and Pattern Recognition, 1 (2005), pp. 589–596.
28. Pearl, J. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufman, San Mateo, 1988.
29. Szeliski R.,Zabih R.,Scharstein D.,Veksler O.,Kolmogorov V., Agarwala A., Tappen M., Rother C.: A comparative study of energy minimization methods for markov random fields with smoothness-based priors. IEEE Transactions on Pattern Analysis and Machine Intelligence 30, 6 (2008), pp. 1068–1080.
30. Cignoni P., Rocchini C., Scopigno R.: Metro: Measuring error on simplifed surfaces. Computer Graphics Forum 17, 2 (1998), pp. 167–174.

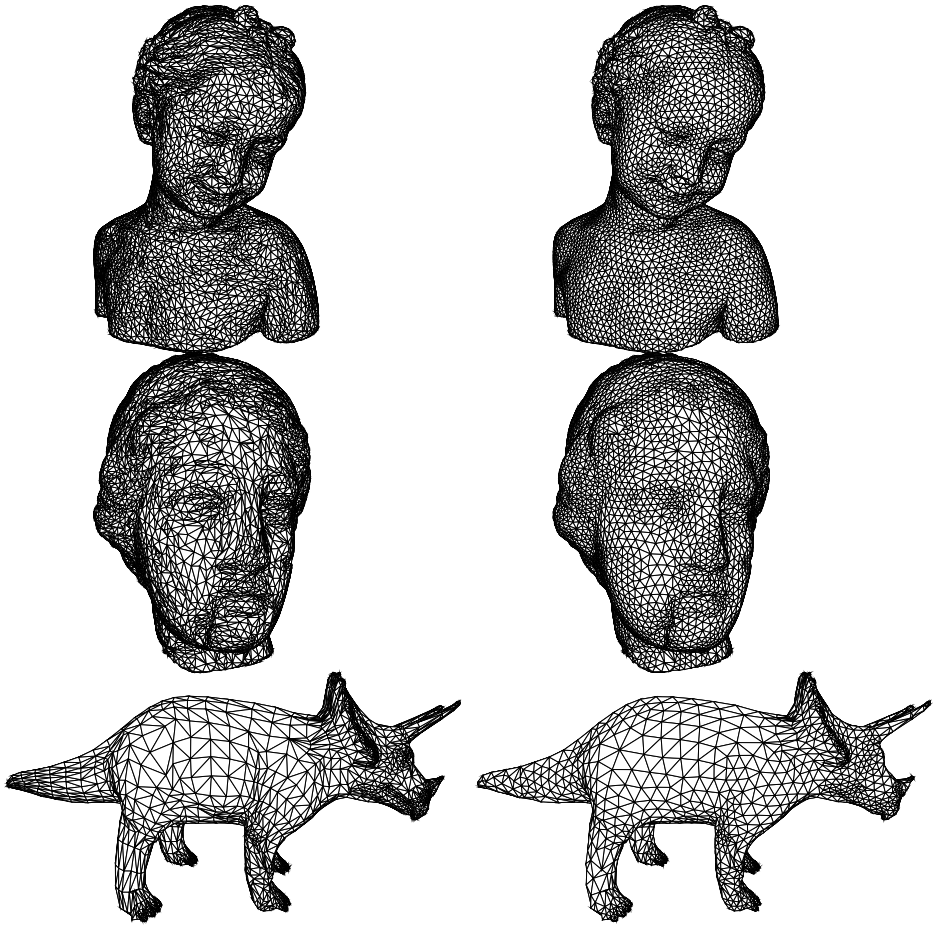


Fig. 5 Results obtained with our method. From top to bottom: bimba, egea, triceratops. Left: input model, right: remeshed model.

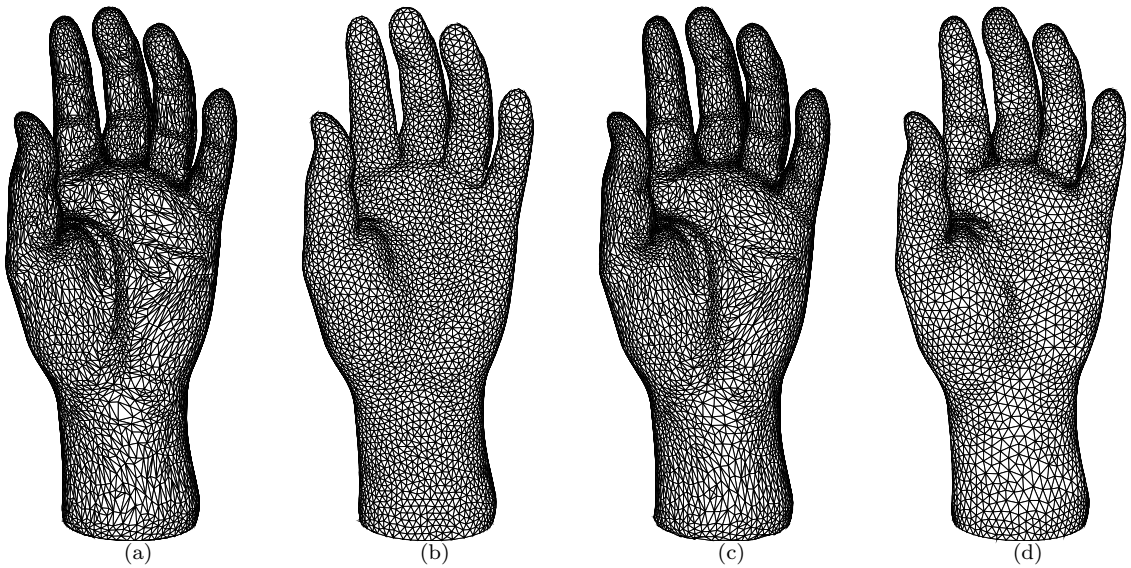


Fig. 6 Comparisons between (a) the original hand model, (b) Valtette et al. [6] (c) Liu et al. [14] and (d) our method.

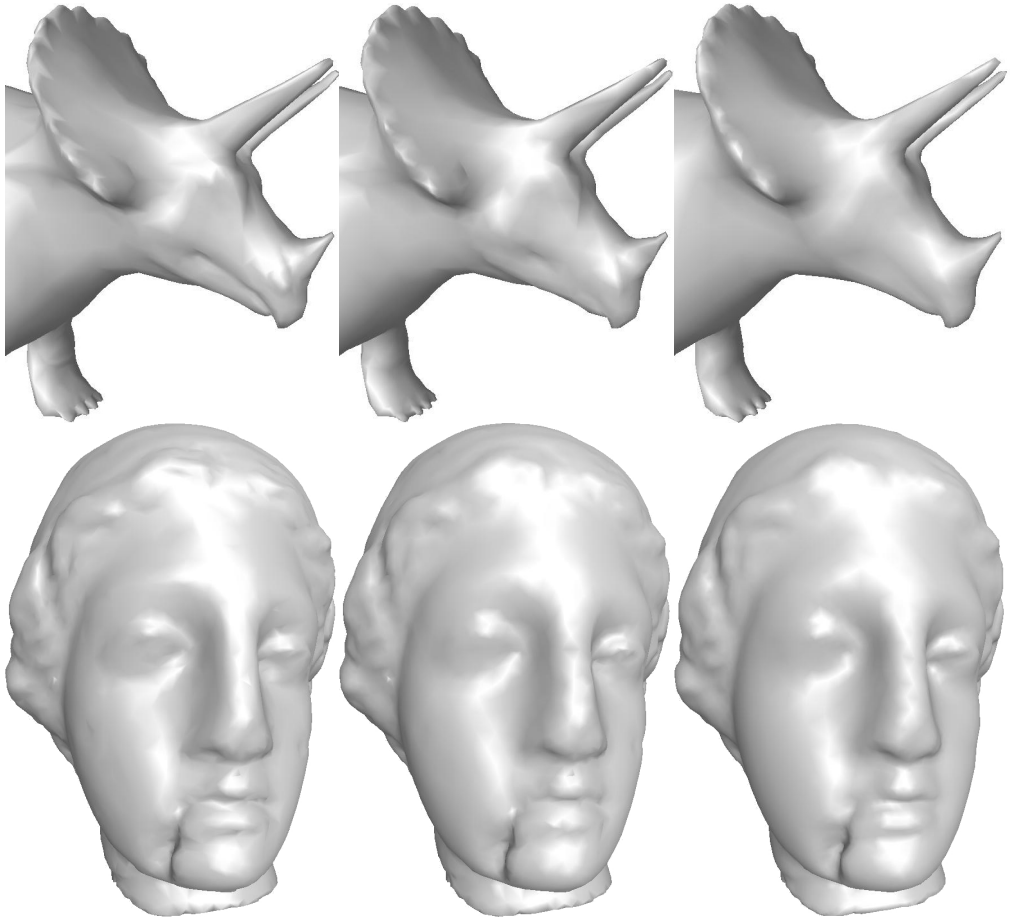
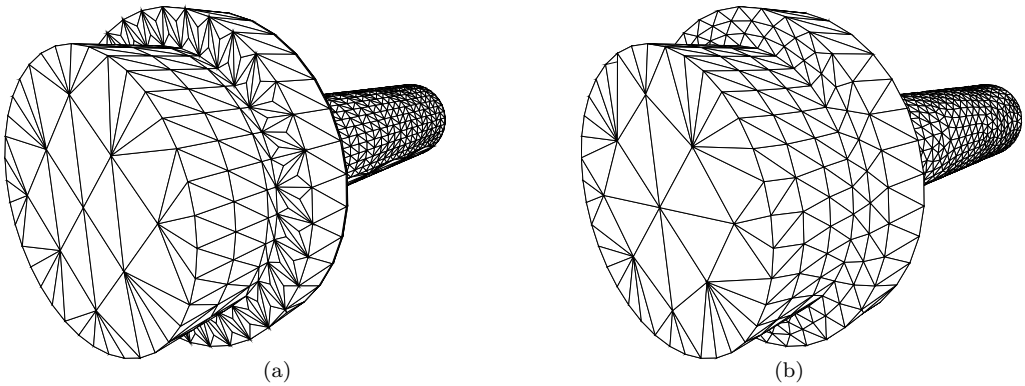


Fig. 7 Comparisons for the triceratops model (top) and the egea model: the original (left), our remeshed version (middle), the Surazhsky and Gotsman [11] remeshed version (right). The features surrounding the triceratops eye are well-preserved by our method.



(a)

#v	1243
Irreg (%)	46
Amin (deg)	25.1
Amax (deg)	95.4
Er_{Haus} (10^{-3})	-
Er_{RMS} (10^{-3})	-

(b)

#v	1157
Irreg (%)	30
Amin (deg)	40.5
Amax (deg)	81.1
Er_{Haus} (10^{-3})	1.7
Er_{RMS} (10^{-3})	0.2

Fig. 8 Comparisons between **(a)** the original CAD model and **(b)** our remeshed version. Note that we obtain a better Amin and Amax angle convergence towards 60^o. A lower vertices number denotes an optimized triangle distribution since less triangles are required to cover the same surface while introducing an insignificantly small geometric error.

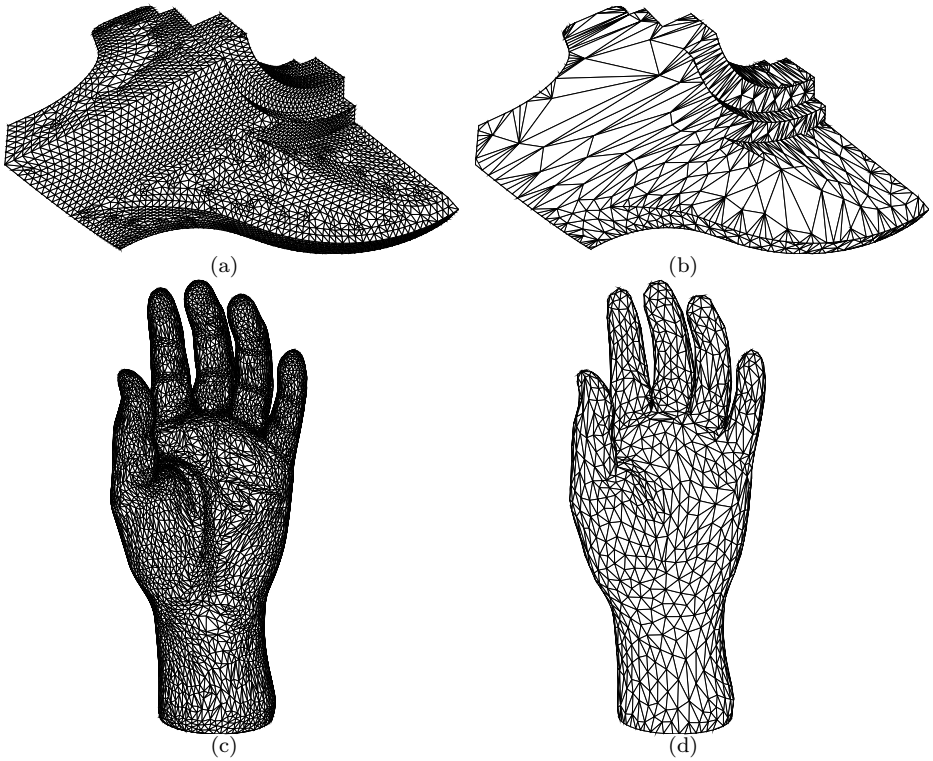


Fig. 9 Results obtained for simplification: (a) original fandisk model; (b) simplified fandisk model (933 vertices; Er_{Haus} (10^{-3}) = 3.0; Er_{RMS} (10^{-3}) = 0.2); (c) original hand model; (d) simplified hand model (1518 vertices; Er_{Haus} (10^{-3}) = 4.5; Er_{RMS} (10^{-3}) = 0.7).

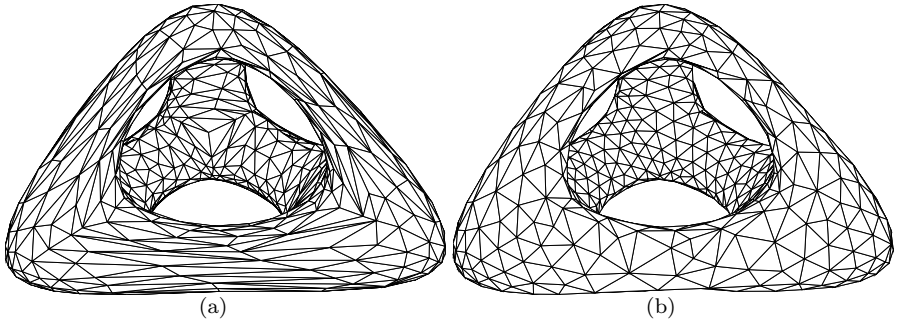


Fig. 10 Results obtained for a genus 3 model: **(a)** the original genus 3 model and **(b)** our remeshed version.

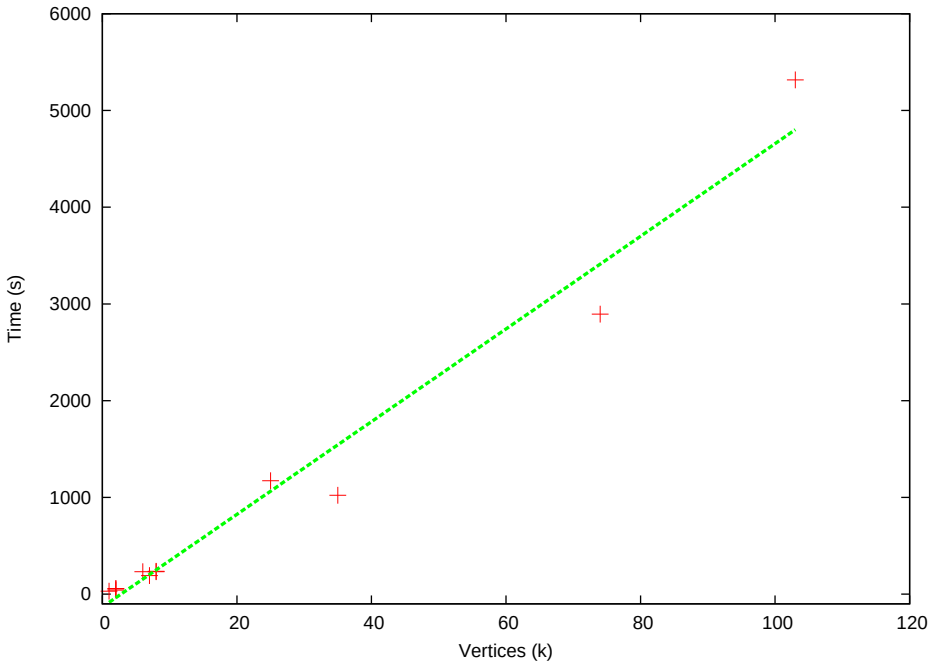


Fig. 11 Empirical complexity of our method: the 8 leftmost (resp. 4 rightmost) points reprensent the running times for our data set (resp. 4 additional meshes obtained by one or two triangle subdivisions of the fandisk, Bimba and Egea models). The dashed line is a linear least square fitting of the points.

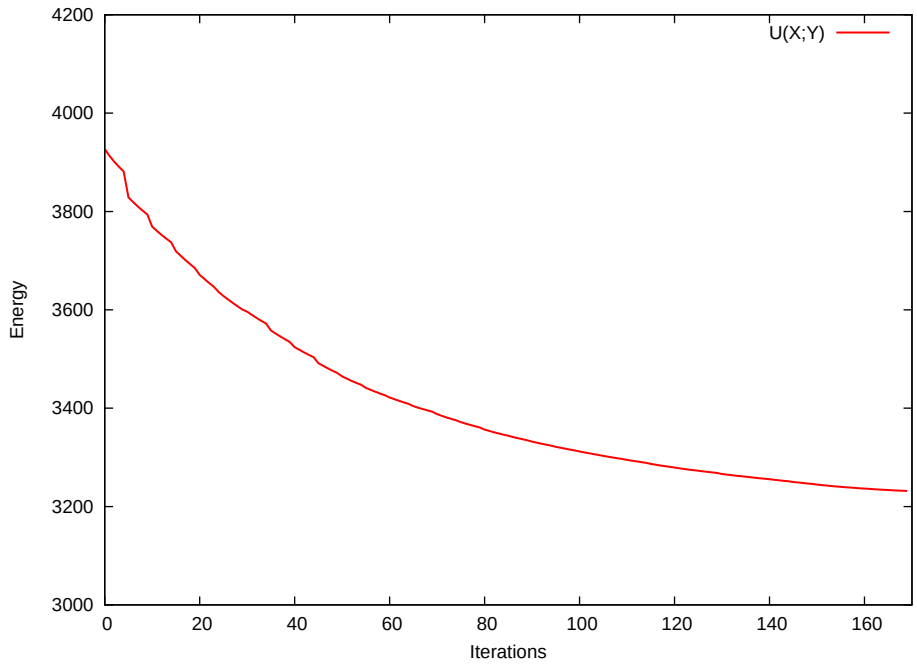
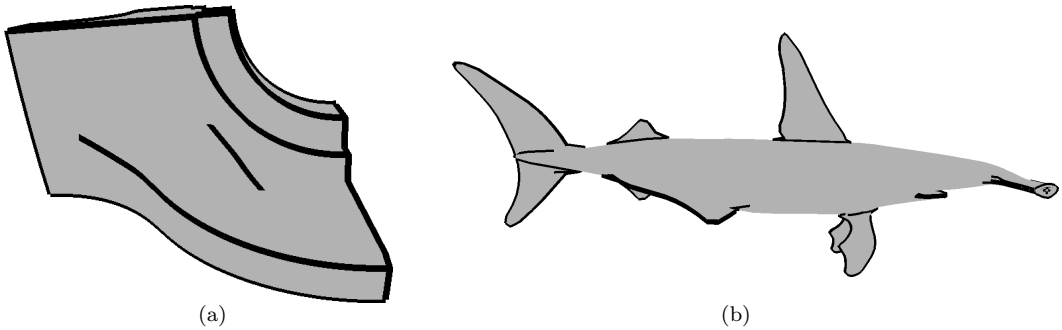


Fig. 12 Evolution of the global energy (cf. equation 4) during 170 iterations for the CAD model presented in figure 8.



(a) (b)

Fig. 13 Extracted feature edges from **(a)** the fandisk model and from **(b)** the shark model. Feature edges (in black) are preserved during our mesh optimization algorithm (see figure 1).