A scale space approach to the processing of point clouds
Master MVA - 2011/2012

Julie Digne
INRIA Sophia Antipolis

Introduction

3D data can now be generated in multiple ways: triangulation laser scanner, stereoscopy, LIDAR... The precision of the acquired surface increases with the technological advances, yet it depends strongly on the acquisition type. We focus here on surfaces obtained by triangulation laser scanner and will describe a processing method derivated from the scale space approach developed for 2D images ([Wit83]). We give in the remainder of this introduction a short description of the acquisition process. Section 2 gives the definition and theoretical results of the scale space for point clouds. Section 3 presents the first application of the scale space: the normal computation and orientation of point clouds. Section 4 presents the second application of the scale space: the reconstruction of an exhaustive point set interpolating triangular meshed surface. Section 5 describes a third application: the seamless merging of scans once they are registered.

A typical acquisition process

A triangulation laser scanner is an acquisition device based on a camera and a laser emitter, that are fixed together. By triangulation (hence the name) of the intersection point between the laser ray and the surface of the object, the 3D coordinate of the surface can be inferred. The coordinate in the CCD camera coordinate system of the impact point yields by system calibration the 3D coordinates of the points. This triangulation is summed up on figure 1. In our case, this camera-laser ray emitter device is set on a revolving arm, so that all sides of an object (with the exception of the base) can be acquired without moving the object. More informations and the data are published online ([DAL*11]).

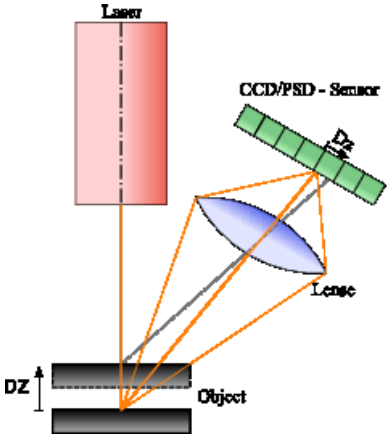


Figure 1: Scheme of a typical triangulation laser scanner

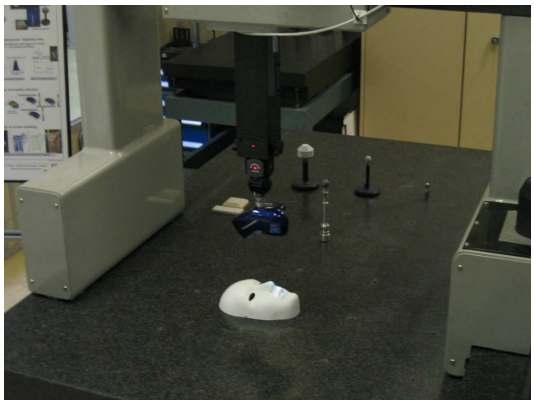


Figure 2: Picture of the complete acquisition system

1 Prerequisites

1.1 Definition of a surface

We shall call "surface" a two-dimensional manifold embed in a higher dimensional space (3 Euclidean space in our case). Saying that $\mathcal{M}$ is 2 dimensional means that for each surface point P there exists a neighborhood V around P where the surface is locally a graph $z = g(x, y)$.

For example, a sphere with radius 1 is a surface. There is no way to write it globally as $z = g(x, y)$ for x, y some coordinate system, but at each point P if one considers the half sphere around P then one can parameterize it as $z = \sqrt{1 - x^2 - y^2}$, so that the surface is locally a graph.

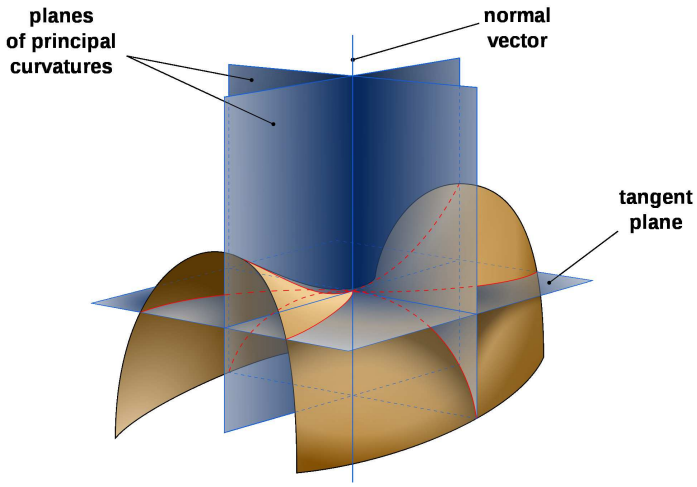


Figure 3: Curvatures of a surface. (Image by Eric Gaba, from Wikimedia Commons)

Here we will assume that the surface is $\mathcal{C}^2$. It means that at each point P one can define a tangent plane orthogonal to the surface normal $\vec{n}$ at P: by taking an orthogonal coordinate system defined on the tangent plane and centered at P, the surface is locally a $\mathcal{C}^2$ graph $z = g(x, y)$. Then by Taylor expansion one can write that on any orthogonal coordinate system on the tangent plane: $z = g(x, y) = ax^2 + by^2 + cxy + o(x^2 + y^2)$.

1.2 Curvatures

The principal curvatures k_1 and k_2 of a $\mathcal{C}^2$ surface are the eigenvalues of D^2g such that $k_1 \geq k_2$. D^2g is the second derivative of g:

$$D^2g = \begin{pmatrix} 2a & c \\ c & 2b \end{pmatrix}.$$

The principal directions $(\vec{t}_1, \vec{t}_2)$ are defined as the eigenvectors associated to the eigenvalues of D^2g. Notice that this definition is independent of the orthogonal basis chosen in the tangent plane. Principal directions and principal curvatures are therefore intrinsic quantities. Then $\vec{t}_1$ and $\vec{t}_2$ are orthogonal (eigenvectors of a symmetric matrix), see fig 3 for a representation of the principal directions.

One should notice that this definition only gives the directions of $(\vec{t}_1, \vec{t}_2)$ but bot their orientations. These orientations depend on the orientation choice of the normal (either inward pointing normal or outward pointing normal.

We will call intrinsic coordinate system the local coordinate system formed by $(P, \vec{t_1}, \vec{t_2}, \vec{n})$. In this coordinate system the surface can be expressed as a graph $z = g(x, y) = -\frac{1}{2}(k_1 x^2 + k_2 y^2) + o(x^2 + y^2)$.

The mean curvature H is defined as $H = \frac{1}{2}(k_1 + k_2)$.

1.3 Geometric interpretation of curvatures

Consider the curve C_1 (resp. C_2) formed by the intersection of the surface with the plane $(P, \vec{t_1}, \vec{n})$ (resp. $(P, t_2, \vec{n})$). Then k_1 (resp. k_2) is the curvature of the planar curve C_1 (resp. C_2). In other words, k_1 is, up to the sign, the inverse of the radius of the osculating circle of curve C_1 at P (see fig. 4).

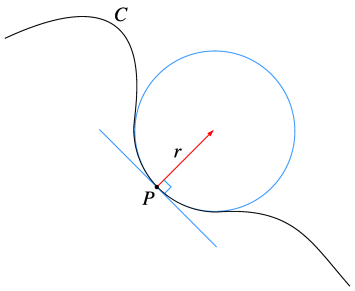


Figure 4: Osculating circle of a planar curve. Image from Wikimedia Commons.

1.4 Mean Curvature Motion

The equivalent of the heat equation for 2D images (isotropic diffusion) is the mean curvature motion for surface (MCM). Let S be a surface and P a point of S with normal $\mathbf{n}$ and mean curvature $H(P)$, then the mean curvature motion writes:

$$\frac{\partial P}{\partial t} = -H(P)\mathbf{n}(P) \tag{1}$$

1.5 Principal Component Analysis (PCA)

Principal Component Analysis is another tool we will use in this course. In a nutshell, PCA finds given a set of variables the directions capturing the highest data variation (see fig. 5).

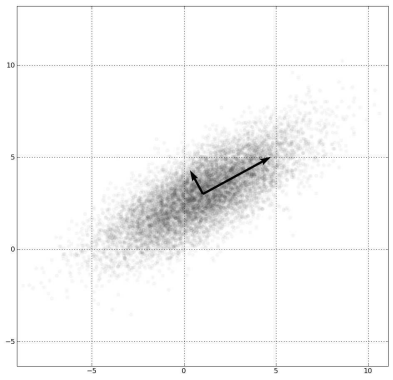


Figure 5: Principal directions found by PCA on a set of 2D points. Image by Ben FrantzDale from Wikimedia Commons.

More precisely if we have a set of points $(P_i)_{i=1\cdots N}$ the PCA first computes the barycenter O of these points:

$$O = \sum_{i=1}^{N} P_i$$

The centered covariance matrix of the P_i is:

$$\Sigma = \sum_{i=1}^{N} (P_i - O)^T (P_i - O)$$

Σ is a $3x3$ real symmetric matrix. It can therefore be diagonalized and its eigenvectors will be orthogonal.

Interpretation Each eigenvalue captures the variation of the point distribution along the corresponding principal direction. Principal Component Analysis therefore comes down to analyzing a distribution of points by looking for the directions that contain the largest variations. Section 2.2 will detail PCA and its use in the scale space theory.

Armed with these definitions we can now turn to defining the scale space for surfaces.

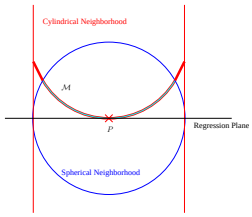


Figure 6: Cylindrical and spherical neighborhood

2 Theoretical results: definition of the scale space

2.1 Continuous Theory

This section describes a consistency result for a numerical approximation of the mean curvature motion: 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,

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

Notice that the sign of z depends on the arbitrary surface orientation.

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:*

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

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 $dM(x,y) = \sqrt{1+z_x^2+z_y^2}dxdy$. One has $z_x = -k_1x + O(r^2)$ and $z_y = -k_2y + O(r^2)$. Thus

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

which yields

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

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

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

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

Consider polar coordinates (ρ,θ) such that $x = \rho\cos\theta$ and $y = \rho\sin\theta$ with $0\rho \le r$ and $0 \le \theta \le 2\pi$. 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

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

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

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

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,

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

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

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

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

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

Surface motion induced by projections on the regression plane The main tool of the scale space is 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 $\vec{v}$,

$$I(\vec{v}) = \int_{B_r} (v_x x + v_y y - v_z \frac{1}{2}(k_1 x^2 + k_2 y^2) + o(r^2))^2 dxdy.$$

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

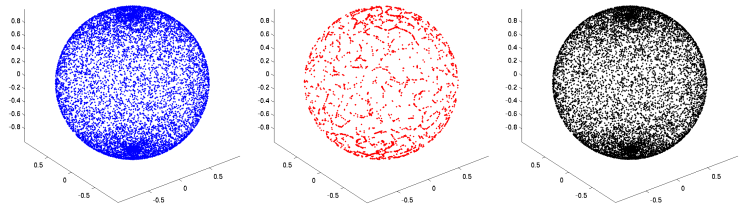
2.2 The discrete algorithm

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. 7. 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. 7. 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 7: 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 equation 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.*

2.3 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. Furthermore the first step of MLS2 is to compute a regression plane, which is sufficient to get the surface motion by mean curvature (Theorem 2).

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. 8. 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. 9 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 8: 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.

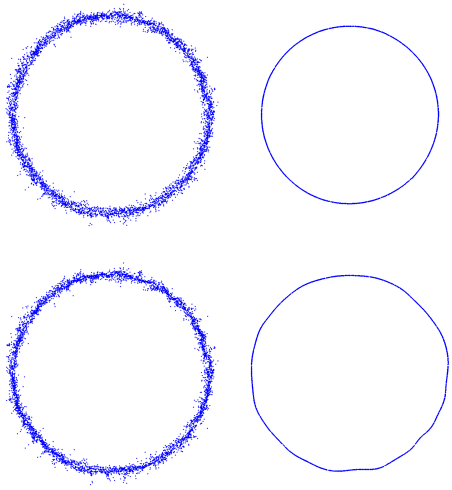


Figure 9: Denoising a noisy circle with (from left to right) $1,100$ iterations of T_r and $1,100$ iterations of MLS2. Even after 100 iterations the oscillations removed by T_r persist with MLS2. The sphere radius decreases with T_r, which is consistent with MCM.

3 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 non oriented points. These points are usually isolated

Algorithme 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 n of p by local PCA;
n̄ ← unit mean of neighboring oriented normals;
if (n̄ · n)^2 > t then
    if n̄ · n > 0 then
        n(p) = n;
    else
        n(p) = −n;
    Return true;
else
    Return false;
```

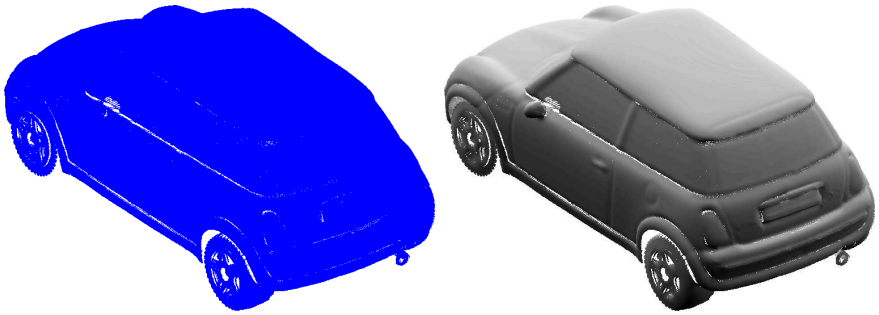


Figure 10: A raw point set (left) and its orientation (right). Points in the right figure are given a gray value equal to the scalar product of their normal and the lighting orientation.

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.

Algorithme 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)*;

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

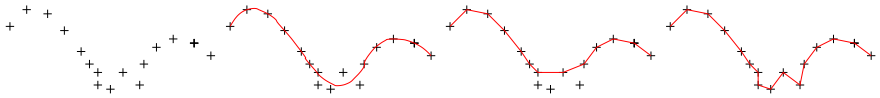
Algorithme 3: Scale space meshing

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.

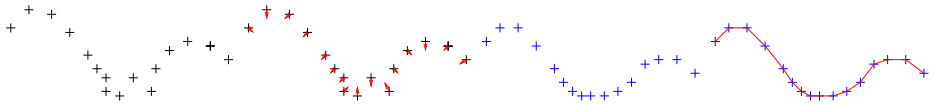
In this algorithm, at a low scale a mesh is built from the set of smoothed points. The only requirement for the meshing technique is that is should interpolate the points provided the underlying surface is smooth enough. A simple way to do that is the Ball Pivoting Algorithm ([BMR*99]). The idea beneath this meshing method is very simple and elegant: three points should be linked by a triangle if and only if one can fit a ball with radius r through the points and that this ball contains no other point. Starting with such



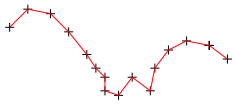
(a) Original samples (b) Level set method (c) Direct Meshing (d) Scale space meshing

Figure 11: Comparison of three meshing methods

a triplet of points, a ball is *pivoted* around each edge of the triangle until it meets another point, if the ball is empty then a new triangle is formed. This is in fact a heuristic to build a subset . It avoids building the complete delaunay tetrahedralization. Also it does not create triangle larger than a given radius. It can also handle very large datasets.



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



(e) Scale space mesh

Figure 12: 2D example of the steps performed by the scale space meshing algorithm.

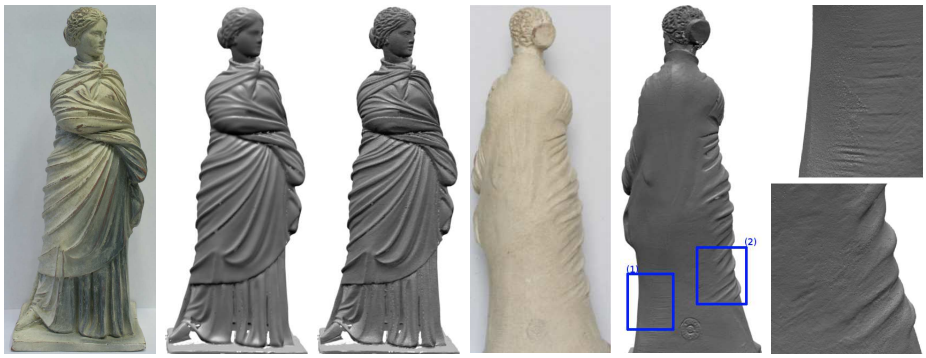
4.1 Comparative experiments on high resolution data

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. 13(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. We do not address teh scan registration here, so we will use single sweeps for our meshing experiments and show that considerable texture information can be recovered from each sweep. Figs. 13 and 14 show the application of scale space meshing with

a mesh rendering at fine and coarse scale. We can see on Figs. 13 and 14 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. 15) 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 is 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. 15(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. 8) that MLS2 does not deliver a scale space and keeps the smoothed out noise. Fig. 20 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 17. Fig. 16 shows the scale space reconstruction of one scan of a fine scale object (i.e. the mesh back-projected at all scales). Fig 18 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. Although the results by both methods are visually close to scale space meshing, 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. 19). 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. 13 displays the many acquisition holes at the bottom of the Tanagra



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

Figure 13: 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.

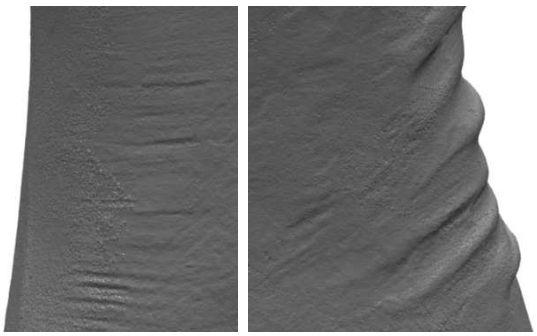
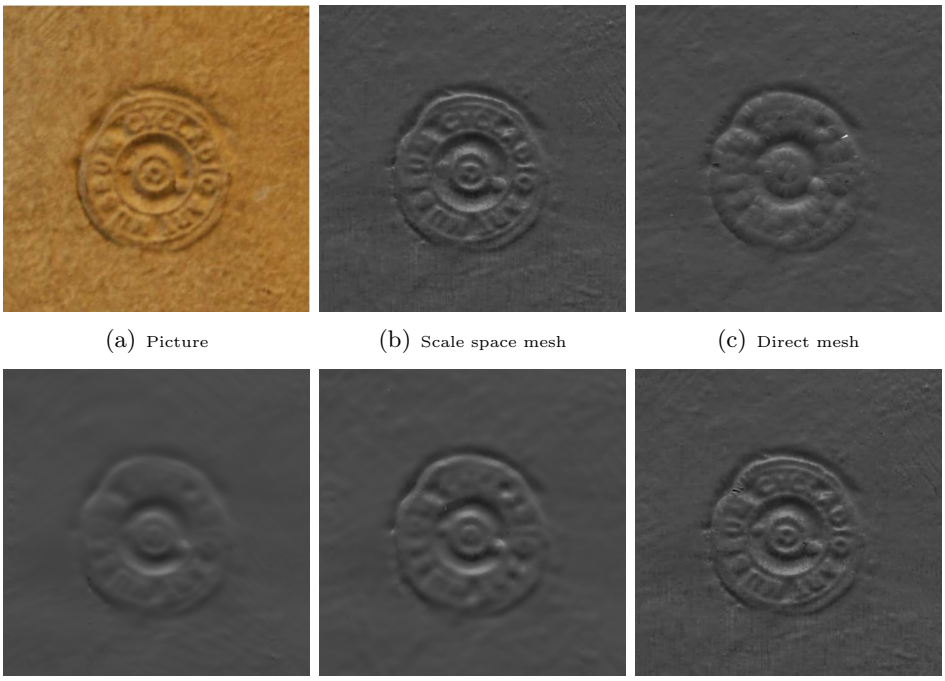


Figure 14: Expansion of the details (1) (left) and (2) (right) selected on fig. 13(e)



(a) Picture (b) Scale space mesh (c) Direct mesh

(d) Poisson Mesh (e) MLS2 + BPA (f) MLS2 scale space

Figure 15: Comparison between several meshing methods on a 1cm high logo. The direct mesh (15(c)) creates many spurious triangles. The Poisson reconstruction [KBH06] clearly smoothes out all details (15(d)). Filtering the logo by order two MLS and meshing the points by the ball pivoting algorithm (15(e)) also creates a smooth mesh. Fig 15(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 15(b) in detail quality but the computation time is double. See figs12-11 for an explanation of this difference.

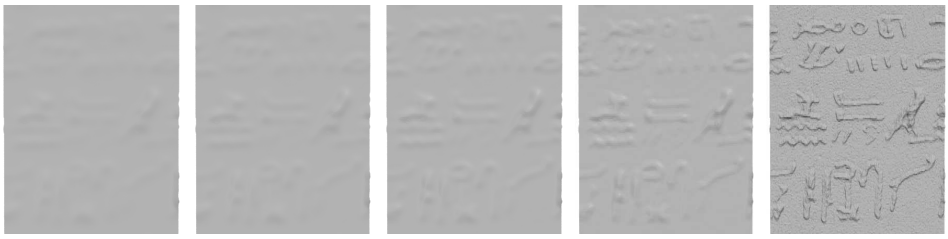
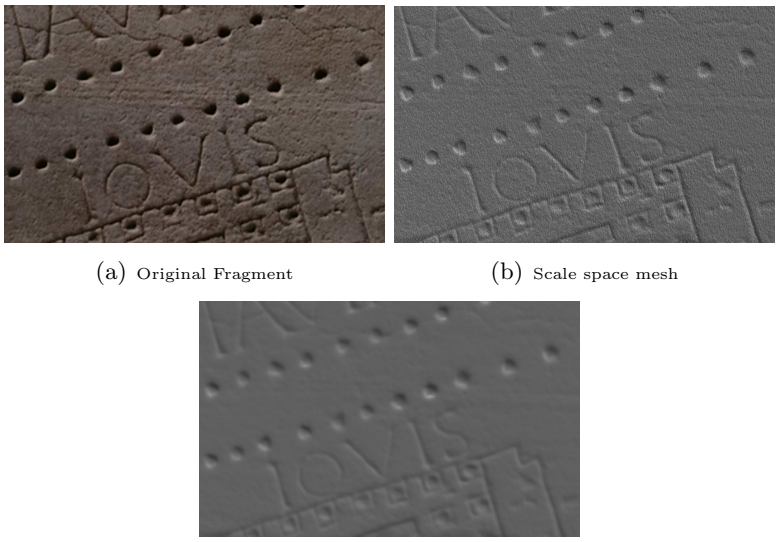


Figure 16: Back-projecting the mesh of a single scan of a fine-scale object (engravings are around $0.1mm$ deep). From left to right: mesh built after 4 scale space iterations; back-projection to levels $3, 2, 1$ and 0 (final mesh).



(a) Original Fragment (b) Scale space mesh

(c) Level Set mesh

Figure 17: Closeup of a piece of the (FUR) database reconstructed by Scale Space Meshing and Poisson Reconstruction.

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.

Fig. 15 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. Fig. 11 shows that not only these methods, but even direct meshing methods can miss small details. Fig. 12 illustrates why scale space meshing allows one to recover those details: standard meshing at a smooth scale is simply easy because details have been unfolded. It is then trivial to propagate back 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

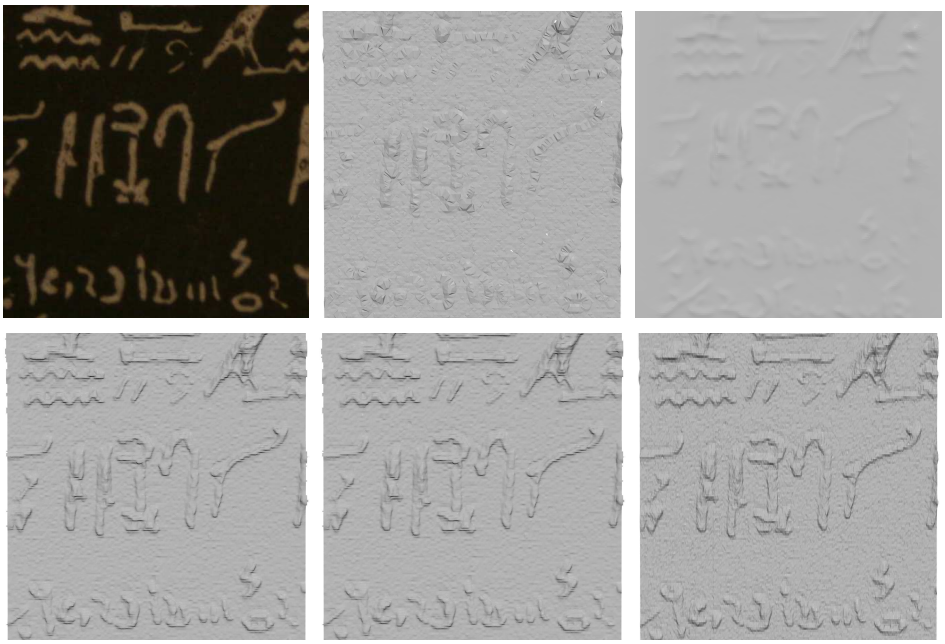
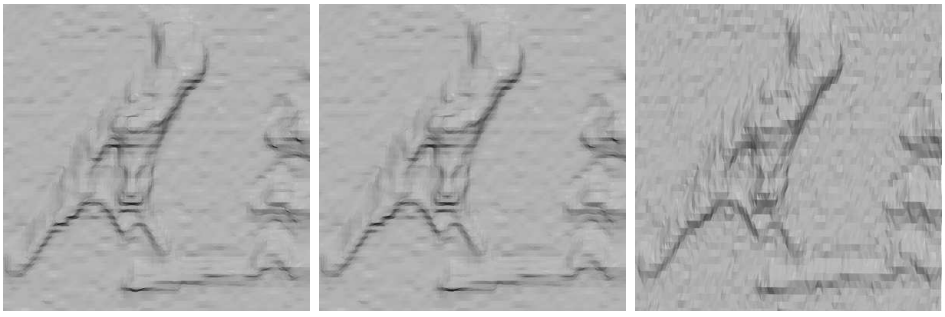


Figure 18: Comparison of the Rosette reconstruction (Picture (a)) using Ball Pivoting Algorithm (b), Poisson Reconstruction (c), RIMLS (d), APSS (e), and scale space meshing (f). APSS and RIMLS yield results that are really close to ours, yet both methods need an isosurface extraction done with the marching cubes, which creates strong artefacts (see a closeup Fig 19). 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.



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

Figure 19: 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.

Figure 20: 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)

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 21: 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

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. 21. 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. 15(c) where BPA looses thin details.

Limitations Scale space meshing 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. 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.

4.2 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.

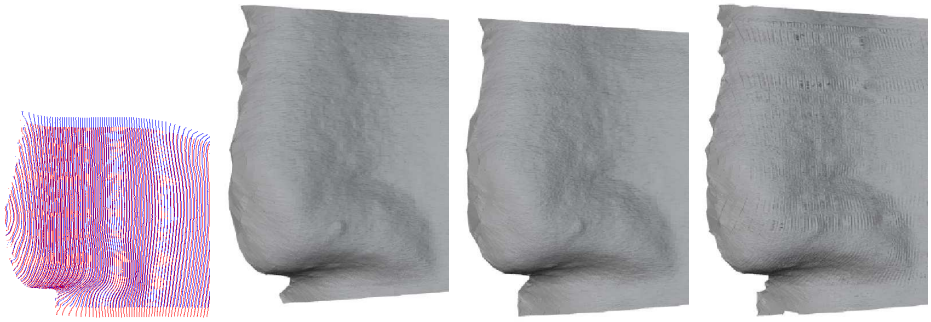
5 Application 3: Merging of Scans [DMAL10]

Triangulation laser scanner can deliver high precision scans of real objects. Yet, although each scan has a very high precision, this precision can be lost again when merging multiple scans and meshing them together. This loss of precision entails a loss of visible texture, which explains the smooth and glassy aspect of most rendered scanned objects. On the other hand the merging of the multiple scans (often called *super-resolution*) is absolutely necessary. A patch of the object may well be acquired tens and even hundreds of times on well exposed parts. Indeed, many sweeps with varying

trajectories are necessary to acquire the less exposed parts of the object. The main goal of the merging considered here is not to gain more detail and texture or to denoise the data point cloud by super-resolution: recent triangulation scanners yield scan sweeps with excellent quality. Unfortunately this quality is at risk of being damaged by the merging procedure itself. Thus, more trivially, the goal is to secure that the texture of each scan is not lost again due to slight matching errors which force a smoothing before a joint meshing. Fig. 22 illustrates the problem. With two overlapping shifted scan grids, as seen in (a), the aliasing risk is high. Meshing each scan separately yields two almost identical surfaces and textures (b, c). Nevertheless, a joint meshing (d) provokes strong tiling and aliasing effects, due to very small local offsets between both scans, in spite of the fact that they have been globally well registered. The challenge is therefore to merge both scans in such a way that the rendering quality does not decrease. The numerical problem is made more acute by two facts. First, not just two, but up to hundred scans may overlap in some region. Second, scans boundaries appear everywhere, as illustrated in fig. 23 and make the fusion near these boundaries still more problematic.

Each point of each scan has three-dimensional coordinates given either in a global coordinate system if the acquisition device is calibrated, or in a local coordinate system if the device is not calibrated. In the case of non-calibrated devices, the scans must be registered in a common coordinate system, and the registration problem becomes a rigid transform estimation. This problem has been widely investigated and has found efficient solutions [BM92] [RL01]. Yet if the scans had some internal local warping (which is usually the case), the rigid transform framework is not sufficient. A whole theory of non-rigid scan registration has therefore been developed [BR04],[BR07]. If the acquisition device is well calibrated the delivered scans are well registered, up to a given precision. Yet, as we already mentioned, a tiny residual mismatch can provoke strong artifacts similar to aliasing patterns (see fig. 22) and forbids a direct meshing of the union of all data point clouds. This problem is generally solved by applying a method which meshes an implicit zero level set of a distance function to the raw points. The distance function is approximated by its Fourier coefficients [Kaz05] or by radial basis functions [KBH06]. The problem is that these methods result in a serious loss of accuracy when the final result is compared to each scan separately.

Experiments are run on sets of scans of an object that have been either previously optimally registered by rigid or non-rigid methods, or registered through a high precision calibration of the acquisition tool. To demonstrate that no texture content will be lost, the goal is to mesh the entire point cloud. This means that all raw acquired points of all scans will ideally be vertices of the mesh. This requirement guarantees a complete preservation of all the acquired information, including noise and fine textures. Of course



(a) Overlapping scans (b) Mesh of scan 1 (c) Mesh of scan 2 (d) Mesh of scans 1&2

Figure 22: Example of two overlapping scans (a), points of each scan are first meshed ((b)-(c)) separately. The result can be compared to the meshing of points of both scans together (d)

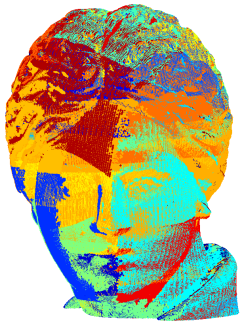


Figure 23: Example of overlapping scans. This head is such a complex structure that not less than 35 scans were acquired to fill in most holes.

such a mesh is not numerically economic, but it is necessary for two goals: first to get high quality rendering of complex shapes such as archeological objects, and second to precisely explore all remanent artifacts such as the holes, inherent in any scanning process. For scanning control purposes, it is anyway quite rewarding to be able to *see exactly* what has been scanned.

5.1 Scan Merging Principle

The general idea behind the scan merging method experimented here is to preserve point positions in non overlapping areas, and to make a fusion of the scans on overlapping regions while keeping all raw points. The fusion involves a smooth-base/height-function decomposition for each scan. The decomposition of a surface as the sum of a smooth base and of a height

function was proposed for a different purpose in [KST09], and [ZTS09], where the height function was used to segment the mesh and extract features as contours of the height function. The underlying idea is that a surface $\mathcal{S}$ can be decomposed into a smooth base B and a height function h, so that:

$$\mathcal{S} = B + h$$

B can be seen as the low frequency surface and h can be seen as the high frequency term. Given several surfaces $\mathcal{S}_1 = B_1 + h_1, S_2 = B_2 + h_2, \cdots, \mathcal{S}_N = B_N + h_N$, the idea is to fuse the bases, but to keep exactly the h_i terms, thus preserving all fine details. In other terms, a common basis B for all surfaces must be found, the high frequencies of all scans adopting this common basis thereafter. This strategy is comparable to the one used for morphing applications in [PKG06]. In a way, the idea is similar to [SCOT03], where the high frequency error due to quantization was transformed into a low frequency error much less noticable.

The data merging using a high/low frequency decomposition has long been a classic method in image processing [BA83]. This article introduced the idea of separating each image into various frequency bands by a Gaussian pyramid. The low frequency bands were merged separately to obtain a smooth blending of different images. The method has been successfully used to create panoramas from multiple images [BL07] and texture 3D models [Bau02]. Two major differences are that in [BA83] all frequency bands are merged, whereas the method described here only merges the low frequencies while keeping the high frequencies intact. Another important difference is the usage of a nonlinear heat equation instead of a linear frequency decomposition.

The next section addresses the robust decomposition of a surface into a base and a height.

Low/High frequency surface decomposition Since the pioneering article [Tau95] it is known that mesh high frequencies are removed by the application of the intrinsic heat equation $\frac{\partial P}{\partial t} = \Delta P$. Yet, our scanned surfaces are given as point clouds and not as meshes. A numerical scheme of the heat equation for raw point clouds must be used. This question has been addressed in [BSW09] and [PKG06]. The scale space introduces in section 2 will be used here.

Consider the projection operator T_r that projects each point p onto the regression plane of the neighbors of p enclosed in a ball of radius r. Then it can be proven that this motion is tangent to the intrinsic heat equation. The iteration of T_r yields a scale space (a representation of the shape at various smoothing scales). In all experiments r is set so that the ball $\mathcal{B}_r$ centered at P contains about 30 neighbors at almost all points, and the number of scale space iterations n is set to 4. The first parameter (30) is fixed so

Figure 24: The unmerged head with aliasing artifacts (left), its smooth base (middle) and the merged result (right)

that a reliable regression plane is always computed. The second parameter, namely the number of iterations 4, is chosen to guarantee a smooth enough basis in all cases. It can be increased without damage. When iterating the projection operator with an initial surface $\mathcal{S}_0$, the surface $\mathcal{S}_t$ is iteratively smoothed. To each point P_t of $\mathcal{S}_t$ corresponds a point P_0 of $\mathcal{S}_0$, and the height function can be taken to be the vector $h(P_t) = P_t - P_0$.

An alternative definition for the height would be the scalar function $h(P_t) = (P_t - P_0) \cdot \vec{n}$, where $\vec{n}$ is the normal to $\mathcal{S}_t$ at P_t. Yet, the results with both height variants being fairly identical, the simplest definition was kept: it separates each data point into a smooth base point and a high frequency vector.

Finding a common smooth basis for all surfaces Choosing a common basis for all scans is the next question. A natural constraint on the method is to keep fixed the points belonging to regions where only one scan is available. Finding the common basis then becomes straightforward: It is enough to apply the same number n of iterations of T_r with the same parameter r to all the sets after they have been put together. This global filtering assumes that the high frequency term of the set $\mathcal{S} = \cup_i \mathcal{S}_i$ contains the registration error: when filtering $\mathcal{S}$ the registration error is filtered away (see fig. 24).

Algorithm The method is summarized in Algorithm 4. The algorithm is based on two applications of the intrinsic heat equation scheme (here the iterated projection on the regression plane) with the same parameters and the same number of iterations. All registered scans are given in the same global coordinate system. The first application (Line 2) is done on the separate scans yielding the intrinsic high frequencies of each scan. The

second application (Line 6) is done on all scans together. When filtering all scans together (lines 5 and 6) the registration error is suppressed and we get a common low scale registration or basis, the set of points $b(P)$. Adding back to them the high frequency component $\overrightarrow{b_i(P)P}$ restores all details from all scans.

Algorithme 4: Scale Space Merging

Data : N point sets (scans) $(\mathcal{S}_i)_{i=1\cdots N}$, a number of projection filter iterations n and a radius r

Result : The set of merged scans: $\mathcal{Q}$

1 **for** $i = 1 \cdots N$ **do**

2 Apply n steps of the projection filter T_r to the set $\mathcal{S}_i$;

3 Store for each point $P \in \mathcal{S}_i$ with corresponding filtered point $b_i(P)$ the high frequency vector $\vec{\delta}(P) = \overrightarrow{b_i(P)P}$;

4 $\mathcal{S} \leftarrow \cup_{i=1}^{N} \mathcal{S}_i$;

5 Apply for each $P \in \mathcal{S}$ n steps of the projection filter T_r, yielding a point $b(P)$;

6 **for** $P \in \mathcal{S}$ **do**

7 $Q = b(P) + \vec{\delta}(P)$;

8 Add Q to $\mathcal{Q}$;

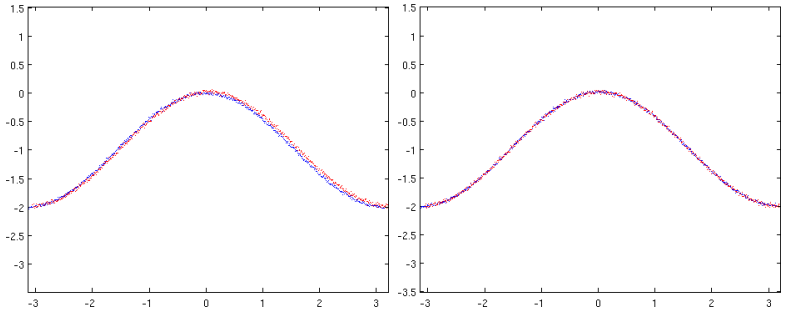
9 Return $\mathcal{Q}$;

An important feature of the method is that each region $\mathcal{A}$ of the shape that has been acquired by one scan only is not altered. Indeed, inside such a region, applying the separate scale space or the common scale space is strictly equivalent, since there is only one scan in the neighborhood of the points of $\mathcal{A}$. Then the point is first filtered to $b_i(P) = b(P)$, and therefore moved back to its original position P at Line 8. So in areas with only one scan, point positions are not changed. The only effect of the algorithm is the merging of overlapping scans.

One-dimensional study It is easy to illustrate the method in 1-D on simple $1D$ shapes. Our goal was to check that the proposed method superimposes two simulated scans without any smoothing effect. To do so, two noisy straight lines A and B were synthesized from the same model and then merged by the algorithm. The noise of each set A, B, $A \cup B$ was estimated as the root mean square error to their regression lines before and after merging. The results in Tab. 25 show that the merging did not cause any denoising. Indeed, the RMSE does not decrease by the merging procedure. Figs 26 and 27 show other 1D examples of the merging procedure where the bases are actually slightly different, in accordance with the real situation encountered on real scans.

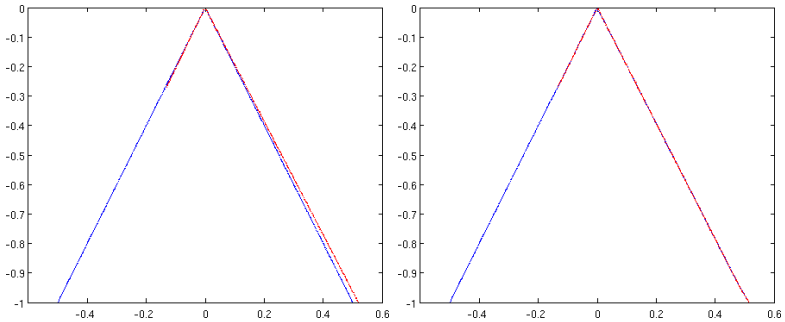
RMSE	Both lines	Line A	Line B
Before Merging	X	$9.95e-04$	$9.76e-04$
After Merging	$9.85e-04$	$9.94e-04$	$9.75e-04$

Figure 25: Noise estimates on each separate scan A and B before and after their merging.



(a) Without Merging (b) With Merging

Figure 26: Two noisy sine functions before and after merging.



(a) Without Merging (b) With Merging

Figure 27: Two noisy edges before and after merging.

5.2 Implementation and Results

The inputs of the merging algorithm are the outputs of our laser triangulation scanner. This device being accurately calibrated, the scans are in principle already registered so that no extra software registration is needed. Nevertheless, the ICP algorithm is applied to see if it can remove the aliasing and tiling artifacts. In this case ICP failes: the positions computed by ICP oscillate around the input scan positions, and the resulting meshes are no better. The registration process was implemented on a 1.5 Ghz processor with $48GB$ RAM. An octree structure was first built to allow for fast access to the neighbors of each given point. Table 28 gives the computation times for various shapes of various sizes with varying numbers of scans. Notice the high number of scans necessary to get a good covering of the object. It entails that several dozens of scans have to be merged on the more exposed parts.

Point set	**points**	**scans**	**Time(s)**	**height**
Dancer	$5,524,627$	94	321	$17cm$
Mime	$8,611,522$	102	140	$11cm$
Greek Mask	$8,961,736$	78	106	$12cm$
Nefertiti	$15,554,528$	115	819	$18cm$
Tanagra	$17,496,999$	160	1258	$22cm$
Rosetta	$36,201,537$	32	$45min$	$30cm$

Figure 28: Computation time for the proposed merging. It is significantly faster than the scanning time itself

Figs 29, 30 and 31 present the results on these data. For all point sets, two different renderings are displayed: the first one is a ball pivoting [BMR*99] meshing of all raw scan points without any merging. The scans were preregistered by the calibrated acquisition device and no software post-registration was needed. The second rendering is again a ball-pivoting meshing, but applied to the merged point set. The rendering was made using the POV-RAY ray-tracer. The conclusion is common to all experiments: even if the scans are actually very accurately registered, the tiny warps of the grids always create some aliasing visible as grid or tiling effects. After the merging procedure (which only slightly affects the low frequencies), these undesirable effects disappear almost completely. In the procedure more than 99.9% of the raw points were kept. Thus, the final result indeed is highly faithful to the raw scan. Yet a careful attention shows some remains of aliasing (Fig. 30, last column). The area of these is actually small, being inferior to the area of the holes. They could easily be removed by a selective local smoothing. Some of the bigger pieces, like Nefertiti, show no defect at all.

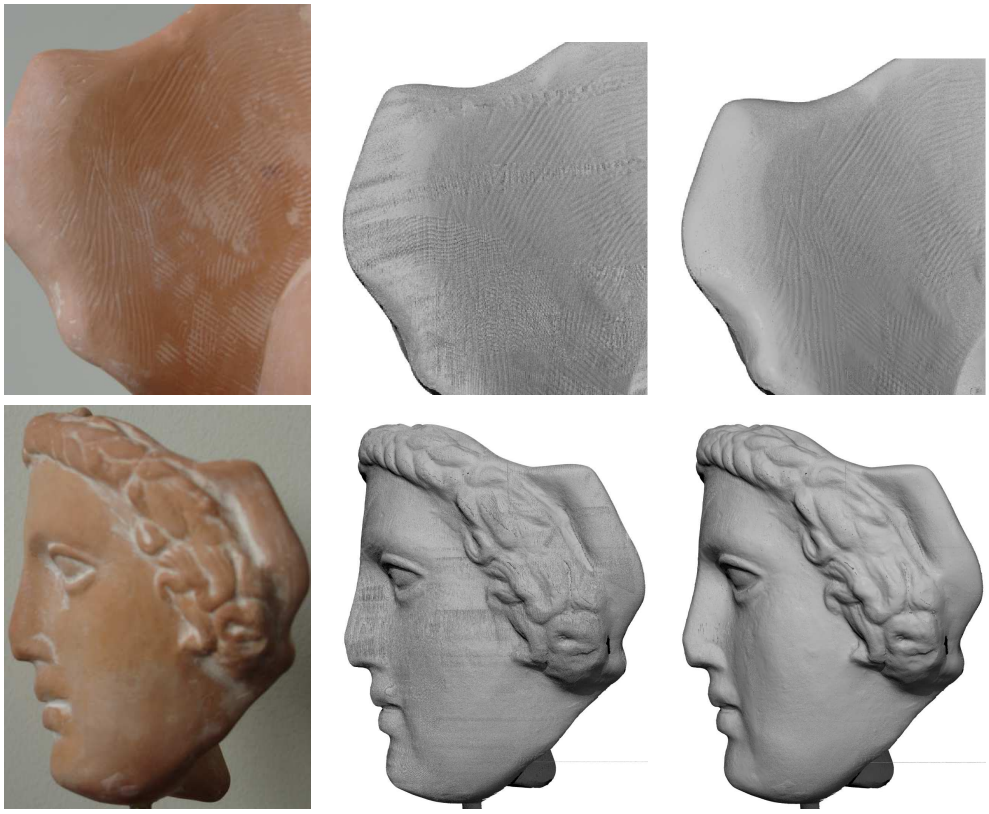


Figure 29: Merging of the mask scans seen from the back side (top row) and left side (bottom row). Left: picture, middle: without merging, right: with merging

Figure 30: Merging of the Dancer With Crotales. From left to right: picture, without merging, with merging, an example of merging failure taken from the back of the object (top: unmerged, bottom merged)

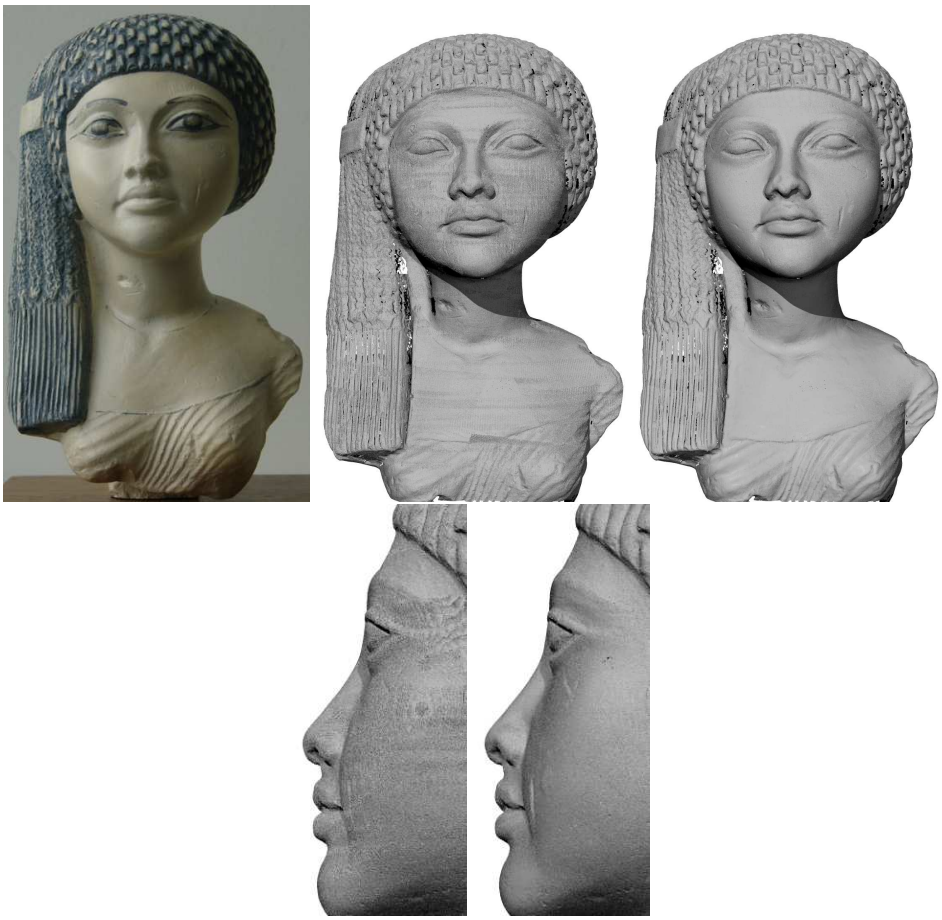


Figure 31: Merging of the Nefertiti (1st: picture, 2nd,4th: without merging, 3rd,5th: with merging)

Figure 32: Comparison of the rendering of a single scan (ground truth) and the merging of all scans that overlap in the same region (Left: ground truth, middle: joint mesh of all scans without merging, right: joint mesh with merging).

Comparison To better judge the texture preservation, the rendering of a scan alone (ground truth) was computed and compared to the rendering of all scans in the same region on Fig 32. This shows that the visual information loss after scale space merging is very low compared to the one due to a simple joint meshing.

It is crucial to compare the raw merging method results with results obtained by the level set reconstruction method of the unmerged scans point set. The result of the level set method applied to the Tanagra head (fig. 33 b), obviously introduces an important smoothing and loses texture in comparison to the merging result (fig. 33, a). But even with that smoothing the result still keeps several artifact lines due to the scan offsets: these offsets become visible at the scans boundaries. See the nearly straight long lines on the surface, mostly vertical and horizontal. It can also be asked if an efficient denoising method could actually restore the raw set. Fig. 33-c, shows the result of the application of the bilateral filter [FDCO03] to the union of the scans. This iterated filtering was applied up to the point where aliasing artifacts were no more visible. Clearly, this entails a much too strong smoothing of detail and texture.

The scan merging is a very local method which is therefore computationally efficient (see Tab. 28). Yet, if the input data are not already well registered the merging could obviously fail. The method corrects the slight misalignments only in the normal direction. A tangential drift in the original registration could therefore cause a loss of sharpness or a loss of small details. Nevertheless, this degradation seems to pass unnoticed. Indeed, for the type of data used in these experiments, the registration error is very small. For a point cloud with side-length $99mm$ the observed average point offset after merging was $0.081mm$, with standard deviation 0.012. The tangential offset could not be measured. The explanation of the relative visual success of the method is that even a tiny normal offset causes a dramatic change in triangles orientation, and therefore completely jeopardizes the vi-

(a) Merged Result (b) Poisson Reconstruction (c) Bilateral filter

Figure 33: Comparisons of the merging (a) with a level set reconstruction method ([KBH06]) of the unmerged scans point set (b) and a filtering of the unmerged scans point set (c). The level set method obviously introduces a serious smoothing, yet does not eliminate the scanning boundary lines. The bilateral filter, applied until all aliasing artifacts have been eliminated, over-smoothes some parts of the shape.

sual quality of the triangulation. An equally small tangential offset seems to be visually undetectable. Thus, the merging method corrects the normal error, and makes the tangential error unnoticeable.

The proposed merging can be seen as a local non rigid registration. Therefore it can be compared to the result given by state of the art non rigid registration methods [BR07]. To perform the comparison, the problem arose that the scans did not systematically contain strongly identified features. Most scans of the mask point set were simply rejected by the non rigid registration method described in [BR07]. In order to perform a serious comparison anyway, two sweeps of the fragment 31u of the Stanford FUR project were used. The computation times were, however, considerably different: it took more than $2h30$ to register non rigidly these meshes. On the same computer, using only the raw points and not the meshes, the merging took only $84s$. The final meshes were built using Poisson Reconstruction [KBH06] in both cases. The registration artifacts (two horizontal lines limiting the overlap area, fig 34) are much less visible with the scan merging than with the non rigid registration.

Conclusion

This course showed that the scale space approach yields a generalized way of processing various problems linked to the processing of point clouds. This vision approach has many other possibilities: registering point sets in a SIFT-like ways. It is a sound way of extending traditional point clouds

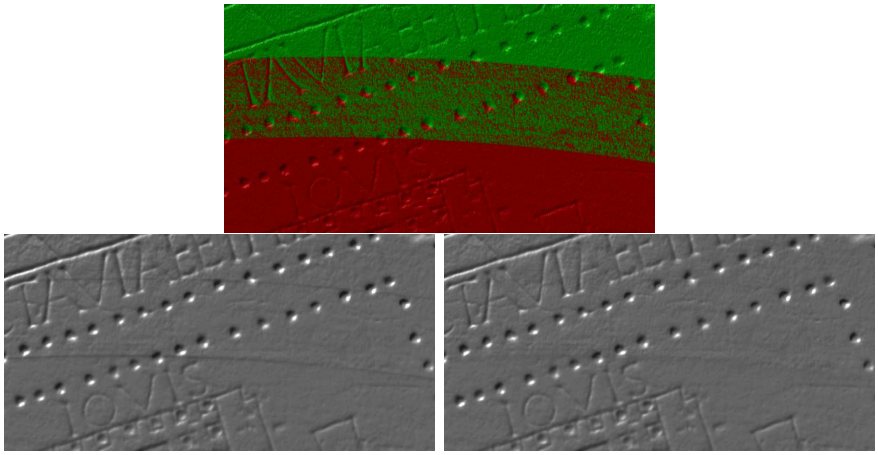
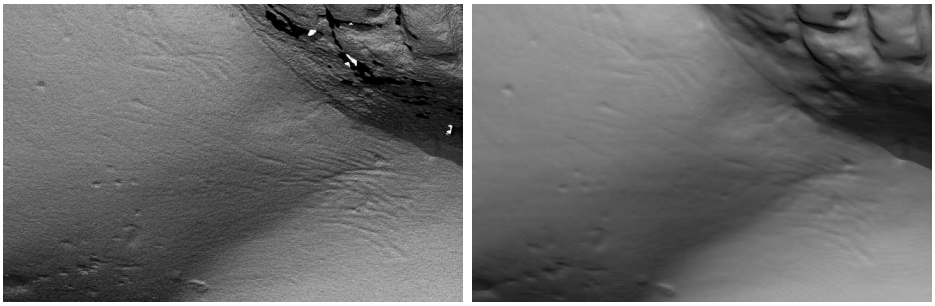


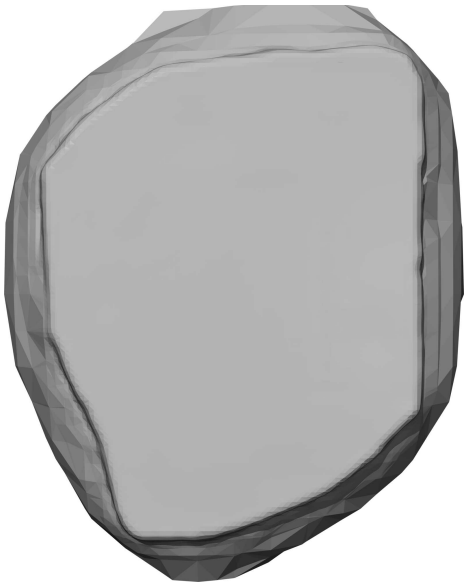
Figure 34: Comparison of registration of two scans (colored in different colors on the first figure) using Global Non Rigid Alignment [BR07] and scale space merging. Meshes were reconstructed using [KBH06].



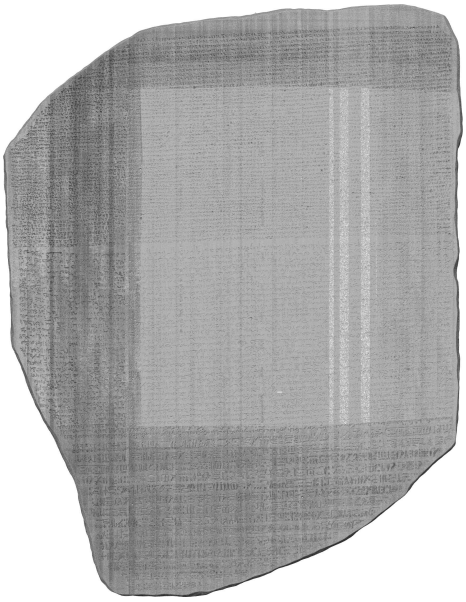
(a) Result of the merging (b) Result of the Poisson reconstruction

Figure 35: Comparison of the mesh obtained by merging and by Poisson reconstruction on a detail of nefertiti's cheek. In this case, Poisson Reconstruction suppresses registration artefacts but smoothes out the details.

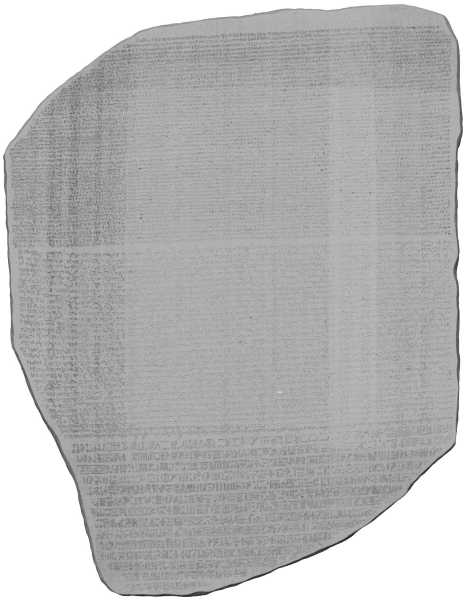
(a) Picture of the object



(b) Poisson Reconstruction of the input sets



(c) Without merging



(d) With merging

Figure 36: Comparison of rosetta meshes. A characteristic of this object is that the engravings in this object are very shallow (around 50μ), which is why Poisson fails. The artefacts in fig 36(c) are due to the 3D aliasing (this is fixed by scale space merging) but also to the resolution variation. Indeed the borders of the object were acquired using multiple orientations while the middle was acquired using only one orientation. This is why we have such a precision difference that is not fixed by the algorithm.

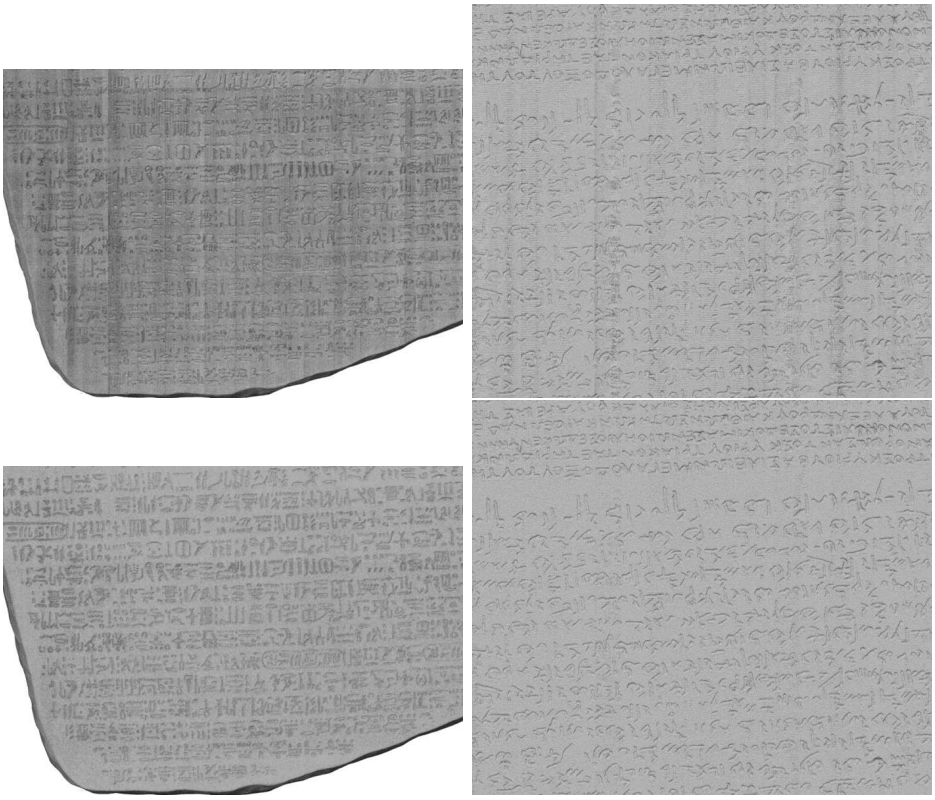


Figure 37: Details of the Rosetta object without merging (top row) and with fusion (bottom row)

processing methods to point sets.

6 A short summary of the algorithmic problems for processing 3D point clouds

All along this work an efficient fixed-radius neighbor finding algorithm was needed. Indeed this is one of the trickiest problems in surface processing: given a point P find the coordinates of all samples lying within a given distance of P. Because of sampling irregularities it may happen that this neighborhood contains either no other point than P or the whole set.

The very naive implementation would need to traverse once the whole set of N points for each of the N points, yielding an algorithm with N^2 complexity. In case of point sets with more than a million points it yields prohibitive computation time. This is why, tree structures are usually used to partition the space into cells. In a nutshell, this partition permits to avoid traversing cells that are too far away from the query point ([Sam90]).

Here an octree was used: it simply divides the bounding box of side L into eight cells of size $L/2$. The splitting point is of course the center of the parent cell. All points are included in *leaf cells*, i.e. cells with no child that have depth d, where d is the octree depth specified by the user.

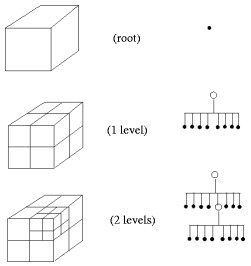


Figure 38: An octree

Given a set of points and a depth d, the octree building proceeds as follows:

- An initial root node is built so that it contains all points;
- When a point P is added to a cell C:
  - If the cell has depth d it is a leaf and the point is simply added to the list of points of C;
  - Otherwise, look for the cell child C_i that would contain the point;
  - If C_i does not exist create it and add the point to it.

This way, only nodes *actually* containing points are created.

6.1 Iterating over the octree

Iterating over the octree means from a given point position find a set of neighbors. In this work *radius fixed neighborhoods* are used meaning that given a point P and a point set $\mathcal{M}_S$, the set of P's neighbors is the subset of $\mathcal{M}_S$ such that:

$$\{\mathbf{m} \in \mathcal{M}_S | \|P - \mathbf{m}\|^2 < r\}$$

.

The iterating method is basically the one from [FP02]. Given a point P, a radius r and a set $\mathcal{M}_S$ it proceeds as follows:

- In case it is not given, the leaf cell C containing P is found.
- Going upwards in the tree, it finds the cell C_0 containing C and whose edge length L is such that $\frac{L}{4} < r \leq \frac{L}{2}$
- Then the set of neighbors is included in C_0 and C_0's neighboring cells at the same depth (there is a most 8 cells to consider then).
- For each of those cells:
  - Look at all non-empty children nodes
  - If the hypercube maximum distance to P is below r then all points included in its descendants are added to the set of neighbors
  - If the hypercube maximum distance to P is above r and the minimum distance is below r then, if the node is not a leaf, the node's children are explored recursively. If it is a leaf then all the point distances are computed and compared to r.
  - If the hypercube minimum distance to P is above r the cell is not explored any further.

The whole neighboring cell search is made faster using Location Codes (see [FP02] for details).

6.2 A single octree to deal with an evolving point set

Since the pointset evolves with filtering iterations, an evolving structure has to be built. This is done by simply considering that each cell contains more than 1 set of points. Since the initial pointset must be remembered, it is necessary to store 3 sets per leaf: the set of initial points, the set of points filtered at iteration N and the set of points filtered at step $N + 1$. Then when performing neighbor-search, the appropriate set should be chosen.

- When filtering set 0, the results are stored in set 1

- When filtering set 1, neighbor-search is done in set 1, the results are stored in set 2 and set 1 is emptied
- When filtering set 2, neighbor-search is done in set 2, the results are stored in set 1 and set 2 is emptied

Each point links to the point of set 0 from which it originated.

6.3 Different tools used for processing 3D point clouds

All the algorithms presented in this course were implemented using C++ and the Standard Template Library (STL). It uses the Template Numerical Toolkit and JAMA-C++ libraries[1] for Principal Component Analysis and system solving.

Reading/Writing images for color cloud filtering for example was done using the CImg library [2].

Reading and writing mesh in Stanford PLY format used rply [3].

For perenity reasons, files are always written in ascii PLY and not binary PLY, in order to be able to access the information on any computer in the next years, at the cost of bigger files.

Renderings have been computed using the POV-RAY program[4]. No texture was used (the triangles were colored in white) and a simulated diffuse light was set.

MVA projects

Bilateral filtering of point sets [FDCO03]

This paper presents a generalization of the image filter to meshes. The bilateral filter aims at denoising a surface without removing the sharp features. This bilateral fiter takes as input a meshed surface and outputs a denoised surface with preserved sharp edges. Though it is designed to work on meshes, the adaptation to point clouds is straightforward using either the k-nearest neighbors or a ball neighborhood. An extension of this project would be to compare the results of the filter using the two kinds of neighborhoods.

Resampling of point sets: Parameterization-free projection for geometry reconstruction [LCOLTE07]

In [LCOLTE07], an iterative algorithm that projects an arbitrary point set onto an input point set P is presented. The set of pojected points minimizes

[1] http://math.nist.gov/tnt/index.html
[2] http://cimg.sourceforge.net/
[3] http://w3.impa.br/ diego/software/rply/
[4] Persistence of Vision Raytracer http://www.povray.org

the sum of weighted distances to the points of P. No local normal information is needed and the surface approximation order is in $O(h^2)$ where h is the support size for the weighting function. Applications include resampling: it is observed that by projection a point set with fewer points than P on P a density regularization occurs.

The goal of this project is to implement and test intensively the point set resampling and the behavior of the resampling with respect to sharp features and details.

References

[BA83] Burt P. J., Adelson E. H.: A multiresolution spline with application to image mosaics. *ACM Trans. Graph. 2*, 4 (1983), 217–236.

[Bau02] Baumberg A.: Blending images for texturing 3d models. In *Proc. Conf. on British Machine Vision Association* (2002), pp. 404–413.

[BL07] Brown M., Lowe D. G.: Automatic panoramic image stitching using invariant features. *Int. J. Comput. Vision 74*, 1 (2007), 59–73.

[BM92] Besl P. J., McKay N. D.: A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell. 14*, 2 (1992), 239–256.

[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.

[BR04] Brown B., Rusinkiewicz S.: Non-rigid range-scan alignment using thin-plate splines. In *3DPVT'04* (2004). printed.

[BR07] Brown B., Rusinkiewicz S.: Global non-rigid alignment of 3-D scans. *ACM Transactions on Graphics (Proc. SIGGRAPH) 26*, 3 (Aug. 2007).

[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.

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

[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.

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

[DAL*11] Digne J., Audfray N., Lartigue C., Mehdi-Souzani C., Morel J.-M.: Farman Institute 3D Point Sets - High Precision 3D Data Sets. *Image Processing On Line* (2011).

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

[FDCO03] Fleishman S., Drori I., Cohen-Or D.: Bilateral mesh denoising. *ACM Trans. Graph. 22*, 3 (2003), 950–953.

[FP02] Frisken S. F., Perry R.: Simple and efficient traversal methods for quadtrees and octrees. *Journal of graphics tools 7(3)*, 3 (2002), 1–11.

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

[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.

[Kaz05] Kazhdan M.: Reconstruction of solid models from oriented point sets. In *SGP '05* (Switzerland, 2005), Eurographics Association, p. 73.

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

[KST09] Kolomenkin M., Shimshoni I., Tal A.: On edge detection on surfaces. In *CVPR* (2009), pp. 2767–2774.

[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.

[LCOLTE07] Lipman Y., Cohen-Or D., Levin D., Tal-Ezer H.: Parameterization-free projection for geometry reconstruction. *ACM Trans. Graph. 26* (2007).

[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).

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

[RL01] Rusinkiewicz S., Levoy M.: Efficient variants of the icp algorithm. In *Proc. 3DIM 2001* (2001), pp. 145–152.

[Sam90] Samet H.: *The design and analysis of spatial data structures.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.

[SCOT03] Sorkine O., Cohen-Or D., Toledo S.: High-pass quantization for mesh encoding. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (Aire-la-Ville, Switzerland, 2003), SGP '03, Eurographics Association, pp. 42–51.

[Tau95] Taubin G.: A signal processing approach to fair surface design. In *SIGGRAPH '95* (USA, 1995), ACM Press, pp. 351–358.

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

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

[ZTS09] Zatzarinni R., Tal A., Shamir A.: Relief analysis and extraction. *ACM Trans. Graph. 28*, 5 (2009), 1–9.