Engineering with Computers (2014) 30:289–300
DOI 10.1007/s00366-012-0291-9

ORIGINAL ARTICLE

Fitting polynomial surfaces to triangular meshes with Voronoi squared distance minimization

Vincent Nivoliers · Dong-Ming Yan · Bruno Lévy

Received: 14 March 2012 / Accepted: 4 October 2012 / Published online: 6 November 2012

Abstract This paper introduces Voronoi squared distance minimization (VSDM), an algorithm that fits a surface to an input mesh. VSDM minimizes an objective function that corresponds to a Voronoi-based approximation of the overall squared distance function between the surface and the input mesh (SDM). This objective function is a generalization of the one minimized by centroidal Voronoi tessellation, and can be minimized by a quasi-Newton solver. VSDM naturally adapts the orientation of the mesh elements to best approximate the input, without estimating any differential quantities. Therefore, it can be applied to triangle soups or surfaces with degenerate triangles, topological noise and sharp features. Applications of fitting quad meshes and polynomial surfaces to input triangular meshes are demonstrated.

Keywords Squared distance minimization · Centroidal Voronoi tessellation · Subdivision surface fitting

V. Nivoliers · D.-M. Yan · B. Lévy
Project ALICE/Institut National de Recherche en Informatique et en Automatique (INRIA) Nancy Grand-Est, LORIA, Nancy, France
e-mail: yandongming@gmail.com

B. Lévy
e-mail: bruno.levy@inria.fr

V. Nivoliers (✉)
Institut National Polytechnique de Lorraine (INPL), Nancy, France
e-mail: vincent.nivoliers@loria.fr

D.-M. Yan
Geometric Modeling and Scientific Visualization Center, King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia

1 Introduction

In this paper, our goal is to transform an input mesh $\mathcal{T}$ into an alternative representation $\mathcal{S}$ (see Fig. 1). In our setting, the result $\mathcal{S}$ may be a mesh made of quadrangular elements or a subdivision surface. We suppose that an initial position of $\mathcal{S}$ is given (referred to as a *template*). The template may be initialized from the bounding box of the input mesh $\mathcal{T}$, as in Fig. 1, but more complicated input geometries may require more general templates, as shown further. The problem reduces then to deforming the template $\mathcal{S}$ to fit the input mesh $\mathcal{T}$, by finding a new position for the vertices of $\mathcal{S}$ (or for its control points in the case of a subdivision surface).

As in several approaches, we formalize the fitting problem as minimizing an objective function that measures the distance between the two surfaces $\mathcal{S}$ and $\mathcal{T}$. Our contribution is a new approximation of the overall squared distance between surfaces, based on the concept of centroidal Voronoi tessellations (CVT), as well as a solution mechanism to minimize it.

In contrast to previous approaches, our approximation offers theoretical guarantees on the error it introduces with respect to the exact squared distance function. Minimizing our objective function results in a good approximation of the input in terms of Hausdorff distance. Our method does not need to estimate any differential quantity such as curvatures on $\mathcal{T}$. This makes our method well suited when the target surface $\mathcal{T}$ is ill formed, with degenerate triangles and wrong or absent connectivity, as in range scan data.

We neither address here the problem of automatically generating the initial template mesh $\mathcal{S}$, nor dynamically modifying it to adapt to the target mesh $\mathcal{T}$. Although our method will not fail even in case of topology mismatch and can often be initialized with the bounding box of $\mathcal{T}$, a bad template may result in pinchouts and overlaps when $\mathcal{T}$

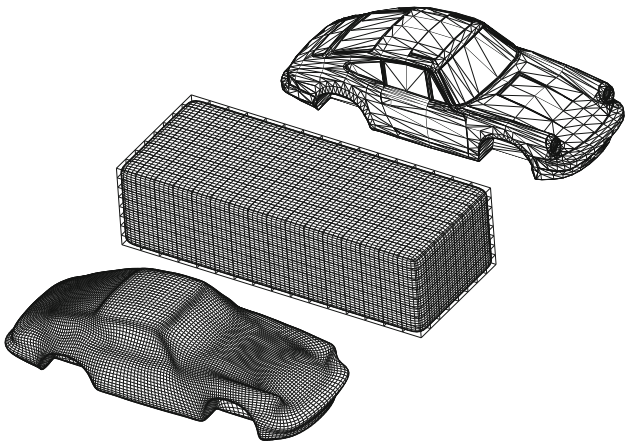


Fig. 1 Given an input mesh $\mathcal{T}$ (*top* 2,065 vertices and 4,114 facets) and a control mesh (898 vertices and 896 quads) in an initial position (*center*), VSDM minimizes the squared distance between $\mathcal{T}$ and the polynomial surface $\mathcal{S}$ defined by the control mesh. The Hausdorff distance between the result (*bottom*) and the input mesh (*top*) is 0.554% of the bounding box diagonal. Other views of the same data are shown further in the paper

contains large protrusions. Another limitation of our approach is that we do not prove the C^2 continuity of our objective function. Yet our minimization procedure empirically behaves well when using a quasi-Newton solver.

2 Background and previous work

2.1 Methods based on parameterization

Fitting splines was the motivation of early works in mesh parameterization for objects homeomorphic to a disc [10]. For fitting splines to objects of arbitrary genus, it is possible to use a parameterization defined over a base complex [9, 26], polycube maps [27, 28] or global parameterization methods [15, 24]. More details about these approaches is given in the survey about quad meshing in computer graphics [3]. The relations between the curvature of the surface and the metric defined by the parameterization is studied in [13] and used to compute an anisotropic mesh that minimizes the approximation error. Since they require the estimates of differential quantities (gradients, curvature, shape operator...), the methods above cannot be applied to meshes with degeneracies (skinny triangles, multiple components, holes, sharp creases). Our method that directly minimizes the squared distance does not suffer from this limitation.

2.2 Methods based on point-to-point distances

To remesh surfaces, "shrink-wrap" methods [7, 12] iteratively project the template onto the input mesh while minimizing a regularization criterion. A similar idea can be applied to subdivision surfaces [20, 21] using an exact algorithm to find closest points on the subdivision surface and the exact evaluation of the subdivision surface. The "dual domain relaxation" method [32] uses some variants of Laplace surface editing to fit a template to the input mesh. Since they are based on point-to-point distances, the methods above can mostly do small corrections on the geometry, and have difficulties converging when the initialization is far away from the target surface. In contrast, VSDM can successfully fit a control mesh to a surface.

2.3 Squared distance minimization (SDM)

SDM was proposed by Pottmann et al. [23] for curve and surface fitting. The SDM approach generalizes the *Iterated Closest Point* (ICP) methods initially introduced for surface registration. A survey and performance study of ICP methods was published in [25].

In the continuous setting, the SDM framework fits a surface $\mathcal{S}$ to another surface $\mathcal{T}$ by minimizing an approximation of the objective function $E(\mathcal{S})$:

$$E(\mathcal{S}) = F_{\mathcal{S}\to\mathcal{T}}(\mathcal{S}) + \lambda R(\mathcal{S}) \tag{1}$$

where:

$$F_{\mathcal{S}\to\mathcal{T}}(\mathcal{S}) = \int_{\mathcal{S}} \| \mathbf{x} - \Pi_{\mathcal{T}}(\mathbf{x}) \|^2 \, d\mathbf{x}$$

In this equation, $\Pi_{\mathcal{T}}(\mathbf{x})$ denotes the projection of $\mathbf{x}$ onto $\mathcal{T}$, i.e., the point of $\mathcal{T}$ nearest to $\mathbf{x}$. The term $R(\mathcal{S})$ is a regularization term aiming at preserving the smoothness of $\mathcal{S}$ by minimizing its Laplacian. The regularization factor λ lets the user choose a tradeoff between the smoothness of $\mathcal{S}$ and the fitting criterion.

Due to the difficulty of the exact computation of F, various approximations are usually introduced. First, the integral over $\mathcal{S}$ is replaced by a sum over a sampling $\mathbf{X} = [\mathbf{x}_i]_{i=1}^n$. Then the nearest neighbor of a sample $\mathbf{x}_i$ is replaced by the nearest sample $\mathbf{y}_j$ in a sampling $\mathbf{Y} = [\mathbf{y}_j]_{j=1}^m$ of $\mathcal{T}$. Using these two samplings, $F_{\mathcal{S}\to\mathcal{T}}$ can be computed in several ways, depending on how the distance between $\mathbf{x}_i$ and $\mathbf{y}_j$ is computed:

Point distance (PD) uses $\|\mathbf{x}_i - \mathbf{y}_j\|^2$ (order 0 approximation).
Tangent distance (TD) uses the squared distance to the nearest point on the tangent plane of $\mathcal{T}$ at $\mathbf{y}_j$ (order 1 approximation).
Second order squared distance (SD) uses the curvature information of $\mathcal{T}$ at $\mathbf{y}_j$ to derive a second order approximation of the squared distance around $\mathbf{y}_j$.

These three approximations are shown in Fig. 2. Wang et al. [29] showed that SDM can be characterized in the SD setting as a quasi-Newton method and they applied it to B-spline curve fitting. Cheng et al. [4, 5] proposed a subdivision surface fitting algorithm based on SDM.

3 Voronoi squared distance minimization

The existing approximations of SDM require an accurate estimation of the curvature tensor on $\mathcal{T}$ to be efficiently minimized, which may be unavailable if $\mathcal{T}$ is a triangle soup or a CAD mesh with many skinny triangles. VSDM introduces a new approximation of F, which also exhibits nice convergence rates while being free of curvature computations. This is done by removing one of the approximations made, by efficiently computing the integral over $\mathcal{S}$ in Eq. 1. As in the other approaches though, the nearest neighbor of a point $\mathbf{x}$ is still approximated using a sampling $\mathbf{Y}$ of $\mathcal{T}$. These computations are based on the concept of CVT, and will be explained in Sect. 3.4.

In addition, we note that $F_{\mathcal{S}\to\mathcal{T}}$ vanishes whenever $\mathcal{S}$ matches a subset of $\mathcal{T}$ (instead of the totality of $\mathcal{T}$). Therefore, to avoid degenerate minimizers that partially match $\mathcal{T}$, we propose to minimize a symmetrized version of SDM given by $F_{\mathcal{S}\to\mathcal{T}} + F_{\mathcal{T}\to\mathcal{S}}$. The benefit of the symmetrized formulation is demonstrated later (Sect. 3.3).

Finally, this new formulation also allows us to provide an approximation bound of the error made by our approximation as a function of the sampling quality of $\mathbf{Y}$ (used for $\tilde{F}_{\mathcal{S}\to\mathcal{T}}$) and $\mathbf{X}$ (used for $\tilde{F}_{\mathcal{T}\to\mathcal{S}}$). This bound is given in Sect. 3.2.

3.1 Definition

We consider two point sets $\mathbf{X}$ and $\mathbf{Y}$ that sample the surface $\mathcal{S}$ being deformed and the input surface $\mathcal{T}$, respectively. From now on, we will use the vertices of $\mathcal{S}$ as the sampling $\mathbf{X}$. In other words, the vertices $\mathbf{X}$ of $\mathcal{S}$ are the unknowns of the optimization problem. When the mesh for $\mathcal{T}$ is regular enough, and has a resolution at least equivalent to that of $\mathcal{S}$, its face centers are used as samples. This allows us to have a well-defined normal at these sample points when needed. Otherwise, the sampling $\mathbf{Y}$ of the input surface $\mathcal{T}$ is automatically computed by starting from a random sampling [25] and optimizing its regularity using restricted CVT [30].

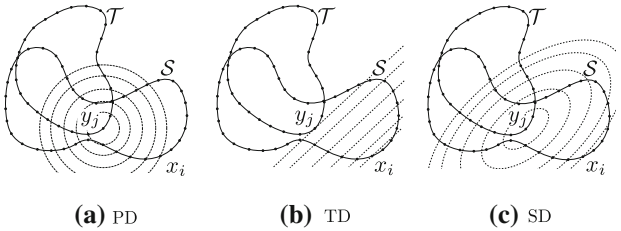


Fig. 2 Illustration of different approximations of SDM. The *dashed lines* represent iso-lines of the distance function to the sample

VSDM minimizes an approximation of the following objective function:

$$F(\mathbf{X}) = F_{\mathcal{T}\to\mathcal{S}}(\mathbf{X}) + F_{\mathcal{S}\to\mathcal{T}}(\mathbf{X}) + \lambda R(\mathbf{X}). \tag{2}$$

Let us consider the first term $F_{\mathcal{T}\to\mathcal{S}}$.Using a sampling $\mathbf{X}$ of $\mathcal{S}$, we make the following approximation $\|\mathbf{y} - \Pi_{\mathcal{S}}(\mathbf{y})\| \simeq \min_i \|\mathbf{y} - \mathbf{x}_i\|$. Replacing the integrand of $F_{\mathcal{T}\to\mathcal{S}}$ gives:

$$\begin{aligned} F_{\mathcal{T}\to\mathcal{S}} &= \int_{\mathcal{T}} \|\mathbf{y} - \Pi_{\mathcal{S}}(\mathbf{y})\|^2 d\mathbf{y} \\ &\simeq \int_{\mathcal{T}} \min_i \|\mathbf{y} - \mathbf{x}_i\|^2 d\mathbf{y} \\ &= \sum_i \int_{\Omega_i \cap \mathcal{T}} \|\mathbf{y} - \mathbf{x}_i\|^2 d\mathbf{y} \end{aligned}$$

where Ω_i denotes the 3D Voronoi cell of $\mathbf{x}_i$ in the Voronoi diagram of $\mathbf{X}$. We shall now give the definition of the approximation $\tilde{F}$ of F minimized by VSDM:

$$\tilde{F}(\mathbf{X}) = \tilde{F}_{\mathcal{T}\to\mathcal{S}}(\mathbf{X}) + \tilde{F}_{\mathcal{S}\to\mathcal{T}}(\mathbf{X}) + \lambda \underbrace{\mathbf{X}^t \mathbf{L}^2 \mathbf{X}}_{R(\mathbf{X})} \tag{3}$$

where:

$$\tilde{F}_{\mathcal{T}\to\mathcal{S}} = \sum_{\mathbf{x}_i \in \mathbf{X}} \int_{\mathcal{T}\cap\Omega_i} \| \mathbf{y} - \mathbf{x}_i \|^2 \, d\mathbf{y}$$

$$\tilde{F}_{\mathcal{S}\to\mathcal{T}} = \sum_{\mathbf{y}_j \in \mathbf{Y}} \int_{\mathcal{S}\cap\Omega_j} \| \mathbf{x} - \mathbf{y}_j \|^2 \, d\mathbf{x}$$

The matrix $\mathbf{L}$ is the uniform graph Laplacian of $\mathcal{S}$. The influence of the regularization factor λ is illustrated in Fig. 3. Ω_i denotes the Voronoi cell of $\mathbf{x}_i$ in the Voronoi

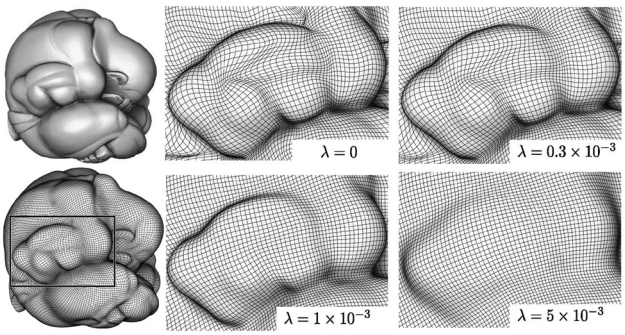


Fig. 3 Influence of the regularization factor λ on subdivision surface fitting

diagram of $\mathbf{X}$, and Ω_j the Voronoi cell of $\mathbf{y}_j$ in the Voronoi diagram of $\mathbf{Y}$ (see Fig. 4).

3.2 Convergence to the continuous objective function

The VSDM approximation replaces the nearest point on $\mathcal{S}$ with the nearest sample of $\mathbf{X}$ (in the term $F_{\mathcal{T}\to\mathcal{S}}$) and the nearest point on $\mathcal{T}$ with the nearest sample of $\mathbf{Y}$ (in the term $F_{\mathcal{S}\to\mathcal{T}}$). The accuracy of the approximation depends on the *density* of the point sets $\mathbf{X}$ and $\mathbf{Y}$ used to sample $\mathcal{S}$ and $\mathcal{T}$, respectively. The density of a sampling is formalized by the notion of ε-sampling [2]. A point set $\mathbf{X}$ is an ε-sampling of a surface $\mathcal{S}$ if for any point $\mathbf{x}$ of $\mathcal{S}$ there is a point $\mathbf{x}_i$ in $\mathbf{X}$ such that $\|\mathbf{x}_i - \mathbf{x}\| < \varepsilon \mathrm{lfs}(\mathbf{x})$ where $\mathrm{lfs}(\mathbf{x})$ denotes local feature size (distance to medial axis of $\mathcal{S}$). $\tilde{F}_{\mathcal{T}\to\mathcal{S}}$ satisfies the following property (proved in Appendix 1).

Property 1 *Given* $\mathbf{X}$, an *ε-sampling of $\mathcal{S}$, we have:*

$$\lim_{\varepsilon\to 0} \tilde{F}_{\mathcal{T}\to\mathcal{S}}(\mathbf{X}) = F_{\mathcal{T}\to\mathcal{S}}(\mathbf{X})$$

The same property is satisfied by the symmetric term $\tilde{F}_{\mathcal{S}\to\mathcal{T}}$ if $\mathbf{Y}$ is an ε-sampling of $\mathcal{T}$. Therefore, if $\mathbf{X}$ and $\mathbf{Y}$ are dense enough, $\tilde{F}$ is a good approximation of F [in the order of $o(\varepsilon^2)$, see Appendix 1]. Note that this property is only satisfied for a smooth surface. However, a good behavior is also observed for surfaces with sharp creases (see, e.g., Fig. 7).

3.3 Need for the symmetrized objective function

The term $\tilde{F}_{\mathcal{T}\to\mathcal{S}}$ of $\tilde{F}$ corresponds to the objective function minimized by Restricted CVT. Therefore, omitting the term $\tilde{F}_{\mathcal{S}\to\mathcal{T}}$ results in the objective function $\tilde{F}_{\mathcal{T}\to\mathcal{S}} + \lambda R(\mathbf{X})$, that can be minimized by a straightforward modification of the CVT quasi-Newton algorithm used in [19, 30], i.e., by adding the term $\lambda\, \mathbf{X}^t\, \mathbf{L}^2\, \mathbf{X}$ to the objective function and $2\,\lambda\, \mathbf{L}^2\, \mathbf{X}$ to the gradient. However, as noted before, the function $F_{\mathcal{T}\to\mathcal{S}}$ reaches a minimum whenever $\mathcal{S}$ is a superset of $\mathcal{T}$. Therefore, a minimizer of $F_{\mathcal{T}\to\mathcal{S}}$ may have spurious parts, as shown in Fig. 5. These spurious parts correspond to the set $\mathcal{S} - \Pi_{\mathcal{S}}(\mathcal{T})$, that does not yield any term in $F_{\mathcal{T}\to\mathcal{S}}$. In terms of the discretization $\tilde{F}_{\mathcal{S}\to\mathcal{T}}$, they correspond to Voronoi cells that have an empty intersection with $\mathcal{T}$.

3.4 Solution mechanism

To minimize the function $\tilde{F} = \tilde{F}_{\mathcal{T}\to\mathcal{S}} + \tilde{F}_{\mathcal{S}\to\mathcal{T}} + \lambda R(\mathbf{X})$ in Eq. 3, VSDM uses the L-BFGS algorithm [17, 22]. L-BFGS is a Newton-type algorithm that uses successive evaluation of the function and its gradient to compute an approximation of the inverse of the Hessian. Although only the gradient is required in the computation, the objective function needs to be C^2 to ensure the proper convergence of the L-BFGS algorithms. We discuss here about the continuity of the three terms of $\tilde{F}$:

- $R(\mathbf{X})$ is a quadratic form (C^∞);
- $\tilde{F}_{\mathcal{T}\to\mathcal{S}}$ corresponds to the *quantization noise power*, which is the objective function minimized by a CVT. It is of class C^2, except in some rarely encountered degenerate configurations (see [19] for a proof);
- $\tilde{F}_{\mathcal{S}\to\mathcal{T}}$ is obtained by permuting the roles of the constant and variables in $\tilde{F}_{\mathcal{T}\to\mathcal{S}}$. We will study its continuity in future work. Experimentally, it is regular enough for obtaining a stable behavior of L-BFGS.

In practice, implementing L-BFGS requires evaluating $\tilde{F}(\mathbf{X}^{(k)})$ and $\nabla\tilde{F}(\mathbf{X}^{(k)})$ for a series of iterates $\mathbf{X}^{(k)}$ (see Algorithm 1):

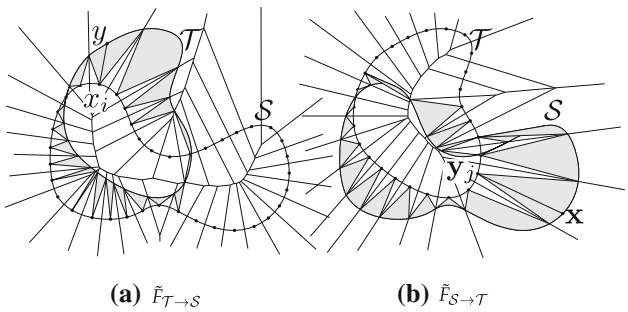


Fig. 4 Illustration of the terms of the VSDM objective function. The *shaded regions* represent for each sample $\mathbf{x}_i$ (resp. $\mathbf{y}_j$) the portion $\mathcal{T} \cap \Omega_i$ (resp. $\mathcal{S} \cap \Omega_j$) of the other surface whose squared distance with respect to the sample is integrated

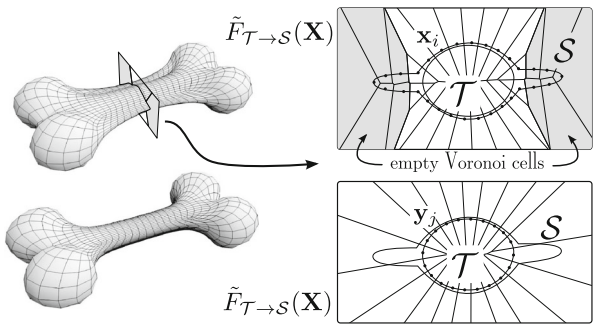


Fig. 5 *Top* the minimizer of $\tilde{F}_{\mathcal{T}\to\mathcal{S}}$ has spurious parts that cannot be eliminated since their Voronoi cells do not intersect the input mesh $\mathcal{T}$. *Bottom* the symmetrized $\tilde{F} = \tilde{F}_{\mathcal{T}\to\mathcal{S}} + \tilde{F}_{\mathcal{S}\to\mathcal{T}}$ detects and eliminates them

(1) $\mathbf{X}^{(0)} \leftarrow$ vertices of $\mathcal{S}^{(0)}$
(2) $\mathbf{Y} \leftarrow$ ε-sampling of $\mathcal{T}$; Compute $Vor(\mathbf{Y})$
while *minimum not reached* **do**
  (3) Compute $Vor(\mathbf{X}^{(k)})$, $Vor(\mathbf{X}^{(k)})|_{\mathcal{T}}$ and $Vor(\mathbf{Y})|_{\mathcal{S}}$
  (4) Compute $\tilde{F}_{\mathcal{T}\to\mathcal{S}}(\mathbf{X}^{(k)})$ and $\nabla\tilde{F}_{\mathcal{T}\to\mathcal{S}}(\mathbf{X}^{(k)})$
  (5) Compute $R(\mathbf{X}^{(k)})$ and $\nabla R(\mathbf{X}^{(k)})$
  (6) Compute $\tilde{F}_{\mathcal{S}\to\mathcal{T}}(\mathbf{X}^{(k)})$ and $\nabla\tilde{F}_{\mathcal{S}\to\mathcal{T}}(\mathbf{X}^{(k)})$
  (7) Compute $\tilde{F}(\mathbf{X})$ and $\nabla\tilde{F}(\mathbf{X})$
  (8) $\mathbf{X}^{(k+1)} \leftarrow \mathbf{X}^{(k)} + \mathbf{p}^{(k)}$; Update $\mathcal{S}$ from $\mathbf{X}^{(k+1)}$
end

Algorithm 1: *fitting a polygon mesh using VSDM.*

In order to make our work reproducible, we further detail the main steps:

(2) the sampling $\mathbf{Y}$ of $\mathcal{T}$, used by $\tilde{F}_{\mathcal{S}\to\mathcal{T}}$, is computed by the CVT algorithm in [30], with the same number of vertices as in $\mathbf{X}$;

(3) the Restricted Voronoi Diagrams $\mathrm{Vor}(\mathbf{Y})|_{\mathcal{S}}$, and $Vor(\mathbf{X}^{(k)})|_{\mathcal{T}}$ are computed as in [30];

(4) $F_{\mathcal{T}\to\mathcal{S}}$ is the CVT objective function. Its gradient is given by Eq. 4:

$$\nabla|_{\mathbf{x}_i}\tilde{F}_{\mathcal{T}\to\mathcal{S}} = 2m_i(\mathbf{x}_i - \mathbf{g}_i) \tag{4}$$

where m_i and $\mathbf{g}_i$ denote the volume and the centroid of the restricted Voronoi cell $\Omega_i \cap \mathcal{T}$ [8];

(5) The expression of $\nabla R(\mathbf{X})$ is simply given by:

$$\nabla R(\mathbf{X}) = \nabla\mathbf{X}^t\mathbf{L}^2\mathbf{X} = 2\mathbf{L}^2\mathbf{X} \tag{5}$$

(6) the term $\tilde{F}_{\mathcal{S}\to\mathcal{T}}$ is obtained by exchanging the roles of $\mathcal{S}$ and $\mathcal{T}$ in $\tilde{F}_{\mathcal{T}\to\mathcal{S}}$ and using the point set $\mathbf{Y}$ instead of $\mathbf{X}$. The computation of this term and its gradient are explained in the next paragraph;

(8) $\mathbf{p}^{(k)}$ denotes the step vector computed by L-BFGS.

The function $\tilde{F}_{\mathcal{S}\to\mathcal{T}}$ depends on the Voronoi diagram of $\mathbf{Y}$ restricted to $\mathcal{S}$ (see Fig. 6). Each restricted Voronoi cell $\Omega_j \cap \mathcal{S}$ (polygons) is decomposed into a set of triangles. One of them $T = (\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3)$ is highlighted. Each triangle T of the decomposition of $\Omega_j \cap \mathcal{S}$ contributes the following terms to $\tilde{F}_{\mathcal{S}\to\mathcal{T}}$ and $\nabla\tilde{F}_{\mathcal{S}\to\mathcal{T}}$:

$$\begin{aligned}\tilde{F}_{\mathcal{S}\to\mathcal{T}} &= \sum_{T\in\Omega_j\cap\mathcal{S}} \tilde{F}^T_{\mathcal{S}\to\mathcal{T}} \\ &= \sum_{T\in\Omega_j\cap\mathcal{S}} \frac{|T|}{6} \sum_{1\le k\le l\le 3} (\mathbf{c}_k - \mathbf{y}_j)\cdot(\mathbf{c}_l - \mathbf{y}_j)\ \frac{d\tilde{F}^T_{\mathcal{S}\to\mathcal{T}}}{d\mathbf{X}} \\ &= \sum_{k=1}^{3} \frac{d\tilde{F}^T_{\mathcal{S}\to\mathcal{T}}}{d\mathbf{c}_k}\frac{d\mathbf{c}_k}{d\mathbf{X}}\end{aligned} \tag{6}$$

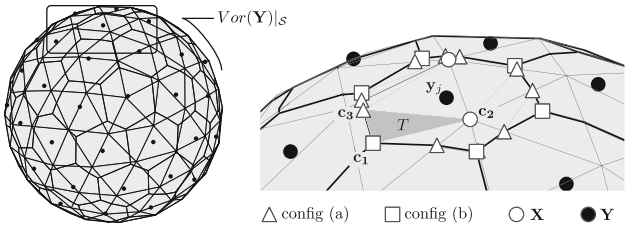


Fig. 6 Computing the gradient of the symmetric term $\nabla\tilde{F}_{\mathcal{S}\to\mathcal{T}}$°: configurations of the vertices of $Vor(\mathbf{Y})|_{\mathcal{S}}$

where $d\mathbf{A}/d\mathbf{B} = (\partial a_i / \partial b_j)_{i,j}$ denotes the Jacobian matrix of $\mathbf{A}$.

The set of possible configurations for a vertex $\mathbf{c}_k$ is similar to the combinatorial structure of the L_p-CVT function [14], with the exception that the roles of the variables and constants are exchanged. Each configuration yields a Jacobian matrix that propagates the derivatives of $\tilde{F}^T_{\mathcal{S}\to\mathcal{T}}$ from the $\mathbf{c}_k$'s to the $\mathbf{x}_i$'s. There are three possible configurations (see overview in Fig. 6):

→ ◯ $\mathbf{c}$ is a vertex $\mathbf{x}_i$ of $\mathcal{S}$:
then $d\ \mathbf{c}/d\ \mathbf{x}_i = \mathbf{I}_{3\times 3}$ and the derivatives are directly propagated;
→ △ $\mathbf{c}$ has configuration (a):
$\mathbf{c}$ corresponds to the intersection between a bisector (thick black edge) and an edge $[\mathbf{x}_1, \mathbf{x}_2]$ of $\mathcal{S}$. The Jacobian matrices $d\ \mathbf{c}/d\ \mathbf{x}_1$ and $d\ \mathbf{c}/d\ \mathbf{x}_2$ are given in Appendix 2, Eq. 13;
→ □ $\mathbf{c}$ has configuration (b):
$\mathbf{c}$ corresponds to the intersection between three bisectors (thick black edges) and a facet $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ of $\mathcal{S}$. The Jacobian matrices $d\ \mathbf{c}/d\ \mathbf{x}_1$, $d\ \mathbf{c}/d\ \mathbf{x}_2$ and $d\ \mathbf{c}/d\ \mathbf{x}_3$ are given in Appendix 2, Eq. 14.

By injecting these vertex derivatives in Eq. 6, we obtain the gradient of $\tilde{F}_{\mathcal{S}\to\mathcal{T}}$.

In Eq. 6 we assumed that the set $\{T \in \Omega_j \cap \mathcal{S}\}$ remains constant for an elementary displacement of the variables $\mathbf{X}$. In fact, when a vertex $\mathbf{x}_i$ of $\mathcal{S}$ enters or leaves the Voronoi cell of a sample $\mathbf{y}_j$, the combinatorics of the restricted cells change, and so do the sets $\{T \in \Omega_j \cap \mathcal{S}\}_j$. We conjecture that $\nabla\tilde{F}_{\mathcal{S}\to\mathcal{T}}$ remains continuous in these situations, except in some rarely encountered degenerate configurations, such as when a face of $\mathcal{S}$ is contained in a Voronoi face of the diagram of $\mathbf{Y}$. The stability we experimented using the L-BFGS solver with this gradient tends to strengthen this conjecture. The complete gradient of our objective function is computed using Eqs. 4, 5 and 6 and provided to the L-BFGS solver for optimization.

3.5 Fitting polynomial surfaces

$\mathbf{P}^{(0)} \leftarrow$ vertices of $\mathcal{C}^{(0)}$
$\mathbf{Y} \leftarrow \varepsilon$-sampling of $\mathcal{T}$; Compute $Vor(\mathbf{Y})$
while *minimum not reached* **do**
  $\mathbf{X} \leftarrow \mathbf{M}\mathbf{P}^{(k)}$; Update $\mathcal{S}$ from $\mathbf{X}$
  Compute $\tilde{F}(\mathbf{X})$ and $\nabla\tilde{F}(\mathbf{X})$ as in Algo. 1, steps (3) to (7)
  Compute $\nabla G(\mathbf{P}) = \mathbf{M}^t \nabla\tilde{F}(\mathbf{X})$
  $\mathbf{P}^{(k+1)} \leftarrow \mathbf{P}^{(k)} + \mathbf{p}^{(k)}$
end

Algorithm 2: *Fitting a polynomial surface.*

We now consider the problem of fitting a polynomial surface defined by its control mesh $\mathcal{C}$. At each iteration, we compute a polygonal approximation $\mathcal{S}$ of the polynomial surface. The vertices $\mathbf{X}$ of $\mathcal{S}$ are given as linear combinations of the control points $\mathbf{P}$ as $\mathbf{X} = \mathbf{M}\mathbf{P}$, where $\mathbf{X} = [x_1 y_1 z_1 \ldots x_n y_n z_n]^t$ denotes the coordinates of the vertices of $\mathcal{S}$ and $\mathbf{P}$ those of the control points. One may use the exact evaluation of the polynomial surface, or simply use the approximation obtained by subdividing the control mesh several times with De Casteljau's rule.

Polynomial surface fitting is done by minimizing the function $G(\mathbf{P}) = \tilde{F}(\mathbf{M}\mathbf{P})$. This can implemented with a change of variable in the VSDM algorithm (see Algorithm 2).

3.6 Feature-sensitive fitting

Using the algorithm above for fitting polynomial surfaces may result in over-smoothing sharp creases (Fig. 7, center). However, this can be improved by injecting *normal anisotropy* [14] into the objective function $\tilde{F}$ (Fig. 7, right). This changes the terms $\tilde{F}_{\mathcal{T}\to\mathcal{S}}$ and $\tilde{F}_{\mathcal{S}\to\mathcal{T}}$ as follows:

$$\tilde{F}^s_{\mathcal{T}\to\mathcal{S}} = \sum_{\mathbf{x}_i\in\mathbf{X}} \sum_{T\subset\mathcal{T}\cap\Omega_i} \int_T \| \mathbf{A}_s(\mathbf{N}_T)(\mathbf{y}-\mathbf{x}_i) \|^2 \, d\mathbf{y}$$

$$\tilde{F}^s_{\mathcal{S}\to\mathcal{T}} = \sum_{\mathbf{y}_j\in\mathbf{Y}} \sum_{T\subset\mathcal{S}\cap\Omega_j} \int_T \| \mathbf{A}_s(\mathbf{N}_j)(\mathbf{x}-\mathbf{y}_j) \|^2 \, d\mathbf{x}$$

where:

$$\mathbf{A}_s(\mathbf{N}) = (s-1)\begin{pmatrix} \mathbf{N}_x[\mathbf{N}]^t \\ \mathbf{N}_y[\mathbf{N}]^t \\ \mathbf{N}_z[\mathbf{N}]^t \end{pmatrix} + \mathbf{I}_{3\times 3}$$

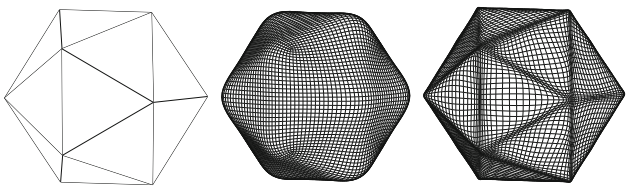


Fig. 7 Influence of the feature-sensitive fitting on meshes with sharp creases (*from left to right* original mesh, result without and with normal anisotropy)

where the parameter $s \in (0, +\infty)$ specifies the importance of normal anisotropy. The normals are sampled from the input surface $\mathcal{T}$ in both terms, $\mathbf{N}_T$ is the normal of the triangle T, and $\mathbf{N}_j$ the normal to $\mathcal{T}$ at $\mathbf{y}_j$. Normal anisotropy is used in all the examples shown below.

3.7 Implementation

To compute the 3D Delaunay triangulation, we used CGAL [1]. For the restricted Voronoi Diagram computation (Sect. 3.4) and the normal anisotropy (previous subsection), we used the implementation provided with [14]. The numerical solver uses the L-BFGS implementation from [18].

4 Results

We shall now show some results obtained with VSDM. In the results herein, the regularization term is set to $\lambda = 0.2 \times 10^{-3}$, the normal anisotropy is set to $s = 50$ and subdivision surfaces are approximated by subdividing the control mesh twice. Figures 8, 9, and 10 show the results obtained with an initial toroidal grid. Note on Fig. 12 how the spacing of the iso-parameter line adapts to the features. Scanned meshes from AimAtShape can also be efficiently processed (see Fig. 13). The method is independent on the connectivity of the data, and can process input meshes with many connected components, degenerate triangles and overlaps, such as registered raw meshes directly obtained from a 3D scanner (Figs. 14, 15).

For each model, the result was obtained in less than 3 min on a 2-GHz machine, except for Figs. 14 and 15 for which the timings were, respectively, 7 and 15 min.

4.1 Failure cases

Our algorithm may fail in several cases. As we do not modify the combinatorial structure of the fitted surface $\mathcal{S}$, when $\mathcal{S}$ and $\mathcal{T}$ have different genus the result will not be satisfactory, though our method will not crash. We will

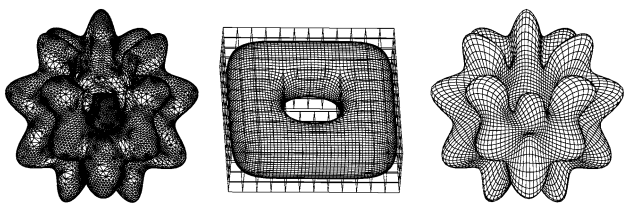


Fig. 8 Fitting a polynomial surface to an object with toroidal topology. *Left* input mesh (16.8k vertices, 33.6k facets), *center* initial control mesh (512 vertices and 512 quads) and surface, *right* result. The Hausdorff distance between the resulting surface and the input mesh is 1.221 % (relative to the diagonal length of the bounding box, measured by Metro [6])

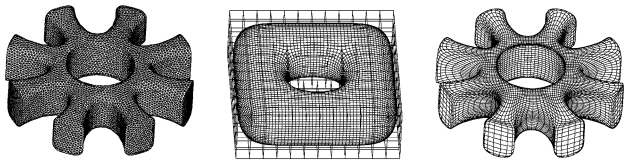


Fig. 9 Another example, using the same initial toroidal control mesh as in Fig. 8. *Left* input mesh (10k vertices, 20k facets), *center* result, *right* result (control mesh with 512 vertices and 512 quads). Hausdorff distance is 0.473% bbox. diag

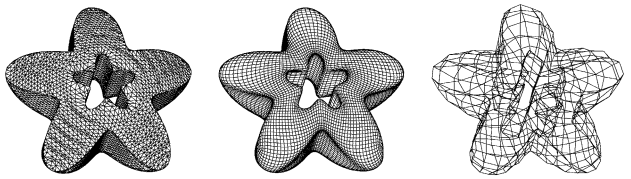


Fig. 10 Another example, still using the same initial toroidal control mesh. *Left* input mesh (5.2k vertices and 10.4k facets), *center* result, *right* result (control mesh with 512 vertices and 512 quads). The Hausdorff distance is 0.699 % bbox. diag

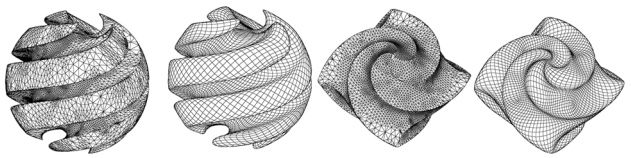


Fig. 11 Other examples with geometrical shapes. Initialization from bounding box (386 vertices and 384 quads). *Left* input mesh of sharp sphere (10.4k vertices, 20.9k facets) and result. *Right* input mesh of octa-flower (7.9k vertices and 15.8k facets) and result. Hausdorff distances are 1.072 and 0.706 % bbox. diag., respectively

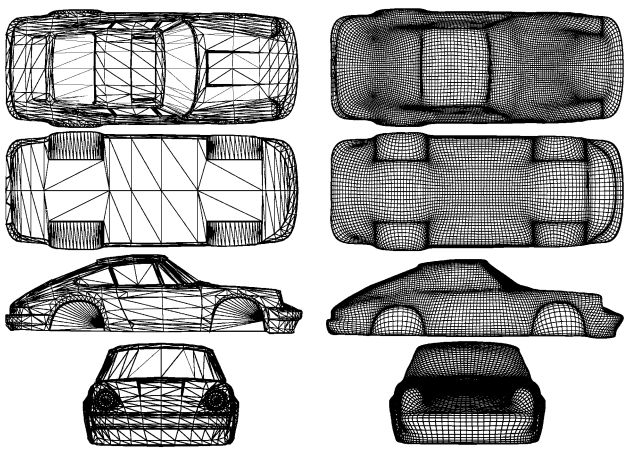


Fig. 12 Different views of the example shown in Fig. 1. The Hausdorff distance between the input and result is 0.554 % of the bounding box diagonal

discuss automatic methods for the generation of an initial template below. Besides, our algorithm may be sensitive to defects in the input data, such as:

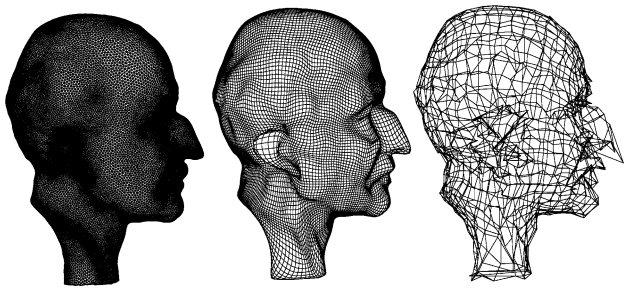


Fig. 13 Fitting a Catmull–Clark subdivision surface to the statue of Max Planck (52.8k vertices and 105.6k facets). Initialization from bounding box (6,257 vertices and 6,255 quads). The Hausdorff distance is 0.386 % of the bounding box diagonal

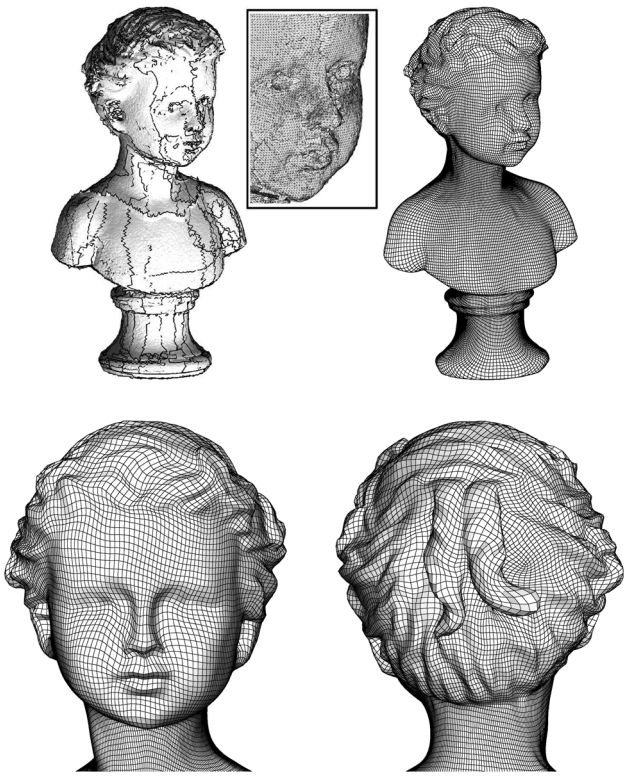


Fig. 14 Fitting a Catmull–Clark subdivision surface to registered scanned meshes (*top left* 327.6k vertices and 630k facets). Note the irregularity of the input data and overlaps (*thick black lines*). The fitted control mesh, initialized from the bounding box, has 7.1k vertices and 7.2k facets. The Hausdorff distance is 1.43 % of the bounding box diagonal. This mesh was generated in 7 min

Bad template Our method does not change the combinatorial structure of the provided initial template. When $\mathcal{S}$ and $\mathcal{T}$ vary too much in shape, pinches or excessive distortion may appear in the result, as in Fig. 16.

Bad initial position We do not claim reaching a global minimum for $\tilde{F}$. Our algorithm converges to a local minimum, which depends on the relative positions of $\mathcal{S}$ and $\mathcal{T}$. Automatically adjusting the orientation of $\mathcal{S}$ and

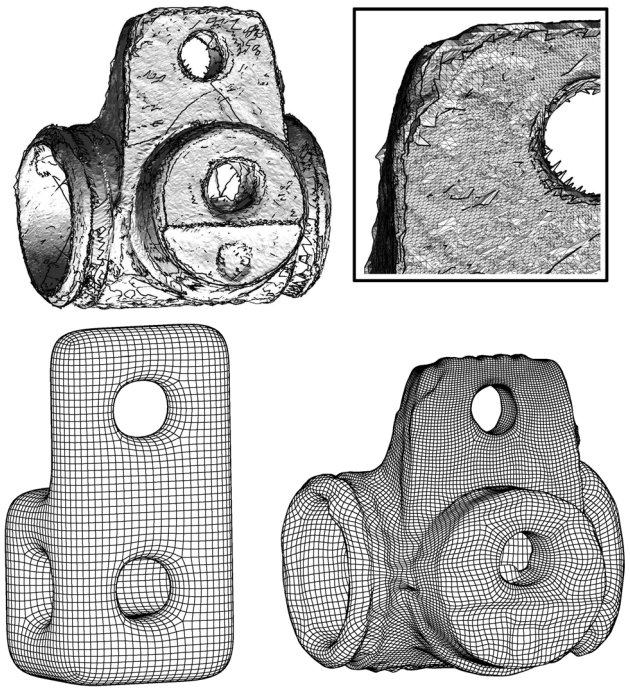


Fig. 15 Fitting a Catmull–Clark subdivision surface to registered scanned meshes (*top left* 747.6k vertices and 1.43m facets). The input data has many degenerate triangles, overlaps and spurious spikes (*top right*). The manually designed template control mesh (*bottom left*) has 5.8k vertices and 5.8k facets. The Hausdorff distance is 7.66 % of the bounding box diagonal. This high approximation error is mainly due to the presence of holes in the target model, due to parts of the object which were not directly visible from the scanner used to acquire the data. This mesh was generated in 15 min

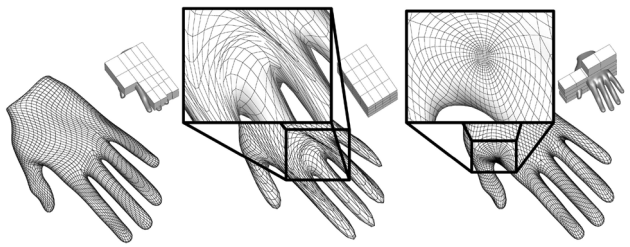


Fig. 16 Fitting a hand model with different initializations. With a well-designed and aligned template (*left*) the fitting is correct. Using the bounding box as a template (*center*) leads to pinches and distortions in the result. When the template is extremely misaligned with the target (*right*), one of the fingers gets flattened leading to an excessive density next to the thumb

$\mathcal{T}$ is a difficult problem, and a misaligned initial template may also lead to pinches or distortion (Fig. 16).

Thin sheets Even when using normal anisotropy, the orientation of the surfaces is not considered in the objective function. Therefore, several layers of $\mathcal{T}$ may be sampled by a single sheet of $\mathcal{S}$ (Fig. 17).

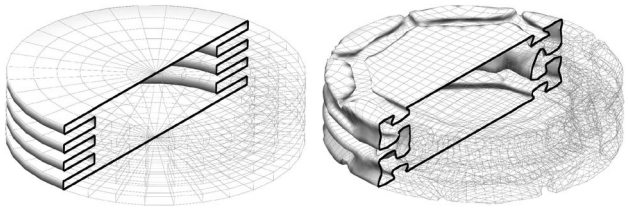


Fig. 17 Fitting of the bounding box of a model with thin features (a *cut* is shown on the *left*). The result is shown on the *right*

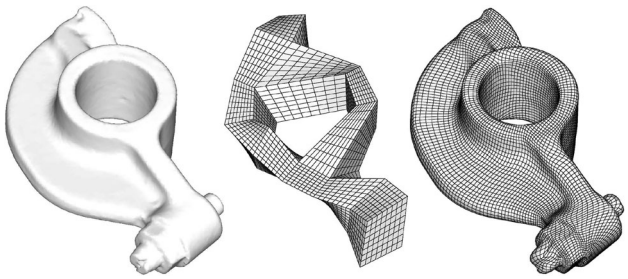


Fig. 18 Surface fitting with a control mesh generated using the method in [31]. The Hausdorff distance is 0.44% of the bounding box diagonal. *Left* input mesh (20k vertices, 40k facets), *center* initial control mesh (3k vertices), *right* fitted control mesh with one subdivision step (11k vertices)

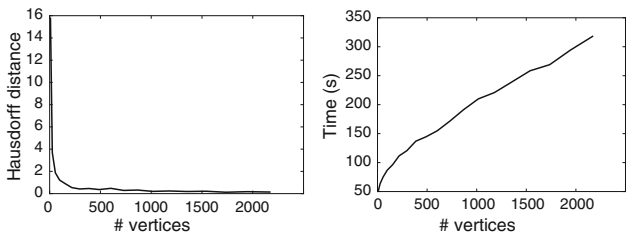


Fig. 19 Evolution of the Hausdorff distance (*left* expressed as a percentage of the bounding box diagonal) and the optimization time (*right*) with the number of vertices of the template on the octa-flower model (Fig. 11)

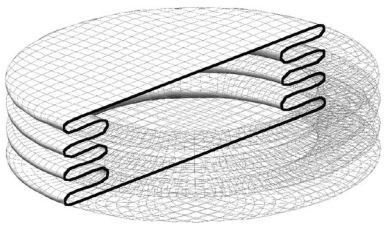


Fig. 20 A correct fitting from the bounding box on a model with thin features by taking into account the normal orientation

4.2 Initial control mesh generation

Our algorithm can also be initialized using existing automatic or assisted method to build the control mesh. We

used the skeleton-based quadrangulation approach [31] to generate the coarse control mesh presented in Fig. 18. Another approach is the automatic generation of polycube maps [11, 16].

4.3 Performance

Figure 19 shows the evolution of the Hausdorff distance and the computation time as the resolution of the initial template increases. Experimentally, the computation time seems to be linear or at worst in $O(n\log n)$ with respect to the number n of vertices of the template. Most of the computation time is spent in the computation of the restricted Voronoi diagrams. The Hausdorff distance decreases very rapidly, though not strictly.

4.4 Discussion and future work

An appropriate initial control mesh must be provided to our method. The automatic generation of this mesh is in general a challenging problem for which a few solutions have already been proposed [11, 16, 31]. We also believe that an automatic mesh modification routine triggered during the optimization process to improve the mesh quality would be a promising approach. We will explore these ideas in future work.

We are currently investigating an additional term in our objective function considering the alignment and respective orientation of the normals, to address the problem of thin features (Fig. 17). We obtained good preliminary results (Fig. 20).

Acknowledgments The authors wish to thank Sylvain Lefebvre for a discussion (about an unrelated topic) that inspired this work, Rhaleb Zayer, Xavier Goaoc, Tamy Boubekeur, Yang Liu and Wenping Wang for many discussions, Loic Marechal, Marc Loriot and the AimAtShape repository for data. This project is partly supported by the European Research Council grant GOODSHAPE ERC-StG-205693 and ANR/NSFC (60625202,60911130368) Program (SHAN Project).

Appendix 1: Convergence to squared distance: error bound

This Appendix proves that if $\mathbf{X}$ is an ε-sampling of $\mathcal{S}$ then:

$$\lim_{\varepsilon\to 0} \tilde{F}_{\mathcal{T}\to\mathcal{S}}(\mathbf{X}) = F_{\mathcal{T}\to\mathcal{S}}(\mathbf{X}) \tag{7}$$

Lemma 1 *Let* $\mathbf{y}$ *be a point of* $\mathcal{T}$ *and* $\mathbf{x}_i$ *its nearest point in* $\mathbf{X}$ (*see* Fig. 21). *Let* $d = \| \mathbf{y} - \Pi_{\mathcal{S}}(\mathbf{y}) \|$ *and* $\tilde{d} = \| \mathbf{y} - \mathbf{x}_i \|$. *Then for* $\varepsilon < 2$ *the following bound is sharp*:

$$\tilde{d}^2 - d^2 \leq \varepsilon^2 \mathrm{lfs}(\Pi_{\mathcal{S}}(\mathbf{y}))(\mathrm{lfs}(\Pi_{\mathcal{S}}(\mathbf{y})) + d)$$

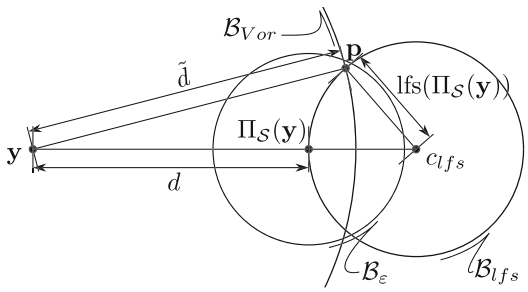


Fig. 21 Configuration of the nearest point of $\Pi_{\mathcal{S}}(\mathbf{y})$

Proof Let $\mathcal{B}_{Vor}$ be the ball centered at $\mathbf{y}$ passing through $\mathbf{x}_i$. This ball contains no point of $\mathbf{X}$.

Let $\mathcal{B}_{lfs}$ be the ball tangent to $\mathcal{S}$ at $\Pi_{\mathcal{S}}(\mathbf{y})$ on the opposite side of $\mathbf{y}$ and of radius $\mathrm{lfs}(\Pi_{\mathcal{S}}(\mathbf{y}))$ and c_{lfs} its center. By definition of lfs, this ball has no point of $\mathcal{S}$ in its interior and therefore contains no point of $\mathbf{X}$.

Finally, let $\mathcal{B}_{\varepsilon}$ be the ball centered at $\Pi_{\mathcal{S}}(\mathbf{y})$ of radius $\varepsilon\mathrm{lfs}(\Pi_{\mathcal{S}}(\mathbf{y}))$. Since $\mathbf{X}$ is an ε-sampling this ball has to contain a point of $\mathbf{X}$.

Since $\Pi_{\mathcal{S}}(\mathbf{y})$ is the nearest point of $\mathbf{y}$ on $\mathcal{S}, \mathbf{y}, \Pi_{\mathcal{S}}(\mathbf{y})$ and c_{lfs} are aligned and the problem is completely symmetric around the line joining them. Figure 21 shows a cut containing this axis.

The bound follows from the fact that $\mathcal{B}_{\varepsilon} \not\subset \mathcal{B}_{Vor} \cup \mathcal{B}_{lfs}$. Let p be a point of $\mathcal{B}_{Vor} \cap \mathcal{B}_{lfs}$. This point exists since $\varepsilon < 2$ and $\mathcal{B}_{\varepsilon}$ is not included in $\mathcal{B}_{Vor}$. Using this point the previous condition can be reformulated as $p \in \mathcal{B}_{\varepsilon}$.

Using triangular identities in $(\Pi_{\mathcal{S}}(\mathbf{y}), c_{lfs}, \mathbf{p})$, we have:

$$\|\Pi_{\mathcal{S}}(\mathbf{y}) - \mathbf{p}\|^2 = 2\mathrm{lfs}(\Pi_{\mathcal{S}}(\mathbf{y}))^2(1 - \cos\alpha) \tag{8}$$

with α being the $(\mathbf{p}, c_{lfs}, \Pi_{\mathcal{S}}(\mathbf{y}))$ angle. Using the same identities in ($\mathbf{x}$, c_{lfs}, $\mathbf{p}$) we obtain:

$$\begin{aligned}\tilde{d}^2 &= (d + \mathrm{lfs}(\Pi_{\mathcal{S}}(\mathbf{y})))^2 + \mathrm{lfs}(\Pi_{\mathcal{S}}(\mathbf{y}))^2 \\ &\quad - 2(d + \mathrm{lfs}(\Pi_{\mathcal{S}}(\mathbf{y})))\mathrm{lfs}(\Pi_{\mathcal{S}}(\mathbf{y}))\cos\alpha \\ \tilde{d}^2 - d^2 &= 2(d + \mathrm{lfs}(\Pi_{\mathcal{S}}(\mathbf{y})))\mathrm{lfs}(\Pi_{\mathcal{S}}(\mathbf{y}))(1 - \cos\alpha)\end{aligned}$$

Using Eq. 8, $(1 - \cos\alpha)$ can be replaced:

$$\tilde{d}^2 - d^2 = (d + \mathrm{lfs}(\Pi_{\mathcal{S}}(\mathbf{y})))\frac{\| \Pi_{\mathcal{S}}(\mathbf{y}) - \mathbf{p} \|^2}{\mathrm{lfs}(\Pi_{\mathcal{S}}(\mathbf{y}))} \tag{9}$$

Finally, since $\mathbf{p}$ is inside $\mathcal{B}_{\varepsilon}$, we have:

$$\|\Pi_{\mathcal{S}}(\mathbf{y}) - \mathbf{p}\| \leq \varepsilon\mathrm{lfs}(\Pi_{\mathcal{S}}(\mathbf{y})) \tag{10}$$

This finally provides the result:

$$\tilde{d}^2 - d^2 \leq \varepsilon^2\mathrm{lfs}(\Pi_{\mathcal{S}}(\mathbf{y}))(\mathrm{lfs}(\Pi_{\mathcal{S}}(\mathbf{y})) + d) \tag{11}$$

This bound is sharp since it is reached whenever $\mathcal{S}$ is exactly $\mathcal{B}_{lfs}$ and $\mathbf{x}_i$ is located at $\mathbf{p}$.

This lemma leads to a global bound:

Proposition 1 *If $\mathcal{S}$ is different from a plane and bounded, then*:

$$\tilde{F}_{\mathcal{T}\to\mathcal{S}}(\mathbf{X}) - F_{\mathcal{T}\to\mathcal{S}}(\mathbf{X}) \leq \varepsilon^2 |\mathcal{T}| \sigma_{\mathcal{S}}(\sigma_{\mathcal{S}} + d_{\mathcal{H}}(\mathcal{T},\mathcal{S}))$$

where $\sigma_{\mathcal{S}} = \sup\{lfs(\mathbf{x}), \mathbf{x} \in \mathcal{S}\}$.

Proof Since $\mathcal{S}$ is not a plane and bounded, σ exists. In addition, the definition of the Hausdorff distance gives us $d \leq d_{\mathcal{H}}(\mathcal{T},\mathcal{S})$.

$$\begin{aligned} e &= \tilde{F}_{\mathcal{T}\to\mathcal{S}}(\mathbf{X}) - F_{\mathcal{T}\to\mathcal{S}}(\mathbf{X}) \\ &= \int_{\mathcal{T}} \min_i \| \mathbf{y} - \mathbf{x}_i \|^2 \, d\mathbf{y} - \int_{\mathcal{T}} \| \mathbf{y} - \Pi_{\mathcal{S}}(\mathbf{y}) \|^2 \, d\mathbf{y} \\ &= \int_{\mathcal{T}} \min_i \| \mathbf{y} - \mathbf{x}_i \|^2 - \| \mathbf{y} - \Pi_{\mathcal{S}}(\mathbf{y}) \|^2 \, d\mathbf{y} \qquad (12) \\ &\leq \int_{\mathcal{T}} \varepsilon^2 \sigma_{\mathcal{S}}(\sigma_{\mathcal{S}} + d_{\mathcal{H}}(\mathcal{T},\mathcal{S})) d\mathbf{y} \\ &\leq \varepsilon^2 |\mathcal{T}| \sigma_{\mathcal{S}}(\sigma_{\mathcal{S}} + d_{\mathcal{H}}(\mathcal{T},\mathcal{S})) \end{aligned}$$

Appendix 2: Gradients of the symmetric term $\nabla\tilde{F}_{\mathcal{S}\to\mathcal{T}}$

This appendix gives the gradients of the symmetric term $\tilde{F}_{\mathcal{S}\to\mathcal{T}}$ (Sect. 3.3). They are used by our Quasi-Newton solution mechanism (see Sect. 3.4). This term is computed using the Voronoi diagram of the samples $\{\mathbf{y}_j\}_{j=1}^m$ of $\mathcal{T}$, restricted to the surface $\mathcal{S}$. In this setting, the variables are the vertices $\{\mathbf{x}_i\}_{i=1}^n$ of the triangles of $\mathcal{S}$. Recall that the restricted Voronoi diagram computes for each seed $\mathbf{y}_j$ a triangulated portion of $\mathcal{S}$ corresponding to the intersection of $\mathcal{S}$ with the Voronoi cell of $\mathbf{y}_j$. One of these triangles T is highlighted in Fig. 22. Each triangle contributes a term in the objective function and in the gradient (see Eq. 6 in Sect. 3.4). Since the vertices of the triangles are not the actual variables, one needs to use the Jacobians of these vertices and combine them with the chain rule in order to get the gradient of $\tilde{F}_{\mathcal{S}\to\mathcal{T}}$ with respect to the variables $\{\mathbf{x}_i\}_{i=1}^n$. To compute the Jacobians, three different configurations (a), (b), and (c) need to be distinguished (see Fig. 22). Configuration (c) (surface vertex) is trivial. The gradients are computed as follows for the two other configurations (a) and (b). For the interested reader, the rest of this appendix details the derivations.

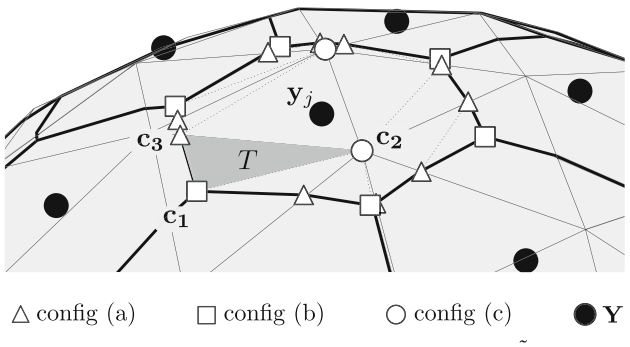
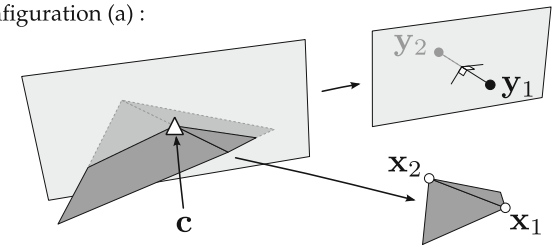


Fig. 22 Computing the gradient of the symmetric term $\nabla\tilde{F}_{\mathcal{S}\to\mathcal{T}}$: configurations of the vertices of $Vor(\mathbf{Y})|_{\mathcal{S}}$

Configuration (a) :

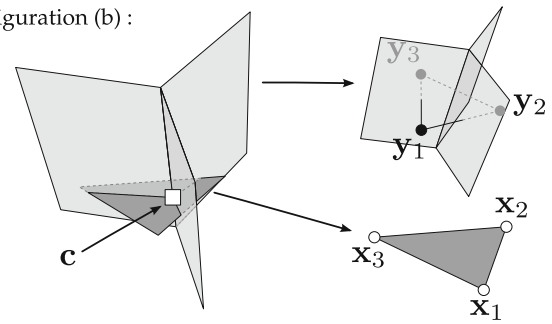


$\mathbf{c}$ corresponds to the intersection between the bisector of $[\mathbf{y}_1,\mathbf{y}_2]$ and an edge $[\mathbf{x}_1, \mathbf{x}_2]$ of $\mathcal{S}$.

$$\frac{d\mathbf{c}}{d\mathbf{x}_1} = \mathbf{e}\mathbf{w}_1^t + (1-u)\mathbf{I}_{3\times 3}$$
$$\frac{d\mathbf{c}}{d\mathbf{x}_2} = \mathbf{e}\mathbf{w}_2^t + u\mathbf{I}_{3\times 3}$$

$$\text{where:} \begin{cases} \mathbf{e} = (\mathbf{x}_2 - \mathbf{x}_1) \\ \mathbf{n} = (\mathbf{y}_2 - \mathbf{y}_1) \\ k = \mathbf{n}\cdot\mathbf{e} \\ h = \frac{1}{2}\mathbf{n}\cdot(\mathbf{y}_1 + \mathbf{y}_2) \\ u = \frac{1}{k}(h - \mathbf{n}\cdot\mathbf{x}_1) \\ \mathbf{w}_1 = \frac{1}{k^2}(h - \mathbf{n}\cdot\mathbf{x}_2)\mathbf{n} \\ \mathbf{w}_2 = -\frac{1}{k^2}(h - \mathbf{n}\cdot\mathbf{x}_1)\mathbf{n} \end{cases} \qquad (13)$$

Configuration (b) :



$\mathbf{c}$ corresponds to the intersection between the three bisectors of $[\mathbf{y}_1, \mathbf{y}_2]$, $[\mathbf{y}_2, \mathbf{y}_3]$, $[\mathbf{y}_3, \mathbf{y}_1]$ and a facet $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ of $\mathcal{S}$.

$$\frac{\mathrm{d}\mathbf{c}}{\mathrm{d}\mathbf{x}_1} = \mathbf{e}\mathbf{w}_1^t$$
$$\frac{\mathrm{d}\mathbf{c}}{\mathrm{d}\mathbf{x}_2} = \mathbf{e}\mathbf{w}_2^t$$
$$\frac{\mathrm{d}\mathbf{c}}{\mathrm{d}\mathbf{x}_3} = \mathbf{e}\mathbf{w}_3^t$$

$$\text{where: } \begin{cases} \mathbf{e} = (\mathbf{y}_1 - \mathbf{y}_2) \times (\mathbf{y}_1 - \mathbf{y}_3) \\ \mathbf{n} = (\mathbf{x}_1 - \mathbf{x}_2) \times (\mathbf{x}_1 - \mathbf{x}_3) \\ k = \mathbf{n} \cdot \mathbf{e} \\ \mathbf{w}_1 = ((\mathbf{x}_2 - \mathbf{x}_3) \times (\mathbf{x}_1 - \mathbf{c}) + \mathbf{n})/k \\ \mathbf{w}_2 = ((\mathbf{x}_3 - \mathbf{x}_1) \times (\mathbf{x}_1 - \mathbf{c}))/k \\ \mathbf{w}_3 = ((\mathbf{x}_1 - \mathbf{x}_2) \times (\mathbf{x}_1 - \mathbf{c}))/k \end{cases} \quad (14)$$

What follows is an explanation of the derivations. For configurations (a) and (b), the point can always be expressed as the intersection between a plane $\mathcal{P}$ and a line $\mathcal{L}$:

– The plane $\mathcal{P}$ defined from a point $\mathbf{p}$ and a normal $\mathbf{n}$
– The line $\mathcal{L}$ defined from a point $\mathbf{v}$ and a direction $\mathbf{e}$

The intersection $\mathbf{c}$ is defined as:

$$\mathbf{c} = \mathbf{v} + u\mathbf{e} \text{ with } u = \frac{(\mathbf{p} - \mathbf{v}).\mathbf{n}}{\mathbf{e}.\mathbf{n}} \quad (15)$$

To ease further notations, we define $k = \mathbf{e}.\mathbf{n}$. In addition, the Jacobian of $\mathbf{c}$ with respect to $\mathbf{x}_i$ will be denoted $\frac{dc}{d\mathbf{x}_i}$.

Derivation for configuration (a)

The plane $\mathcal{P}$ is a Voronoi cell facet. This plane is the bisector of two Voronoi seeds $\mathbf{y}_1$ and $\mathbf{y}_2$. Its parameters are therefore:

- $\mathbf{n} = \mathbf{y}_2 - \mathbf{y}_1$
- $\mathbf{p} = \frac{\mathbf{y}_1 + \mathbf{y}_2}{2}$

This plane is constant with respect to the variables. The edge separates two vertices of $\mathbf{X}$ namely $\mathbf{x}_1$ and $\mathbf{x}_2$. Its parameters are:

- $\mathbf{e} = \mathbf{x}_2 - \mathbf{x}_1$
- $\mathbf{v} = \mathbf{x}_1$

Considering a small displacement $\mathrm{d}\mathbf{X}$ of $\mathbf{X}$, the variables become $\mathbf{x}_i + \mathrm{d}\mathbf{x}_i$. The new location of the intersection point becomes $\mathbf{c} + \mathrm{d}\mathbf{c}$. We have:

$$\mathrm{d}\mathbf{c} = \mathrm{d}\mathbf{x}_1 + u\mathrm{d}\mathbf{e} + \mathrm{d}u\mathbf{e} = u\mathrm{d}\mathbf{x}_2 + (1-u)\mathrm{d}\mathbf{x}_1 + \mathrm{d}u\mathbf{e}$$

$$\begin{aligned} k^2\mathrm{d}u &= -\mathrm{d}\mathbf{x}_1.\mathbf{n}k - (\mathbf{p} - \mathbf{v}).\mathbf{n}(\mathrm{d}\mathbf{x}_2 - \mathrm{d}\mathbf{x}_1).\mathbf{n} \\ &= \mathrm{d}\mathbf{x}_1.\mathbf{n}(\mathbf{p} - \mathbf{x}_2).\mathbf{n} - \mathrm{d}\mathbf{x}_2.\mathbf{n}(\mathbf{p} - \mathbf{x}_1).\mathbf{n} \end{aligned}$$

The Jacobians of $\mathbf{u}$ are as follows:

$$\frac{\mathrm{d}u}{\mathrm{d}\mathbf{x}_1} = \left[\frac{1}{k^2}(\mathbf{p} - \mathbf{x}_2).\mathbf{n}\right]\mathbf{n}^t, \quad \frac{\mathrm{d}u}{\mathrm{d}\mathbf{x}_2} = -\left[\frac{1}{k^2}(\mathbf{p} - \mathbf{x}_1).\mathbf{n}\right]\mathbf{n}^t$$

Finally, the Jacobians of $\mathbf{c}$ are:

$$\frac{\mathrm{d}\mathbf{c}}{\mathrm{d}\mathbf{x}_1} = \mathbf{e}\frac{\mathrm{d}u}{\mathrm{d}\mathbf{x}_1} + (1-u)I_{3\times 3}, \quad \frac{\mathrm{d}\mathbf{c}}{\mathrm{d}\mathbf{x}_2} = \mathbf{e}\frac{\mathrm{d}u}{\mathrm{d}\mathbf{x}_2} + uI_{3\times 3} \quad (16)$$

Derivation for configuration (b)

The plane $\mathcal{P}$ is a facet of $\mathcal{S}$ defined by its three vertices $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$. Its parameters are:

- $\mathbf{n} = (\mathbf{x}_1 - \mathbf{x}_2) \times (\mathbf{x}_1 - \mathbf{x}_3)$
- $\mathbf{p} = \mathbf{x}_1$.

The edge is the intersection of two bisectors defined by three constant seeds $\mathbf{y}_1$, $\mathbf{y}_2$ and $\mathbf{y}_3$. Its parameters are:

- $\mathbf{e} = (\mathbf{y}_1 - \mathbf{y}_2) \times (\mathbf{y}_1 - \mathbf{y}_3)$
- $\mathbf{v} = \mathbf{c}$

Note that in this case $\mathbf{c}$ can be used to define a point on the line since the line is constant. Therefore:

$$\mathrm{d}\mathbf{c} = \mathrm{d}u\mathbf{e}$$
$$\mathrm{d}\mathbf{n} = \mathrm{d}\mathbf{x}_1 \times (\mathbf{x}_2 - \mathbf{x}_3) + \mathrm{d}\mathbf{x}_2 \times (\mathbf{x}_3 - \mathbf{x}_1) + \mathrm{d}\mathbf{x}_3 \times (\mathbf{x}_1 - \mathbf{x}_2)$$

$$k^2\mathrm{d}u = k(\mathrm{d}\mathbf{p}.\mathbf{n} + (\mathbf{p} - \mathbf{v}).\mathrm{d}\mathbf{n}) - \overbrace{(\mathbf{v} - \mathbf{p}).\mathbf{n}}^{(\mathbf{v},\mathbf{p}) \in \mathcal{P}^2 \Rightarrow 0}\mathrm{d}k$$
$$k\mathrm{d}u = (\mathbf{p} - \mathbf{v}).\mathrm{d}\mathbf{n} + \mathrm{d}\mathbf{x}_1.\mathbf{n}$$
$$= (\mathbf{x}_1 - \mathbf{v}).\sum_{i=1}^{3} \mathrm{d}\mathbf{x}_i \times (\mathbf{x}_{(i+1)[3]} - \mathbf{x}_{(i+2)[3]}) + \mathrm{d}\mathbf{x}_1.\mathbf{n}$$
$$= \sum_{i=1}^{3} \mathrm{d}\mathbf{x}_i.((\mathbf{x}_{(i+1)[3]} - \mathbf{x}_{(i+2)[3]}) \times (\mathbf{x}_1 - \mathbf{v})) + \mathrm{d}\mathbf{x}_1.\mathbf{n}$$

The Jacobians of u are:

$$\frac{\mathrm{d}u}{\mathrm{d}\mathbf{x}_1} = \frac{1}{k}[(\mathbf{x}_2 - \mathbf{x}_3) \times (\mathbf{x}_1 - \mathbf{v}) + \mathbf{n}]^t$$
$$\frac{\mathrm{d}u}{\mathrm{d}\mathbf{x}_2} = \frac{1}{k}[(\mathbf{x}_3 - \mathbf{x}_1) \times (\mathbf{x}_1 - \mathbf{v})]^t$$
$$\frac{\mathrm{d}u}{\mathrm{d}\mathbf{x}_3} = \frac{1}{k}[(\mathbf{x}_1 - \mathbf{x}_2) \times (\mathbf{x}_1 - \mathbf{v})]^t$$

And finally the Jacobians of $\mathbf{c}$ are

$$\frac{\mathrm{d}\mathbf{c}}{\mathrm{d}\mathbf{x}_i} = \mathbf{e}\frac{\mathrm{d}u}{\mathrm{d}\mathbf{x}_i}$$

References

1. (2011) CGAL. http://www.cgal.org
2. Amenta N, Bern M (1999) Surface reconstruction by Voronoi filtering. Discret Comput Geom 22(4):481–504
3. Bommes D, Lévy B, Pietroni N, Puppo E, Silva C, Tarini M, Zorin D (2012) State of the art in quad meshing (submitted)

4. Cheng KS, Wang W, Qin H, Wong KY, Yang HP, Liu Y (2007) Design and analysis of methods for subdivision surface fitting. IEEE TVCG 13(5)
5. Cheng KSD, Wang W, Qin H, Wong KYK, Yang HP, Liu Y (2004) Fitting subdivision surfaces using SDM. In: Pacific graphics conference proceedings, pp 16–24
6. Cignoni P, Rocchini C, Scopigno R (1998) Metro: measuring error on simplified surfaces. Comp Graphics Forum 17(2):167–174
7. Delingette H (1999) General object reconstruction based on simplex meshes. IJCV 32(2):111–146
8. Du Q, Faber V, Gunzburger M (1999) Centroidal Voronoi tessellations: applications and algorithms. SIAM Rev 41(4):637–676
9. Eck M, Hoppe H (1996) Automatic reconstruction of B-spline surfaces of arbitrary topological type. In: Proceedings of ACM SIGGRAPH, pp 325–334
10. Floater M (1997) Parametrization and smooth approximation of surface triangulations. Comput Aided Geom Des 14(3):231–250
11. He Y, Wang H, Fu CW, Qin, H (2009) A divide-and-conquer approach for automatic polycube map construction. Comput Graphics 33(3):369–380
12. Kobbelt L, Vorsatz J, Labsik U, Seidel HP (2001) A shrink wrapping approach to remeshing polygonal surfaces. Comput Graphics Forum 18(3)
13. Kovacs D, Myles A, Zorin D (2010) Anisotropic quadrangulation. In: Proceedings of the 14th ACM Symposium on solid and physical modeling, pp 137–146
14. Lévy B, Liu Y (2010) L_p centroidal Voronoi tessellation and its applications. ACM TOG (Proc SIGGRAPH) 29(4):Article No. 119
15. Li WC, Ray N, Lévy B (2006) Automatic and interactive mesh to T-spline conversion. In: Symposium on geometry processing, pp 191–200
16. Lin J, Jin X, Fan Z, Wang CCL (2008) Automatic polycube-maps. In: Proceedings of the 5th international conference on advances in geometric modeling and processing, pp 3–16
17. Liu D, Nocedal J (1989) On the limited memory BFGS method for large scale optimization. Math Program Ser A and B 45(3):503–528
18. Liu Y (2010) HLBFGS: an hybrid L-BFGS optimization framework. http://research.microsoft.com/en-us/UM/people/yangliu/software/HLBFGS/
19. Liu Y, Wang W, Lévy B, Sun F, Yan DM, Lu L, Yang C (2009) On centroidal Voronoi tessellation—energy smoothness and fast computation. ACM Trans Graphics 28(4):Article No. 101
20. Ma W, Ma X, Tso SK, Pan Z (2004) A direct approach for subdivision surface fitting. Compt Aided Des 36(6)
21. Marinov M, Kobbelt L (2005) Optimization methods for scattered data approximation with subdivision surfaces. Graph Models 67(5):452–473
22. Nocedal J, Wright S (2006) Numerical optimization. Springer, Berlin
23. Pottmann H, Leopoldseder S (2003) A concept for parametric surface fitting which avoids the parametrization problem. Comp Aided Geom Des 20(6)
24. Ray N, Li W, Lévy B, Scheffer A, Alliez P (2006) Periodic global parameterization. ACM Trans Graphics 25(4):1460–1485
25. Rusinkiewicz S, Levoy M (2001) Efficient variants of the ICP algorithm. IEEE third international conference on 3-D digital imaging and modeling, pp 145–152.
26. Schreiner J, Asirvatham A, Praun E, Hoppe H (2004) Inter-surface mapping. Proc SIGGRAPH ACM TOG 23(3):870–877
27. Tarini M, Hormann K, Cignoni P, Montani C (2004) Polycube-maps. Proc of SIGGRAPH ACM TOG 23(3):853–860
28. Wang H, He Y, Li X, Gu X, Qin H (2008) Polycube splines. Comp Aided Des 40(6):721–733
29. Wang W, Pottmann H, Liu Y (2006) Fitting B-spline curves to point clouds by curvature-based SDM. ACM Trans Graphics 25(2):214–238
30. Yan DM, Lévy B, Liu Y, Sun F, Wang W (2009) Isotropic remeshing with fast and exact computation of restricted Voronoi diagram. Proc SGP Comp Graphics Forum 28(5):1445–1454
31. Yao C, Chu H, Ju T, LEE T (2009) Compatible quadrangulation by sketching. Comput Animat Virtual Worlds 20(2-3):101–109
32. Yeh IC, Lin CH, Sorkine O, Lee TY (2011) Template-based 3D model fitting using dual-domain relaxation. IEEE TVCG 17(8):1178–1190