Scale Space Meshing of Raw Data Point Sets

Julie Digne[1], Jean-Michel Morel[1], Charyar-Mehdi Souzani[2] and Claire Lartigue[2]
[1] CMLA, ENS Cachan, CNRS, UniverSud,
61 Avenue du Président Wilson,
F-94230 Cachan
[2] LURPA, ENS Cachan, Univ. Paris Sud 11,
61 Avenue du Président Wilson,
F-94230 Cachan

November 12, 2009

Abstract

This paper develops a scale space strategy for the meshing and segmentation of complete raw data points sets. The scale space is based on the intrinsic heat equation (mean curvature motion, MCM). A simple iterative scheme implementing MCM directly on the raw points is described, and a mathematical proof of its consistency with MCM given. Points evolved by this MCM implementation can be trivially backtracked to their initial raw position. A consequence is that the reversible MCM scheme permits to first orient, and then mesh reliably a raw textured surface. The accuracy gain is demonstrated on archaeological objects by comparisons with other meshing methods. The obtained discrete 3D scale space also complies with its traditional role: It permits to segment the original raw surface into ridges and valleys computed at coarse scales, and to draw the meaningful inflexion lines on the raw meshed surface.

1 Introduction

A growing number of applications involve creating numerical models for existing objects acquired by triangulation laser scanner. Those scanners are called triangulation laser scanner because of the triangle formed by the detected point, the laser emitter and the camera. Scanners can either produce a direct triangulation of points sampled on the surface, or the raw set of points with no connectivity information. In this paper, only raw input data will be considered, namely a set of unorganized and non-oriented points given by their x, y, z coordinates. The focus is indeed to build a highly accurate meshing method for these raw data set, with two scopes: the visualization of the finest surface details, and a robust segmentation in ridges, valleys, and scanning holes. Fig. 1 shows a high precision laser acquisition system devised for our experiments on small artistic and archaeological models. The acquisition error is around 20μ, allowing in principle to recover all of the object's texture and finest details.

The main tool that we shall use is a raw data set point smoothing operator consistent with the intrinsic heat equation. The intrinsic heat equation, or mean curvature motion (MCM), is the most standard way to smooth out a surface. It will be given an implementation that permits to backtrack the evolved surface to the initial raw data set point. At first sight, the proposed MCM implementation is an instance of the moving least square surface method, which performs a local surface regression for each point, and projects the point on the regression surface. Yet, this process will be made iterative. Indeed, mathematical and experimental arguments will show that the iterated degree one (planar) regression consistently implements the MCM. The proposed MCM implementation can therefore be summarized in a few words: it is the iterated projection on the regression plane of a spherical neighborhood. Theorem 3 will state that by these iterations, each raw data set point moves forward at the speed of the surface mean curvature in the direction of the surface normal. By the iterated projection algorithm each initial raw data point can be tracked forward in the surface smoothing process. But is can also be trivially tracked backward. As a consequence, we shall prove that all detected surface features and structures can be transported back on the raw data set point. To the best of our knowledge, several applications

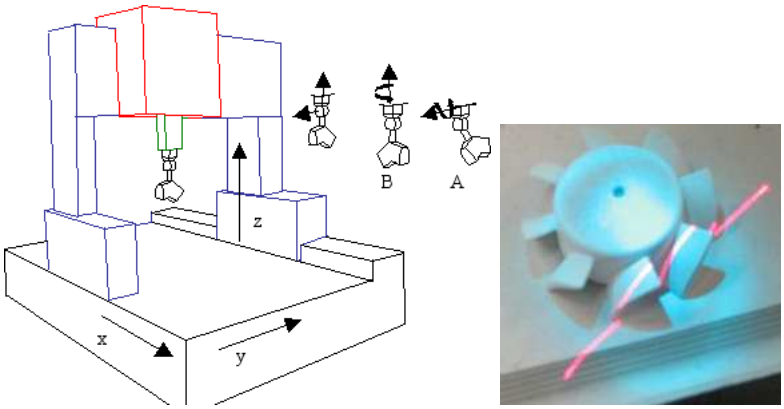


Figure 1: Our laser acquisition system at LURPA (ENS Cachan)

of this easy reverse scale space had not yet been noticed. The main application is to obtain directly a topologically faithful orientation and mesh for the whole raw data set point. A second application is the accurate detection of holes in the raw data, useful for further scanning attempts. A third application is an easy ridge-valley segmentation of the raw data point set. Comparative experiments will illustrate that a direct meshing gives poor results, while the back transported mesh gives an accurate surface rendering, whose vertices are simply all initial raw points. Obviously, such a complete mesh is not economical, but permits an accurate rendering of fine art or archaeological pieces.

The use of the mean curvature motion, forward and backward, is a direct 3D extension of the scale space paradigm in image processing. Image scale space is introduced in the founding Witkin paper [Wit83]. It consists of applying the heat equation $\frac{\partial u}{\partial t} = \Delta u$ to the image, which entails a rapid image simplification. The main image features (the edges) are detected a coarse scale (large t) and then back tracked to their fine scale position. A main difficulty of this scale space edge detection is the fact that non straight edges and corners are displaced by the heat equation. Thus the back tracking of edges and corners is not easy, and has therefore provoked a huge literature. The back propagation of a surface segmentation is comparatively easier because, as we shall see, the MCM is implementable as a point evolution from the raw data. It can therefore be followed forward and backward.

The remainder of this introduction is a review of the state of the art on raw data point set processing, and of the various computation methods of surface curvatures and normals.

1.1 Building a mesh

Given an initial oriented point cloud most processing methods (be it for rendering, for geometry detection or for any other purpose) begin with building a mesh. The methods are mostly based on defining a signed distance field [HDD+92], [KBH06], [Kaz05]. The signed distance function can be estimated at any point by computing the distance between the point and the regression plane of its k-nearest neighbors [HDD+92]. Since the neighbors are assumed previously oriented, the distance's sign is straightforward. All of these methods indeed need point orientation. However, a level set method which does not need the surface orientation was recently introduced in [ACSTD07]. Instead of looking for an implicit function f satisfying $\nabla f = \vec{n}$, this method finds the implicit function f whose gradient direction is best aligned with the normal field.

Other successful methods approximate the distance function using its decomposition on a local basis: radial basis functions [KBH06] or a Fourier basis [Kaz05]. Once the distance function is defined, extracting the surface corresponds to extracting the zero level set of the distance function. This can be done by using the marching cubes algorithm [LC87] which gives the triangulation of the shape, or by sampling the zero level set with particle systems [WH94], [CA97]. These methods yield meshes that approximate well the shape, but they always include an approximation entailing some surface smoothing and the loss of fine texture. Acquisition holes are also filled in by those methods, the signed distance function giving

a natural close up of the surface. Nonetheless, for some applications, filling the holes is questionable. For example, if the goal is to build a closed loop scanning process, the acquisition holes should be detected rather than filled in, so as to guide the laser head towards them. High quality laser acquisition systems have a potential to acquire very fine geometric texture details, provided these details are not lost in the reconstruction. On such data we shall see that an ideal detail-preserving mesh can have all raw data point set as vertices.

1.2 Raw data point set processing

Yet, it is impossible to apply a classic meshing method directly to the raw data point set. The literature has therefore considered more and more sophisticated smoothing and interpolation methods. In [Lev03] and [Lev98], the concept of "Moving Least Square Surfaces" (MLS surfaces) was introduced. MLS surfaces are defined as the set of stationary points of an operator projecting each point to the surface. The MLS algorithm estimates at each point a degree n polynomial from a set of its weighted neighbors. The obtained least square surface can be used to project the point on the MLS surface, or to sub-sample the surface by removing step by step the points with least influence [ABCO$^+$03]. The proof of the approximation power of MLS was published in [Lev98]. Variations of the MLS algorithm for denoising point sampled surfaces and preserving edges were proposed in [FCOS05] (see also [GTE$^+$06], [OGG09], [LCOL07]).

Another product of smoothing methods is the detection of geometric features. Detecting the geometry is important for various applications, one of them being a resampling adapted to surface geometry. In [MD04], [MD03], the authors simplify a point cloud by using geodesic Voronoi diagrams and fast marching methods. More theoretical work on point clouds include [MS02] where theoretical results were presented for bounding the error on distances defined on point sampled surfaces.

Scale space processing was recently extended to meshes and point clouds. The main difficulty is the computation of the surface instrinsic Laplacian (or mean curvature motion) to apply the intrinsic diffusion equation $\frac{\partial x}{\partial t} = \Delta x$. For meshes, the standard discretization of the Laplacian operator is through the cotangent formula [MDSB02]. For point clouds, in [PKG06], each point is moved in the direction of its normal proportionally to its curvature. The curvature is either estimated by a polynomial regression or by projection on a fitted least square surface (in other terms, by MLS). The reverse operator is built by storing the displacements of each point at each step. A similar scale space approach will be used here, but with quite different scopes. In [PKG06], the proposed applications are morphing and shape editing. The present paper instead focuses on raw meshing and raw surface segmentation. A more technical difference stands in the implementation of the scale space: the curvature in our MCM implementation is not explicitly computed, the robust motion being obtained by a simple planar projection operator.

In [UH08], another MCM discretization is proposed. The surface Laplacian is computed by building an operator A_θ at each point position and for every direction θ in the tangent plane. A_θ moves a point p proportionally to the curvature H_θ of the section curve in direction θ. By integrating over θ, it yields a mean curvature motion. The non-uniform subsampling problem is cleverly treated by using a non uniform kernel. The resulting scale space is used to detect characteristic scales of the shape, and regions of interest.

1.3 Computing curvatures

Computing curvatures reliably on a given surface is crucial to various applications, the main ones being to perform anisotropic filtering, i.e. filtering preserving sharp edges ([HP04], [MDSB02]), and to resample the surface according to its curvature ([PGK02]).

On meshes, the curvature estimation problem has already been investigated in [MDSB02] where the famous cotangent formula is proven and extended. [Tau95] derive an analytic expression for estimating the directional curvatures in the edge directions. In [Rus04], [TRZS04], the tensor curvature was estimated on each face of a mesh surface. Other mesh curvature computation techniques include the use of the normal cycle theory [CSM03]. For a summary and comparison of mesh curvature estimation methods, see [MSR07]. It is also possible to estimate curvatures by building curves contained in the surface and passing through the considered point [Tan05].

To determine the curvature of a given point, direct methods fit a surface (a polynomial or a quadric) locally to each neighborhood and then compute the fundamental forms in their explicit form. This allows

to compute the Weingarten map whose eigenvalues and eigenvectors are the principal curvatures and principal directions ([SF04], [LFM96] among others). In [PGK02], the authors simplify meshes using a geometrically coherent method (i.e. points lying in flat areas are removed but points lying in highly curved areas are kept). To determine if points should be removed or not, the curvature was replaced by a new quantity, the *surface variation*. This quantity is defined as the ratio of the least eigenvalue of the covariance matrix versus the sum of all eigenvalues. An interesting feature is that it is directly computed from the raw data set point. However, the surface variation is a rather complicated function of the principal curvatures and loses their sign. Indeed, one can prove:

Theorem 1. *In the local coordinate system the surface variation σ defined in [PGK02] satisfies*

$$\sigma = \frac{r^2}{16}\left(\frac{k_1^2 + k_2^2}{2} - \frac{1}{3}k_1 k_2\right) + o(r^2) \tag{1}$$

where r is the neighborhood radius, and k_1, k_2 are the principal curvatures.

In [BC94], another way of computing the curvature from an oriented raw data set without surface fitting was presented. It relies on expressing the fundamental forms of a 3D surface as covariance matrices. The authors claim that the covariance matrix of point normals projected on the regression plane yields the principal curvatures and their directions. Now, this method requires that the point cloud be previously oriented and therefore does not completely compute the curvature on the raw point set.

Other approaches avoiding surface regression include the computation of integral invariants ([PWHY09], [PWY+07]). They are based on the idea that differentiation is not robust in a discrete and potentially noisy data set, whereas integration is much more resistent to noise. The proofs link the computation of the area of the intersection of the surface with a ball to the principal curvatures. Another possibility is to adapt the curvature estimation of [Tau95] to the case of point clouds as in [LP05]. Instead of considering the edge direction, since no edge information is given for the point cloud, they consider all directions from the center point to one of its neighbors.

More recently, measuring the covariance of Voronoi cells was shown to allow the computation of the principal curvature directions. The soundness of this estimation is proved in [MOG09]. MLS surfaces were also used to derive analytic expressions for the curvatures of point set surfaces [YQ] .

1.4 Feature extraction

In accordance with the edge detection paradigm in image processing, it is classic to perform a 3D shape analysis by extracting the crest lines (the real edges) on meshes or point clouds. Ridge lines are the loci of points where the maximal curvature takes a positive maximum along its curvature line. Valley lines are the loci of points where the minimal principal curvature attains a negative minimum along its curvature line. These points can be linked to form lines (see among others [OBS04], [AGB05], [LFM96], [YBS05], [SF04]). Most methods use a quadric or polynomial regression. In [GWM01], the lines are detected by neighborhood covariance analysis. Indeed, from a point neighborhood, the centroid and centered covariance can be computed. Comparing the ratios of the covariance matrix eigenvalues gives the geometry of the neighborhood (see also [MOG09]). In [HMG00], edges of a mesh are first classified according to their importance (this importance is an increasing function of the adjacent faces angle).

A multiscale approach was proposed in [PKG03]. Nearby feature points are first detected. In the neighborhood of these points surfaces are fitted, and depending on the number of fitted surfaces, points are projected to the nearest surface. Intersection points of these surfaces are finally classified as edge or corner points. By increasing the processing radius, one could track feature lines and keep only the ones at a given scale. Though dealing with scales, this method does not introduce a scale space framework. A similar idea for points classification and point projection was used in [DIOHS08].

Although these papers introduce a ridge/valley line detection, none of them proposes a ridge and valley segmentation. In [IFP95] the idea was suggested, though: indeed points lying near ridges or near valleys were labeled and this labeling was used to obtain a better rendering of the ridge and valley lines. But crest lines as defined by these methods require the computation of degree three surface derivatives. Here we will focus on other interesting and well defined line features: namely the curvature level lines and level sets, analogous to the image grey level lines. Of particular interest are the zero-crossings of the curvature, which are technically similar to the zero-crossings of the Laplacian in image processing.

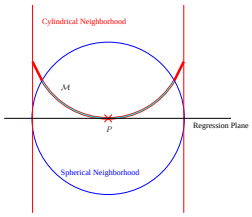


Figure 2: Comparison between cylindrical and spherical neighborhoods

These zero-crossings define inflexion lines, easy to compute from the raw data point set. They reliably segment the surface into ridges and valleys.

The paper is divided as follows: Sect. 2 gives mathematical results proving the consistency of the proposed scale space algorithm. Input data are briefly presented in Sect. 4. Sections 5, 6 and 7 describe the three main applications of the scale space: a point cloud orientation method, a faithful mesh construction for the raw data set and a raw shape segmentation method.

2 Continuous Theory

This section investigates a new way of estimating the surface curvature based on the local covariance analysis. Interestingly for our scopes, it only requires a degree one surface regression to compute a curvature related operator. In this theoretical section the surface (denoted by $\mathcal{M}$) supporting the data point set is always assumed to be smooth (at least C^2). The samples on the surface $\mathcal{M}$ are denoted by $\mathcal{M}_S$.

Let $P(x_P, y_P, z_P)$ be a point of the surface $\mathcal{M}$. Locally we can express the surface as the graph $z = f(x, y)$ of a function f. At each non umbilical point P, consider the principal curvatures k_1 and k_2 linked to the principal directions $\vec{t}_1$ and $\vec{t}_2$, with $k_1 > k_2$ where $\vec{t}_1$ and $\vec{t}_2$ are orthogonal vectors. (At umbilical points, any orthogonal pair $(\vec{t}_1, \vec{t}_2)$ can be taken.) Set $\vec{n} = \vec{t}_1 \times \vec{t}_2$ so that $(\vec{t}_1, \vec{t}_2, \vec{n})$ is an orthonormal basis. The quadruplet $(P, \vec{t}_1, \vec{t}_2, \vec{n})$ is called the local intrinsic coordinate system, and the Taylor expansion of f yields

$$z = f(x, y) = -\frac{1}{2}(k_1 x^2 + k_2 y^2) + o(x^2 + y^2) \tag{2}$$

Notice that the sign of z is irrelevant, since it only depends on the arbitrary surface orientation.

2.1 Spherical neighborhoods vs cylindrical neighborhoods

Consider two kinds of neighborhoods in $\mathcal{M}$ for P defined in the local intrinsic coordinate system $(P, \vec{t}_1, \vec{t}_2, \vec{n})$:

- a neighborhood $B_r = B_r(P) \cap \mathcal{M}$ is the set of all points Q of $\mathcal{M}$ with coordinates (x, y, z) satisfying $(x - x_P)^2 + (y - y_P)^2 + (z - z_P)^2 < r^2$
- a cylindrical neighborhood $C_r = C_r(P) \cap \mathcal{M}$ is the set of all points $Q(x, y, z)$ on $\mathcal{M}$ such that $(x - x_P)^2 + (y - y_P)^2 < r^2$.

For commodity the cylindrical neighborhood will used in the first estimates of each proof. The difference between both neighborhoods will be proved negligible by the next lemma.

Lemma 1. *Integrating on $\mathcal{M}$ any function $f(x, y)$ such that $f(x, y) = O(r^n)$ on a cylindrical neighborhood $C_r(P)$ instead of a spherical neighborhood $B_r(P)$ introduces an $o(r^{n+3})$ error. More precisely:*

$$\int_{\mathcal{B}(r)} f(x, y) dM = \int_{x^2+y^2<r^2} f(x, y) dx dy + O(r^{3+n}). \tag{3}$$

Proof. The surface area element of a point $M(x, y, z(x, y))$ on the surface $\mathcal{M}$, expressed as a function of x, y, dx and dy is $dM(x, y) = \sqrt{1 + z_x^2}dx\sqrt{1 + z_y^2}dy$. One has $z_x = -k_1x+O(r^2)$ and $z_y = -k_2y+O(r^2)$. Thus

$$dM(x, y) = \sqrt{(1 + k_1^2x^2 + O(r^3))(1 + k_2^2y^2 + O(r^3))}dxdy$$

which yields

$$dM(x, y) = (1 + O(r^2))dxdy. \tag{4}$$

Using (4), the integrals we are interested in become

$$\int_{B_r} f(x, y)dM = (1 + O(r^2)) \int_{B_r} f(x, y)dxdy; \tag{5}$$

and

$$\begin{aligned} \int_{C_r} f(x, y)dM &= (1 + O(r^2)) \int_{B_r} f(x, y)dxdy \\ &= (1 + O(r^2)) \int_{x^2+y^2<r^2} f(x, y)dxdy. \end{aligned} \tag{6}$$

This last form is more amenable to analytic computations, which explains why Lemma 1 introduces it. Consider polar coordinates (ρ, θ) such that $x = \rho\cos\theta$ and $y = \rho\sin\theta$ with $-r \le \rho \le r$ and $0 \le \theta \le \pi$. Then for $M(x, y, z)$ belonging to the surface $\mathcal{M}$, we have $z = -\frac{1}{2}\rho^2(k_1\cos^2\theta + k_2\sin^2\theta) + O(r^3)$. Fixing θ we obtain a curve with equation $z = -\frac{1}{2}\rho^2k(\theta) + O(r^3)$, where $k(\theta) = k_1\cos^2\theta + k_2\sin^2\theta$. With this notation, the condition that (x, y, z) belongs to the neighborhood $B_r(P)$ can be rewritten as $\rho^2+z^2 < r^2$, that is

$$\rho^2 + \frac{1}{4}k(\theta)^2\rho^4 < r^2 + O(r^5)$$

Computing the boundaries $\pm\rho(\theta)$ of this neighborhood yield $\rho(\theta)^2 + \frac{1}{4}k(\theta)^2\rho(\theta)^4 - r^2 + O(r^5) = 0$ and therefore

$$\rho(\theta)^2 = \frac{-1 + \sqrt{1 + k(\theta)^2(r^2 + O(r^5))}}{\frac{1}{2}k(\theta)^2}.$$

This yields $\rho(\theta) = r - \frac{1}{8}k(\theta)^2r^3 + o(r^3)$. We shall use this estimate for the error term E appearing in

$$\begin{aligned} \int_{\mathcal{B}(r)} f(x, y)dxdy &= \int_{[0,2\pi]} \int_{[0,\rho(\theta)]} f(x, y)\rho d\rho d\theta \\ &= \int_{[0,2\pi]} \int_{[0,r]} f(x, y)\rho d\rho d\theta - E \\ &= \int_{C_r\cap M} f(x, y)dxdy - E, \end{aligned}$$

with $E =: \int_{[0,2\pi]} \int_{[\rho(\theta),r]} f(x, y)\rho d\rho d\theta$. Thus

$$|E| \le \frac{\pi}{4} \sup_{x^2+y^2\le r^2} |f(x, y)|k(\theta)^2r^3,$$

which yields $|E| \le \frac{\pi|k_1|^2}{4} \sup_{x^2+y^2\le r^2} |f(x, y)|r^3$. In particular if $f(x, y) = O(r^n)$, then $|E| \le O^{3+n}$. Finally we have

$$\int_{\mathcal{B}(r)} f(x, y)dxdy = \int_{C_r\cap M} f(x, y)dxdy + O(r^{3+n}). \tag{7}$$

Combining (5), (6) and (7) yields the announced result (3). □

2.2 Curvature Estimation

The next theorem deals with the simplest local smoothing operator based on raw points and consistent with curvature, the barycenter.

Theorem 2. *In the local intrinsic coordinate system, the barycenter of a neighborhood $B_r(P)$ where P is the origin of the neighborhood has coordinates $x_O = o(r^2)$, $y_O = o(r^2)$ and $z_O = -\frac{Hr^2}{4} + o(r^2)$, where $H = \frac{k_1+k_2}{2}$ is the mean curvature at P.*

Proof. By Lemma 1 applied to the numerator and denominator of the following fraction, we have

$$\begin{aligned} z_O &= -\frac{\int_{B_r} z dM}{\int_{B_r} dM} = -\frac{\int_{x^2+y^2<r^2} z(x,y)dxdy + O(r^5)}{\int_{x^2+y^2<r^2} dxdy + O(r^3)} \\ &= -\frac{\int_{x^2+y^2<r^2} \left[\frac{1}{2}(k_1x^2+k_2y^2) + o(x^2+y^2)\right] dxdy}{\int_{x^2+y^2<r^2} dxdy} + O(r^3) \\ &= -\frac{1}{2\pi r^2}\int_{\rho=0}^{r}\int_{\theta=0}^{2\pi} \rho^2(k_1\cos^2\theta + k_2\sin^2\theta)\rho d\rho d\theta + o(r^2) \\ &= -\frac{r^2}{8\pi}(k_1\pi + k_2\pi) + o(r^2) \\ &= -\frac{Hr^2}{4} + o(r^2.) \end{aligned}$$

A similar but simpler computation yields the estimates of x_O and y_O. □

Theorem 2 states that projecting a point onto the neighborhood barycenter approximates the mean curvature motion. We shall discuss later on why, in spite of Theorem 1, the barycenter cannot be used for implementing the mean curvature motion.

2.3 Surface motion induced by projections on the regression plane

The main tool of the scale space will be a simple projection of each surface point P on the local regression plane found by local covariance analysis. The projected point is called P'. Let us first compare the normal to the local regression plane with the point normal $\vec{n}(P)$

Lemma 2. *The normal $\vec{v}$ to the PCA regression plane at $P \in \mathcal{M}$ is equal to the surface normal at point P, up to a negligible factor: $\vec{v} = \vec{n}(P) + O(r)$.*

Proof. The local PCA regression plane of point P is defined as the plane passing through the barycenter of the neighborhood $\mathcal{B}_r(P)$ and with normal $\vec{v}$ minimizing:

$$I(\vec{v}) = \int_{\mathcal{B}_r(P)} |\langle \vec{v}, PP' \rangle|^2 dP' \text{ s.t. } \|v\| = 1$$

Let the coordinates of $\vec{v}$ be (v_x, v_y, v_z). We have

$$I(\vec{v}) = \int_{B_r} (v_x x + v_y y + v_z \frac{1}{2}(k_1x^2 + k_2y^2) + o(r^2))^2 dxdy.$$

Considering the particular value $\vec{v} = (0,0,1)$ shows that the minimal value I_{min} of $I(\vec{v})$ satisfies $I_{min} \leq O(r^6)$. In consequence the minimum (v_x, v_y, v_z) satisfies $v_x \leq O(r)$ and $v_y \leq O(r)$. Thus $v_z \geq 1 - O(r)$ and therefore $\vec{v} = \vec{n}(P) + O(r)$. □

By Lemma 2, projecting P onto the regression plane induces a motion which is asymptotically in the normal direction: $P'P$ is almost parallel to $\vec{n}(P)$. A consequence is that the simple operation of projecting each surface point P onto its local regression plane approximates a 3D scale space (mean curvature motion) as shown in the next theorem.

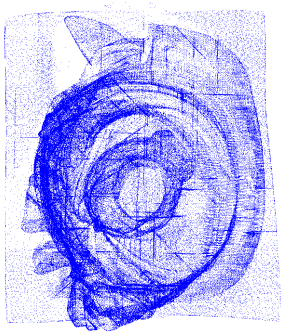


Figure 3: Visualization of the raw point cloud showing the irregular sampling of our Tanagra input data. The cloud has been uniformly and randomly subsampled to visualize the sampling variations. Otherwise, the cloud would look completely dense). This is a top view of the figurine in fig. 11(a)

Theorem 3. *Let T_r be the operator defined on the surface $\mathcal{M}$ transforming each point P into its projection P' on the local regression plane. Then*

$$T_r(P) - P = -\frac{Hr^2}{4}\vec{n}(P) + o(r^2). \tag{8}$$

Thus, this operator is tangent to the mean curvature motion (for theoretical results on the Mean Curvature Motion for surfaces, see, for example [ATW93]).

Proof. By Theorem 2 the barycenter O of B_r has local coordinates $\vec{PO} = (o(r^2), o(r^2), -\frac{Hr^2}{4} + o(r^2))$. On the other hand $\vec{PP'}$ is proportional to $\vec{v}$. Thus by Lemma 2 $\vec{PP'} = \lambda(O(r), O(r), 1 - O(r))$. To compute λ, we use the fact that P' is the projection on the regression plane of P, and that O belongs by definition ot this plane. This implies that $\vec{PP'} \perp \vec{OP'}$ and therefore

$$\lambda^2 O(r^2) + \lambda(1 - O(r))(H\frac{r^2}{4} + o(r^2) + \lambda(1 - O(r))) = 0,$$

which yields $\lambda = -\frac{Hr^2}{4} + o(r^2)$ and therefore

$$\vec{PP'} = (O(r^3), O(r^3), -\frac{Hr^2}{4} + o(r^2)) = -\frac{Hr^2}{4}\vec{n}(P) + o(r^2).$$

□

3 Discrete Theory

The main difference with the previous theory is the irregular sampling. Indeed, the raw triangulation scanner sampling density is highly linked to the surface geometry and can vary a lot (fig 3).

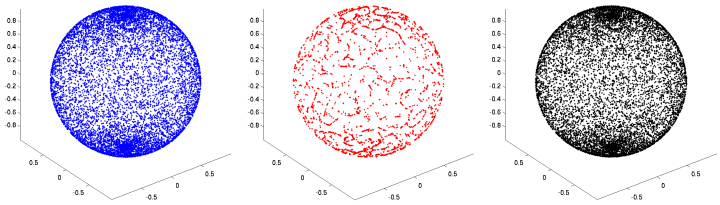
The previous theorems assume that the surface is a uniform Lebesgue mesure. Their applicability to a raw data set point requires some adjustment and some discussion. A constant sampling density is assumed by the theorem and can be approximated by weighting each point by a weight inversely proportional to its initial density, as proposed in [UH08]. More precisely, let p be a point and $\mathcal{N}_r(p)$ its neighborhood on the surface, defined as the set of all points q in $\mathcal{M}_s$ such that $\|p-q\| < r$. Each point q should ideally have a weight $0 \leq w(q) \leq 1$ computed so that for all p, $\sum_{q\in\mathcal{N}_r(p)} w(q) = 1$. This amounts to solve a huge linear system. For this reason, we shall be contented with ensuring $\sum_{q\in\mathcal{N}_r(p)} w(q) \simeq 1$ by taking $w(p) = \frac{1}{\sharp(B_p(r))}$. Let O be the weighted barycenter of this neighborhood. In $\mathbb{R}^3$, the coordinates are written with superscripts e.g. the coordinates of a point u are (u^1, u^2, u^3). Thus, for $i = 1, 2, 3$, $O^i = \frac{1}{\sum_{q\in\mathcal{N}_r(p)} w(q)} \sum_{q\in\mathcal{N}_r(p)} w(q)q^i$. The centered covariance matrix $\Sigma = (m_{ij})_{i,j=1,\cdots,3}$ is defined as

$m_{ij} = \sum_{q \in \mathcal{N}_r(p)} w(q)(q^i - O^i) \cdot (q^j - O^j)$ for $i, j = 1, 2, 3$. Let $\lambda_0 \leq \lambda_1 \leq \lambda_2$ be the eigenvalues of Σ with corresponding eigenvectors v_0, v_1, v_2. For $k = 0, 1, 2$,

$$\lambda_k = \sum_{q \in \mathcal{N}_r(p)} w(q) \langle (q - O), v_k \rangle^2. \tag{9}$$

Each eigenvalue gives the variance of the point set in the direction of the corresponding eigenvector. Since v_1 and v_2 are the vectors that capture most variations, they define the PCA regression plane. The normal v_0 to this plane is the direction v minimizing $\sum_{q \in \mathcal{N}_r(p)} w(q) \langle (p_i - O), v \rangle^2$. We showed that projecting the point onto its local regression plane is a good approximation of the mean curvature motion and that, asymptotically, it is the same as projecting the point to the barycenter of the neighborhood.

Discussion: Both Theorems 2 and 3 permit *a priori* to implement the mean curvature motion on the raw data point set. The numerical application of these theorems depends nonetheless on the assumption that the Lebesgue measure on the surface is well approximated by its sample density. This is not true for the barycenter method of Theorem 2. Iterating the barycenter method with a small neighborhood and a slightly varying sample density leads to a local clustering of the samples. Indeed, a too local neighborhood in an irregular sampling always has some local asymmetry. Thus, the barycenter method provokes a normal motion, but also a non negligible tangential motion to the surface. More crudely said, the algorithm sending each point to the barycenter of its neighborhood is nothing but the well known Mean Shift filter [Che95], which is used for data clustering. This is illustrated in fig. 4. Even though the point distribution on the sphere is probabilistically uniform, sample clustering occurs. Theorem 2 is effective in the sense that globally the sampled sphere evolves in a sampled sphere. But the samples are not moving only in the normal direction. We therefore needed a filter which preserves sampling irregularity while keeping the asymptotic mean curvature motion property. When applying the projection filter, no point cluster is created, since there is no tangent shift, but only a motion along the normal direction. Theorem 3 is in that case effective with an fairly small neighborhood. This fact is easily explained. Take any irregular sampling of the tangent plane to the surface. Then the linear regression will *always* find back the right plane if all samples are not aligned. This good behaviour is experimentally illustrated in fig. 4.



(a) Original samples on a sphere

(b) 4 iterations of Mean Shift

(c) Four iterations of the projection filter

Figure 4: Comparison of the clustering effect for the mean shift filter and the projection filter on a randomly sampled sphere. Clusters appear when the mean shift is iterated, whereas the sampling density is preserved with the projection filter filter. The undesired mean shift clustering effect is due to a tangential motion to the surface caused by the irregular sampling. This tangential motion is avoided with the normal motion, which estimates correctly the normal direction by computing the regression plane, even with irregular sampling

Computing curvatures As a consequence of Theorem 3, the curvature of a point can be computed without any surface fitting step. It is enough to compute $\frac{4}{r^2}\langle n, P - P' \rangle$. On Figs 5, and 6, the curvature is computed on various types of shapes.

Figure 5: Curvature of the Tanagra raw point set

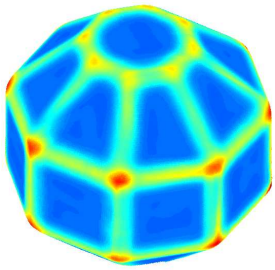


Figure 6: Curvature of the scan of diamond shaped mire used in our lab (5cm diameter)

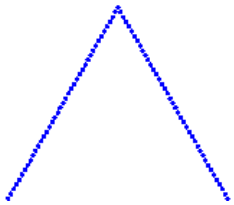


Figure 7: Initial surface: a sharp edge with angle $\frac{\pi}{3}$. This is a 3D surface, and the figures depict its orthogonal projection in the direction of the edge

Back propagation The notion of scale space has been invented in image processing for detecting edges at a coarse scale, and then tracking them back to their fine scale location. The coarse scale is computed by convolving the image with Gaussians, or equivalently by applying the heat equation. The edge backward tracking has always been problematic, because the heat equation is not reversible: it is not easy to decide where an edge detected at scale t came from at scale 0. In the case of surfaces, however, the reverse normal motion defines a natural backtracking. The mean curvature motion writes

$$\frac{dP}{dt} = H(P)\vec{n}(P) \tag{10}$$

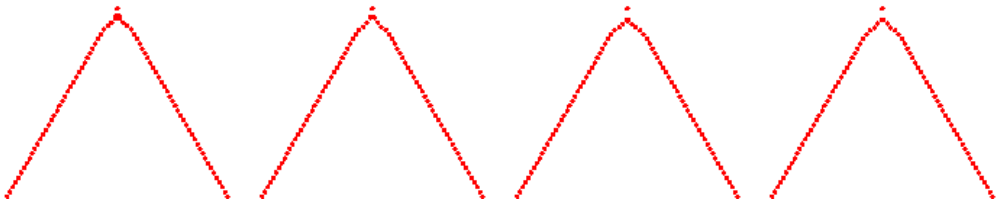
where $H(P)$ is the mean curvature at P (whose sign depends on the normal orientation), and $\vec{n}(P)$ the normal. Thus, a normal motion can be defined for every point P_0 on the initial surface as a solution of (10) considered as an ordinary differential equation with initial point P_0. Thus, the backward scale space is trivial, provided the forward MCM implementation actually implements the evolution of each raw data set point P_0. Let us consider a point P_t and its evolution P_{t+1} at steps t and $t+1$. Now, we can build the sequence $d_P(t) = P_{t+1} - P_t$ and the reverse scale space operator $\mathcal{P}_t^{-1}(P_{t+1}) = P_{t+1} - d_P(t)$, this operator allows to go backward in the scale space evolution from step $t+1$ to 0. This is exactly the construction proposed in [PKG06]. If we only need to go from step t to the initial data 0, without any intermediate step, the operator is even simpler to build, since we only need to store for each point $P(t)$ its initial position $\mathcal{P}_t^{-1}(P_t) = P_0$. This reverse scale space operator will be called *back propagation, or back transportation.*

3.1 Projection on a higher order regression surface

[CP03] proved that a degree n polynomial fitting estimates all k^{th}*order* differential quantity to accuracy $O(h^{n-k+1})$. At first sight this suggests implementing the mean curvature motion by performing a degree 2 regression instead of the plane regression. This would yield a direct curvature estimate by computing explicitly the fundamental form. However, this estimation cannot be turned into an iterative projection operator. In [PKG06] the Moving Least Squares Projection (projecting the point onto the locally fitted least squares surface) was proposed as a scale space operator, but no proof was made of the consistency of these iterated projections with the diffusion equation. Fitting a degree two polynomial to the point set leaves error terms of third order or more. This means that the motion induced by such a projection is proportional to partial differential operator with order larger than 2, whereas the PDE we are approximating relies on second order spatial derivatives. This fact can be experimentally checked by comparing iterations of the 2nd order MLS projection with iterations of our first order projection operator on a sharp edge (7). Figs 8 and 9 show the edge evolution. As proved in Theorem 3, the first order projection implements a mean curvature driven motion where highly curved points evolve faster than low curved points, and flat points do not move. On the contrary the motion induced by the MLS projection is no smoother. It enhances the edge and creates a higher order singularity.

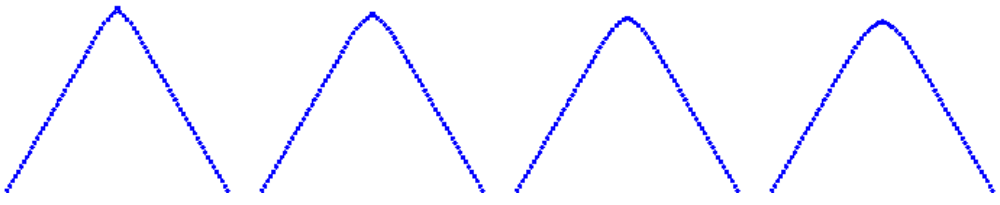
4 Input Data

The algorithms described in the next sections are devised for highly accurate point clouds acquired by a laser scanner. Three objects acquired by our scanner device will be used in the following experiment. The



(a) Iteration 1 (b) Iteration 2 (c) Iteration 3 (d) Iteration 4

Figure 8: Iterated MLS projection operator (4 iterations). It creates a singularity on the edge. Iterated MLS can be used as a scale space only with degree 1 surface, because the iterated operator is consistent with the mean curvature motion



(a) Iteration 1 (b) Iteration 2 (c) Iteration 3 (d) Iteration 4

Figure 9: Scale space operator: the order 1 iterated MLS (four iterations). The edge is smoothed out nicely. Indeed, this operator is consistent with the mean curvature motion

first one is a Tanagra figurine. This object is a mould of a fourth century B.C. Greek figurine obtained at the Museum of Cycladic Arts, Athens (fig. 11(a)). It is 22cm high and the point cloud contains $6 \cdot 10^6$ points. The second one is a geometric 5cm long diamond shaped mire containing $2.5 \cdot 10^5$ points, and the third one is a reproduction of a Nefertiti head figurine (see fig. 20) acquired at the Louvre Museum, Paris containing $3 \cdot 10^6$ points. Thanks to a very accurate calibration of the laser scanner device, the output is a well registered point cloud containing a negligible warp. Since the data point set processing relies on detecting the points contained in a ball centered at each given point, an octree structure is built to to accelerate the neighborhood computations.

Tests will also be made on objects of the Stanford Fragment Urbis Romae database. In that case a registration is needed to have a point cloud representing the whole object. Since we do not address the sweep registration problem in this paper, we will use single sweeps for our meshing experiments. It is interesting to note that even if the mesh obtained using two badly registered point clouds is not good enough for visualization, the detected feature lines are still coherent provided that the registration wrap is not too important (see fig. 23 for feature extraction on a surface containing more than one sweep).

The points acquired are non-oriented. An important feature of the scale space operator we just defined is that it does not need a previous surface orientation to proceed. However, if we want to infer the curvature sign from the scale space, we shall need a coherent orientation. The orientation will be obtained thanks to the scale space. This is the object of the next section.

5 First application: scale space raw data point orientation

Given an initial non oriented raw point cloud the surface orientation is a much needed information. Finding normal directions is very easy, since a local PCA yields the direction corresponding to the least eigenvalue of the local covariance matrix. This direction is a good approximation of the normal direction. We must then pick one of two possible orientations, and this choice must be locally coherent. The idea is to start by picking a random orientation for one point and then propagate it to the neighboring points. Now, sharp edges or a messy surface may fool such a propagation. If, however, the surface is smoothed enough, the propagation of the normal is safe. Thus the overall technique to orient the raw data set will be to smooth it by the scale space, to orient the smoothed surface, and to transport back this coherent orientation to the initial data points.

The first tool to realize this program is a simple propagation method for a point p whose neighborhood $\mathcal{N}_r(p)$ contains some previously oriented points. The orientation is transmitted from a point to the next if their normal directions are similar. The algorithm is summed up below:

OrientateFromNeighbors(p,r,t)

Algorithm 1: OrientateFromNeighbors(p,r,t)

Data: p an unoriented point, a threshold $0 < t < 1$, a radius r, the set $\mathcal{N}_r(p)$ of p's neighbors within radius r

Result: true if the point was oriented, false otherwise

```
Compute p's normal direction n by local PCA;
n̄ ← normalized mean of already oriented neighbors' normals;
if (n̄ · n)^2 > t then
    if n̄ · n > 0 then
        n(p) = n;
    else
        n(p) = −n;
    end
    Return true;
else
    Return false;
end
```

Scale space Point Cloud Orientation algorithm The input parameters are the radius r and a threshold $0 \leq t \leq 1$.

Algorithm 2: Scale space Point Cloud Orientation

Data: A point cloud $\mathcal{P}$, a radius r, an update parameter $a > 1$
1 Iterate the projection filter T_r and keep track of each raw data point sample (Mean Curvature Motion);
2 Find a point p_0 in a flat area, pick its orientation and mark it as oriented. Add its neighbors to the pile $\mathcal{S}$;
3 **while** $\mathcal{S}$ *is not empty or* $\mathcal{S}$ *does not become constant* **do**
4 | Take p the first point in $\mathcal{S}$;
5 | **if** *orientateFromNeighbors(p,r,t)* **then**
6 | | Mark the point as oriented and remove p from $\mathcal{S}$;
7 | **end**
8 | Add p's neighbors to $\mathcal{S}$;
9 **end**
10 Add all remaining unoriented points to $\mathcal{S}$;
11 **while** $\mathcal{S}$ *is not empty and* $\sharp\mathcal{S}$ *does not become constant* **do**
12 | $r = \alpha r$;
13 | **for** p *in* $\mathcal{S}$ **do**
14 | | Perform *orientateFromNeighbors(p,r,t)*;
15 | **end**
16 **end**

Steps from 10 to the end are necessary because adding neighbors of points to the pile might not be enough to cover the whole cloud due to sampling irregularities. Once this procedure is over, there might remain non oriented points. These points are usually isolated points, and it is simplest to ignore them. Actually, in all our experiments the number of remaining non oriented points was below 0.1%. At each step the radius is multiplied by an $\alpha > 1$ factor. In step 12, the radius r is changed. Thus all normals are not computed with the same radius. This is why we must reverse the scale space to come back to the original point cloud. At scale 0, recompute the normal direction by local PCA for all points and pick the orientation which has positive scalar product with the previous normal. The whole process is summed up in algorithm 3. It is a first straightforward application of the scale space framework, where the information is computed at a coarse scale and propagated back to the finest scale.

Algorithm 3: Point Cloud Reorientation

Data: An oriented point cloud $\mathcal{P}$, a radius r
Result: A point cloud with normals computed at the same scale
1 **for** $p \in \mathcal{P}$ **do**
2 | Compute the normal $\vec{v}$ of p's neighbors $\mathcal{N}_r(p)$ by local PCA;
3 | **if** $\langle \vec{n}(p), \vec{v} < 0$ **then**
4 | | $\vec{n}(p) = -v$;
5 | **else**
6 | | $\vec{n}(p) = v$;
7 | **end**
8 **end**

Having oriented the surface allows us to define positive and negative curvature level sets and 0-curvature level lines, as will be shown in section 7.

6 Scale space meshing

We now discuss how to build a mesh on a high precision point cloud. Direct meshing is not possible because of the surface oscillation due to texture. The idea is to perform meshing on the smoothed surface and to transport this mesh back on the original point cloud. We need an efficient triangulation technique such as [BMR+99], [CSD04]. The only requirement on the algorithm is that the mesh should interpolate the points. The final vertices must be a subset of the original points almost identical to the raw data

set point. This is not the case with level set methods ([KBH06], [HDD+92], [Kaz05],[LC87], [CA97] and [LGS06]). The whole pipeline was implemented using the Ball Pivoting Algorithm (BPA) [BMR+99]. The method proceeds as follows:

Algorithm 4: Scale Space Meshing Algorithm

Data: A point set with computed normals
Result: A mesh of the original 3D data point set
1 Iterate (four times) the projection filter T_r and keep track of each raw data point sample: this is the forward mean curvature motion;
2 Mesh the smoothed samples;
3 Transport the mesh back to the original points (thus reverting the mean curvature motion).

To set the radius automatically, we can get a good approximation of what a good radius would be while we compute the octree to sort the points. Indeed the root of the octree is the bounding box of all points. Let us call L_{max} the length of its largest side. Then, each cell represents a 3D cube with size $L_{max}/2^d$ where d is the depth of the cell. Counting the number of points in that cell gives an approximation of the number of neighbors of a point contained in this cell for a spherical neighborhood of radius $r_d = L_{max}/2^{d+1}$. Performing this approximation in all non empty cells at the same depth gives an approximation of the number of neighbors for spherical neighborhoods with radius r_d. To perform the projection filter, the minimum number of neighbors of a point is 3 (including the point itself), because we need to estimate a regression plane. But having only 3 neighbors will lead to instable regression plane estimations. To have a robust geometric processing, our experiments led us to consider 30 neighbors a good value. Of course, since the same radius is used for all points, it may occur that for some points, the chosen radius does not ensure enough neighbors to perform the plane regression, those points are irrelevant and should be eliminated. Since we deal with a dense point cloud, removing them should not affect the point cloud. In all our experiments we removed less than 0.1% of points. With this automatically set radius, only a few scale space iterations are necessary: in all our experiments four projection iterations were used.

Algorithm 5: Setting the radius automatically

Data: An octree with depth d containing the point cloud (root is at depth d and leaves are at depth 0) L_{max} size of the octree bounding box. A minimum number of neighbors N_{min}
Result: A radius r

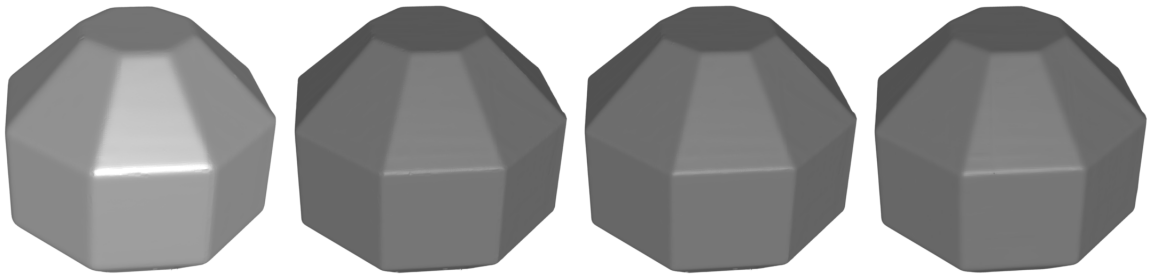
```
1  np = 0;
2  l = 0;;
3  while np < N_min do
4  |   n = 0;
5  |   np = 0;
6  |   for all non empty cells at depth l do
7  |   |   n = n + 1;
8  |   |   np = np + cell → Npoints;
9  |   end
10 |   np = np/n;
11 |   l ← l + 1;
12 end
13 r = L_max/2^(d−l+1);
```

Transporting back the connectivity information (step 3) can in theory lead to a self crossing mesh. Indeed, if two points lie too close to each other they may "switch position" in the scale space iterations, leading to a complicated surface topology. This problem can be solved by detecting all pairs of intersecting triangles. Then any remeshing algorithm can solve the problem by switching edges in quadrilaterals. However, this additional step was not implemented for two good reasons. First, the existence of a few intersecting triangles is no serious visual inconvenience. Second, we did not find any such crossing in all of our experiments.

Fig. 10 illustrates the mesh rendering of a simple geometric pattern by a back propagated mesh. Fig 10(a) shows the coarse scale mesh, i.e. the mesh obtained after four scale space iterations. The sharp edges have been smoothed out. Nonetheless, a direct back-projection of this mesh at the original scale

allows to recover the sharp edges (fig 10(b)). The result can be compared to the meshes obtained by direct meshing (Fig. 10(c), simple Ball Pivoting Algorithm) and 10(d) (level set method [KBH06]). In that simple case with no texture and little noise, there is no significant difference between 10(b), 10(c), and 10(d)). Scale space meshing recovers edges as well as other state of the art methods.

Figs. 11 shows the application of scale space meshing with a mesh rendering at fine and coarse scale. We can see on Fig. 11 that the surface texture is lost at a coarse scale, but completely and accurately recovered by simply propagating the mesh information to the initial points at fine scale. Comparing the back projected mesh to the result of a direct meshing of the initial samples (Fig. 12) shows that the scale space triangulation is much more precise. In fact, a direct meshing is not applicable. It creates, among other artifacts, many spurious triangles. Fig. 13 shows a comparison between the reconstruction obtained by VRIP reconstruction method (see [CL96]) and scale space meshing. The scale space method produces a much more precise mesh, as can be seen on the close up of figs 14 and 15



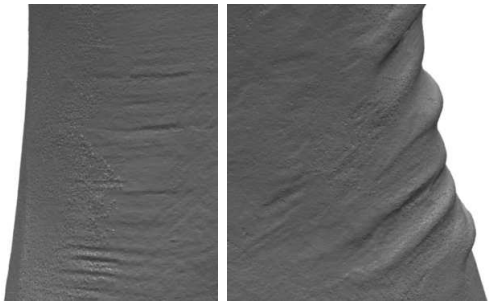
(a) Coarse scale mesh (b) Back-projected mesh (c) Direct Meshing (d) Poisson Meshing

Figure 10: Multi-resolution mesh reconstruction from the Diamond point set illustrating the recovery of sharp edges. In that case the object has no texture and almost no noise. Thus all methods give the same result, which proves the consistency of mesh back propagation method

Fig. 11 displays the many acquisition holes at the bottom of the Tanagra figurine, in the tunic's folds or near the right foot. By the scale space meshing these holes are not filled in and can be detected and characterized by their border Jordan curve (see Fig. 16). Since the Ball Pivoting Algorithm is used for triangulation, no triangle larger than a given threshold has been created. Indeed, to form a triangle, three points must lie on a sphere of given radius r. Thus, low density areas are considered holes. Fig. 12 illustrate the loss of details with level sets methods. Level set methods create a smoothed zero level surface of the signed distance to the raw data set point. They do not contain the raw data set points and lose track of them. Fig. 18 shows that not only these methods, but even direct meshing methods can miss small details. Fig. 17 illustrate why scale space meshing allows to recover those details: standard meshing at a smooth scale is simply easy because details have been unfolded. It is then trivial to back propagate the vertices of the smooth mesh to their initial positions. This yields a direct triangulation of the original raw data set.

The quantitative performance of each algorithm can be evaluated by meshing simple shapes. Test point sets were built by sampling perfect geometric shapes (for example a sinusoidal surface). The root mean square distance of the triangle barycenters of the mesh to the real surface were compared for each meshing method. This distance is computed by the Newton-Raphson method. The first surface "Wave 1" has equation $z = 0.2\cos(5x)$, "Wave 1" has equation $z = 0.2\cos(5x) * \cos(5y)$, the third surface is a regularly sampled sphere and the last one is a sum of two close and narrow Gaussians $z = -\exp(-\frac{(x-0.1)^2}{0.01} - \exp(-\frac{(x+0.1)^2}{0.01}$. The RMSE results are shown in Table 19. It is obvious from these results that the Poisson reconstruction or any level set method cannot be applied to recover a surface with very thin details. On shapes containing no sharp edges, direct BPA and scale space meshing perform comparably. On the thin structure created by adding two very close Gaussians, the loss of precision of BPA is clear. This phenomenon is similar to the one observable in Fig. 12(c) where BPA does not recover thin details.

(a) initial object (22 cm high) (b) Coarse scale mesh (c) Back-projected mesh (d) Photo of the back side (e) Detail selection

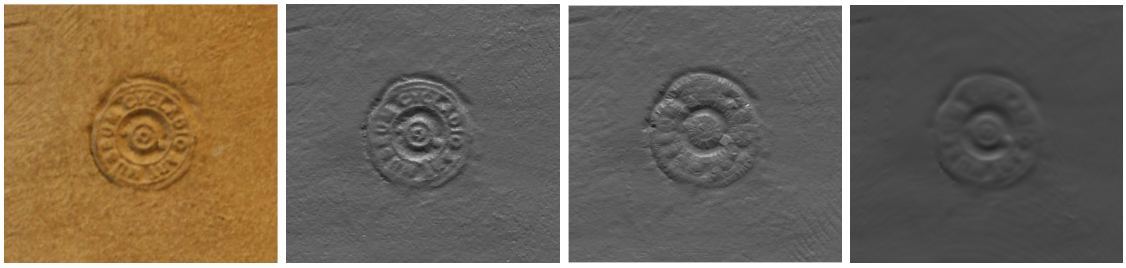


(f) Detail expansion (1) (g) Detail expansion (2)

Figure 11: Multi-resolution mesh reconstruction of the Tanagra point set illustrating the recovery of fine texture. All back propagated textures are present on the original

7 Scale Space geometric features extraction

Having a method to compute curvature directly on the data point set, and a method to backtrack raw points, has consequences on two closely related problems: the ridge-valley classification and the computation of inflexion lines. Inflexion lines are defined as the curvature zero-crossings: they are therefore equivalent to the "zero-crossing of Laplacian"proposed by Hildreth and Marr [MH80]. These curves are closed and segment the shape into ridges (positive mean curvature points) and valleys (negative mean curvature points). In order to achieve this scale-dependent segmentation, several scale space iterations yield a scale-dependent curvature. The sign of this multiscale curvature yields a binary classification in ridge and valley regions (see 20). Of course, the presence of fine textures hinders the detection of large scale ridges and valleys. They are instead easily computed after several scale space iterations, and transported back on the initial raw data set point. We have defined inflexion lines as zero-crossings of the curvature. They therefore separate ridges (positive curvature) from valleys (negative curvature). At each scale, curvature level lines can be drawn on the scale space mesh of the raw original samples as defined in Section 6. For $\alpha \in \mathbb{R}$, to extract a α-level line, we detect edges whose extremities P_1 P_2 have curvatures $H(P_1)$, such that $(H(P_1) - \alpha) \cdot (H(P_2) - \alpha) < 0$. Inflexion lines vertices are linearly interpolated along



(a) Picture of the Logo

(b) Back-propagated mesh

(c) Direct mesh

(d) Reconstruction obtained by Poisson Reconstruction

Figure 12: Comparison between the direct mesh and the back projected mesh. The width of this logo is approximately 1cm. Direct meshing creates many wrong triangles. Compare the details in 12(b) and 12(c). See figs 17-18 for an explanation of this difference.

these edges and linked using mesh the connectivity information. The algorithm is summed in algorithm 6, an edge e will be called α-crossing if its extremities verify $(H(P_1) - \alpha) \cdot (H(P_2) - \alpha) < 0$.

Algorithm 6: Extracting the curvature level lines

Data: A meshed point cloud with curvature information for each point
Result: $\mathcal{L}$ a set of inflexion lines

```
while There remain unmarked α-crossing edges do
    l an empty list of points;
    Find an α-crossing edge e_0;
    Mark e_0;
    Choose T one of the adjacent triangles to e_0;
    e ← e_0;
    Add e to the right of l;
    while e is not a border edge and e is not e_0 do
        e_prev = e Find e the other α-crossing edge of T;
        Mark e;
        Add e to the right of l;
        Pick T adjacent to e but not to e_prev;
    end
    L ← L ∪ l;
end
```

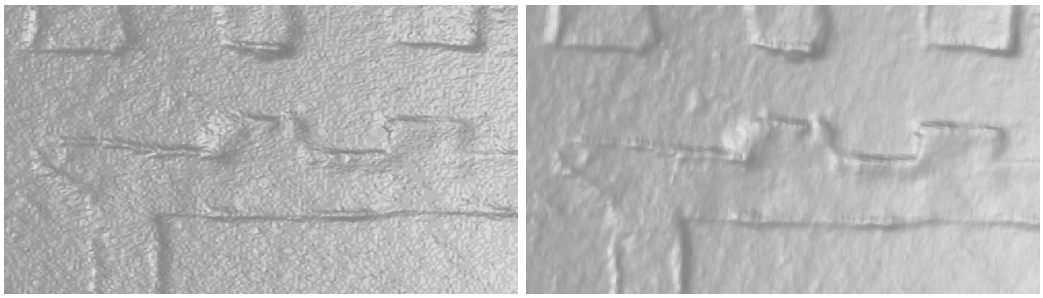
The algorithm is based on the fact that each triangle with one α-crossing edge has two α-crossing edges. Since the inflexion lines surround the surface crest lines, crest lines can be considered as a sort of skeleton of well chosen level lines, as can be seen on Fig. 22. Furthermore, inflexion lines are well defined closed Jordan curves. The only open inflexion lines are those that end on a hole. At the bottom of the Tanagra figurine some open inflexion line end up on the acquisition holes (see Fig. 16).

8 Conclusion

The increasing accuracy of 3D triangulation scanners requires an effort to reconsider the whole rendering chain, and to obtain high quality visualization, actually better than those obtained by photography. The present paper has proposed a strategy to mesh the raw original surface, therefore ensuring a faithful rendering and an accurate hole detection. Future work will be on the testing of a closed scanning loop with our experimental scanner. The scanner will be steered towards the detected holes to get a data

(a) Picture of the Object (b) Back-projected mesh (c) Reconstructed Mesh available on the FUR website

Figure 13: Comparison on a piece of the Fragment Urbis Romae (FUR) database. Texture and details are better recovered on the back-propagated mesh (middle). Compare with the VRIP reconstruction method available on the FUR website (right)



(a) Back-projected mesh (b) Vrip Reconstructed Mesh

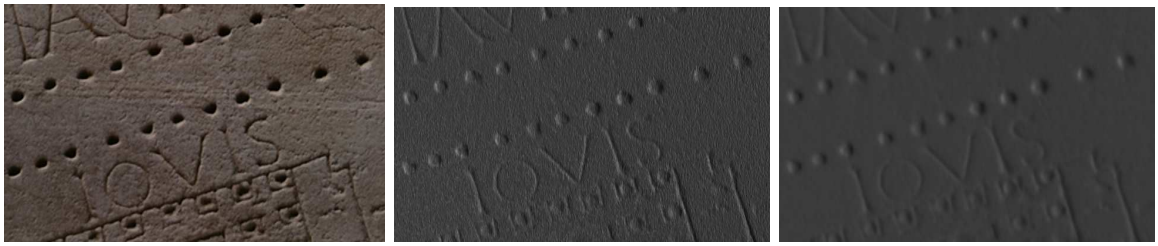
Figure 14: Closeup of a piece of the (FUR) database reconstructed by scale space meshing (left) and VRIP method (right)

point set as complete as geometrically possible. Future work will also reconsider the registration and fusion algorithms for several scanning sweeps in textured areas. The final goal would be to perform accurate supperresolution from several sweeps.

Acknowledgements The authors would like to acknowledge the Stanford Digital Forma Urbis Romae Project (http://formaurbis.stanford.edu) for the fragments and photographs of figures 13, 14 and 15 and especially professor Marc Levoy for allowing us to use the fragments data. Those photographs and raw datas are property of both the Sovraintendenza in Rome and Stanford University.

References

[ABCO+03] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva, *Computing and rendering point set surfaces*, IEEE Transactions on Visualization and Computer Graphics **9** (2003), no. 1, 3–15.



(a) Original Fragment (b) Back-projected mesh (c) Level Set mesh

Figure 15: Closeup of a piece of the (FUR) Database reconstructed by Scale Space Meshing and Poisson Reconstruction Method

Figure 16: The holes in the Tanagra are detected and surrounded by Jordan curves after scale space meshing

[ACSTD07] P. Alliez, D. Cohen-Steiner, Y. Tong, and M. Desbrun, *Voronoi-based variational reconstruction of unoriented point sets*, SGP '07: Proceedings of the fifth Eurographics symposium on Geometry processing (Aire-la-Ville, Switzerland, Switzerland), Eurographics Association, 2007, pp. 39–48.

[AGB05] Elena V. Anoshkina Alexander G. Belyaev, *Mathematics of surfaces xi, ima conference on the mathematics of surfaces 2005*, ch. Detection of Surface Creases in Range Data, Springer, 2005.

[ATW93] Fred Almgren, Jean E. Taylor, and Lihe Wang, *Curvature-driven flows: a variational approach*, SIAM J. Control Optim. **31** (1993), no. 2, 387–438. MR MR1205983 (94h:58067)

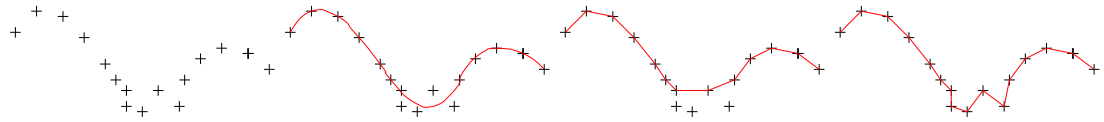
[BC94] J. Berkmann and T. Caelli, *Computation of surface geometry and segmentation using covariance techniques*, IEEE Trans. Pattern Anal. Mach. Intell. **16** (1994), no. 11, 1114–1116.

[BMR+99] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva, and Gabriel Taubin, *The ball-pivoting algorithm for surface reconstruction*, IEEE Transactions on Visualization and Computer Graphics **5** (1999), 349–359.

[CA97] Patricia Crossno and Edward Angel, *Isosurface extraction using particle systems*, IEEE Visualization '97 (Roni Yagel and Hans Hagen, eds.), 1997, pp. 495–498.

(a) Original samples (b) Points projection motion (c) Resulting samples (d) Mesh of the resulting samples (red line) (e) Back projected Mesh (red line)

Figure 17: 2D example of the steps performed by the scale space meshing algorithm



(a) Original samples (b) Level set method (mesh will be sampled on the 0 level set=red curve) (c) Direct Meshing (red line) (d) Scale Space Meshing (red line)

Figure 18: Comparison of three meshing methods

[Che95] Yizong Cheng, *Mean shift, mode seeking, and clustering*, IEEE Trans. Pattern Anal. Mach. Intell. **17** (1995), no. 8, 790–799.

[CL96] Brian Curless and Marc Levoy, *A volumetric method for building complex models from range images*, SIGGRAPH '96: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (New York, NY, USA), ACM Press, 1996, pp. 303–312.

[CP03] F. Cazals and M. Pouget, *Estimating differential quantities using polynomial fitting of osculating jets*, SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing (Aire-la-Ville, Switzerland, Switzerland), Eurographics Association, 2003, pp. 177–187.

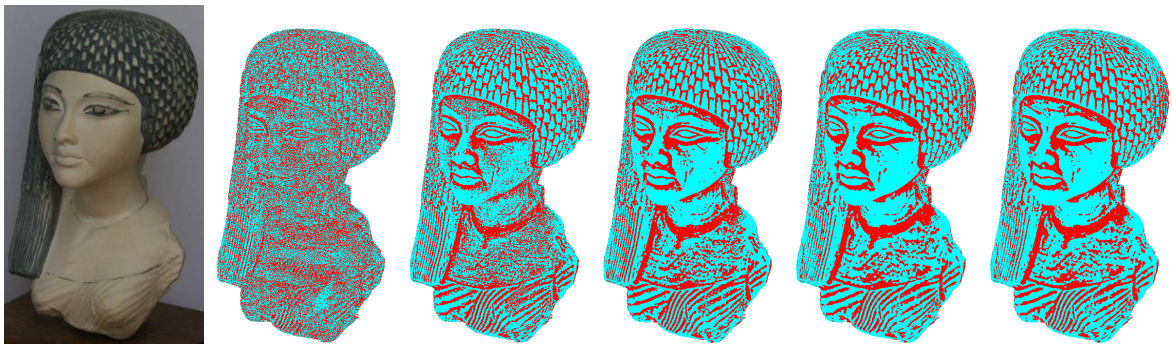
[CSD04] David Cohen-Steiner and Frank Da, *A greedy delaunay-based surface reconstruction algorithm*, Vis. Comput. **20** (2004), no. 1, 4–16.

[CSM03] David Cohen-Steiner and Jean-Marie Morvan, *Restricted delaunay triangulations and normal cycle*, SCG '03: Proceedings of the nineteenth annual symposium on Computational geometry (New York, NY, USA), ACM, 2003, pp. 312–321.

[DIOHS08] Joel Daniels II, Tilo Ochotta, Linh K. Ha, and Claudio T. Silva, *Spline-based feature curves from point-sampled geometry*, Vis. Comput. **24** (2008), no. 6, 449–462.

Method	Wave 1	Wave 2	sphere	sharp
Scale Space	0.00019	0.00028	0.000041	0.00045
BPA	0.00018	0.00024	0.00004	0.0012
Poisson	0.0015	0.043	0.00024	0.004

Figure 19: Quantitative comparison of three meshing methods, scale space meshing, Ball Pivoting Algorithm and Poisson Reconstruction



(a) Initial object (b) Initial segmentation (c) First projection (d) Second projection (e) Third projection (f) Back projection at the original position

Figure 20: Classification of the Nefertiti point set. Four steps of the scale space are applied to the point set. The final coarse scale classification is transported back to the original point positions. The final classification captures the global geometric properties of the shape at the desired scale

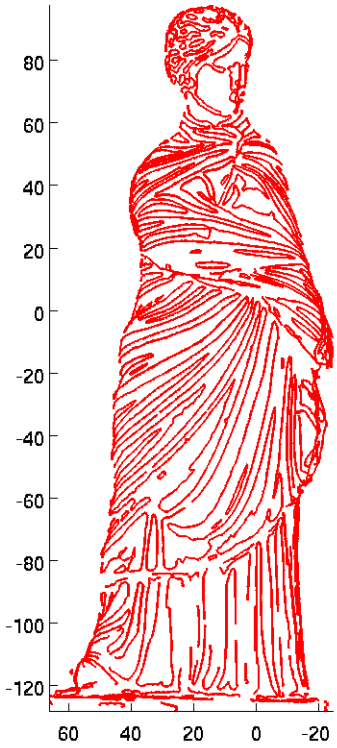


Figure 21: Tanagra inflexion lines (front part), dividing the folds in ridges and valleys.

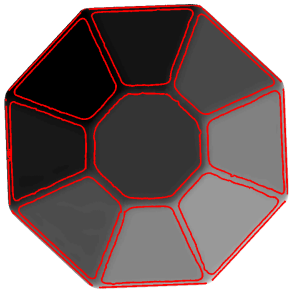


Figure 22: Extraction of level line 0.12 of the diamond point set, segmenting the points into flat and non flat areas

[FCOS05] Shachar Fleishman, Daniel Cohen-Or, and Claudio T. Silva, *Robust moving least-squares fitting with sharp features*, ACM Trans. Graph. **24** (2005), no. 3, 544–552.

[GTE+06] Joao Paulo Gois, Eduardo Tejada, Tiago Etiene, Luis Gustavo Nonato, Antonio Castelo, and Thomas Ertl, *Curvature-driven modeling and rendering of point-based surfaces*, Computer Graphics and Image Processing, Brazilian Symposium on **0** (2006), 27–36.

[GWM01] Stefan Gumhold, Xinlong Wang, and Rob McLeod, *Feature extraction from point clouds*, Proc. 10th International Meshing Roundtable, 2001, 2001.

[HDD+92] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle, *Surface reconstruction from unorganized points*, SIGGRAPH '92: Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques (New York, NY, USA), ACM Press, 1992, pp. 71–78.

[HMG00] Andreas Hubeli, Kuno Meyer, and Markus Gross, *Mesh edge detection*, Proc. Workshop Lake Tahoe (Lake Tahoe City, California, USA), January 2000.

[HP04] Klaus Hildebrandt and Konrad Polthier, *Anisotropic filtering of non-linear surface features*, Computer Graphics Forum **23** (2004), 391–400.

[IFP95] Victoria Interrante, Henry Fuchs, and Stephen Pizer, *Enhancing transparent skin surfaces with ridge and valley lines*, VIS '95: Proceedings of the 6th conference on Visualization '95 (Washington, DC, USA), IEEE Computer Society, 1995, p. 52.

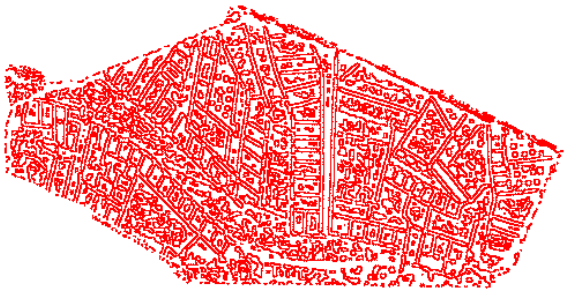
[Kaz05] Michael Kazhdan, *Reconstruction of solid models from oriented point sets*, SGP '05: Proceedings of the Third Eurographics Symposium on Geometry processing (Aire-la-Ville, Switzerland, Switzerland), Eurographics Association, 2005, p. 73.

[KBH06] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe, *Poisson surface reconstruction*, SGP '06: Proceedings of the fourth Eurographics Symposium on Geometry processing (Aire-la-Ville, Switzerland, Switzerland), Eurographics Association, 2006, pp. 61–70.

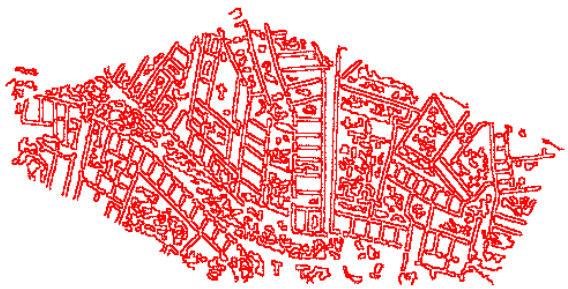
[LC87] William E. Lorensen and Harvey E. Cline, *Marching cubes: A high resolution 3d surface construction algorithm*, SIGGRAPH '87: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (New York, NY, USA), ACM Press, 1987, pp. 163–169.

[LCOL07] Yaron Lipman, Daniel Cohen-Or, and David Levin, *Data-dependent mls for faithful surface approximation*, SGP '07: Proceedings of the fifth Eurographics symposium on Geometry processing (Aire-la-Ville, Switzerland, Switzerland), Eurographics Association, 2007, pp. 59–67.

(a) Original Object

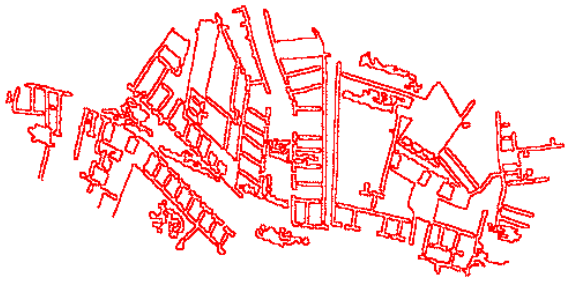


(b) Showing inflexion lines with length above 2mm



(c) Showing inflexion lines with length above 3cm

(d) Showing inflexion lines with length above 6cm



(e) Showing inflexion lines with length above 22cm

Figure 23: Extraction of inflexion lines of the FUR fragment 10g. Some of the detected lines are due to surface irregularity and not to surface carving. They are in general short and are filtered out by a length threshold

[Lev98] David Levin, *The approximation power of moving least-squares*, Math. Comput. **67** (1998), no. 224, 1517–1531.

[Lev03] ———, *Mesh-independent surface interpolation*, Geometric Modeling for Scientific Visualization (Hamann Brunnett and Mueller, eds.), Springer-Verlag, 2003, pp. 37–49.

[LFM96] R. Lengagne, P. Fua, and O. Monga, *Using crest lines to guide surface reconstruction from stereo*, ICPR '96: Proceedings of the 1996 International Conference on Pattern Recognition (ICPR '96) Volume I (Washington, DC, USA), IEEE Computer Society, 1996, p. 9.

[LGS06] F. LEVET, X. GRANIER, and C. SCHLICK, *Fast sampling of implicit surfaces by particle systems*, Shape Modeling International (SMI 2006) - short paper, Fast sampling of implicit surfaces by particle systems, jun 2006.

[LP05] Carsten Lange and Konrad Polthier, *Anisotropic smoothing of point sets*, Comput. Aided Geom. Des. **22** (2005), no. 7, 680–692.

[MD03] Carsten Moenning and Neil A. Dodgson, *A new point cloud simplification algorithm*, Proc. 3rd Int. Conf. on Visualization, Imaging and Image Processing, 2003, pp. 1027–1033.

[MD04] ———, *Intrinsic point cloud simplification*, Proceedings of the 14th GraphiCon '04, 2004.

[MDSB02] M. Meyer, M. Desbrun, P. Schr^der, and A. Barr, *Discrete differential geometry operators for triangulated 2-manifolds*, International Workshop on Visualization and Mathematics, 2002.

[MH80] D. Marr and E. Hildreth, *Theory of edge detection*, Proceedings of the Royal Society of London. Series B, Biological Sciences, **207** (1980), no. 1167, 187–217.

[MOG09] Quentin Mérigot, Maks Ovsjanikov, and Leonidas J. Guibas, *Robust Voronoi-based Curvature and Feature Estimation*, SIAM/ACM Joint Conference on Geometric and Physical Modeling (San Francisco États-Unis d'Amérique), 2009 (Anglais).

[MS02] Facundo Memoli and Guillermo Sapiro, *Distance functions and geodesics on point clouds*, Tech. report, Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455, 2002.

[MSR07] Evgeni Magid, Octavian Soldea, and Ehud Rivlin, *A comparison of gaussian and mean curvature estimation methods on triangular meshes of range image data*, Comput. Vis. Image Underst. **107** (2007), no. 3, 139–159.

[OBS04] Yutaka Ohtake, Alexander Belyaev, and Hans-Peter Seidel, *Ridge-valley lines on meshes via implicit surface fitting*, ACM Trans. Graph. **23** (2004), no. 3, 609–612.

[OGG09] A. C. Oztireli, G. Guennebaud, and M. Gross, *Feature preserving point set surfaces based on non-linear kernel regression*, Computer Graphics Forum **28** (April 2009), 493–501(9).

[PGK02] Mark Pauly, Markus Gross, and Leif P. Kobbelt, *Efficient simplification of point-sampled surfaces*, VIS '02: Proceedings of the Conference on Visualization '02 (Washington, DC, USA), IEEE Computer Society, 2002, pp. 163–170.

[PKG03] M. Pauly, R. Keiser, and M. Gross, *Multi-scale feature extraction on point-sampled surfaces*, Computer Graphics Forum, vol. 22, september 2003, pp. 281–289.

[PKG06] Mark Pauly, Leif P. Kobbelt, and Markus Gross, *Point-based multiscale surface representation*, ACM Trans. Graph. **25** (2006), no. 2, 177–193.

[PWHY09] Helmut Pottmann, Johannes Wallner, Qi-Xing Huang, and Yong-Liang Yang, *Integral invariants for robust geometry processing*, Comput. Aided Geom. Des. **26** (2009), no. 1, 37–60.

[PWY+07] Helmut Pottmann, Johannes Wallner, Yong-Liang Yang, Yu-Kun Lai, and Shi-Min Hu, *Principal curvatures from the integral invariant viewpoint*, Comput. Aided Geom. Des. **24** (2007), no. 8-9, 428–442.

[Rus04] Szymon Rusinkiewicz, *Estimating curvatures and their derivatives on triangle meshes*, 3DPVT '04: Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium (Washington, DC, USA), IEEE Computer Society, 2004, pp. 486–493.

[SF04] Georgios Stylianou and Gerald Farin, *Crest lines for surface segmentation and flattening*, IEEE Transactions on Visualization and Computer Graphics **10** (2004), no. 5, 536–544.

[Tan05] Xiaojing Tang, *A sampling framework for accurate curvature estimation in discrete surfaces*, IEEE Transactions on Visualization and Computer Graphics **11** (2005), no. 5, 573–583, Member-Agam, Gady.

[Tau95] G. Taubin, *Estimating the tensor of curvature of a surface from a polyhedral approximation*, ICCV '95: Proceedings of the Fifth International Conference on Computer Vision (Washington, DC, USA), IEEE Computer Society, 1995, p. 902.

[TRZS04] Holger Theisel, Christian Rossl, Rhaleb Zayer, and Hans-Peter Seidel, *Normal based estimation of the curvature tensor for triangular meshes*, PG '04: Proceedings of the Computer Graphics and Applications, 12th Pacific Conference (Washington, DC, USA), IEEE Computer Society, 2004, pp. 288–297.

[UH08] Ranjith Unnikrishnan and Martial Hebert, *Multi-scale interest regions from unorganized point clouds*, Workshop on Search in 3D (S3D), IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), June 2008.

[WH94] Andrew P. Witkin and Paul S. Heckbert, *Using particles to sample and control implicit surfaces*, SIGGRAPH '94: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques (New York, NY, USA), ACM Press, 1994, pp. 269–277.

[Wit83] Andrew P. Witkin, *Scale-space filtering.*, 8th Int. Joint Conf. Artificial Intelligence (Karlsruhe), vol. 2, August 1983, pp. 1019–1022.

[YBS05] Shin Yoshizawa, Alexander Belyaev, and Hans-Peter Seidel, *Fast and robust detection of crest lines on meshes*, SPM '05: Proceedings of the 2005 ACM Symposium on Solid and Physical Modeling (New York, NY, USA), ACM Press, 2005, pp. 227–232.

[YQ] P. Yang and X. Qian, *Direct computing of surface curvatures for point-set surfaces*, Proceedings of 2007 IEEE/Eurographics Symposium on Point-based Graphics(PBG).