Jet_fitting_3: A Generic C++ Package for Estimating the Differential Properties on Sampled Surfaces via Polynomial Fitting

Frédéric Cazals, Marc Pouget

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

`Jet_fitting_3`: *A Generic C++ Package for Estimating the Differential Properties on Sampled Surfaces via Polynomial Fitting*

Frédéric Cazals — Marc Pouget

N° 6093

Janvier 2007

Thème SYM

ISSN 0249-6399 ISRN INRIA/RR--6093--FR+ENG

Jet_fitting_3: A Generic C++ Package for Estimating the Differential Properties on Sampled Surfaces via Polynomial Fitting

Frédéric Cazals [*] , Marc Pouget [†]

Thème SYM — Systèmes symboliques
Projets Geometrica et Vegas

Rapport de recherche n° 6093 — Janvier 2007 — 18 pages

Abstract: Surfaces of $\mathbb{R}^3$ are ubiquitous in science and engineering, and estimating the local differential properties of a surface discretized as a point cloud or a triangle mesh is a central building block in Computer Graphics, Computer Aided Design, Computational Geometry, Computer Vision. One strategy to perform such an estimation consists of resorting to polynomial fitting, either interpolation or approximation, but this route is difficult for several reasons: choice of the coordinate system, numerical handling of the fitting problem, extraction of the differential properties.

This paper presents a generic C++ software package solving these problems. On the theoretical side and as established in a companion paper, the interpolation and approximation methods provided achieve the best asymptotic error bounds known to date. On the implementation side and following state-of-the-art coding rules in Computational Geometry, genericity of the package is achieved thanks to three template classes accounting for (a) the type of the input points (b) the internal geometric computations and (c) the linear algebra operations. An instantiation within the Computational Geometry Algorithms Library (CGAL, version 3.3) and using CLAPACK is also provided.

Key-words: Differential Geometry, Computational Geometry, Sampled Surfaces, C++ design.

[*] INRIA Sophia-Antipolis, Geometrica project
[†] LORIA, INRIA Lorraine, Vegas project

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,
615, rue du Jardin Botanique, BP 101, 54602 Villers-Lès-Nancy (France)
Téléphone : +33 3 83 59 30 00 — Télécopie : +33 3 83 27 83 19

`Jet_fitting_3`: Un package C++ générique pour l'estimation des propriétés différentielles sur des surfaces échantillonnées

Résumé : Les surfaces de $\mathbb{R}^3$ sont des objets centraux en sciences et en ingénierie, et la question de l'estimation de quantités différentielles locales d'une surface lisse connue sous forme d'un nuage de points ou d'un maillage se pose de façon récurrente en infographie, CAGD, géométrie algorithmique ou encore vision. Une stratégie pour effectuer de telles estimations consiste à ajuster un polynôme, par interpolation ou approximation, mais celle-ci est délicate pour plusieurs raisons: le choix du système de coordonnées, les problèmes numériques, et enfin l'extraction des quantités différentielles.

Ce papier présente un package C++ générique résolvant ces problèmes. D'un point de vue théorique, et comme cela est établi dans un travail joint, les méthodes d'interpolation et d'approximation implantées correspondent aux équivalents asymptotiques les plus fins connus à ce jour. Suivant les bonnes pratiques de la géométrie algorithmique, l'implementation C++ est générique grâce aux trois classes template suivantes: (a) la classe correspondant au type des points donnés en entrée, (b) la classe utilisée pour les représentations géométriques internes, (c) la classe fournissant les routines d'algèbre linéaire. Une instantiation de ces classes au sein de CGAL 3.3, basée sur CLAPACK pour l'algèbre linéaire, est mise à disposition.

Mots-clés : C++, Géométrie Différentielle, Géométrie computationnelle, Surfaces Echantillonnées

1 Introduction

1.1 Estimating Differential Quantities

Surface segmentation in Computer Aided Geometric Design and reverse engineering, surface smoothing / denoising in Computer Graphics, or surface reconstruction in Computational Geometry are example applications where one faces the problem of estimating the differential properties of a smooth surface known through a discretization —collection of sample points or triangular mesh. Depending on the particular constraints, first order (tangent space), second order (principal curvatures and directions), or third order informations (derivatives of the principal curvatures along the curvature lines) are required. Due to its importance, this problem triggered an important activity in several communities, and the reader is referred to [Pet01] for a report up to 2001. Given a point at which the differential quantities are sought, all methods developed require gathering points or triangles in a neighborhood of that point. Methods may be classified in two families. On one hand, strategies relying on *discrete differential geometry* only use the information provided by the mesh [PP93, DMSB00, CSM03]. On the other hand, fitting methods rely on smooth differential geometry and approximation, and hence directly accommodate point clouds.

Each category has a champion in terms theoretical guarantees provided by the estimation procedure, when the sampling becomes infinitely dense. In the former category, the normal cycle theory used in [CSM03] provides a linear error bound on the estimate of the integral of the Weingarten map of surface discretized by a mesh. In the later, the polynomial fitting strategy developed by the authors in [CP05a] allows a precise specification of the coefficients of the local representation of the surface as a height function —see Thm 1 below. The polynomial fitting method has three advantages : first, differential properties of any order can be retrieved; second, the coefficients estimated feature the best error bounds known so far; third, it accommodates point clouds —a mesh is not mandatory.

1.2 Contributions and Paper Overview

This paper describes the C++ software package `Jet_fitting_3` which implements the polynomial fitting method [CP05a], that is (i) the mathematics and numerical issues involved (ii) the C++ implementation to be released with the 3.3 version of the Computational Geometry Algorithms Library (CGAL) `www.cgal.org`. (CGAL is the standard library for robust and efficient geometric software. A package is accepted for integration after a review process similar to that of a journal paper, and upon acceptation by the Editorial Board. Release 3.3 is scheduled for 04/2007.)

The paper is organized as follows. Section 2 specifies the objects manipulated and recalls the principles of the fitting algorithm, while algorithmic and mathematical details are developed in section 3. Software design issues are discussed in section 4, while illustrations are provided in section 5.

2 Estimating differential properties by polynomial fitting

2.1 Jets, Monge Form and Polynomial Fitting

Smooth surfaces, d-jets and the Monge form. To present the method, we shall need the following notions. Consider a smooth surface. About one of its points, consider a coordinate system whose z-axis is not contained in the tangent space of the surface at that point. In such a frame, the surface can locally be written as the graph of a bivariate function. Letting *h.o.t* stand for *higher order terms*, one has :

$$z(x,y) = J_{B,d}(x,y) + h.o.t \quad \text{with } J_{B,d}(x,y) = \sum_{k=0}^{k=d}(\sum_{i=0}^{i=k} \frac{B_{k-i,i}x^{k-i}y^i}{i!(k-i)!}). \tag{1}$$

The degree d polynomial $J_{B,d}$ is the Taylor expansion of the function z, we call it its d-jet. Notice that a d-jet contains $N_d = (d+1)(d+2)/2$ coefficients.

While the first order properties of the surface are related to its tangent space, its second order properties are encoded by the principal curvatures and directions [dC76]. When the principal curvatures are distinct —the point is not a so-called umbilic, the principal directions d_1, d_2 are well defined, and these (non oriented) directions together with the normal vector n define two direct orthonormal frames. If v_1 is a unit vector of direction d_1, there exists a unique unit vector v_2 so that (v_1, v_2, n) is direct; and the other possible frame is $(-v_1, -v_2, n)$. These coordinate systems are called the *Monge* frames, and in any of them, the Taylor expansion of the surface has the following canonical expression, called the *Monge form*:

$$z(x,y) = \frac{1}{2}(k_1x^2 + k_2y^2) + \frac{1}{6}(b_0x^3 + 3b_1x^2y + 3b_2xy^2 + b_3y^3) \tag{2}$$

$$+ \frac{1}{24}(c_0x^4 + 4c_1x^3y + 6c_2x^2y^2 + 4c_3xy^3 + c_4y^4) + h.o.t \tag{3}$$

Recall that coefficients k_1, k_2 are the principal curvatures, b_0, b_3 are the directional derivatives of k_1, k_2 along their respective curvature lines, while b_1, b_2 are the directional derivatives of k_1, k_2 along the other curvature lines.

The Monge coordinate system can be computed from any d-jet with $d \geq 2$, and so are the Monge coefficients. The Monge representation characterizes the surface in a canonical way, and is the output of our algorithm.

Interpolating or approximating the d-jet. Consider a sampled smooth surface, and assume we are given a collection of points P about a given sample p. We aim at estimating the order-d differential properties of the surface at point p from the point set $P^+ = P \cup \{p\}$ –we note $N = \mid P^+ \mid$. The estimation algorithms developed in [CP05a] consist of fitting the d-jet, using bivariate polynomial interpolation or approximation on the point set P^+. More

precisely, in a well chosen coordinate system, the fitting consists of finding the coefficients $A_{i,j}$ of the degree d polynomial $J_{A,d} = \sum_{k=0}^{k=d}(\sum_{i=0}^{i=k} \frac{A_{k-i,i}x^{k-i}y^i}{i!(k-i)!})$.

Denote $p_i = (x_i, y_i, z_i),\ i = 1, \ldots, N$, the coordinates of the sample points of P^+. For interpolation the linear equations to solve are $J_{A,d}(x_i, y_i) = z_i,\ i = 1, \ldots, N$, and for approximation one has to minimize $\sum_{i=1}^{N}(J_{A,d}(x_i, y_i) - z_i)^2$. To provide the linear algebra formulation of the problem, define:

$$\begin{aligned} A =&(A_{0,0}, A_{1,0}, A_{0,1}, \ldots, A_{0,d})^T, \\ Z =&(z_1, z_2, \ldots, z_N)^T, \\ M =&(1, x_i,\ y_i,\ \frac{x_i^2}{2}, \ldots,\ \frac{x_i y_i^{d-1}}{(d-1)!},\ \frac{y_i^d}{d!})_{i=1,\ldots,N}. \end{aligned}$$

Using these notations, solving the fitting problems by interpolation is tantamount to solving $MA = Z$, while the approximation strategy requires solving $\min ||MA - Z||_2$.

The following theorem precisely states the order of convergence of the polynomial fitting method. Given a parameter h measuring the *sampling density*, the following theorem, providing the best asymptotic estimates known to date, is proved in [CP05a] :

Theorem. 1. *A polynomial fitting of degree d estimates any k^{th}-order differential quantity to accuracy $O(h^{d-k+1})$:*

$$A_{i,k-i} = B_{i,k-i} + O(h^{d-k+1}). \tag{4}$$

In particular:

- *the coefficients of the unit normal vector are estimated with accuracy $O(h^d)$.*
- *the coefficients of the second fundamental form and the shape operator are approximated with accuracy $O(h^{d-1})$, and so are the principal curvatures and directions (as long as they are well defined, i.e. away from umbilics).*

Algorithm. Based on the above concepts, the algorithm consists of four steps.

1. We perform a principal component analysis (PCA) on P^+. This analysis outputs three orthonormal eigenvectors and the associated eigenvalues. If the surface is well sampled, we expect the PCA to provide one small and two large eigenvalues, the eigenvector associated to the small one approximating the normal vector.

2. We perform a change of coordinates to move the samples into the coordinate system defined by the PCA eigenvectors. We then resort to polynomial fitting, so as to either interpolate or approximate the d-jet $J_{B,d}$ of the surface. This fitting reduces to linear algebra operations.

3. From the d-jet $J_{A,d}$, we compute the Monge basis (d_1, d_2, n).

4. Finally, we compute the Monge coefficients.

For the fitting, we do not assume the z-axis of the fitting to be the normal of the surface. Hence we keep the first order coefficients of the polynomial $J_{A,d}$. It is proved that such a choice improves the estimations [CP05a].

Note that we do not aim at identifying exactly special points such as umbilics where both principal curvatures are equal. This would require the original surface and the fitted surface to coincide. This implying in turn that the original surface is a graph of a bivariate polynomial of a lower degree than the fitted polynomial and that the coordinate system used for the fitting has the same height direction.

2.2 Degenerate cases

For an interpolation of degree d, there are special configuration of the sample points for which the interpolation problem is not well defined —if the point lie on an algebraic curve of degree at most d. In such a case, the condition number of the system to solve is infinite. In practice, this condition number is passed to the user for an assessment of the accuracy of the result.

3 Mathematical and Algorithmic Details

In this section, we detail the mathematics involved, in order to justify the C++ design choices.

As illustrated on Fig. 3, we deal with three relevant direct orthonormal basis: the world-basis W, fitting-basis F, and the Monge-basis $M = (d_1, d_2, n)$. If X, Y are two such basis, the matrix used to transport a vector from X to Y is denoted $P_{X\to Y}$. If V_X (resp. V_Y) are the coordinates of a vector V with respect to the basis X (resp. Y), then : $V_Y = P_{X\to Y} V_X$. Note that as all considered basis are orthonormal, the matrix to change from one to another is orthonormal, that is its inverse is its transpose : $P_{Y\to X} = P_{X\to Y}^{-1} = P_{X\to Y}^{T}$. For computations, note that the columns of $P_{X\to Y}$ are the coordinates of the vectors of X in the basis Y.

3.1 Computing a Basis for the Fitting

Input : sample points
Output : the fitting-basis

We perform a Principal Component Analysis of the sample points, which requires a linear algebra method to perform an eigen analysis of a 3×3 symmetric matrix. This analysis gives an orthonormal basis whose z-axis is chosen as the eigenvector associated to the smallest eigenvalue [1]. Notice that one may have to swap the orientation of a vector to get a direct

[1] Another possibility is to choose as z-axis the axis of the world-basis with the least angle with the axis determined with the PCA. Then the change of basis reduces to a permutation of axis.

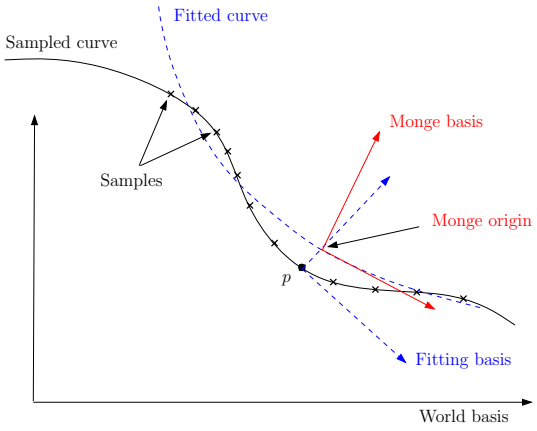

Figure 1: The three basis involved in the estimation.

basis. We call fitting-basis and denote F this basis. Notice also that the quality of the sampling is encoded in the PCA eigenvalues, as a good sampling is characterized by a small eigenvalue (indicating the normal vector) and two similar bigger ones (indicating the tangent space).

3.2 Solving the Interpolation / Approximation Problem

Input : sample points, fitting-basis
Output : coefficients $A_{i,j}$ of the fitted polynomial (in the fitting-basis)

The fitting problem is solved in the fitting basis, whose origin is the point p. Therefore, sample points are expressed in this frame thanks to a translation $(-p)$ and a multiplication by $P_{W\to F}$.

We solve the system $MA = Z$ —in the least square sense for approximation. There is a preconditioning of the matrix M so as to improve the condition number. Assuming the $\{x_i\}$, $\{y_i\}$ are of order h, the preconditioning consists of performing a column scaling by dividing each monomial $x_i^k y_i^l$ by h^{k+l}. The parameter h is chosen as the mean value of the $\{x_i\}$ and $\{y_i\}$. In other words, the new system is $M'Y = (MD^{-1})(DA) = Z$ with D the diagonal matrix $D = (1, h, h, h^2, \ldots, h^d, h^d)$, so that the solution A of the original system is $A = D^{-1}Y$.

There is always a single solution to the system $M'Y = Z$ since for under constrained systems we also minimize $||Y||_2$. The method uses a singular value decomposition of the $N \times N_d$ matrix $M' = USV^T$, where U is a $N \times N$ orthogonal matrix, V is a $N_d \times N_d$ orthogonal matrix and S is a $N \times N_d$ matrix with the singular values on its diagonal. Denoting r the rank of M', we can decompose $S = \begin{pmatrix} D_r & 0_{r,\ N_d-r} \\ 0_{N-r,\ r} & 0_{N-r,\ N_d-r} \end{pmatrix}$. The number r, which is the number of non zero singular values, is strictly lower than N_d if the system is

under constrained. In any case, the unique solution which minimize $||Y||_2$ is given by :

$$Y = V \begin{pmatrix} D_r^{-1} & 0_{N_d-r,\ r} \\ 0_{r,\ N-r} & 0_{N_d-r,\ N-r} \end{pmatrix} U^T Z. \tag{5}$$

The condition number of the matrix M' is the ratio of the maximal and the minimal singular values. It is infinite if the system is under constrained, that is the smallest singular value is zero.

Remark. A system $MA = Z$ can be solved using a variety of methods. A QR decomposition can be substituted to the SVD. One can also use the normal equation $M^T MA = MTZ$ and apply methods for square systems such as LU, QR or Cholesky since $M^T M$ is symmetric definite positive when M has full rank. The advantages of the SVD is that it works directly on the rectangular system and gives the condition number of the system. For more on these alternatives, see [GvL83].

The method to solve the system is provided by the template parameter $SvdTraits$ so it is up to the user to choose amongst these techniques. See section 4.

3.3 Computing the Principal Curvatures and Directions

Input : coefficients $A_{i,j}$ of the fitted polynomial
Output : Monge-basis wrt fitting-basis and world-basis

The surface on which we are computing local differential quantities is the graph of $J_{A,d}$ and the point on this surface where the estimation is sought has coordinates $(0, 0, J_{A,d}(0, 0)$. Computations are done in the fitting-basis. The partial derivatives, evaluated at $(x, y) = (0, 0)$, of the fitted polynomial $J_{A,d}(x, y)$ are $A_{i,j} = \frac{\partial^{i+j} J_{A,d}}{\partial^i x \partial^j y}$, hence

$$J_{A,d}(x, y) = A_{0,0} + A_{1,0}x + A_{0,1}y + \frac{1}{2}(A_{2,0}x^2 + 2A_{1,1}xy + A_{0,2}y^2) \tag{6}$$

$$+ \frac{1}{6}(A_{3,0}x^3 + 3A_{2,1}x^2y + \ldots) + \ldots \tag{7}$$

Equipped with these coefficients, differential quantities up to order two are retrieved as follows:

- The origin, that is the point of the fitted surface where the estimation is performed, is $(0, 0, A_{0,0})$.
- The normal is $n = (-A_{1,0}, -A_{0,1}, 1)/\sqrt{1 + {A_{1,0}}^2 + {A_{0,1}}^2}$.
- Curvature related properties are retrieved resorting to standard differential calculus [dC76, Chap. 3]. More precisely, the Weingarten operator $W = -I^{-1}II$ is first computed in the basis of the tangent plane $\{(1, 0, A_{1,0}), (0, 1, A_{0,1})\}$, where I, II are

the first and second fundamental forms :

$$I = \begin{pmatrix} 1 + A_{1,0}^2 & A_{1,0}A_{0,1} \\ A_{1,0}A_{0,1} & 1 + A_{0,1}^2 \end{pmatrix} \quad II = \begin{pmatrix} \frac{A_{2,0}}{\sqrt{1+{A_{1,0}}^2+{A_{0,1}}^2}} & \frac{A_{1,1}}{\sqrt{1+{A_{1,0}}^2+{A_{0,1}}^2}} \\ \frac{A_{1,1}}{\sqrt{1+{A_{1,0}}^2+{A_{0,1}}^2}} & \frac{A_{0,2}}{\sqrt{1+{A_{1,0}}^2+{A_{0,1}}^2}} \end{pmatrix} \tag{8}$$

We compute an orthonormal basis of the tangent plane using the Gram-Schmidt algorithm. Denoting P the matrix of the orthonormal vectors in the fitting basis, the matrix of the Weingarten map in this orthonormal basis of the tangent space is given by $W' = P^{-1}WP$. In this orthonormal basis, the matrix of the Weingarten map is symmetric and we diagonalize it. One finally gets the principal curvatures which are the eigenvalues of W, and the associated principal directions d_1, d_2.

This gives an orthonormal direct basis $(d_1, d_2, n = d_1 \wedge d_2)$ called the Monge-basis those vectors are given by their coordinates in the fitting-basis.

The coordinates of d_1, d_2 and n expressed in the world-basis are obtained by multiplying their coordinates in the fitting-basis by $P_{F\to W}$. The same holds for the origin point which has in addition to be translated by p, i.e. the coordinates of the origin point are $P_{F\to W}(0, 0, A_{0,0}) + p$.

3.4 Computing higher order Monge coefficients

Input : coefficients $A_{i,j}$ of the fitted polynomial, Monge-basis wrt fitting-basis ($P_{F\to M}$)
Output : third and fourth order coefficients of the Monge form

Having established the equation of the fitted polynomial in the fitting basis, we derive an implicit equivalent, say $f(x, y, z) = 0$, in the Monge basis. Then, from the implicit function theorem [dC76], we know that the surface $f(x, y, z) = 0$ can locally be written as the graph of a height function $z = g(x, y)$. The Taylor expansion of g at $(0, 0)$ are the Monge coefficients sought. More formally:

Theorem. 2. *Third and fourth order coefficients of the Monge form with respect to derivatives of the implicit equation of the fitted polynomial $f(x, y, z) = 0$ in the Monge basis are*

given by:

$$b_0 = g_{3,0} = -\frac{f_{3,0,0} f_{0,0,1} - 3\, f_{1,0,1} f_{2,0,0}}{f_{0,0,1}^2}$$

$$b_1 = g_{2,1} = -\frac{-f_{0,1,1} f_{2,0,0} + f_{2,1,0} f_{0,0,1}}{f_{0,0,1}^2}$$

$$b_2 = g_{1,2} = -\frac{f_{1,2,0} f_{0,0,1} - f_{1,0,1} f_{0,2,0}}{f_{0,0,1}^2}$$

$$b_3 = g_{0,3} = -\frac{f_{0,3,0} f_{0,0,1} - 3\, f_{0,1,1} f_{0,2,0}}{f_{0,0,1}^2}$$

$$c_0 = g_{4,0} = \frac{-3\, f_{0,0,2} f_{2,0,0}^2 - f_{4,0,0} f_{0,0,1}^2 + 6\, f_{2,0,1} f_{0,0,1} f_{2,0,0} + 4\, f_{1,0,1} f_{0,0,1} f_{3,0,0} - 12\, f_{1,0,1}^2 f_{2,0,0}}{f_{0,0,1}^3}$$

$$c_1 = g_{3,1} = \frac{-6\, f_{1,0,1} f_{0,1,1} f_{2,0,0} + 3\, f_{0,0,1} f_{1,1,1} f_{2,0,0} + 3\, f_{1,0,1} f_{0,0,1} f_{2,1,0} + f_{0,1,1} f_{0,0,1} f_{3,0,0} - f_{3,1,0} f_{0,0,1}^2}{f_{0,0,1}^3}$$

$$\begin{aligned} c_2 = g_{2,2} = -(&f_{2,2,0} f_{0,0,1}^2 - f_{2,0,1} f_{0,0,1} f_{0,2,0} - 2\, f_{1,0,1} f_{0,0,1} f_{1,2,0} + 2\, f_{1,0,1}^2 f_{0,2,0} \\ &- f_{0,2,1} f_{0,0,1} f_{2,0,0} + 2\, f_{0,1,1}^2 f_{2,0,0} - 2\, f_{0,1,1} f_{0,0,1} f_{2,1,0} + f_{0,0,2} f_{0,2,0} f_{2,0,0}) / f_{0,0,1}^3 \end{aligned}$$

$$c_3 = g_{1,3} = -\frac{f_{1,3,0} f_{0,0,1}^2 + 6\, f_{1,0,1} f_{0,1,1} f_{0,2,0} - 3\, f_{0,1,1} f_{0,0,1} f_{1,2,0} - 3\, f_{1,1,1} f_{0,0,1} f_{0,2,0} - f_{1,0,1} f_{0,0,1} f_{0,3,0}}{f_{0,0,1}^3}$$

$$c_4 = g_{0,4} = -\frac{f_{0,4,0} f_{0,0,1}^2 + 3\, f_{0,0,2} f_{0,2,0}^2 - 6\, f_{0,2,1} f_{0,0,1} f_{0,2,0} - 4\, f_{0,1,1} f_{0,0,1} f_{0,3,0} + 12\, f_{0,1,1}^2 f_{0,2,0}}{f_{0,0,1}^3}$$

Proof. The implicit equation of the fitted polynomial surface in the fitting-basis with origin the point $(0, 0, A_{0,0})$ is $Q = 0$ with

$$Q = -w - A_{0,0} + \sum_{i,j} \frac{A_{i,j} u^i v^j}{i! j!}. \tag{9}$$

Notice the term $-A_{0,0}$, which stems from the fact that the surface is translated so as to go through the intersection with the z-axis.

The equation in the Monge-basis is obtained by substituting (u, v, w) by $P_{M \to F}(x, y, z)$, let us denote $f(x, y, z)$ this implicit equation. By the implicit function Thm and the definition of the Monge-basis, we locally have at $(0, 0, 0)$

$$f(x, y, z) = 0 \Leftrightarrow z = g(x, y), \tag{10}$$

and the Taylor expansion of g at $(0, 0)$ are the Monge coefficients sought. Let us denote the partial derivatives evaluated at the origin of f and g by $f_{i,j,k} = \frac{\partial^{i+j+k} f}{\partial^i x \partial^j y \partial^k z}$ and $g_{i,j} = \frac{\partial^{i+j} g}{\partial^i x \partial^j y}$. By the definition of the Monge basis, one has $g_{0,0} = g_{1,0} = g_{0,1} = g_{1,1} = 0$ and $g_{2,0} = k_1$, $g_{0,2} = k_2$. The partial derivative of order n of f depends on the matrix $P_{M \to F}$ and the partial derivatives of order at most n of $J_{A,d}$ (that is coefficients $A_{i,j}$ for $i + j \leq n$). The third and fourth order Monge coefficients of are computed applying the chain rule while differentiating the equation $f(x, y, g(x, y)) = 0$. For instance, differentiating once, twice and

thrice with respect to x gives

$$f_{1,0,0} + f_{0,0,1}g_{1,0} = 0 \tag{11}$$

$$f_{2,0,0} + 2f_{1,0,1}g_{1,0} + f_{0,0,2}g_{1,0}^2 + f_{0,0,1}g_{2,0} = 0 \tag{12}$$

$$f_{3,0,0} + 3f_{2,0,1}g_{1,0}^2 + 3f_{1,0,1}g_{2,0} + 3f_{1,0,2}g_{1,0}^2 + f_{0,0,3}g_{1,0}^3 + 3f_{0,0,2}g_{1,0}g_{2,0} + f_{0,0,1}g_{3,0} = 0 \tag{13}$$

In addition, at $(x, y) = (0, 0)$, equation (12) gives $g_{2,0} = -f_{2,0,0}/f_{0,0,1}$ and equation (13) leads to $f_{0,0,1}g_{3,0} = -(f_{3,0,0} + 2f_{1,0,1}g_{2,0})$, we then obtain equation (9). Other equations are obtained by similar computations. □

4 Software Design

In this section, we outline the C++ design and refer the reader to the CGAL manual [CP07] for details. For readers not familiar with generic geometric code, we begin with some generalities.

4.1 CGAL: templated geometric code

CGAL consists of generic classes implementing a number of geometric algorithms. Each implementation distinguishes the combinatorial from the numerics. For example, a triangulated surface consists of a collection of triangles with the proper incidences between them (the combinatorial part describing the surface topology), and of a collection of points (coordinates) associated to the vertices of the triangulation (the numerical part). As an algorithm operating on a surface should accommodate several types of points, a high level of genericity is achieved thanks to function and class templates. Quoting CGAL's manual:

≪ The algorithms in CGAL's basic library are implemented as function templates or class templates, usually having a template parameter whose name contains the word Traits. This template parameter represents a concept and so has a corresponding set of requirements that define the interface between the algorithm and the geometric (or numeric) primitives it uses. Any concrete class that serves as a model for this concept is a traits class for the given algorithm or data structure. ≫

4.2 User interface specifications

In a nutshell, the `Jet_fitting_3` package has one class, `Monge_via_jet_fitting`, to perform the computation via the method `operator()`. The return value of this method is the nested class `Monge_form` which stores the Monge basis and coefficients.

Input parameters. The fitting strategy performed by the class `Monge_via_jet_fitting` requires the following parameters:

- the degree d of the fitted polynomial ($d \geq 1$),
- the degree d' of the Monge coefficients sought, with $1 \leq d' \leq \min(d, 4)$,
- a range of N input points on the surface, with the precondition that $N \geq N_d = (d+1)(d+2)/2$. Note that if $N = N_d$, interpolation is performed; and if $N > N_d$, approximation is used.

Output. As explained in Section 1, the output consists of a coordinate system, the Monge basis, together with the Monge coefficients which are stored in the `Monge_form` class. In addition, more information on the computational issues are stored in the `Monge_via_jet_fitting` class.

The `Monge_form` class provides the following information.

- Origin. This is the point on the fitted polynomial surface where the differential quantities have been computed. In the approximation case, it differs from the input point p : it is the projection of p onto the fitted surface following the z-direction of the fitting basis.
- Monge Basis. The Monge basis (d_1, d_2, n) is orthonormal direct, and the maximal, minimal curvatures are defined wrt this basis. If the user has a predefined normal n_0 (e.g. the sample points come from an oriented mesh) then if $n_0.n > 0$ then max-min is correct; if not, i.e. $n_0.n < 0$, the user should switch to the orthonormal direct basis $(d_1', d_2', n') = (d_2, d_1, -n)$ with the maximal curvature $k_1' = -k_2$ and the minimal curvature $k_2' = -k_1$ (the method `comply_wrt_given_normal()` does this job). If $n_0.n = 0$ or is small, the orientation of the surface is clearly ill-defined, and the user may proof-check the samples used to comply with its predefined normal.
- Monge Coefficients. The coefficient of the Monge form are $(k_1, k_2 (\leq k_1), b_0, b_1, b_2, b_3, c_0, c_1, c_2, c_3, c_4)$ for $d' = 4$.

In addition, the class `Monge_via_jet_fitting` stores

- the condition number of the fitting system,
- the coordinate system of the PCA (in which the fitting is performed) and the associated eigenvalues.

4.3 Template Parameters

The `Monge_via_jet_fitting` class is actually templated by three parameters, that is `Monge_via_jet_fitting<DataKernel, LocalKernel, SvdTraits>`.

4.3.1 Template parameter `DataKernel`

This concept provides the types for the input sample points, together with $3d$ vectors and a number type. Typically, one can use `CGAL::Cartesian<double>`.

4.3.2 Template parameter `LocalKernel`

This concept defines the vector and number types used for local computations and to store the PCA basis —fitting basis.

Input points of type `DataKernel::Point_3` are converted to `LocalKernel::Point_3`. For output of the `Monge_form` class, these types are converted back to `Data_Kernel` ones. Typically, one can use `CGAL::Cartesian<double>` which is the default.

4.3.3 Template parameter `SvdTraits`

This concept provides the number, vector and matrix types for algebra operations required by the fitting method. The only method `solve` is a linear solver. A default is provided by the class `Lapack_svd` using the singular value decomposition of the CLAPACK library. But the user may pass another method of his choice by providing another traits class. See section 3.2.

4.3.4 Compatibility requirements

To solve the fitting problem, the sample points are first converted from the `DataKernel` to the `LocalKernel` —this is done using the `CGAL::Cartesian_converter`. Then change of coordinate systems and linear algebra operations are performed with this kernel. This implies that the number types `LocalKernel::FT` and `SvdTraits::FT` must be identical. Second the Monge basis and coefficients, computed with the `LocalKernel`, are converted back to the `DataKernel` (this is done using the `CGAL::Cartesian_converter` and the `CGAL::NT_converter`).

5 Illustrations

We illustrate our algorithm on triangulated surfaces. For a given point, its neighbors are easily retrieved from the connectivity of the mesh. Concerning the accuracy of the method for samplings of surfaces with known differential quantities, the reader is referred to [CP05a]. The complexity of the computation is linear with respect to the number of vertices. On an Intel 2Ghz duo core processor, an average of 5000 point estimations are performed by second for only up to second order quantities and 2500 point estimations for up to fourth order quantities. Our test models are a version of Michelangelo's David (96 Kpoints, see `graphics.stanford.edu/projects/mich`), and the following Bézier surface (66 Kpoints) which is the graph of the function :

$$\begin{aligned} h(u,v) =& 116u^4v^4 - 200u^4v^3 + 108u^4v^2 - 24u^4v - 312u^3v^4 + 592u^3v^3 - 360u^3v^2 + 80u^3v \\ &+ 252u^2v^4 - 504u^2v^3 + 324u^2v^2 - 72u^2v - 56uv^4 + 112uv^3 - 72uv^2 + 16uv. \end{aligned}$$

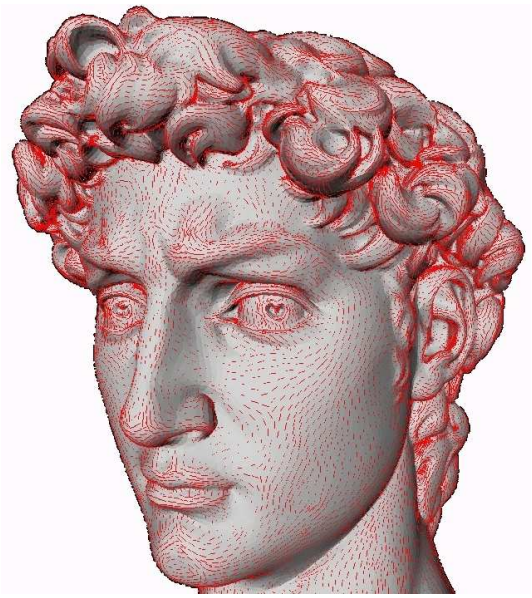

Figure 2: Principal directions associated with k_{max} scaled by k_{min}.

5.1 Principal directions and curvatures

We first illustrate the calculation of the principal directions of curvature, which requires estimating differential quantities up to order two at each vertex of the mesh. On the figures 2 and 3, a red segment in the direction of the maximal principal curvature is drawn at each vertex.

5.2 Ridges

As a second illustration, we present global features on smooth surfaces: ridges. More precisely, a ridge is a curve along which one of the principal curvatures has an extremum along its curvature line. As ridges features extrema of a principal curvatures, their extraction requires estimating differential quantities up to the third order —and actually up to the fourth order to decide whether the extremum is a maximum or a minimum. Next, these local informations have to be patched together at the surface level, so as to trace the ridges lines. See [CP05b] for the algorithm used in producing these illustrations.

As curves of extremal curvature, ridges encode important informations used in segmentation, registration, matching and surface analysis. The lines of minimum/maximum of the minimal/maximal principal curvatures are drawn on the Bézier surface, figure 5. Only part of the ridges, the most visually salient ones, called crests, are shown on the David model, see Fig. 4.

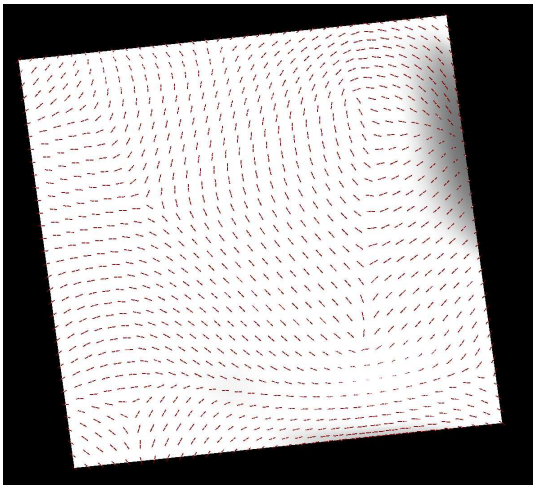

Figure 3: Principal directions associated with k_{max} on the Bézier surface.

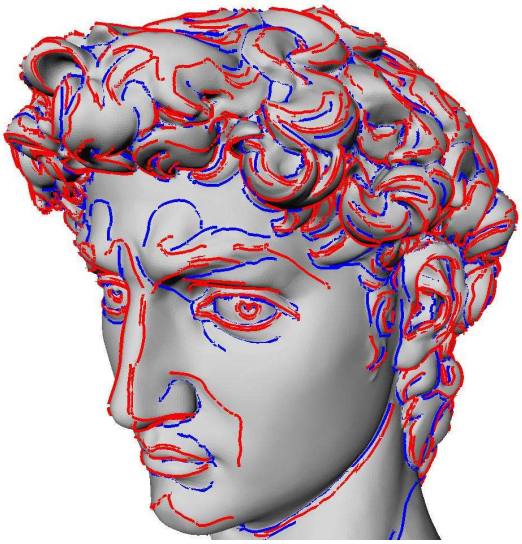

Figure 4: Crest ridges on the David. Red and blue lines distinguish crest associated to the maximal and minimal principal curvatures.

References

[CP05a] F. Cazals and M. Pouget. Estimating differential quantities using polynomial fitting of osculating jets. *Computer Aided Geometric Design*, 22(2), 2005. Con-

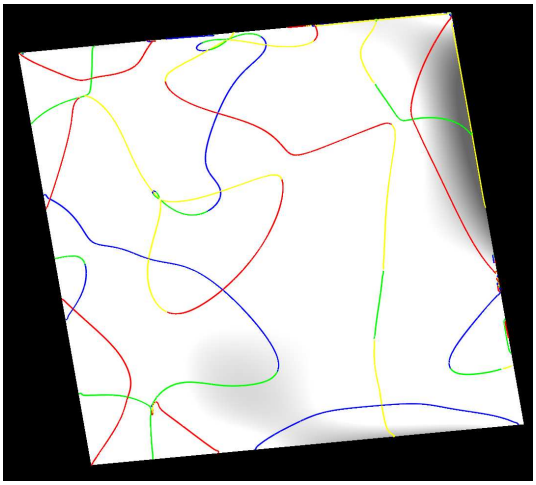

Figure 5: Ridges on the Bézier surface. Blue (green) lines are ridges associated to the maximum (resp. minimum) of the maximal principal curvature, red (yellow) lines are ridges associated to the minimum (resp. maximum) of the minimal principal curvature.

ference version: Symp. on Geometry Processing 2003.

[CP05b] F. Cazals and M. Pouget. Topology driven algorithms for ridge extraction on meshes. Rapport de recherche 5526, INRIA, 2005.

[CP07] F. Cazals and M. Pouget. Estimation of local differential properties. Cgal manual, INRIA, 2007. To be released with CGAL 3.3. See also the authors' webpages : http://www-sop.inria.fr/geometrica/team/Frederic.Cazals/software.

[CSM03] D. Cohen-Steiner and J.-M. Morvan. Restricted delaunay triangulations and normal cycle. In *ACM Symposium on Computational Geometry*, 2003.

[dC76] M. de Carmo. *Differential Geometry of Curves and Surfaces*. Prentice Hall, Englewood Cliffs, NJ, 1976.

[DMSB00] M. Desbrun, M. Meyer, P. Schröder, and A. Barr. Discrete differential-geometry operators in nD, 2000. Preprint, The Caltech Multi-Res Modeling Group.

[GvL83] G. Golub and C. van Loan. *Matrix Computations*. Johns Hopkins Univ. Press, Baltimore, MA, 1983.

[Pet01] S. Petitjean. A survey of methods for recovering quadrics in triangle meshes. *ACM Computing Surveys*, 34(2), 2001.

[PP93] U. Pinkall and K. Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2(1):15–36, 1993.

Contents

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399