Implementation of Integral based Digital Curvature Estimators in DGtal*

David Coeurjolly[1], Jacques-Olivier Lachaud[2], Jérémy Levallois[1,2]

[1] Université de Lyon, CNRS
INSA-Lyon, LIRIS, UMR5205, F-69621, France.
[2] Université de Savoie, CNRS
LAMA, UMR5127, F-73776, France.
david.coeurjolly@liris.cnrs.fr, jacques-olivier.lachaud@univ-savoie.fr, jeremy.levallois@liris.cnrs.fr

Abstract In many geometry processing applications, differential geometric quantities estimation such as curvature or normal vector field is an essential step. In [1], we have defined curvature estimators on digital shape boundaries based on Integral Invariants. In this paper, we focus on implementation details of these estimators.

Keywords: Digital geometry, curvature estimation, integral invariants

1 Introduction

In many shape processing applications, the estimation of differential quantities on the shape boundary such as normal vectors, curvatures or principal directions are crucial. When evaluating a differential estimator on discrete or digital data, because of digitization approximation, we need a way to mathematically link the estimated quantity to the expect Euclidean one. In Digital Geometry, we usually consider multigrid convergence principles: when the shape is digitized on a grid with resolution tending to zero, the estimated quantity should converge to the expected one [2].

In [1], we have proposed digital versions of integral invariant estimators for which convergence results can be obtained when the kernel size tends to zero. In this paper, we first remind the general principle of [1]. Then we focus on implementation details of these estimators in `DGtal` library, and finally we show some results.

1.1 Principle

In geometry processing, interesting mathematical tools have been developed to design differential estimators on smooth surfaces based on integral invariants. The principle is simple: we move a convolution kernel along the shape surface and we compute integrals on the intersection between the shape and the convolution kernel, as follow in dimension 3:

$$V_r(x) \stackrel{def}{=} \int_{B_r(x)} \chi(p) dp \tag{1}$$

where $B_r(x)$ is the Euclidean ball of radius r, centered at x and $\chi(p)$ the characteristic function of X. In dimension 2, we simply denote $A_r(x)$ such quantity (represented in orange color on Fig. 1).

*This work has been mainly funded by DigitalSnow ANR-11-BS02-009 research grants

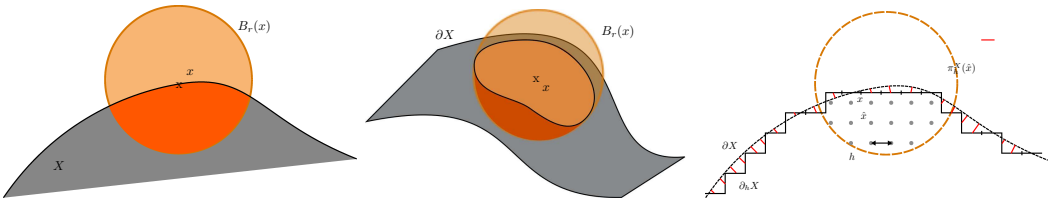

Figure 1: Integral invariant computation in dimension 2 (*left*) and 3 (*middle*), and notations in dimension 2 (*right*)

Authors of [4][5] have demonstrated that some integral quantities provide interesting curvature information when the kernel size tends to zero. Indeed, thanks to Taylor expansion at x of the surface ∂X approximated by a parametric function $y = f(x)$ in 2d and $z = f(x, y)$ in 3d and with a fixed radius r, we obtain convergent local curvature estimators $\tilde{\kappa}_r(X, x)$ and $\tilde{H}_r(X, x)$ of quantities $\kappa(X, x)$ and $H(X, x)$ respectively:

$$\tilde{\kappa}_r(X,x) \stackrel{def}{=} \frac{3\pi}{2r} - \frac{3A_r(x)}{r^3}, \quad \tilde{H}_r(X,x) \stackrel{def}{=} \frac{8}{3r} - \frac{4V_r(x)}{\pi r^4} \tag{2}$$

$$\tilde{\kappa}_r(X,x) = \kappa(X,x) + O(r), \quad \tilde{H}_r(X,x) = H(X,x) + O(r) \tag{3}$$

where $\kappa_r(X, x)$ is the 2d curvature of ∂X at x and $H_r(X, x)$ is the 3d mean curvature of ∂X at x.

In [1], we were interested in studying the behavior of integral invariants in digital geometry using a multigrid framework with digitization on grids with grid step h tending to zero. We define a digital version of Eq. (2) and (3) estimators.

$$\forall 0 < h < r, \quad \hat{\kappa}_r(Z,x,h) \stackrel{def}{=} \frac{3\pi}{2r} - \frac{3\widehat{\text{Area}}(B_{r/h}(\frac{1}{h}\cdot x) \cap Z, h)}{r^3}. \tag{4}$$

where $\hat{\kappa}_r$ is an integral digital curvature estimator of a digital shape $Z \subset \mathbb{Z}^2$ at point $x \in \mathbb{R}^2$ and step h. $B_{r/h}(\frac{1}{h}\cdot x) \cap Z, h)$ means the intersection between Z and a Ball B of radius r digitized by h centered in x.

In the same way, we have in 3d:

$$\forall 0 < h < r, \quad \hat{H}_r(Z',x,h) \stackrel{def}{=} \frac{8}{3r} - \frac{4\widehat{\text{Vol}}(B_{r/h}(\frac{1}{h}\cdot x) \cap Z', h)}{\pi r^4}. \tag{5}$$

where $\hat{H}_r$ is an integral digital mean curvature estimator of a digital shape $Z' \subset \mathbb{Z}^3$ at point $x \in \mathbb{R}^3$ and step h.

We have demonstrated that Eq. (4) and (5) are multigrid convergent. With hypothesis about the shape geometry and the radius of the convolution kernel, we guarantee a theoretical convergence speed in $O(h^{\frac{1}{3}})$, confirmed with experimental results. We have also discussed about an integral digital Gaussian curvature and principal curvature directions estimators obtained by digital approximation of the covariance matrices on Eq. (1) integral. Convergence results rely on the fact that digital moments converge in the same manner as volumes [3]. Note that as other convolution estimators, results of our algorithm depend on the size of the convolution kernel.

2 Implementation in `DGtal`

We have implemented integral invariant curvature estimators in `DGtal` library. `DGtal` is an open-source C++ library that provides geometry structures, algorithms and tools for digital data (`http://libdgtal.org`). Project `DGtal` gathers in a unified setting many data structures and algorithms of digital geometry and related fields (digital topology, image processing, discrete geometry, arithmetic). For example, `DGtal` provides us parametric or implicit multigrid shape construction in dimension 2 and 3, as well as object loaders. Furthermore, we benefit from available implementations of existing 2d curvature estimators. It allows us to make comparisons easy.

In our framework, objects are embedded into a Khalimsky space topological model. This topological structure gives us cells from 0 to n dimension (in 3d, cells are called *pointels*, *linels*, *pixels* and *voxels* for respectively cells of dimension 0, 1, 2 and 3). We call surfels cells of $n-1$ dimension and spel cells of n dimension. Khalimsky cells are defined into a Khalimsky Domain (cubical grid). For more information about the topology in `DGtal`, please refer to documentation (`http://libdgtal.org/doc/stable/packageTopology.html`)

2.1 User interface

The user part is rather simple by using only `IntegralInvariantMeanCurvatureEstimator` or `IntegralInvariantGaussianCurvatureEstimator`. These classes are parametrized by a Khalimsky space of a digital shape and a functor who return a value for a given Khalimsky cell. Indeed, the $\chi(p)$ characteristic function is defined as a functor (here with values in $\{0,1\}$ for each spel of the Khalimsky space).

At initialization (`init()`), we specify the current grid step h, and an Euclidean radius r_e in order to construct the convolution kernel. As described in Sect. 3, this parameter r_e determines the level of feature detected by the estimator.

For the evaluation (`eval()`), the user has two possibilities: evaluate the curvature at a specific surfel of the digitized shape surface, or at a range of surfels. This choice can be important because optimizations are available with the second option (see in Sect. 2.2). For the first one, the estimator return a curvature value at the given surfel. If you choose the second possibility, the estimator will try to optimize computations by using previous results thanks to displacement kernel masks. But it requires to set a range of 0-adjacent surfels. If the range of surfels does not follow that rule, no optimization is performed.

2.2 Implementation details

The following paragraph gives details on `IntegralInvariantMeanCurvatureEstimator` and `IntegralInvariantGaussianCurvatureEstimator` classes.

To sum up, we need a convolution kernel which will be integrated along a digitized shape boundary. At initialization, we first create a convolution kernel (`Ball2D` in 2d, `Ball3D` in 3d) of Euclidean radius r_e and digitized at the grid step h. `DigitalSurfaceConvolver` is in charge of centering the convolution kernel on the surfel on the digitized shape border. Its objective is to compute $B_{r/h}(\frac{1}{h}\cdot x)\cap Z$ and $B_{r/h}(\frac{1}{h}\cdot x)\cap Z'$ from Eq. (4) and (5).

At evaluation, it will place the convolution kernel in order to lie his center with the cell to estimate. After, it will iterate on all spels of the kernel, and compute the integral:

$$V(x) = \sum_{spel\in B_{r/h}(\frac{1}{h}\cdot x)\cap Z} f(x)g(s-x) \tag{6}$$

where x is the current spel, f is the shape functor (set by user), g is the kernel functor (return 1 on all spels from the kernel). `IntegralInvariantMeanCurvatureEstimator` uses this result to get $\widehat{\text{Area}}(B_{r/h}(\frac{1}{h}\cdot x)\cap Z, h)$ in 2d and $\widehat{\text{Vol}}(B_{r/h}(\frac{1}{h}\cdot x)\cap Z', h)$ in 3d. With Eq. (4) and (5), it finally returns curvature quantities. Using this approach, we obtain a computation cost in $O((r/h)^d)$ per surface element.

For `IntegralInvariantGaussianCurvatureEstimator`, the initialization is the same but at evaluation `DigitalSurfaceConvolver` computes a covariance matrix of $B_{r/h}(\frac{1}{h}\cdot x)\cap Z'$. Eigenvectors and eigenvalues analysis of this covariance matrix helps us to extract principal curvatures information. We can easily obtain mean or Gaussian curvature from them.

When we move the convolution kernel to 0-adjacent cells, we see that only few cells need to be updated from the previous convolution. As seen in the top left illustration of Fig. 2, only the green and the red part are interesting, the grey part is the same from the previous result. This effect can produce a lot of computations, so we only need to remove the green part, and add the red one (by additivity of the convolution).

$$\widehat{\text{Area}}(B_{r/h}(\frac{1}{h}\cdot x+\vec{\delta})\cap Z,h) = \widehat{\text{Area}}(B_{r/h}(\frac{1}{h}\cdot x)\cap Z,h) - \widehat{\text{Area}}(B1_{r/h}(\frac{1}{h}\cdot x)\cap Z,h) + \widehat{\text{Area}}(B2_{r/h}(\frac{1}{h}\cdot x+\vec{\delta})\cap Z,h).$$

where $\vec{\delta}$ is a translation vector in 0-adjacency, $B1_{r/h}$ is the mask to remove, and $B2_{r/h}$ the mask to add (both depending to $\vec{\delta}$).

We pre-compute at initialization of estimators a set a displacement masks from the full convolution kernel (see Fig. 2 at right for an example in 2d). At evaluation, we use the full kernel for the first computation. If the next surfel is adjacent to the previous one, we use a pair of masks (depending to the shift) to remove/add interested quantities to the last. In the worst case, all surfels set by user to the estimator are not adjacent and it will compute the curvature always with the full convolution kernel, which means no optimizations. The computation cost per surface element can be reduced from $O((r/h)^d)$ (size of the complete kernel) to $O((r/h)^{d-1})$, and the extra-cost of pre-computing displacement masks is meaningless: for example, with a full 2d kernel support size of 91893, the optimization built 8 supplementary masks of size ≈ 450.

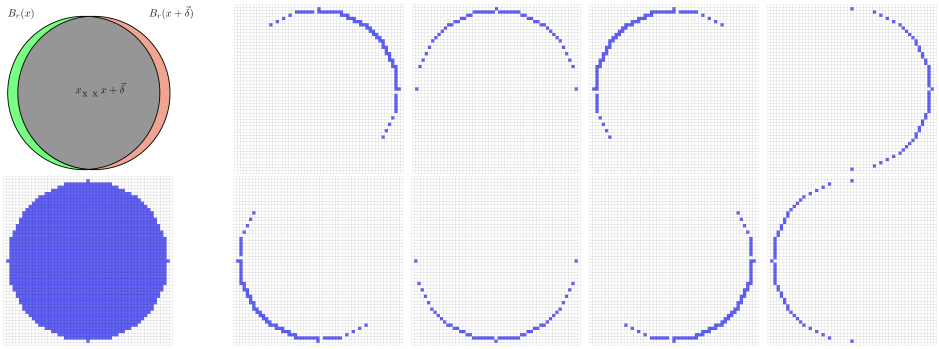

Figure 2: Illustration of the optimization with partial masks of a 2d ball for a given h. *On the top left: Illustration of a displacement of a full kernel from x to $x+\vec{\delta}$, on the bottom left: full 2d kernel support, on the right: 0-adjacent 2d partial displacement masks of full kernel support.*

3 Results

For all results presented in this section, yellow color is the highest curvature value, blue is the lowest curvature value, and red means in-between max and min values. In Fig. 3, we can see in order mean and Gaussian curvature mapped in a blobby cube[1]. We can also extract principal curvature information. As mentioned before, the user needs to choose a convolution kernel size r_e, which will influence the level of details captured (see Fig. 3) and computation time.

Euclidean radius r_e of a convolution kernel determines the level of details of estimated curvature. It is an intuitive behavior, because a smaller convolution kernel will be more influenced by small features, but will ignore large features and dependent to the noise level. A larger convolution kernel will result the opposite effect.

[1]Implicit surface is $81x^4+81y^4+81z^4-45x^2-45y^2-45z^2-6=0$.

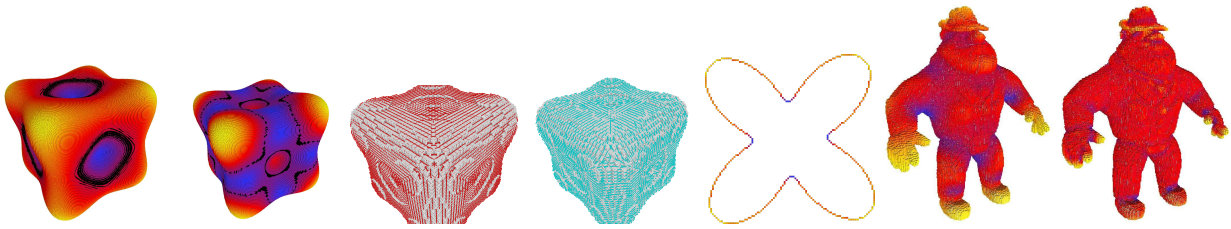

Figure 3: Illustration of curvature estimation.*From left to right: mean curvature and Gaussian curvature (back color means zero curvature surfels), first and second principal curvature directions mapped on a blobby cube on a blobby cube, 2d curvature mapped on a* `Flower2D`*, 3d Gaussian curvature on Al with a convolution kernel size of 14 and with a convolution kernel size of 7.*

References

[1] D. Coeurjolly, J.-O. Lachaud, and J. Levallois. Integral based curvature estimators in digital geometry. In *Discrete Geometry for Computer Imagery*, 2013.

[2] D. Coeurjolly, J.-O. Lachaud, and T. Roussillon. *Digital Geometry Algorithms, Theoretical Foundations and Applications of Computational Imaging*, volume 2 of *LNCVB*, chapter Multigrid convergence of discrete geometric estimators, pages 395–424. Springer, 2012.

[3] R. Klette and J. Žunić. Multigrid convergence of calculated features in image analysis. *Journal of Mathematical Imaging and Vision*, 13:173–191, 2000.

[4] H Pottmann, J Wallner, Q Huang, and Y Yang. Integral invariants for robust geometry processing. *Computer Aided Geometric Design*, 26(1):37–60, 2009.

[5] H Pottmann, J Wallner, Y Yang, Y Lai, and S Hu. Principal curvatures from the integral invariant viewpoint. *Computer Aided Geometric Design*, 24(8-9):428–442, 2007.

ISSN: 1885-4508