Normals and Curvature Estimation for Digital Surfaces Based on Convolutions

Sébastien Fourey[1] and Rémy Malgouyres[2]

[1] GREYC, UMR6072 – ENSICAEN, 6 bd maréchal Juin 14050 Caen CEDEX, France
Sebastien.Fourey@greyc.ensicaen.fr
[2] LAIC, EA2146 – Université Clermont 1, BP 86, Aubière CEDEX France
Remy.Malgouyres@laic.u-clermont1.fr

Abstract. In this paper, we present a method that we call *on-surface convolution* which extends the classical notion of a 2D digital filter to the case of digital surfaces (following the *cuberille* model). We also define an averaging mask with local support which, when applied with the iterated convolution operator, behaves like an averaging with large support. The interesting property of the latter averaging is the way the resulting weights are distributed: they tend to decrease following a "continuous" geodesic distance within the surface. We eventually use the iterated averaging followed by convolutions with differentiation masks to estimate partial derivatives and then normal vectors over a surface. We provide an heuristics based on [14] for an optimal mask size and show results.

1 Introduction

Estimation of geometrical properties and quantities of objects known through their digitizations is an important goal of discrete geometry. One of the classical problems is simply to measure the length of a curve (or a perimeter) in the digital plane [6,4]. One may also quote the estimation of tangents or normals to a curve [11], normal vectors over a surface [12], or area of a digital surface [3,9,20].

In 2D, a whole set of methods rely on the *digital straight segments* recognition algorithm [5] used to find maximal line segments in a curve, which may in turn be used to estimated the curve's length or its tangent vectors [11]. These methods have been extended to the 3D case with digital plane recognition [17]. Directional tangent estimation based on segments recognition was used in [19] to compute normal vectors on a digital surface and in [10] for the nD case. All of these methods are sensitive to noise.

In the case of digital surfaces, another method was introduced by Papier and Françon ([16,15]) to estimate the normal vector field. It is based on a weighted averaging of the canonical normals in a neighborhood of each surfel. Their method generalizes to large neighborhoods the approach proposed by Chen *et al.* in [2] and is very close to the one we propose here, although it differs for at least two points: First, the size of the neighborhood taken into account, called the "order of the umbrellas" in their paper, is a parameter of the method and no hints to find an optimal size is provided. Here, we rely on a new theoretical result in the 1D functional case to set this parameter. Tests on spheres and tori show that the chosen size provides experimental *convergence* (see Section 4.1). Second, umbrellas in Papier's method grow following a breadth-first traversal of the surfels v-adjacency graph, whereas our method may be seen as the result of an averaging process using masks which grow in a more *geodesic* and isotropic way (see Section 4.2).

The normal estimation method introduced here is based on the notion of *on-surface convolution* (Section 3) which extends to digital surfaces the classical 2D filters used in image processing. Using an averaging mask defined locally, we apply an iterated convolution operation on the centers of the surfels. Then, we use two orthogonal differentiation operators on the resulting centers to estimate partial derivatives, and by a cross product we obtain normal vectors. As previously mentioned, we follow the criterion given in [14] to set the optimal number of iterations depending on the digitization step (voxel size).

In the last section, we consider the problem of curvature estimation, which requires second order derivatives estimation, and show an encouraging experiment.

2 Digital Objects and Discrete Surfaces

2.1 Digital Objects

In this paper, we simply call a *digital object* a subset of $\mathbb{Z}^3$, the classical three dimensional grid. Such an object is seen as a set of unit cubes called *object voxels* centered on points with integer coordinates. *Background voxels* are voxels that do not belong to the object.

2.2 Digital Surface

A digital object can be visualized slice per slice, i.e., as a series of 2D binary images. More often, what is rendered is a 2D view of its external surface (see [1,2]). In this case, the basic rendering primitives are the *surfels.* Surfels are unit squares that are shared by two 6-adjacent voxels: one belonging to the object and one belonging to the background. There are exactly six types of surfels according to the direction of their normal vectors. Thus, a surfel can be uniquely defined by the data of its center's coordinates and its orientation. In the sequel, a surfel is a pair $(p, \boldsymbol{n})$ where $p \in \mathbb{R}^3$ (the center) and $\boldsymbol{n} \in \{(\pm 1, 0, 0), (0, \pm 1, 0), (0, 0, \pm 1)\}$ (the normal vector). Eventually, a *digital surface* is a set of surfels which is the set of all the surfels of a digital object.

We will use in the sequel the two functions σ and ν which associate to a surfel $s = (p, \boldsymbol{n})$, respectively, its centre $\sigma(s) = p$ and its normal vector $\nu(s) = \boldsymbol{n}$.

2.3 Adjacency Relations between Surfels an Surfels Neighborhoods

We can define two adjacency relations between surfels: the e-adjacency and the v-adjacency relations.

As depicted in Figure 1, a surfel x might share one of its four edges with at most three other surfels. In the cases depicted in Figure 1(a)(b)(c), only one surfel y shares the bold edge with x. We say that this surfel y is *e-adjacent* to x (y is also called an *e-neighbor* of x). In the case of Figure 1(d), we choose[1] to define the surfel y as the e-neighbor of x associated with the bold edge. Thus, we have defined with examples the e-adjacency relation ("e" for *edge*) so that a surfel has exactly four e-neighbors: one per edge.

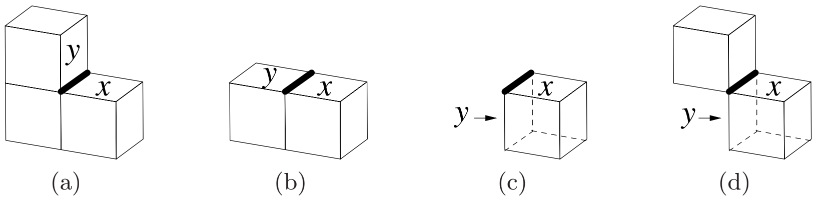


Fig. 1. The e-neighbor y of a surfel x given one of its edges (bold line)

Next, we define a *loop* in a digital surface Σ as an e-connected component of the set of the surfels of Σ which share a given vertex w. For example, if Σ is the surface of the object depicted in Figure 2(a) (which is made of three voxels), then the vertex w defines two loops: one that contains the six gray surfels, and another one in the back with three surfels. Eventually, we say that two surfels are *v-adjacent* (v for *vertex*) if they belong to a common loop of Σ.

These two adjacency relations allow us to define the *e-neighborhood* (resp. *v-neighborhood*) of a surfel x denoted by $N_e(x)$ (resp. $N_v(x)$) as the set of surfels that are v-adjacent (resp. e-adjacent) to x. Figure 2(b) shows an example of a v-neighborhood.

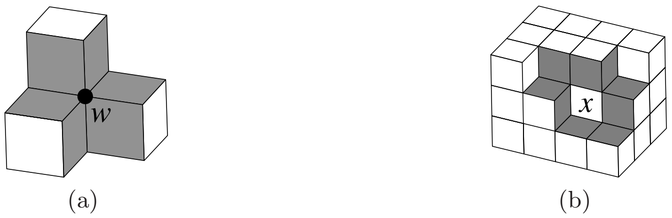


Fig. 2. (a) A loop of surfels (in gray), (b) A v-neighborhood

3 On-Surface Convolution

The work presented in the next sections illustrates the use of *on-surface convolution*, which we introduce here. In the sequel of the paper, Σ is a digital surface and S is a vector space over $\mathbb{R}$. We define the space of *digital surface filters over* Σ as the set of functions from $\Sigma \times \Sigma$ to $\mathbb{R}$.

[1] By this choice, the interior of a surface component is 6-connected [7].

Definition 1 (Convolution operator). *For $f : \Sigma \mapsto S$ and $F : \Sigma \times \Sigma \mapsto \mathbb{R}$, we define the operator Ψ as follows:*

$$\begin{aligned} \Psi_{f,F} : \Sigma &\longrightarrow S \\ x &\longmapsto \sum_{y \in \Sigma} F(x,y) \cdot f(y) \end{aligned} \tag{1}$$

Intuitively, Ψ acts like a convolution of the values of f on the surface with a convolution kernel whose values should depend on the relative positions of two surfels. We also define the iterated operator $\Psi^{(n)}$.

Definition 2 (Iterated convolution operator). *The* iterated convolution operator *is defined for $n \in \mathbb{Z}$ by:*

$$\begin{cases} \Psi^{(0)}_{f,F} = f \\ \Psi^{(n)}_{f,F} = \Psi_{\Psi^{(n-1)}_{f,F},F} \quad \textit{if } n > 0. \end{cases} \tag{2}$$

Next, we define an averaging and two derivative filters which we will use in Section 4.1 to estimate the normal field on a digital surface.

3.1 The Averaging Filter

We define here a local averaging mask $W_{\text{avg}} : \Sigma \times \Sigma \mapsto \mathbb{R}$. This mask should be seen as a wrapping of the 2D classical mask (Figure 3(a)) which follows the *local* shape of the digital surface. We define this wrapping in such a way that the weights remain balanced, the way they are in the 2D pattern, but considering the local orientation within a v-neighborhood.

Let x and y be two surfels of Σ such that $y \in N_v(x)$. If y is e-adjacent to x, then y has exactly two e-neighbors in $N_v(x)$, say s and t. We define $\delta_x(y)$ as the number of surfels in $\{s,t\}$ which are e-adjacent to x. If y is v-adjacent but not e-adjacent to x then there is a single loop L of Σ that contains both x and y. In this case, we define $\delta_x(y) = \text{card}(L) - 3$.

The number $\delta_x(y)$ is used to take into account the number of surfels in a loop containing x which are not e-adjacent or equal to x. Within a loop, all these surfels will end up with a total contribution of $\frac{1}{16}$. If there are no such surfels, the weight $\frac{1}{16}$ is split among the two n-neighbors of x in the loop.

Now, let x be a surfel of Σ. For any surfel $y \in \Sigma$ we define the *weight* $W_{\text{avg}}(x,y)$ as follows:

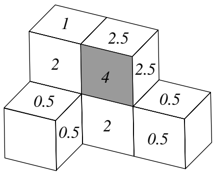
$$W_{\text{avg}}(x,y) = \begin{cases} \frac{1}{4} & \text{if } y = x, \\ \frac{1}{8} + \frac{\delta_x(y)}{32} & \text{if } y \in N_e(x), \\ \frac{1}{16 \cdot \delta_x(y)} & \text{if } y \in N_v(x) \setminus N_e(x), \\ 0 & \text{if } y \notin N_v(x). \end{cases} \tag{3}$$

One can check that for all $x \in \Sigma$, we have $\displaystyle\sum_{y \in \Sigma} W_{\text{avg}}(x,y) = 1$.

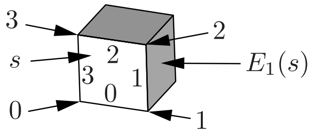
See Figure 3(b) for an example of a v-neighborhood and the associated values of W_{avg}.

1	2	1
2	4	2
1	2	1

(a) A 2D mask.



(b) A surfel x (in gray) and the values of $16 \cdot W_{\text{avg}}(x,y)$ for the surfels y of $N_v(x) \cup \{x\}$.



(c) Ordering of vertices and edges for a given surfel s.

Fig. 3. Illustrations of the masks definition

3.2 The First Order Derivative Filters

We introduce here two directional derivative masks which may be used with the convolution operator to obtain two orthogonal differentiation operators.

For each surfel s of Σ we define a numbering of the surfel vertices and edges as illustrated by Figure 3(c), following the coherent orientation around the outward normal. We denote by $E_i(s)$ the e-neighbor of s that shares with s its i^{th} edge. Then, we define the derivative masks $D_{\text{u}}(x,y)$ and $D_{\text{v}}(x,y)$ for $x, y \in \Sigma$ as follows:

$$D_{\text{u}}(x,y) = \begin{cases} \frac{1}{2} & \text{if } y = N_0(x), \\ -\frac{1}{2} & \text{if } y = N_2(x), \\ 0 & \text{otherwise.} \end{cases} \qquad D_{\text{v}}(x,y) = \begin{cases} \frac{1}{2} & \text{if } y = N_1(x), \\ -\frac{1}{2} & \text{if } y = N_3(x), \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Given the derivative masks D_{u} and D_{v}, we may define two derivative operators $\Psi_{f,D_{\text{u}}}$ and $\Psi_{f,D_{\text{v}}}$ which act on a function f defined on Σ.

4 Normal Estimation

In this section, we address the problem of estimating the normal vectors on the surface of a digital object. This estimation is achieved using iterated convolutions of the surfel centers with the averaging mask W_{avg}, followed by a step of differentiation using the derivative filters D_{u} and D_{v} defined in Section 3.2.

4.1 Surface Normal Estimation

For different purposes, such as shading methods for visualization or for area estimation, it is interesting to compute normal vectors on the surface of a digital object. In so doing we expect that the computed normals will be as closed as possible to the normals on the surface of a continuous object from which the digital object could have been obtained by a digitization process. In other words, given a continuous object, we expect that the estimated normals can get as close as required to the real normals as soon as the digitization step is chosen sufficiently small. Roughly speaking, this property is what is called *multigrid convergence* in the literature. Several surface normal estimation methods have been proposed, among which we may cite [13,19,10] is past DGCI's, but no method has been proved to be convergent in that sense.

Here, we show that the normal vectors of a digital surface can also be estimated by first averaging the positions of the surfel centers using the iterated convolution operator, then computing approximations of two partial derivatives to obtain vectors in the tangent space to the surface, and finally computing a normal vector by a simple cross product of the tangent vectors.

More formally, using the iterated convolution operator Ψ, the averaging mask W_{avg}, and the derivative masks D_{u} and D_{v} we define a function $\Gamma^{(n)} : \Sigma \longrightarrow \mathbb{R}^3$ for $n \in \mathbb{Z}$ such that $\Gamma^{(n)}(s)$ is the *estimated normal vector* of Σ at the center of s (after n on-surface convolutions). We define the function $\Gamma^{(n)}$ for $n \in \mathbb{Z}$ by:

$$\Gamma^{(n)}(s) = \frac{\Delta_u^{(n)}(s) \wedge \Delta_v^{(n)}(s)}{\|\Delta_u^{(n)}(s) \wedge \Delta_v^{(n)}(s)\|} \text{ with } \Delta_u^{(n)} = \Psi_{\Psi^{(n)}_{\sigma,W_{\mathrm{avg}}},D_{\mathrm{u}}} \quad \text{(resp. for } \Delta_v^{(n)}\text{)} \qquad (5)$$

As the initial averaging process is iterated, the size of the neighborhood taken into account grows accordingly and the precision of the estimate increases as we get closer to the optimal number of iterations. (This number is discussed in Section 4.3.)

From the definition of the operator $\Psi^{(n)}$, we see that $\Gamma^{(n)}(s)$ is the result of a computation which involves all the surfels of Σ whose distance to s is at most n in the v-adjacency graph of Σ. In fact, the size of the neighborhood taken into account when computing $\Gamma^{(n)}(s)$ grows with n but the weights tend to follow a geodesic distance which does not coincide with the distance in the v-adjacency graph. This point, which we claim is a good point, is discussed in the next section. (The way the weights are distributed when the number of iterations increases is illustrated by Figures 5(b) and 4(b).) We will present in Section 4.3 the results of some experiments.

4.2 Comparison with Papier's Averaging Process

In [16,15], Papier *et al.* define averaging weights on possibly large neighborhoods obtained using a breadth-first visiting algorithm of the surfels v-adjacency graph. Their approach generalizes the one of [2] who used only the only e-neighborhood to estimate the normals by averaging the elementary normals (among the six possible ones). Both methods are based on the averaging of the canonical normals $\nu(s)$ of the surfels to estimate the exact normals. This point slightly differs from our method since we are not averaging the normal vectors but simply the surfel centers.

Furthermore, when considering a large neighborhood in an averaging process, one should expect that the boundary of the neighborhood is equidistant to its center, according to the geodesic distance within the continuous surface which has been digitized. The weights should also decrease according to this geodesic distance. As depicted in Figure 5(a) on a digitized plane with normal vector

$(1, 1, 1)$, as well as in Figure 4(c) on a digitized paraboloid, neighborhoods obtained by a breadth-first traversal of the surfels graph do not share the former property. Therefore, we think that the neighborhoods used by Papier *et al.* are not optimal.

In our case, the neighborhood taken into account by the averaging process grows after iterations of convolutions with a local mask, designed to adapt itself to the local geometry of the surface (Section 3.1). Although the actual size of the masks resulting of iterated convolutions also follow the v-adjacency graph, we observe that the weights in these masks tends to share the above mentioned properties (isotropy and decreasing according to a "continuous" geodesic distance). In order to illustrate how the averaging mask grows, we use a diffusion process: with $S = \mathbb{R}$ we choose a surfel $s_0 \in \Sigma$ and define the function $\delta_{s_0} : \Sigma \longrightarrow \mathbb{R}$ such that $\delta_{s_0}(s_0) = 1$ and $\delta_{s_0}(s) = 0$ for $s \in \Sigma \setminus \{s_0\}$. Then, we compute $\Psi^{(n)}_{\delta_{s_0}, W_{\text{avg}}}$ for a given n. A result of this diffusion process that we call an *impulse response of the averaging filter* $\Psi^{(n)}_{f, W_{\text{avg}}}$ is depicted in Figure 5(b) for the same plane as mentioned previously. Another example is given on the paraboloid depicted in Figure 4(b) with the surfel s_0 at the saddle point, where one can get convinced that the impulse response of our iterated convolution mask behaves as if following a geodesic distance function on the surface.

4.3 Experiments

We have evaluated the precision of the estimation that can be achieved with our method. For this purpose, we have used digitized spheres and tori with several radii and measured the average angular error between the estimated and the exact normal vectors for all surfels. In these experiments, we use a number n of convolution iterations inspired by the result of [14] (Theorem 1) for continuous functions from $\mathbb{R}$ to $\mathbb{R}$ known through their digitizations. Following the latter result, if h is the width of the pixels used for the digitization process, then a convergence at rate $h^{\frac{2}{3}}$ for the estimation of the first derivative of the function may be obtained by using a convolution mask with a width $w = \lfloor h^{-\frac{4}{3}} \rfloor$. Given this width, we deduce the number n of convolution iterations required when using the mask W_{avg} (see Section 3.1): $n = \frac{w}{2}$ if $w \in 2\mathbb{Z}$ and $n = \frac{w-1}{2}$ if $w + 1 \in 2\mathbb{Z}$.

The results of our experiments are presented in Figure 6. For Figure 6(a), we have used digital spheres with radii from 10 to 100 (i.e. h goes from $\frac{1}{20}$ to $\frac{1}{200}$). It appears clearly that the method achieves a better estimation when the size of the sphere increases (i.e. the digitization step decreases). This tends to show that our estimator is multigrid convergent.

As a comparison, tests conducted by Papier in [15] do not clearly show an improvement of the precision when the radius of the sphere increases. Furthermore, they only used umbrellas of order 1 to 5. It is however clear that the size of the mask should be set according to the resolution, as we do here. With the approach mentioned in the introduction, the best result in [10] is obtained on a sphere with radius 100 and an average error of 1.51° (std. dev. 2.34), when the average error of our method is 0.58° (std. dev. 0.27). On a sphere with radius 50, they obtain 2.19° (std. dev. 3.46) when we have 0.85° (std. dev. 0.33). The

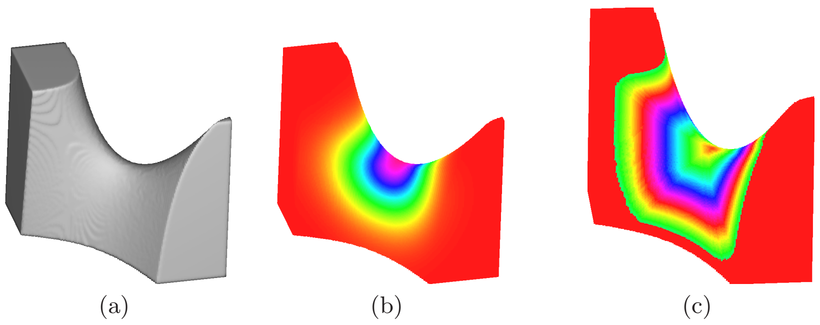
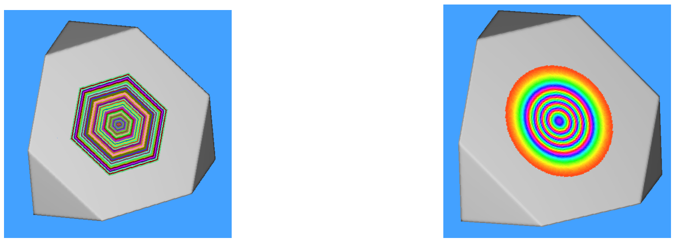
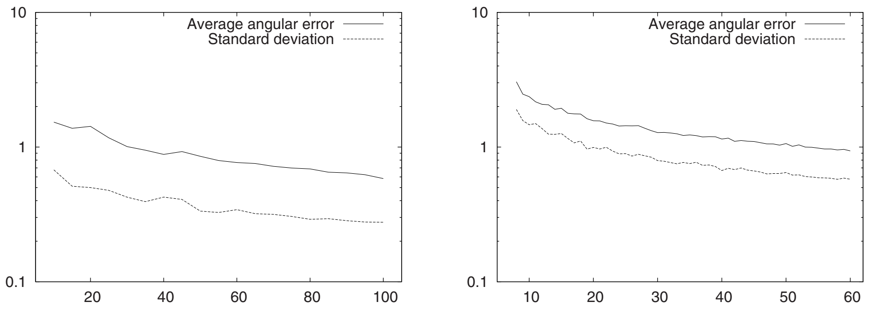


Fig. 4. (a) View of a paraboloid shaded using normals estimated with our method. (b) Response of the iterated averaging filter over the paraboloid compared with a breadth-first traversal (c). (The volume size is 150 × 150 × 150 and has 104926 surfels.)



(a) Breadth-first traversal. (b) Iterated convolution response.

Fig. 5. Breadth-first traversal of the surfels graph and convolution over a digitized plane $x + y + z = 0$



(a) Spheres with increasing radii. (b) Tori with increasing large and small radii. (Large radius on the horizontal axis.)

Fig. 6. Average angular error, in degree and using a logarithmic scale, of the estimated normals on spheres and tori. The error, on each surfel, is computed as the angle between the estimated normal vector and the direction from the center of sphere to the center of the surfel (resp. from the skeleton of the torus).

earlier paper by Tellier and Debled-Rennesson [19] reports the best average error of 2.84° (std. dev. 2.24) for a sphere with radius 25, when our method obtains 1.16° (std. dev. 0.47).

Furthermore, spheres are not general enough to put to the test a surface normal estimator. This at least because a sphere has a constant and positive Gaussian curvature. Therefore, we have tested our method on several tori with increasing radii, rotated along the three axes. Tori are nice for this test because they have both positive and negative Gaussian curvatures. The results are depicted in Figure 6(b), where each torus had a small radius of half its large one. Again, the average error and its standard deviation decrease with an increasing resolution.

5 Curvature Estimation

In this section, we present an attempt to estimate second order quantities after a similar averaging process as the one used to estimate normals. Although the theoretical work done in [14] shows that higher orders estimates may be computed in a similar way for 1D digitized functions, it seems from our first experiments that the precision is not so good when using on-surface convolution.

5.1 Second Derivatives Operators

Because the masks D_{u} and D_{v} defined in Section 3.2 do not generally preserve the orientations of the differentiations when applied to one surfel and one of its neighbors, they should not be applied iteratively to compute second derivatives. Hence, we need to define new differentiation operators.

First, given a surfel s and its i^{th} edge we denote by $EE_i(s)$ the e-neighbor of $E_i(s)$ which is not included in a loop containing s. Intuitively, $EE_i(s)$ is the next surfel one encounters on the surface after $E_i(s)$ when traveling in the direction from s to $E_i(s)$. Furthermore, given the j^{th} vertex of s we denote by $EV_i^j(s)$ the e-neighbor of $N_i(s)$ that shares with s its vertex j. (See Figure 3(c) for the vertices and edges ordering.) Eventually, for $n \in \mathbb{Z}$ and $s \in \Sigma$ we define (remember that $\sigma(s)$ is the center of s)

$$\begin{aligned}
\Delta_{uu}^{(n)}(s) &= \frac{\big(\tilde{\sigma}^{(n)}(EE_0(s))-\tilde{\sigma}^{(n)}(s)\big)-\big(\tilde{\sigma}^{(n)}(s)-\tilde{\sigma}^{(n)}(EE_2(s))\big)}{4}\\
\Delta_{vv}^{(n)}(s) &= \frac{\big(\tilde{\sigma}^{(n)}(EE_1(s))-\tilde{\sigma}^{(n)}(s)\big)-\big(\tilde{\sigma}^{(n)}(s)-\tilde{\sigma}^{(n)}(EE_3(s))\big)}{4}\\
\Delta_{uv}^{(n)}(s) &= \frac{\big(\tilde{\sigma}^{(n)}(EV_0^0(s))-\tilde{\sigma}^{(n)}(EV_0^1(s))\big)-\big(\tilde{\sigma}^{(n)}(EV_2^3(s))-\tilde{\sigma}^{(n)}(EV_2^2(s))\big)}{4}
\end{aligned}$$

where

$$\tilde{\sigma}^{(n)} = \Psi_{\sigma,W_{\mathrm{avg}}}^{(n)} \tag{6}$$

5.2 Curvature Estimation

Given a *regular parametric surface* $S : D \subset \mathbb{R}^2 \longrightarrow \mathbb{R}^3$, $(u,v) \longmapsto S(u,v) = (x(u,v), y(u,v), z(u,v))$ where D is an open and connected subset of $\mathbb{R}^2$, the *Gaussian* (K) and *mean* (H) *curvatures* are defined as follows [18]:

$$K = \frac{e.g - f^2}{E.G - F^2} \qquad\qquad H = \frac{e.G - 2.f.F + g.E}{2(E.G - F^2)} \tag{7}$$

with

$$E = \frac{\partial S}{\partial u}\frac{\partial S}{\partial u},\ F = \frac{\partial S}{\partial u}\frac{\partial S}{\partial v},\ G = \frac{\partial S}{\partial v}\frac{\partial S}{\partial v},\ e = N \cdot \frac{\partial^2 S}{\partial u^2},\ f = N \cdot \frac{\partial^2 S}{\partial u \partial v},\ g = N \cdot \frac{\partial^2 S}{\partial v^2}$$

where N is the unit normal field over S.

For a given number n of averaging iterations, we approximate $\frac{\partial S}{\partial u}$ as $\Delta_u^{(n)}$, $\frac{\partial S}{\partial v}$ as $\Delta_v^{(n)}$, $\frac{\partial^2 S}{\partial u^2}$ as $\Delta_{uu}^{(n)}$, $\frac{\partial^2 S}{\partial v^2}$ as $\Delta_{vv}^{(n)}$, $\frac{\partial^2 S}{\partial v^2}$ as $\Delta_{uv}^{(n)}$, and N as $\Gamma^{(n)}$ (Eq. 5). Eventually, we obtain an estimation of the Gaussian or mean curvature for each surfel of a digital surface using equation 7.

5.3 Experiment

We have conducted an experiment with a large digitized torus which show that the Gaussian curvature may be only roughly estimated. The torus had a 80 voxels large radius, and a 40 voxels small radius. Figure 7 shows the estimated and exact Gaussian curvatures for all the surfels sorted according to their increasing exact Gaussian curvatures. The size of the averaging mask to be used was set according to the result of [14] about higher order derivatives (Theorem 3). Actually, we divided the estimation process in two steps. First, an estimated Gaussian curvature $\tilde{g_1}(s)$ for each surfel $s \in \Sigma$ was computed after $\frac{n}{2}$ averaging

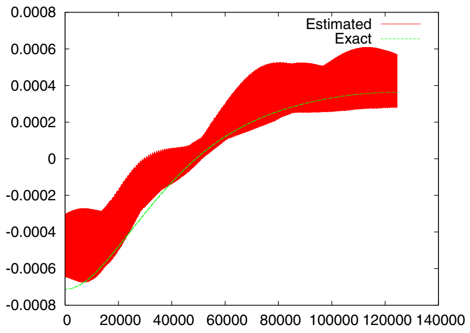


Fig. 7. Estimated and exact Gaussian curvatures on a torus (large radius is 80, small radius is 40). Surfels are numbered on the horizontal axis according to their increasing exact Gaussian curvatures. The estimated values appear as a thick area because they are plotted with lines and vary very quickly.

iterations (i.e., using $\Gamma^{\left(\frac{n}{2}\right)}$ and the operators $\Delta_*^{\left(\frac{n}{2}\right)}$ as decribed before) and we used $\tilde{g} = \Psi_{g_1,W_{\mathrm{avg}}}^{\left(\frac{n}{2}\right)}$ as the actual estimate. Figure 7 shows that the curvature is only approximated.

6 Conclusion

We have defined an averaging operator based on a convolution over the surface of a digital object, as well as directional derivative operators. When combined, these operators may be used to estimate the normal vectors on the surface of a digitized object. The implementation of this method is straightforward (compared with methods which involve, say, DSS recognition). Furthermore, since it relies on an averaging that may be seen as low-pass filtering, this method is less noise sensitive, just as the planar case ([14]).

We also tackled the problem of curvature estimation. This estimator should be investigated in the future since, as far as we know, there is no multigrid convergent second derivative estimator known for digital surfaces.

The complexity of the convolution approach should also be compared with existing geometric estimators.

References

1. Artzy, E., Frieder, G., Herman, G.T.: The theory, design, implementation and evaluation of a three dimensional surface detection algorithm. Computer graphics and image processing 15(1), 1–23 (1981)
2. Chen, L.-S., Herman, G.T., Reynolds, R.A., Udupa, J.K.: Surface shading in the cuberille environment. IEEE Journal of computer graphics and Application 5(12), 33–43 (1985)
3. Coeurjolly, D., Flin, F., Teytaud, O., Tougne, L.: Multigrid convergence and surface area estimation. In: Asano, T., Klette, R., Ronse, C. (eds.) Geometry, Morphology, and Computational Imaging. LNCS, vol. 2616, pp. 119–124. Springer, Heidelberg (2003)
4. Coeurjolly, D., Klette, R.: A comparative evaluation of length estimators of digital curves. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(2), 252–257 (2004)
5. Debled-Rennesson, I., Reveillès, J.-P.: A linear algorithm for segmentation of digital curves. International Journal of Pattern Recognition and Artificial Intelligence 9(4), 635–662 (1995)
6. Dorst, L., Smeulders, A.W.M.: Length estimators for digitized contours. Computer Vision, Graphics, and Image Processing 40(3), 311–333 (1987)
7. Herman, G.T.: Discrete multidimensional Jordan surfaces. CVGIP: Graphical Models and Image Processing 54(6), 507–515 (1992)
8. Herman, G.T., Liu, H.K.: Three-dimensional display of human organs from computed tomograms. Computer Graphics and Image Processing 9, 1–29 (1979)
9. Klette, R., Sun, H.J.: A global surface area estimation algorithm for digital regular solids. Technical report, Computer science department of the University of Auckland (September 2000)

10. Lachaud, J.-O., Vialard, A.: Geometric measures on arbitrary dimensional digital surfaces. In: Nyström, I., Sanniti di Baja, G., Svensson, S. (eds.) DGCI 2003. LNCS, vol. 2886, pp. 434–443. Springer, Heidelberg (2003)
11. Lachaud, J.-O., Vialard, A., de Vieilleville, F.: Fast, accurate and convergent tangent estimation on digital contours. Image and Vision Computing 25(10), 1572–1587 (2007)
12. Lenoir, A.: Des outils pour les surfaces discrètes. PhD thesis, Université de Caen (in French) (1999)
13. Lenoir, A., Malgouyres, R., Revenu, M.: Fast computation of the normal vector field of the surface of a 3-d discrete object. In: Miguet, S., Ubéda, S., Montanvert, A. (eds.) DGCI 1996. LNCS, vol. 1176, pp. 101–112. Springer, Heidelberg (1996)
14. Malgouyres, R., Brunet, F., Fourey, S.: Binomial convolutions and derivatives estimation from noisy discretizations. In: Coeurjolly, D., et al. (eds.) DGCI 2008. LNCS, vol. 4992, pp. 369–377. Springer, Heidelberg (2008)
15. Papier, L.: Polyédrisation et visualisation d'objets discrets tridimensionnels. PhD thesis, Université Louis Pasteur, Strasbourg, France (in French) (1999)
16. Papier, L., Françon, J.: Évalutation de la normale au bord d'un objet discret 3D. Revue de CFAO et d'informatique graphique 13, 205–226 (1998)
17. Sivignon, I., Dupont, F., Chassery, J.-M.: Discrete surfaces segmentation into discrete planes. In: Klette, R., Žunić, J. (eds.) IWCIA 2004. LNCS, vol. 3322, pp. 458–473. Springer, Heidelberg (2004)
18. Stoker, J.J.: Differential geometry. Wiley, Chichester (1989)
19. Tellier, P., Debled-Rennesson, I.: 3d discrete normal vectors. In: Bertrand, G., Couprie, M., Perroton, L. (eds.) DGCI 1999. LNCS, vol. 1568, pp. 447–458. Springer, Heidelberg (1999)
20. Windreich, G., Kiryati, N., Lohmann, G.: Voxel-based surface area estimation: from theory to practice. Pattern Recognition 36(11), 2531–2541 (2003)