Volume xx (200y), Number z, pp. 1–12

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

Abstract

This paper develops a scale space strategy for orienting and meshing exactly and completely a raw point set. The scale space is based on the intrinsic heat equation, also called mean curvature motion (MCM). A simple iterative scheme implementing MCM directly on the raw point set is described, and a mathematical proof of its consistency with MCM is given. Points evolved by this MCM implementation can be trivially backtracked to their initial raw position. Therefore, both the orientation and mesh of the data point set obtained at a smooth scale can be transported back on the original. The gain in visual accuracy is demonstrated on archaeological objects by comparison with several state of the art meshing methods.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations

1. Introduction

A growing number of applications involve creating numerical models for existing objects acquired by triangulation laser scanner or other devices. Our study deals with raw input data acquired by such scanners, namely sets of unorganized and non-oriented points given by their x, y, z coordinates. The proposed method orients and meshes directly the complete raw point set, thus allowing for the visualization of the finest surface details and an accurate delineation of the scanning holes. Contrarily to usual meshing methods, the goal is to keep almost all raw data points as vertices, thus putting in evidence all defects caused by the scanning process. Doing so is important to keep an optimal accuracy. Indeed, the processed point data treated here have a typical acquisition error of 20μ, allowing in principle to recover all textures and details.

The main tool introduced 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 simplest intrinsic method to smooth out a surface. It writes

$$\frac{dP}{dt} = H(P)\vec{n}(P) \tag{1}$$

where $H(P)$ is the mean curvature at P (whose sign depends on the normal orientation), and $\vec{n}(P)$ the normal. This motion will be given a robust implementation working directly on raw data, which can be completely described in few words: *it is the iterated projection of each point on the regression plane of its radial neighborhood.* In contrast to [AMD02] which defined a remeshing method based on a global planar parametrization, here, the plane projection will be local and iterated. Mathematical and experimental arguments will show that this iterated planar regression consistently implements the MCM and actually permits to compute an accurate denoised curvature. Indeed, Theorem 2 states 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 and backward in the surface smoothing process. As a consequence, the surface structure established at a smooth scale can be transported back on the raw data set point. This back transportation yields a topologically faithful orientation at each raw point, and subsequently a mesh whose vertices are almost all raw data points. It also permits an accurate detection of holes in the raw data, useful for further scanning attempts. Comparative experiments will show that a direct meshing gives poor results, while the back transported mesh allows for the uttermost accurate rendering of the surface, the mesh vertices being more than 99% of all initial raw points. Obviously, such a complete mesh is not economical, but it permits an accurate rendering of fine art or archaeological pieces at 20μ precision in the current technology and a detection by visual inspection of the tiniest scanning defects.

The use of the mean curvature motion, forward and backward, is a direct 3D extension of the scale space paradigm in image processing introduced in the founding Witkin paper [Wit83]. It consists of applying the heat equation $\frac{\partial u}{\partial t} = \Delta u$ to the image leading to a rapid image simplification. The main image features (for example the edges) are detected at a coarse scale (large t) and then tracked back to their fine scale position. The next subsection reviews the methods computing curvatures and normals on raw data.

1.1. Building a mesh

Given an initial *oriented* point cloud, most meshing methods start by defining a signed distance field to the inferred surface [HDD*92], [KBH06]. 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 sign of this distance is straightforward. Other successful methods approximate the distance function using its decomposition on a local radial basis functions [KBH06]. Extracting the surface then corresponds to extracting the zero level set of the distance function. This can be done with the marching cubes algorithm [LC87] or any other contouring algorithm.

These level set methods are robust and yield meshes that approach well the shape, *but the approximation entails an implicit surface smoothing and the loss of fine texture.* Acquisition holes are also filled in by these methods, the signed distance function giving a natural close up of the surface. Nonetheless, for scanning applications, the acquisition holes should be detected rather than filled in. The smoothing can be desirable if there are noise and scanning artifacts. However, in the cases we shall examine, texture actually dominates noise. A guarantee that no detail is lost is granted only when almost all raw data points are mesh vertices.

1.2. Raw data point set processing

Considering it impossible to mesh directly the raw data point set, the literature has considered more and more sophisticated smoothing and interpolation methods. Probably the most prominent and efficient one is the "Moving Least Square Surface" (MLS) introduced in [Lev03]. It is defined as the set of stationary points of an operator projecting each raw point on a local quadratic regression of the surface. The order n MLS algorithm estimates at each point a degree n polynomial surface from a set of weighted neighbors. The obtained 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]. Variations of the MLS algorithm for denoising point sampled surfaces and preserving edges were proposed in [FCOS05], [GTE*06], [OGG09], [LCOL07], [GG07]. Interpolating point set surfaces can be achieved by using a singular weighting function ([OGG09], [AA09], [SOS04]), but using the marching cubes to extract the iso-surface looses anyway the input point positions.

At first sight applying MCM to a data point set requires the separate computations of the surface intrinsic Laplacian (mean curvature) and of the normal. For meshes, the standard discretization of the Laplacian operator is done through the cotangent formula [MDSB02]. For point clouds, [BSW09] proposed the construction of a Laplacian operator for functions defined on point clouds (yet no result on real noisy shapes was presented). In [PKG06], 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 were shape morphing and shape editing.

In [UH08], another MCM discretization for point sets was 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.

1.3. Computing curvatures

Computing the principal curvatures reliably on a given surface is crucial for various applications, in particular the anisotropic filtering preserving sharp edges

[HP04], [MDSB02], or the sampling methods adapting the density to the surface curvatures [PGK02]. On meshes, the curvature estimation problem has already been investigated in [MDSB02] where the cotangent formula is proven and extended. [Tau95] derived 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 permits to compute the Weingarten map whose eigenvalues and eigenvectors are the principal curvatures and principal directions ([LFM96] among others). In [BC94] the principal curvatures are computed from an oriented raw data set without surface fitting by expressing the fundamental forms of a 3D surface as covariance matrices. Indeed, the covariance matrix of the point normals projected on the regression plane yields the principal curvatures and their directions. 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, like 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. MLS surfaces were also used to derive analytic expressions for the curvatures of point set surfaces [YQ07]. As far as meshes are concerned, a comparison of various curvature estimations can be found in [SMS*03].

The paper is divided as follows: Section 2 gives mathematical results proving the consistency of the proposed scale space algorithm and Section 3 analyzes the discretization problem. Sections 4, 5 describe the two main applications of the scale space: a point cloud orientation method and a faithful mesh construction for the raw data set. Comparative experiments are presented in Section 6.

2. Continuous Theory

This section investigates a new way of implementing the mean curvature motion by the iteration of a planar surface regression. The surface $\mathcal{M}$ supporting the data point set is assumed to be 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 $\mathcal{M}$. 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. In this system we can express the surface as a C^2 graph $z = f(x, y)$. By Taylor expansion,

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

Notice that the sign of z 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 the forthcoming proofs the cylindrical neighborhood will prove much handier than the spherical one. The next technical lemma justifies its use.

Lemma 1 Integrating on $\mathcal{M}$ any function $f(x, y)$ such that $f(x, y) = O(r^n)$ on a cylindrical neighborhood $\mathcal{C}_r(P)$ instead of a spherical neighborhood $\mathcal{B}_r(P)$ introduces an $o(r^{n+4})$ error. More precisely:

$$\int_{\mathcal{B}_r} f(x, y) dM = \int_{x^2+y^2<r^2} f(x, y) dx dy + O(r^{4+n}). \quad (3)$$

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 $dM(x, y) = \sqrt{1 + z_x^2 + z_y^2} dx dy$. One has $z_x = -k_1 x + O(r^2)$ and $z_y = -k_2 y + O(r^2)$. Thus

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

which yields

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

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

$$\int_{\mathcal{B}_r} f(x,y)dM = (1+O(r^2))\int_{\mathcal{B}_r} f(x,y)dxdy; \quad (5)$$

$$\int_{\mathcal{C}_r} f(x,y)dM = (1+O(r^2))\int_{\mathcal{C}_r} f(x,y)dxdy \quad (6)$$

$$= (1+O(r^2))\int_{x^2+y^2<r^2} f(x,y)dxdy.$$

Consider polar coordinates (ρ,θ) such that $x=\rho\cos\theta$ and $y=\rho\sin\theta$ with $0\rho \leq r$ and $0\leq\theta\leq 2\pi$. For $M(x,y,z)$ belonging to the surface $\mathcal{M}$, we have $z=-\frac{1}{2}\rho^2(k_1\cos^2\theta+k_2\sin^2\theta)+O(r^3)=-\frac{1}{2}\rho^2k(\theta)+O(r^3)$, where $k(\theta)=k_1\cos^2\theta+k_2\sin^2\theta$. The condition that (x,y,z) belongs to the neighborhood $\mathcal{B}_r(P)$ can therefore be rewritten as $\rho^2+z^2<r^2$, which yields

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

For each θ the extremal value $\rho(\theta)$ of this neighborhood satisfies $\rho(\theta)^2+\frac{1}{4}k(\theta)^2\rho(\theta)^4-r^2+O(r^5)=0$. Thus

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

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

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

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

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

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

$$\int_{\mathcal{B}_r} f(x,y)dxdy = \int_{C_r\cap M} f(x,y)dxdy + O(r^{4+n}), \quad (7)$$

and combining (5), (6) and (7) yields (3). □

2.2. Curvature Estimation

By Theorem 1 projecting a point onto the neighborhood barycenter approximates the mean curvature motion. We discuss later on why this result cannot be used for implementing the mean curvature motion.

Theorem 1 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

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

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

2.3. Surface motion induced by projections on the regression plane

The main tool of the proposed scale space will be a simple projection of each surface sample P on the surface local regression plane. This PCA regression plane is defined as the plane orthogonal to the least eigenvector of the centered local covariance matrix, and passing through the centroid of the neighborhood. The projection of P on this plane will be called P'. The next lemma compares the normal to the PCA regression plane with the normal to the surface, $\vec{n}(P)$.

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 characterized as the plane passing through the barycenter of the neighborhood $\mathcal{B}_r(P)$ and with normal $\vec{v}$ minimizing:

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

Denoting by (v_x,v_y,v_z) the coordinates of $\vec{v}$,

$$I(\vec{v}) = \int_{B_r}(v_xx+v_yy-v_z\frac{1}{2}(k_1x^2+k_2y^2)+o(r^2))^2dxdy.$$

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 $\vec{n}(P)$. The

simple projection of each surface point P onto its local regression plane approximates a 3D scale space (mean curvature motion) as shown in the next theorem.

Theorem 2 Let T_r be the operator defined on the surface $\mathcal{M}$ transforming each point P into its projection $P' = T_r(P)$ on the local regression plane. Then

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

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

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

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

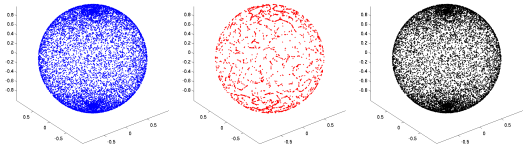
3. The discrete algorithm

The previous theorems assume that the surface is a uniform Lebesgue measure. A constant sampling density is therefore necessary. This constant density will be approximated on discrete data by weighting each point by a weight inversely proportional to its initial density. More precisely, let p be a point and $\mathcal{N}_r(p) = \mathcal{M}_s \cap B_r(p)$. Each point q should ideally have a weight $0 \leq w(q) \leq 1$ such that $\sum_{q\in\mathcal{N}_r(p)} w(q) = 1$. 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 $w(p) = \frac{1}{\sharp(\mathcal{B}_p(r))}$, as proposed in [UH08]. 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 $\vec{v}$ to this plane is the direction $\vec{v}$ minimizing $\sum_{q\in\mathcal{N}_r(p)} w(q)\langle (p_i - O), \vec{v}\rangle^2$.

Effectiveness of Theorems 1 and 2. Both Theorems permit *a priori* to implement the mean curvature motion on the raw data point set *without any previous orientation*. Nevertheless, the numerical application of these theorems depends on the assumption that a uniform Lebesgue measure on the surface is well represented by a uniform sample density. This is not true for the barycenter method of Theorem 1. Iterating the barycenter method with a small neighborhood and a slightly varying sample density leads to a local clustering of the samples. Indeed, the barycenter method provokes a normal motion, but also a non negligible tangential motion to the surface. This motion is precisely the one used in the Mean Shift method [Che95] for data clustering. This undesirable clustering effect is illustrated in fig. 1. Even though the point distribution on the sphere is probabilistically uniform, local clustering occurs by taking local barycenters. In contrast, for the projection filter there is no observable tangential shift on the right image of fig. 1. Theorem 2 is in that case consistent with its numerical implementation. This follows from the obvious fact that any (non aligned) irregular sampling of a plane permits to *exactly* recover the plane by linear regression.



(a) Original samples (b) 4 barycenter iterations (c) 4 projection filter iterations

Figure 1: *Comparison of the iterated barycenter and of the iterated projection filter on a randomly sampled sphere. Both motions are consistent with the mean curvature motion, but the iterated barycenter provokes clustering.*

Back propagation A normal motion by mean curvature can be defined for every point P_0 on the initial surface as a solution of (1) ($\frac{dP}{dt} = H(P)\vec{n}(P)$) 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. Higher order regression surfaces

The authors of [CP03] proved that a degree n polynomial fitting estimates all k^{th}*order* differential quantities to accuracy $O(h^{n-k+1})$. In [PKG06] an order 2 Moving Least Squares (MLS2) method projecting the point onto the locally fitted least squares surface was actually proposed as a scale space operator. Yet these iterated projections cannot be consistent with MCM, because by definition they leave invariant all degree two surfaces.

Can iterated MLS2 give a better estimate of the curvature than the projection filter? Comparative experiments were performed on a randomly and uniformly sampled sphere with added Gaussian noise. The point samples were filtered four times by T_r. By Theorem 2, each filtering step gives an estimate of the mean curvature. The same sampled sphere was filtered by MLS2, the surface mean curvature being computed as the mean curvature of the approximating surface at each point. This estimate is very exact, since the difference on a C^4 surface between a point and its MLS2 estimate can be proved to be $O(r^4)$. Both mean curvature estimates are compared by their mean and standard variations in the table of fig. 2. The result shows that when the noise level increases the planar projection yields a much more stable computation (see the fast decay of the standard variation for the curvature estimate). This experiment is also coherent with the MCM consistency theorem. Indeed, the planar projection yields a point set with (slowly) increasing curvatures (once the noise is removed, i.e., once the standard variation is stable). This explains why T_r at iteration 10 gives a curvature 1.05 and not 1. This result is accurate, having standard variation 0.01.

Fig. 3 is another illustration in 1D of the same phenomena: a circle with radius 1 and added Gaussian noise with variance 0.05 is denoised by iterated T_r and by an iterated MLS2 projection using the same neighborhood radius. In 1D, T_r becomes a simple line regression and the MLS2 surface a degree 2 polynomial curve. The simplest MLS2 method is used: it merely performs a weighted least squares polynomial regression on the local neighborhood. The neighbors weights are equal to $G(d)$ where G is a Gaussian and d is the distance between the neighbor and the center point. The standard variation of G is equal to the neighborhood radius.

Noise	0.01	0.05	0.1
T_r 1	1.00/1.95	1.15/5.57	1.27/4.76
T_r 2	0.99/0.07	1.02/2.16	1.17/4.89
T_r 3	1.00/0.02	1.01/0.16	1.05/2.10
T_r 4	1.01/0.01	1.01/0.05	1.02/0.27
T_r 10	1.04/0.01	1.05/0.01	1.09/0.04
MLS2 1	0.94/0.22	0.11/2.58	−0.42/2.99
MLS2 2	1.01/0.13	1.02/0.49	0.62/1.36
MLS2 3	1.01/0.10	1.02/0.36	1.06/0.68
MLS2 4	1.00/0.08	1.02/0.30	1.05/2.19
MLS2 10	1.00/0.04	1.01/0.14	1.02/0.74

Figure 2: *Comparison of mean curvature estimates on a noisy sphere with radius* 1 *(mean/standard variations) given by iterated planar projection (T_r) and iterated MLS2 regression (iterations 1, 2, 3, 4, 10). The curvature is evaluated at all points as the displacement along the normal induced by the planar projection (as stated in Thm 2) for the planar case, and by the explicit computation of the MLS2 surface mean curvature in the MLS2 case. The same radius is used for both iterations and both regressions.*

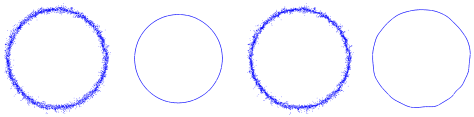


Figure 3: *Denoising a noisy circle with (from left to right)* $1, 100$ *iterations of* T_r *and* $1, 100$ *iterations of MLS2. Even after* 100 *iterations the oscillations removed by* T_r *persist with MLS2. The sphere radius decreases with* T_r*, which is consistent with MCM.*

4. First application: scale space raw data point orientation

Given an initial non oriented raw point cloud the surface orientation is a much needed information before meshing. The eigenvector of the least eigenvalue of the local covariance matrix is a classic approximation of the normal. We must then pick one of two possible orientations, and this choice must be globally coherent. The idea is to start by picking a random orientation for one point and to propagate it to the neighboring points. Now, sharp edges or a messy surface could fool such a propagation. If, however, the surface is smooth 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 one point to the next if their normal directions are similar (algorithm 1).

The input parameters for the scale space orientation (algorithm 2) are the radius r and a threshold $0 \leq t \leq 1$. Steps from 8 to the end are necessary because adding neighbors of points to the stack might not be enough to cover the entire cloud due to sampling irregularities. Once this procedure is over, there might remain

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 the normal direction $\mathbf{n}$ of p by local PCA;
$\bar{\mathbf{n}} \leftarrow$ unit mean of neighboring oriented normals;
if $(\bar{\mathbf{n}} \cdot \mathbf{n})^2 > t$ **then**
  if $\bar{\mathbf{n}} \cdot \mathbf{n} > 0$ **then**
    $\mathbf{n}(p) = \mathbf{n}$;
  else
    $\mathbf{n}(p) = -\mathbf{n}$;
  Return true;
else
  Return false;

non oriented points. These points are usually isolated points, and the simplest choice is to ignore them. In all our experiments the number of remaining non oriented points was below 0.1%. At step 10 the radius is multiplied by an $\alpha > 1$ factor. 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, the normal direction is recomputed by local PCA for all points and the chosen orientation is the one which has positive scalar product with the previous normals. This is an application of the scale space paradigm, where the information is computed at a coarse scale and propagated back to the finest scale.

5. Second application: scale space meshing

We now discuss how to build a mesh on the raw point cloud. Direct meshing is not possible because of the surface oscillation due to fine texture and noise. The idea is again to perform meshing on the smoothed surface and to transport this mesh back on the original point cloud. An efficient triangulation technique, the ball pivoting method [BMR*99] is used in all experiments for the coarse scale triangulation. The crucial faithfulness requirement is that the final vertices of the mesh must be an overwhelming majority of the raw data set points. This conservative requirement, incompatible with level set methods ([KBH06], [HDD*92], [LC87]) is described in Algorithm 3.

Parameters of scale space meshing The radius can be set automatically while computing 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

Algorithm 2: Scale space Orientation

Data: A point cloud $\mathcal{P}$, a radius r, an update parameter $\alpha > 1$

Iterate the projection filter T_r and keep track of each raw data point sample (mean curvature motion);
Find a point p_0 in a flat area, pick its orientation and mark it as oriented. Add its neighbors to the stack $\mathcal{S}$;
while $\mathcal{S}$ *is not empty or* $\mathcal{S}$ *does not become constant* **do**
  Take p the first point in $\mathcal{S}$;
  if *orientateFromNeighbors(p,r,t)* **then**
    Mark the point as oriented and remove p from $\mathcal{S}$;
  Add the neighbors of p to $\mathcal{S}$;
Add all remaining unoriented points to $\mathcal{S}$;
while $\mathcal{S}$ *is not empty and* $\sharp\mathcal{S}$ *does not become constant* **do**
  $r = \alpha r$;
  for p *in* $\mathcal{S}$ **do**
    Perform *orientateFromNeighbors(p,r,t)*;

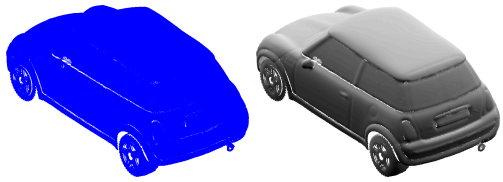


Figure 4: *A raw point set (left) and its orientation (right). Points in the right figure are given a gray value equal to the scalar product of their normal and the lighting orientation.*

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. The projection filter requires at least three neighbors per point to estimate a regression plane, but a robust estimate is experimentally attained with 30 neighbors. Of course, since the same radius is used for all points, it may occur that there are not enough neighbors to perform the plane regression. Those points must be eliminated, but in all the experiments less than 0.1% of points were removed this way. These points are mostly outliers, or isolated points in folds of the acquired object. Al-

though their relative number is low, nonetheless this represents some thousands points that are eliminated.

Algorithm 3: Scale space meshing

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).

Once the minimal number of neighborhood points has been fixed (and it has been fixed once and for all on all experiments to 30), the radius is also fixed and the meshing scale space only depends on the number of scale space iterations. When setting the radius automatically as described above, it was found that four iterations were always enough to smooth the point cloud and build the mesh. Thus, the scale space is conceived as a very local motion securing a reliable tangent space. In all experiments the points barely moved (less than 40μ for the Tanagra point set). The scale space and the ball pivoting reconstruction use the same ball radius. The parameter of the Poisson reconstruction (namely the octree depth) was set to be the largest allowed by our computing equipment (namely a 8 $3Ghz$ processors computer with 48 Go RAM).

Transporting back the connectivity information (step 3) can in theory lead to a self intersecting 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 would be no serious visual inconvenience. Second, no such crossing was found in many experiments on about twenty very large data point sets.

6. Comparative experiments on high resolution data

The algorithms were devised for highly accurate point clouds acquired by a laser scanner. A typical example of the scanned objects is a mould of a fourth century B.C. Tanagra figurine acquired at the Museum of Cycladic Arts, Athens (Fig. 6(a)). It is 22cm high and the point cloud contains $6 \cdot 10^6$ points.

Thanks to a very accurate calibration of the laser scanner device, the output is a well registered non oriented point cloud containing a negligible warp. Tests were also made on objects of the Stanford Fragment Urbis Romae database. In that case a registration of the raw sweeps is needed to have a point cloud representing the whole object. Preferring not to address the sweep registration problem in this paper, we will use single sweeps for our meshing experiments and show that considerable texture information can be recovered from each sweep. Figs. 6 and 6(f) show the application of scale space meshing with a mesh rendering at fine and coarse scale. We can see on Figs. 6 and 6(f) that the surface texture is lost at a coarse scale, but completely and accurately recovered by scale space meshing. Comparing the back projected mesh to the result of a direct meshing of the initial samples (Fig. 7) 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. T_r has been proposed as the simplest smoothing operation implementing the mean curvature motion. It may be objected that the surface could also be directly approximated by the classic order 2 moving least square method (MLS2). The most objective way to compare T_r and $MLS2$ was to implement them with exactly the same neighborhood radius. Fig. 7(f) shows the comparative result on one of the finest details of the Tanagra data set. The results are similar in terms of detail quality, yet the computation time was doubled, and we have seen (Fig. 2) that MLS2 does not deliver a scale space and keeps the smoothed out noise. Fig. 12 shows a comparison between the reconstruction obtained by the VRIP reconstruction method (see [CL96]) and scale space meshing. The scale space method produces a significantly more precise mesh, as can be seen on the close up of Fig 9. Fig. 8 shows the scale space reconstruction of one scan of a fine scale object (i.e. the mesh back-projected at all scales). Fig 10 compares the mesh reconstruction by several meshing algorithm with scale space meshing. The experiment clearly rules out both Ball Pivoting algorithm and Poisson Reconstruction. Two MLS methods were also tested. APSS ([GG07]) and RIMLS ([OGG09]). APSS builds an implicit function by evaluating the distance between each evaluation point and an algebraic spherical fit of the surface. Although this last method is not explicitly devised for meshing, the iso-surface can be extracted using the marching cubes. RIMLS is another modification of the standard MLS procedure. It is based on minimizing an objective function that gives less weight to spatial and normal outliers (i.e., sparse points and features). Here, the marching cubes are explicitly mentioned for extracting the surface. For both methods, the resolution depends on the marching cubes grid resolution: it was set so that increasing it would not change much the visual aspect.

(a) Original samples (b) Projection motion (c) Resulting samples (d) Coarse scale mesh (e) Scale space mesh

Figure 5: *2D example of the steps performed by the scale space meshing algorithm.*

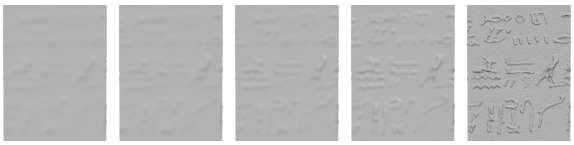


Figure 8: *Back-projecting the mesh of a single scan of a fine-scale object (engravings are around* $0.1mm$ *deep). From left to right: mesh built after* 4 *scale space iterations; back-projection to levels* $3, 2, 1$ *and* 0 *(final mesh).*

(a) RIMLS (b) APSS (c) Scale space

Figure 11: *Detail of the mesh built using (from left to right) RIMLS, APSS and scale space meshing. Notice the horizontal artifacts caused by the isosurface extraction for both APSS and RIMLS methods. The tiny vertical ridges restituted by the scale space meshing are present in the raw set: they reveal the scanning direction.*

Although the results by both methods are visually close to scale space meshing, the proposed scale space meshing is much simpler. Processing directly the raw points, it skips the iso-surface extraction. It is also the only method which preserves input samples and does not add additional vertices (both APSS and RIMLS actually introduce more than twice the number of input samples). Another problem is that the iso-surface extraction by marching cubes introduces aliasing-like artifacts which are avoided by scale space meshing (see Fig. 11). The tiny vertical ridges restituted by the scale space meshing are present in the raw set: they reveal the scanning direction. It is precisely one of the scopes of the method to be able to visually check the tiniest problems in the scanning process.

Fig. 6 displays the many acquisition holes at the bottom of the Tanagra figurine, in the folds of the tunic or near the right foot. By the scale space meshing these holes are not filled in and can be detected. 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 as holes.

Method	Wave 1	Wave 2	sphere	sharp
Scale space	0.19	0.28	0.04	0.04
BPA	0.18	0.24	0.04	1.2
Poisson	1.5	43	0.24	4

Figure 13: *Quantitative comparison of scale space meshing, ball pivoting, and Poisson reconstruction: RMSE of the distance from the triangle barycenters to the real surface. All results are multiplied by* 10^3 *for readability*

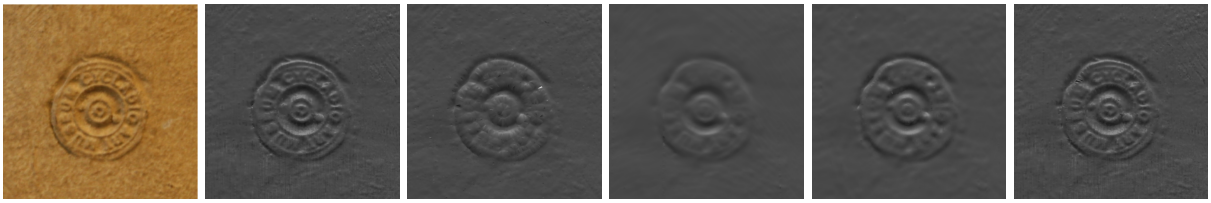
Fig. 7 illustrates the loss of detail with level sets methods. Level set methods extract the zero level set of the signed distance to the surface. Thus, they do not contain the input points and loose track of them.

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 the table of fig. 13. 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. 7(c) where BPA looses thin details.

Limitations The method introduced in this paper builds an exact mesh for any raw input point cloud. Therefore, all imperfections of the input data are visible. Meshes produced by this meshing method are not smoothed at all, and are by no means economical in terms of vertex number. If the goal is to build an economical mesh of a closed surface, with no special intention for detail preservation, then it is not the adequate method. Yet, to fix the scanning imperfections, the raw data has to be visualized. In particular the light offsets caused by misalignment errors between two partial scans become terribly conspicuous.

(a) Initial object (b) Coarse scale (c) Fine scale (d) Picture (e) Selection (f) Details

Figure 6: *Multi-resolution mesh reconstruction of the Tanagra point set (22 cm high) illustrating the recovery of fine texture. All back propagated textures are present on the original object.*



(a) Picture (b) Scale space mesh (c) Direct mesh (d) Poisson Mesh (e) MLS2 + BPA (f) MLS2 scale space

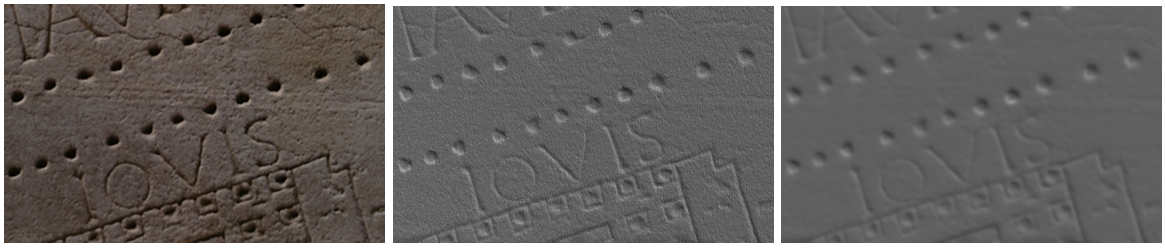
Figure 7: *Comparison between several meshing methods on a* 1*cm high logo. The direct mesh (7(c)) creates many spurious triangles. The Poisson reconstruction [KBH06] clearly smoothes out all details (7(d)). Filtering the logo by order two MLS and meshing the points by the ball pivoting algorithm (7(e)) also creates a smooth mesh. Fig 7(f) shows the result of applying the same scale space strategy with the projection on the order* 2 *MLS surface instead of the regression plane. The result is similar to 7(b) in detail quality but the computation time is double.*

In this case, the scale space meshing method, since it preserves all points, does not fix the misalignment problem, as would Poisson for example. Fixing this problem is handled in [DMAL10], which uses a development of the scale space meshing. The scale space meshing method must therefore be considered as a preliminary visualization method in a scanning loop, permitting to visualize the raw data point set, and to uncover all imperfections at an early acquisition stage. While outliers are automatically eliminated by the rule asking for a dense enough neighborhood, it is clear that, in contrast to level sets methods, holes in the shape are not filled in.

7. Complexity analysis and computation time measures

One scale-space projection requires the following operations: look for neighbors within radius r, build their covariance matrix and their centroid, perform PCA of this 3×3 covariance matrix. Therefore, once the neighbors are found, they are sequentially scanned in order to build the covariance matrix and the centroid. This yields 6 multiplications and additions per point for the covariance matrix update and 3 additions per point for the centroid update. The PCA complexity does not depend on the number of neighbors: it requires 9 operations. Knowing the least eigenvector, the projection is only 12 operations. There is one list scan (9 operations per processed point) and 21 operations once the covariance and centroid are built. Assuming we have 30 neighbors, this yields a total of 200 operations per point. Finally finding the neighbors in the octree is $O(logN)$ (average) and one scale space iteration therefore is $O(N(logN + 200))$ operations, where N is the total number of points in the point cloud.

The computation time needed for meshing the Tanagra point set with six millions points was as follows: Sorting the points in the octree takes $1.2s$. The scale space iterations require 3 min, leading to a total computation time of $19min$ for orientation and of $27min$ for the whole meshing on an 8 $3Ghz$ processors computer with 48 Go RAM. The maximum memory usage was less than $2Go$. These figures should be compared with the time required for directly meshing the oriented point set by the ball pivoting method without any scale space iterations, which took $25min$. Therefore, only a two additional minutes were used to get a much more faithful mesh.



(a) Original Fragment (b) Scale space mesh (c) Level Set mesh

Figure 9: *Closeup of a piece of the (FUR) database reconstructed by Scale Space Meshing and Poisson Reconstruction.*

Figure 10: *Comparison of the Rosette reconstruction (Picture (a)) using Ball Pivoting Algorithm (b), Poisson Reconstruction (c), RIMLS (d), APSS (e), and scale space meshing (f). APSS and RIMLS yield results that are really close to ours, yet both methods need an isosurface extraction done with the marching cubes, which creates strong artefacts (see a closeup Fig 11). Besides, RIMLS and APSS meshes contain around* 268500 *vertices whereas the scale space mesh contains* 132203 *vertices. Notice also that APSS and RIMLS introduce some denoising (visible especially in the nearly flat parts). Scale space meshing is the only method that preserves exactly the input data.*

8. Conclusion

The increasing accuracy of 3D triangulation scanners requires an effort to reconsider the whole rendering chain, and to obtain high quality visualization. The present paper has proposed a strategy to mesh the raw original surface, therefore ensuring a faithful rendering of textures, detail, and the detection of scanning artifacts and holes. Future work will test a closed scanning loop with our experimental scanner to direct the scanner toward the detected holes.

Acknowledgements The authors thank Prof. Marc Levoy and the Stanford Digital Forma Urbis Romae Project (http://formaurbis.stanford.edu) for the data used in figs 9, 12 property of the Sovraintendenza of Rome and of Stanford University. The authors also thank Prof. Tamy Boubekeur for advice and several additional references. The authors acknowledge support by D.G.A., French Ministry of Defense, Centre National d'Etudes Spatiales (R&T MISS), Office of Naval research (grant N00014-97-1-0839) and the E.R.C. advanced grant "Twelve labours".

References

[AA09] Alexa M., Adamson A.: Interpolatory point set surfaces convexity and hermite data. *ACM Trans. Graph. 28* (May 2009), 20:1–20:10. 2

[ABCO*03] Alexa M., Behr J., Cohen-Or D., Fleishman S., Levin D., Silva C. T.: Computing and rendering point set surfaces. *IEEE TVCG 9*, 1 (2003), 3–15. 2

[AMD02] Alliez P., Meyer M., Desbrun M.: Interactive geometry remeshing. *ACM Trans. Graph. 21*, 3 (2002), 347–354. 1

[BC94] Berkmann J., Caelli T.: Computation of surface geometry and segmentation using covariance techniques. *IEEE PAMI 16*, 11 (1994), 1114–1116. 3

[BMR*99] Bernardini F., Mittleman J., Rushmeier H., Silva C., Taubin G.: The ball-pivoting algorithm for surface reconstruction. *IEEE TVCG 5* (1999), 349–359. 7

[BSW09] Belkin M., Sun J., Wang Y.: Constructing laplace operator from point clouds in rd. In *Proc. SODA '09* (USA, 2009), SIAM, pp. 1031–1040. 2

[Che95] Cheng Y.: Mean shift, mode seeking, and clustering. *IEEE PAMI 17*, 8 (1995), 790–799. 5

[CL96] Curless B., Levoy M.: A volumetric method for building complex models from range images. In *SIGGRAPH '96* (USA, 1996), ACM Press, pp. 303–312. 8

[CP03] Cazals F., Pouget M.: Estimating differential quantities using polynomial fitting of osculating jets. In *SGP '03* (Switzerland, 2003), Eurographics, pp. 177–187. 6

[CSM03] Cohen-Steiner D., Morvan J.-M.: Restricted delaunay triangulations and normal cycle. In *Proc. SCG '03* (USA, 2003), ACM, pp. 312–321. 3

[DMAL10] Digne J., Morel J.-M., Audfray N., Lartigue C.: High fidelity scan merging. *CGF 29*, 5 (2010), 1643–1651. SGP2010. 10

[FCOS05] Fleishman S., Cohen-Or D., Silva C. T.: Robust moving least-squares fitting with sharp features. *ACM Trans. Graph. 24*, 3 (2005), 544–552. 2

[GG07] Guennebaud G., Gross M.: Algebraic point set surfaces. *ACM Trans. Graph. 26* (2007). 2, 8

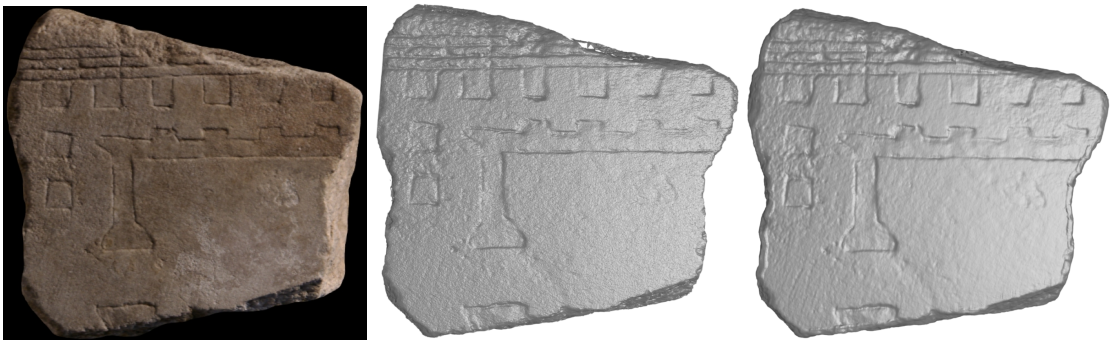


Figure 12: *Comparison on a piece of the Fragment Urbis Romae (FUR) (left: picture). Texture and details are better recovered on the back-propagated mesh (middle) than on the VRIP reconstruction available on the FUR website (right)*

[GTE*06] Gois J. P., Tejada E., Etiene T., Nonato L. G., Castelo A., Ertl T.: Curvature-driven modeling and rendering of point-based surfaces. *Computer Graphics and Image Processing, Brazilian Symposium on* (2006), 27–36. 2

[HDD*92] Hoppe H., DeRose T., Duchamp T., McDonald J., Stuetzle W.: Surface reconstruction from unorganized points. In *SIGGRAPH '92* (USA, 1992), ACM Press, pp. 71–78. 2, 7

[HP04] Hildebrandt K., Polthier K.: Anisotropic filtering of non-linear surface features. *CGF 23* (2004), 391–400. 3

[KBH06] Kazhdan M., Bolitho M., Hoppe H.: Poisson surface reconstruction. In *SGP '06* (Switzerland, 2006), Eurographics, pp. 61–70. 2, 7, 10

[LC87] Lorensen W. E., Cline H. E.: Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87* (USA, 1987), ACM Press, pp. 163–169. 2, 7

[LCOL07] Lipman Y., Cohen-Or D., Levin D.: Data-dependent mls for faithful surface approximation. In *SGP '07* (Switzerland, 2007), Eurographics, pp. 59–67. 2

[Lev03] Levin D.: Mesh-independent surface interpolation. In *Geometric Modeling for Scientific Visualization* (2003), Brunnett H., Mueller, (Eds.), Springer-Verlag, pp. 37–49. 2

[LFM96] Lengagne R., Fua P., Monga O.: Using crest lines to guide surface reconstruction from stereo. In *ICPR '96: Volume I* (USA, 1996), IEEE, p. 9. 3

[LP05] Lange C., Polthier K.: Anisotropic smoothing of point sets. *CAGD 22*, 7 (2005), 680–692. 3

[MDSB02] Meyer M., Desbrun M., Schröder P., Barr A.: Discrete differential geometry operators for triangulated 2-manifolds. In *International Workshop on Visualization and Mathematics* (2002). 2, 3

[MSR07] Magid E., Soldea O., Rivlin E.: A comparison of gaussian and mean curvature estimation methods on triangular meshes of range image data. *CVIU 107*, 3 (2007), 139–159. 3

[OGG09] Oztireli A. C., Guennebaud G., Gross M.: Feature preserving point set surfaces based on non-linear kernel regression. *CGF 28* (2009), 493–501(9). 2, 8

[PGK02] Pauly M., Gross M., Kobbelt L. P.: Efficient simplification of point-sampled surfaces. In *Proc. VIS '02* (USA, 2002), IEEE, pp. 163–170. 3

[PKG06] Pauly M., Kobbelt L. P., Gross M.: Point-based multiscale surface representation. *ACM Trans. Graph. 25*, 2 (2006), 177–193. 2, 5, 6

[PWHY09] Pottmann H., Wallner J., Huang Q.-X., Yang Y.-L.: Integral invariants for robust geometry processing. *CAGD 26*, 1 (2009), 37–60. 3

[PWY*07] Pottmann H., Wallner J., Yang Y.-L., Lai Y.-K., Hu S.-M.: Principal curvatures from the integral invariant viewpoint. *CAGD 24*, 8-9 (2007), 428–442. 3

[Rus04] Rusinkiewicz S.: Estimating curvatures and their derivatives on triangle meshes. In *3DPVT '04* (USA, 2004), IEEE, pp. 486–493. 3

[SMS*03] Surazhsky T., Magid E., Soldea O., Elber G., Rivlin E.: A comparison of gaussian and mean curvatures estimation methods on triangular meshes. In *Proc. ICRA 2003* (2003). 3

[SOS04] Shen C., O'Brien J. F., Shewchuk J. R.: Interpolating and approximating implicit surfaces from polygon soup. In *SIGGRAPH'04* (USA, 2004), ACM Press., pp. 896–904. 2

[Tan05] Tang X.: A sampling framework for accurate curvature estimation in discrete surfaces. *IEEE TVCG 11*, 5 (2005), 573–583. 3

[Tau95] Taubin G.: Estimating the tensor of curvature of a surface from a polyhedral approximation. In *ICCV '95* (USA, 1995), IEEE, p. 902. 3

[TRZS04] Theisel H., Rossl C., Zayer R., Seidel H.-P.: Normal based estimation of the curvature tensor for triangular meshes. In *Proc. PG' 04* (USA, 2004), IEEE, pp. 288–297. 3

[UH08] Unnikrishnan R., Hebert M.: Multi-scale interest regions from unorganized point clouds. In *Workshop on Search in 3D (S3D), IEEE CVPR* (2008). 2, 5

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

[YQ07] Yang P., Qian X.: Direct computing of surface curvatures for point-set surfaces. In *Proc. PBG 07* (2007). 3