Published in Image Processing On Line on 2017–07–28.
Submitted on 2017–06–01, accepted on 2017–06–15.
ISSN 2105–1232 © 2017 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2017.212

The Image Curvature Microscope: Accurate Curvature Computation at Subpixel Resolution

Adina Ciomaga[1], Pascal Monasse[2], Jean-Michel Morel[3]

[1] Univ. Paris Diderot, Laboratoire Jacques-Louis Lions, UMR 7598, UPMC, CNRS, F-75205 Paris, France
[2] LIGM (UMR 8049), École des Ponts, UPE, Champs-sur-Marne, France
[3] CMLA, ENS Cachan, France (morel@cmla.ens-cachan.fr)
(adina@math.univ-paris-diderot.fr, monasse@imagine.enpc.fr, morel@cmla.ens-cachan.fr)

Communicated by Luis Álvarez *Demo edited by* Pascal Monasse

Abstract

We detail in this paper the numerical implementation of the so-called image curvature microscope, an algorithm that computes accurate image curvatures at subpixel resolution, and yields a curvature map conforming with our visual perception. In contrast to standard methods, which would compute the curvature by a finite difference scheme, the curvatures are evaluated directly on the level lines of the bilinearly interpolated image, after their independent smoothing, a step necessary to remove pixelization artifacts. The smoothing step consists in the affine erosion of the level lines through a geometric scheme, and can be applied in parallel to all level lines. The online algorithm allows the user to visualize the image of curvatures at different resolutions, as well as the set of level lines before and after smoothing.

Source Code

The ANSI C++ implementation reviewed by IPOL is given in the file `curv.cpp`. The complete source code, code documentation, and online demo are accessible at the IPOL web page of this article[1]. The *level lines extraction* (files `levelLine.cpp`, `lltree.cpp`) and *level lines smoothing* (file `gass.cpp`) are independent algorithms, out of the scope of this article, and details for their implementation are planned for separate IPOL publications.

Keywords: curvature; level lines; image smoothing; subpixel

[1] https://doi.org/10.5201/ipol.2017.212

1 Introduction

We detail in this paper the Image Curvature Microscope (ICM) algorithm that computes accurate image curvatures, at subpixel resolution, and yields a visualization of the curvature map conforming to our visual perception. The curvatures are evaluated directly on the level lines of an image, after their independent smoothing.

The role of curvatures in visual perception has been long discussed and numerous attempts have been introduced to estimate curvatures accurately. We focus here on the thorough description and implementation of the accurate algorithm sketched in [11].

The role of curvature in visual perception was first raised by Attneave [4], who argued, based on neurological grounds, that visual stimulation is highly redundant and "information is concentrated along contours and further at those points on a contour at which its direction changes most rapidly (peaks of curvature)", as shown in Figure 1.

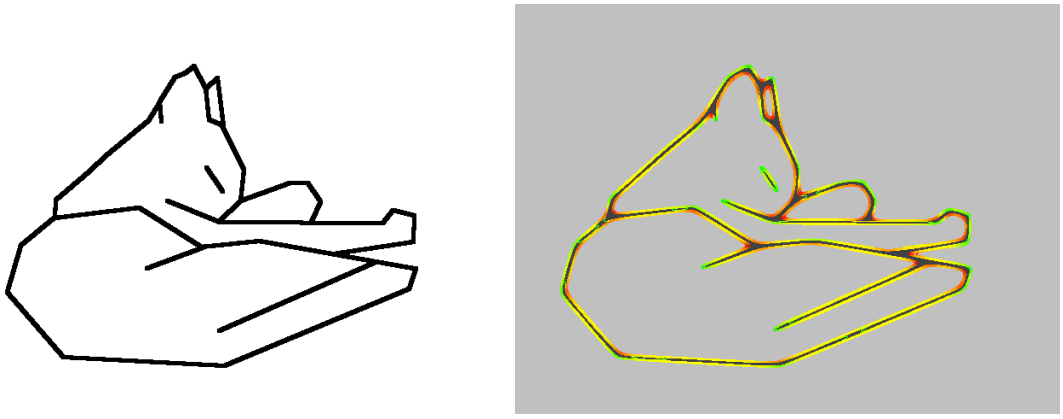


Figure 1: Attneave's cat and its curvature map computed by our algorithm. The drawn lines have a certain width, so they give rise to several level lines (exterior and interior), whose orientations are opposite. The hue encodes the curvature, with yellow meaning zero curvature, and shades of green and red meaning positive and negatives curvatures respectively.

Asada and Brady implemented Attneave's idea and introduced the concept of *curvature primal sketch* [3]. This consists of a multiscale interpretation of shapes: contours of objects are represented by splines whose knots are points of significant changes in curvature, and the knots are found by convolutions with first and second order derivatives of a Gaussian. Their representation follows the *primal sketch* introduced by Marr [19] and the idea of *scale space* promoted by Witkin [27]. This paper led to increasingly sophisticated attempts to compute curvature correctly. There were, however, two major difficulties:

(i) the extraction of the contours on which the curvature should be computed;

(ii) the smoothing of the contours while preserving their topological and topographical properties.

Caselles, Coll and Morel [7] realized the potential of using directly level lines instead of edges and proposed to perform image analysis directly on the topographic map. Ballester, Caselles and Monasse [5] and later Caselles, Meinhardt and Monasse [8] showed how the tree of lower and upper level sets can merge into a single tree of shapes. A fast algorithm computing the topographic map was proposed by Monasse and Guichard [23].

The second difficulty was clarified by several mathematical contributions. Gage and Hamilton [15] studied the evolution of convex planar curves by intrinsic heat equation, also known as *curve shortening*. Their study was completed by Grayson [16] for general Jordan curves. In parallel, Mackworth and Mokthariam [21] proposed a fast numerical scheme for curve shortening. Osher and Sethian

introduced the *level-set method* [25] to extend the motion to higher dimensional hyper-surfaces and, thus, implement motion by mean curvature, for which Evans and Spruck [14] and Chen, Giga and Goto [10] provided the analytical framework. Sapiro and Tannenbaum discovered the affine curve shortening [26] and Alvarez et al. [1] the affine invariant and contrast invariant image smoothing. A remarkably fast geometric algorithm for affine shortening was published by Moisan [22]. Mikula and Ševčovič gave numerical methods for more general curvature motions [20], and Cao and Moisan [6] proposed morphological schemes for the motion of curves by arbitrary powers of the curvature.

Yet, direct computations of curvatures on raw images were still unreliable, as they were based on difference approximations on finite grids, see for example Crandall and Lions [13] or Alvarez and Morel [2]. Although these schemes are somewhat efficient for simulating image motion by mean curvature, they are not reliable when it comes to visualizing the actual curvature map. The result, which is displayed in Figure 2 (middle two images) has spurious oscillations along transversal lines.

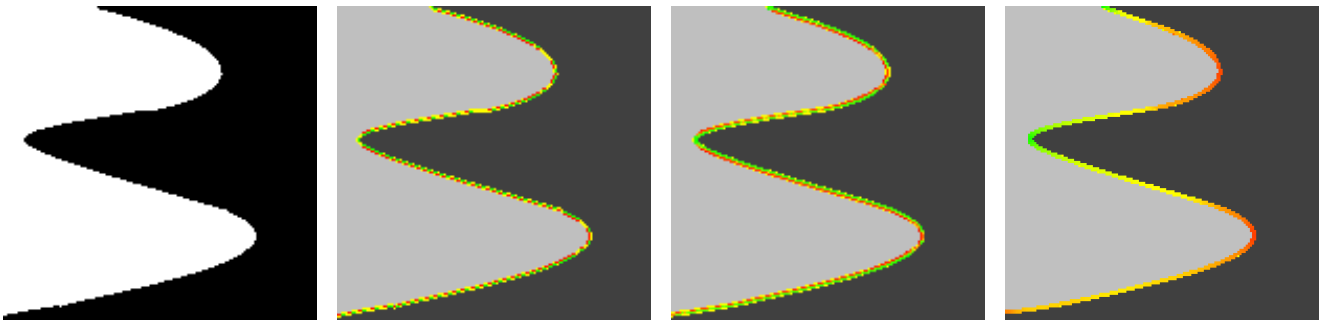


Figure 2: The curvature color display rule: (a) the initial image, (b) curvatures computed by a finite difference scheme, (c) curvatures computed by a finite difference scheme on a smoothed image, (d) curvatures computed by the image curvature microscope algorithm.

Based on the previously mentioned contributions, the authors introduced in [11] an algorithm that computes the image curvatures directly on the bilinear level lines, after their smoothing. The fact that one can compute, at any desired resolution, the bilinear level lines of the image yields a microscopic visualization of the curvature map, as displayed in Figure 2 (right). As visible in the middle two images of Figure 2, a finite difference scheme applied on the original image, or even after image smoothing, does not yield a satisfactory result.

The paper is organized as follows. Section 2 reviews the definition of curvatures in the continuous 2D setting and introduces a formula for the discrete curvature based on its direct computation on level lines. Section 3 details the algorithm pipeline and its implementation, while experiments are discussed in Section 4.

2 Preliminaries

Digital images are given as discrete functions defined on a finite grid $I : \mathcal{R} \cap \mathbb{N}^2 \to \{0, 1, 2, \ldots, 255\}$, where $\mathcal{R} = [0, w) \times [0, h)$ stands for the frame of the image of width $w \in \mathbb{N}^*$ and height $h \in \mathbb{N}^*$. Our goal is to define discrete curvatures associated to a digital image I, while preserving their geometric properties from the continuous framework. To this end, we consider the continuous bilinear interpolation $u : \mathcal{R} \to [0, 255]$, which is a continuous function of the form

$$u(x, y) = axy + bx + cy + d, \tag{1}$$

so that $u = 0$ on $\partial\mathcal{R}$, a, b, c, d are numeric functions depending on $(\lfloor x \rfloor, \lfloor y \rfloor)$, and, for all $i \in \{0, 1, \ldots, w-1\}$, $j \in \{0, 1, \ldots, h-1\}$,

$$u(i + 1/2, j + 1/2) = I(i, j). \tag{2}$$

It is well known that u is piecewise $\mathcal{C}^1(\mathcal{R})$, discontinuities of the gradient may occur along the grid edges, and except for a set of Lebesgue measure zero the function is actually $\mathcal{C}^\infty(\mathcal{R})$.

We denote by $\mathbf{x} = (x, y) \in \mathcal{R}$ a point in $\mathbb{R}^2$, $||\cdot||$ the standard Euclidean norm, and recall that the gradient, the vector orthogonal to the gradient and the Hessian matrix of u at $\mathbf{x}$ are respectively given by

$$Du(\mathbf{x}) = \begin{pmatrix} u_x \\ u_y \end{pmatrix}(\mathbf{x}) \qquad Du^\perp(\mathbf{x}) = \begin{pmatrix} -u_y \\ u_x \end{pmatrix}(\mathbf{x}) \qquad D^2u(\mathbf{x}) = \begin{pmatrix} u_{xx} & u_{xy} \\ u_{yx} & u_{yy} \end{pmatrix}(\mathbf{x}). \tag{3}$$

2.1 The Level Lines Decomposition

2.1.1 Continous Framework

One can decompose u into its upper and lower level sets $\chi_\lambda(u) = \{\mathbf{x} \in \mathcal{R} : u(\mathbf{x}) > \lambda\}$, $\chi^\lambda(u) = \{\mathbf{x} \in \mathcal{R} : u(\mathbf{x}) < \lambda\}$ where $\lambda \in [0, 255]$. The upper, respectively lower, level sets can be ordered in a tree structure, following the geometrical inclusion.

The *level lines* of u are defined as the topological boundaries of the connected components of its level sets. Since u is the bilinear interpolation of I, its level lines are *piecewise $\mathcal{C}^2$ Jordan curves* at the possible exception of levels $\lambda \in \{0, 1, \cdots, 255\}$ and the levels of saddle points of u (a finite set) [9]. In fact, level lines are expressed piecewise by implicit equations of the form

$$a(x - x_s)(y - y_s) = \lambda - \lambda_s \quad \text{or} \quad bx + cy + d = \lambda. \tag{4}$$

The first case corresponds to a factorization of the implicit equation $u(x, y) = \lambda$ using (1) when $a \neq 0$, with $x_s = -c/a$, $y_s = -b/a$ and $\lambda_s = d - ax_sy_s = (ad - bc)/a$. Hence, level lines are locally either pieces of hyperbolae of asymptotes $x = x_s$, $y = y_s$ for all $\lambda \neq \lambda_s$, or straight lines. This may lead to pixelization effects, especially when concatenating level lines along different pixels. The phenomenon can be attenuated by taking half-integer values for λ. In addition, we may assume that level lines meeting the frame of the image are closed curves, by imposing that I has a constant gray level at its boundary, i.e. $\#\big(I\left(\partial(\mathcal{R} \cap \mathbb{N}^2)\right)\big) = 1$.

The level lines inherit the *tree structure* of their upper and lower level sets, and they can be merged into a single tree, as shown in [23]. Indeed, each level line has an unambiguous notion of interior, the bounded connected component of its complement in the plane. One can therefore build a partial order on the family of level lines by writing $\Sigma \prec \Sigma'$ if Σ is in the interior of Σ'. Although the computation of curvatures does not use the tree structure, since each level line is treated independently, the reconstruction of a gray level image on which the curvature map is displayed requires it.

2.1.2 Discrete Level Lines

Level lines are discretized from their continuous bilinear analogue by sampling over a finite number of levels. Firstly, a quantization step q is given, in order to extract all level lines corresponding to $\lambda \in (0.5 + q\mathbb{N}) \cap (0, 255)$. By default, the algorithm chooses half-integer levels $\{0.5, 1.5, \ldots, 254.5\}$, in order to select level lines which are closed curves. However, the structure of level lines can take a rather complicated form at saddle levels λ_s, and one is not guaranteed to avoid all saddle values, hence to select only Jordan curves by this process. Nevertheless, the level line extraction algorithm is resilient to this problem: if a level line corresponding to some λ_s has self-crossings, then the discretization follows the direction given by the level line at level $\lambda + \epsilon$, with $\epsilon > 0$ sufficiently small. As a consequence of this rule, some level line may be cut at a saddle point into two closed curves, or some other may have a double point at a saddle. This choice is consistent with the fattening effect of the smoothing process, performed next: curves with self-crossings are limits from both sides

of families of Jordan curves, and instantly develop a nonempty interior (see [12] for details). An example about curvature computation near saddle points is discussed in Section 3.3.

Secondly, level lines are sampled from their bilinear analogues given by equations of form (4), and registered as *discrete polygonal lines*: for all $\lambda \in (0.5 + q\mathbb{N}) \cap (0, 255)$ there exist a finite set I_λ of level lines such that, for all $i \in I_\lambda$, the level line $\Sigma^{\lambda,i}$ is an ordered set of points $P_j^{\lambda,i}$

$$\Sigma^{\lambda,i} = \{P_j^{\lambda,i}(x_j, y_j)\} \quad \text{with} \quad P_0^{\lambda,i} = P_N^{\lambda,i}, \tag{5}$$

where $(x_j, y_j) \in \mathcal{R}$ for $j = 1, 2, \cdots, N$ are floating point coordinates of points $P_j^{\lambda,i}$.

Discrete level lines maintain the *tree structure*. In addition, they are oriented according to the direction of the gradient: the level line is oriented so as to leave the gradient on its left, which is clockwise for the outer boundary of a lower level set component.

The fact that one can choose (i) any quantization step q and (ii) a sufficiently large number of points when discretizing formula (4) shows that level lines can be extracted at any desired resolution. This is necessary to explore visually the geometric image structure. In particular curvatures computed directly on bilinear level lines, are given at an arbitrary resolution, hence the name of *curvature microscope*.

The level line extraction algorithm and tree structure construction are out of the scope of this publication, considered here only as a preprocessing step. Rigorous definitions and the extraction algorithm are detailed in a monograph [9].

2.2 Curvatures in Digital Images

The curvature is an invariant descriptor associated to a curve, which describes how rapidly a curve pulls away from its tangent line. It is therefore natural to think of curvatures in an image as being associated to the level lines.

2.2.1 Continuous Framework

Let Σ be a bilinear level line and assume that it is a piecewise $\mathcal{C}^2$ Jordan curve (which is the case except for a finite number of them). Consider an arc-length parametrization so that $\Sigma = \{\mathbf{x}(s) : s \in [0, L],\ \|\mathbf{x}'(s)\| = 1\}$. Then Σ has a well defined *vectorial curvature* defined by

$$\kappa(\mathbf{x}) := \mathbf{x}''(s). \tag{6}$$

Notice that the vectorial curvature does not depend on the orientation or starting point of the curve, since writing $\mathbf{y}(s) = \mathbf{x}(s_0 - s)$, we have $\mathbf{y}'(s) = -\mathbf{x}'(s_0 - s)$ but $\mathbf{y}''(s) = \mathbf{x}''(s_0 - s)$.

At the same time, level lines are subsets of iso-level sets $\{\mathbf{x} \in \mathcal{R} : u(\mathbf{x}) = \lambda\}$. In view of the implicit function theorem, at points where $Du(\mathbf{x_0}) \neq 0$, the iso-level set is locally a graph in a neighborhood of $\mathbf{x}_0$, part of a piecewise smooth Jordan curve Σ, passing through $\mathbf{x}_0$ [17]. If u is locally $\mathcal{C}^2$ at $\mathbf{x_0}$, then the *scalar curvature* of u at $\mathbf{x_0}$ is given by

$$\mathrm{curv}(u)(\mathbf{x_0}) := \frac{u_{xx}u_y^2 - 2u_{xy}u_xu_y + u_{yy}u_x^2}{(u_x^2 + u_y^2)^{3/2}}(\mathbf{x_0}), \tag{7}$$

or equivalently

$$\mathrm{curv}(u)(\mathbf{x_0}) = \frac{1}{\|Du\|^3} D^2u(Du^\perp, Du^\perp)(\mathbf{x_0}).$$

Proposition 1. *The scalar curvature (7) is linked to the vectorial curvature (6) of the level line containing* $\mathbf{x_0}$ *by*

$$\kappa(\mathbf{x_0}) = -\mathrm{curv}(u)(\mathbf{x_0}) \frac{Du}{\|Du\|}(\mathbf{x_0}). \tag{8}$$

Proof. Let $\mathbf{t}(s) = \mathbf{x}'(s)$ for $s \in [0, L]$ be the unit tangent vector oriented along increasing s, and $\mathbf{n}(\mathbf{s}), s \in [0, L]$ be the unit normal vector such that the pair $\{\mathbf{t}(s), \mathbf{n}(s)\}$ forms a direct orthonormal basis of the plane (the tangent $\mathbf{t}$ turns counterclockwise toward the normal $\mathbf{n}$). One can define the signed scalar curvature $k(s)$ by

$$\mathbf{x}''(s) = k(s)\mathbf{n}(s). \tag{9}$$

The signed scalar curvature $k(s)$ changes sign with the orientation of the curve. However, it does not coincide with the scalar curvature defined in (7), as we shall see below.
Differentiating the equation $u(\mathbf{x}(s)) = \lambda$ with respect to s, we see that

$$Du(\mathbf{x}(s)) \cdot \mathbf{x}'(s) = 0,$$

hence the normal $\mathbf{n}(s)$ and $Du(\mathbf{x})$ are collinear. Therefore, the direct frame $\{\mathbf{t}, \mathbf{n}\}$ is given either by (i) $\{-Du^\perp/||Du||, Du/||Du||\}$ or (ii) $\{Du^\perp/||Du||, -Du/||Du||\}$. Differentiating twice the equation we get

$$Du(\mathbf{x}(s)) \cdot \mathbf{x}''(s) = -D^2u(\mathbf{x}(s))\ \mathbf{x}'(s) \cdot \mathbf{x}'(s).$$

In case (i), the normal $\mathbf{n}(s)$ has the same direction as $Du(\mathbf{x}(s))$, and the equation becomes

$$k(s) = \mathbf{x}''(s) \cdot \frac{Du}{\|Du\|}\big(\mathbf{x}(s)\big) = -\frac{1}{\|Du\|} D^2u \left(-\frac{Du^\perp}{||Du||}, -\frac{Du^\perp}{||Du||}\right)(\mathbf{x}(s)),$$

whereas, in case (ii), we have

$$k(s) = \mathbf{x}''(s) \cdot \frac{-Du}{\|Du\|}\big(\mathbf{x}(s)\big) = \frac{1}{\|Du\|} D^2u \left(\frac{Du^\perp}{||Du||}, \frac{Du^\perp}{||Du||}\right)(\mathbf{x}(s)).$$

However, note that in both cases,

$$\begin{aligned}\mathbf{x}''(s) &= -\frac{1}{\|Du\|} D^2u \left(\frac{Du^\perp}{||Du||}, \frac{Du^\perp}{||Du||}\right)(\mathbf{x}(s))\ \frac{Du}{\|Du\|}(\mathbf{x}(s)) \\ &= -\mathrm{curv}(u)\big(\mathbf{x}(\mathbf{s})\big)\frac{Du}{\|Du\|}(\mathbf{x}(s)).\end{aligned}$$

□

Remark 1. *The orientation of the curve can be chosen according to the gradient direction: let the normal be* $\mathbf{n} = Du/\|Du\|$*, in which case* $\mathbf{t} = -Du^\perp/\|Du\|$*, so that the frame is direct (case (i) in the proof above). In this case* $k(s) = -curv(u)(\mathbf{x}(\mathbf{s}))$*.*

Equation (8) suggests that the curvature can be computed in two different ways: either as the curvature of a level line extracted from the image and parameterized by arc-length, or as a two-dimensional differential operator. The latter solution, more direct, is the basis for finite difference schemes. However, as shown by Mondelli and Ciomaga [24], such schemes always present some defects because their discrete nature does not satisfy the geometric covariance properties, namely the commutation with any Euclidean transformation in the case of curvature computation. That is the reason why the former solution, evaluation via level lines, though more complex, is adopted in this paper.

2.2.2 Discrete Curvatures

Approaches previously introduced to compute discrete curvatures were based on finite difference schemes approximations for (7), see for example Crandall and Lions [13], Alvarez and Morel [2]. However, these approximations lead to spurious oscillations along horizontal and vertical lines, as they are based on grid values, see Figure 2(b). We give below a formula to compute instead discrete curvatures directly on the level lines. This is based on a discrete approximation of (9).

Recall that each level line is stored as a polygon, defined by its ordered vertices (5), and it is the discretization of a continuous level line. For convenience, we drop the dependence on λ, and denote generically a level line by $\Sigma = \{P_j(x_j, y_j)\}$ with $P_0 = P_N$. At any vertex P_j one can compute a discrete curvature $\kappa(P_j)$, as the inverse of the circumscribed radius. More precisely, we consider for each $j = 0, 1, \cdots, N$ the triangle determined by three consecutive points P_{j-1}, P_j, P_{j+1}, compute the radius r_j of the circumcircle and define the (signed) discrete curvature $\kappa(P_j) := \pm 1/r_j$, with a sign consistent with (9). Remember that the level line is oriented so that the gradient of the image is at the left hand side, meaning that $\mathbf{n} = Du/\|Du\|$ (Figure 3).

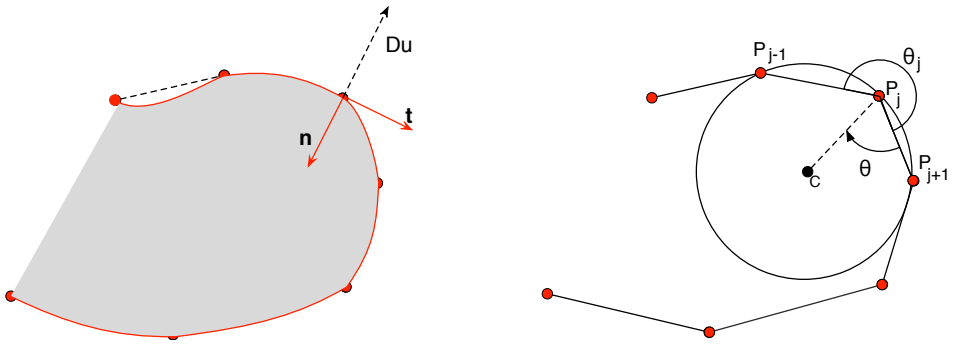


Figure 3: Left: piece of a level line, oriented clockwise (red arrow), with the gradient of the image at its left hand side. Right: discretization of the level line and computation of discrete curvature at each vertex, as inverse of the circumscribed radius.

Proposition 2. *The discrete curvature at point P_j is given by*

$$\kappa(P_j) = \frac{-2\,\sin\theta_j}{\|P_{j-1}P_{j+1}\|} = \frac{-2\,\det(P_jP_{j-1} \quad P_jP_{j+1})}{\|P_{j-1}P_j\|\;\|P_jP_{j+1}\|\;\|P_{j-1}P_{j+1}\|}, \tag{10}$$

where θ_j is the angle $\angle(P_jP_{j-1}, P_jP_{j+1})$, and the numerator of the right hand side includes the term

$$\det\begin{pmatrix} P_jP_{j-1} & P_jP_{j+1}\end{pmatrix} := \det\begin{pmatrix} x_{j-1}-x_j & x_{j+1}-x_j \\ y_{j-1}-y_j & y_{j+1}-y_j\end{pmatrix}. \tag{11}$$

Proof. Let C the center of the circumcircle of the triangle, and $\theta = \angle(P_jP_{j+1}, P_jC)$ (see Figure 3 right). The point C being on two perpendicular bisectors, we have

$$r_j\,\cos\theta = \frac{\|P_jP_{j+1}\|}{2}, \qquad r_j\,\cos(\theta_j+\theta) = \frac{\|P_{j-1}P_j\|}{2}. \tag{12}$$

Let $r_+ = \|P_jP_{j+1}\|$, $r_- = \|P_{j-1}P_j\|$. Eliminating r_j between both equations and expanding the cosine of a sum, we deduce

$$(r_+\cos\theta_j - r_-)\,\cos\theta = r_+\,\sin\theta_j\,\sin\theta,$$

so that

$$\tan\theta = \frac{r_+\cos\theta_j - r_-}{r_+\,\sin\theta_j}.$$

Then we get

$$\frac{1}{|\cos\theta|} = \sqrt{1+\tan^2\theta} = \frac{\sqrt{r_+^2 + r_-^2 - 2\,r_+\,r_-\,\cos\theta_j}}{r_+\,|\sin\theta_j|} = \frac{\|P_{j-1}P_{j+1}\|}{r_+\,|\sin\theta_j|},$$

where we recognized the cosine law in the last equality. Injecting into (12) yields

$$\frac{1}{|\kappa(P_j)|} = r_j = \frac{\|P_{j-1}P_{j+1}\|}{2\,|\sin\theta_j|}.$$

The last equality of the proposition comes from the vector product property

$$\det\begin{pmatrix} P_jP_{j-1} & P_jP_{j+1}\end{pmatrix} = r_+\,r_-\,\sin(\theta_j).$$

To check that the sign is correct, notice that when $\theta_j \in (0,\pi)$, the oriented vector $P_{j-1}P_j$ rotates counterclockwise over P_jP_{j+1}, hence the discrete curvature is positive, whereas when $\theta_j \in (\pi, 2\pi)$, the rotation happens clockwise, leading to a negative discrete curvature. □

Remark 2. *It should be noted that pixel coordinates are oriented clockwise: the positive y axis goes down instead of up. This means that when we want to preserve the usual trigonometric orientation, we have to take the* opposite *of the right-hand side of* (11).

2.3 Level Lines Smoothing

Level lines extracted from the bilinear interpolation suffer from pixelization. In order to remove oscillations of level lines along transversal directions, short time smoothing is necessary.

2.3.1 Continuous Framework

Among all possible smoothing options, curvature driven motions preserve most of the invariance properties required in image analysis. Therefore, one should either apply mean curvature motion (also known in geometric analysis as curve shortening) or affine morphological scale space (equivalently affine shortening). Roughly speaking, the latter can be interpreted as the PDE evolution

$$\frac{\partial \mathbf{x}}{\partial t} = |\kappa|^{-\frac{2}{3}}\kappa(\mathbf{x}), \tag{13}$$

where the family of time evolving curves $(\Sigma_t)_{t\geq 0}$ is parametrized by arc-length $\mathbf{x}(s,t), s \in [0, L(\Sigma_t)]$, and $\kappa(\mathbf{x})$ is the corresponding vectorial curvature.

2.3.2 Discrete Affine Smoothing

We use the geometric scheme proposed by Moisan [22] to implement (13). The algorithm is based on the fact that equation (13) is numerically defined as an alternate filter, which alternates affine erosion and affine dilation, with a small parameter σ; the latter is approximated by computations of affine erosions of convex and concave components of Σ.

The discrete curve $\Sigma = \{P_1, \ldots, P_n\}$ is cut into convex components by detecting its inflection points $E_i = P_{j(i)}$. Next, the affine erosion of each convex component

$$C_i = \{E_i = P_{j(i)}, P_{j(i)+1}, P_{j(i)+2}, \ldots, P_{j(i+1)} = E_{i+1}\} \tag{14}$$

is the enveloppe of the σ-chords, which is also the set of middle points of σ-chords, with σ a small area parameter. By definition, a σ-chord is a segment joining two points of the curve so that the part

delimited with the curve has area σ. In practice, C_i is a polygonal line and C_i^σ is also a polygonal line obtained by taking the middle points of σ-chords, see Figure 4

$$C_i^\sigma = \{E_i, P^\sigma_{j(i)}, P^\sigma_{j(i)+1}, \dots, E_{i+1}\}. \tag{15}$$

Finally, a new curve is recomposed from the new convex and concave parts.

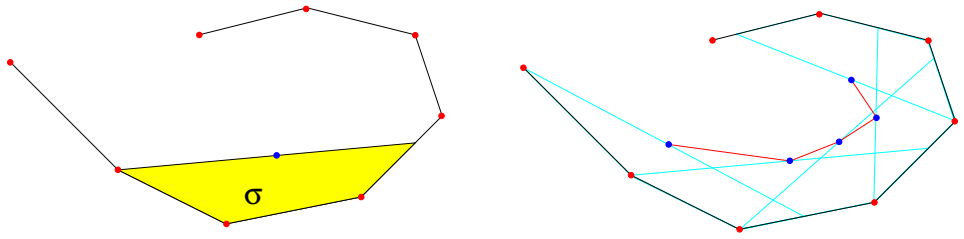


Figure 4: A σ-chord of a component C_i of level line (left) and the resulting polygonal line C_i^σ in red, obtained by taking the middle points of σ-chords (right). Iterating this process is the principle of the curve affine shortening algorithm.

The accuracy of the geometric scheme decreases when σ gets larger, so the process with a small σ is iterated so as to reach the desired scale t of the affine shortening. At each iteration, the evolved components C_i^σ are glued again at the tie points E_i, which have not moved, yielding the polygonal line C^σ. Before input for the next iteration, C^σ is resampled by arc-length. Notice that this geometric scheme commutes with special affine transformations: it relies only on areas and middle points, which are preserved by special affine transformations. More details about the numerical aspects of this scheme are out of the scope of this article and would be the subject of another IPOL publication. Here we focus on the computation of curvature.

3 Algorithmic Pipeline

Most image processing algorithms compute curvatures based on finite difference schemes (FDS) while computing (7), hence curvature depends on the pixel values at corresponding neighborhoods. Consequently, high oscillations along transverse level lines appear, regardless of the resolution of the image, whether prior smoothing has been performed or not. This limitation of FDS for applying curvature motions to images is well known [24].

In fact, when talking about curvatures in a digital image, one actually associates curvatures to the level lines of the image. Therefore, it seems more reasonable to compute curvatures directly on the level lines, instead of a finite grid, by performing a discretization for formula (6). This allows us to compute reliable curvatures, but only after level lines smoothing. The smoothing is necessary since the original level lines suffer from aliasing due to pixelization and to the interpolation itself. This way, the curvatures correspond to our visual perception. But, more fundamentally, *the perception of curvature is and must be multi-scale.* The striking difference between an FDS result and an LLS result is displayed in Figure 2.

3.1 The Algorithm

The algorithm computing the curvature map of any digital image consists of four main steps:

1. *Extraction of bilinear level lines* $\{\Sigma_0^{\lambda,i}\}_{i\in I_\lambda}$. Given a quantization step q, for all levels $\lambda \in (0.5 + q\mathbb{N}) \cap (0, 255)$, level lines are extracted from the bilinear interpolation and discretized as polygonal lines, with uniform sub-pixel sampling. The sampling is performed using (4) and the curve is oriented so as to leave the image gradient at the left hand side. Note that, although

we have to extract lines at half-integer levels to avoid singularities, we need to fix these levels in order to reconstruct back the original image from the family of level lines. This is done by shifting the gray level λ, depending on the level line orientation: decreasing by 0.5 if it is positive and increasing by 0.5 otherwise.

2. *Smoothing of level lines.* Level lines are then smoothed at a low scale t by affine shortening. A new family of level lines $\{\Sigma_t^{\lambda,i}\}_{i\in I_\lambda}$ is obtained, preserving the same orientation and order as the original ones. Notice that for our purpose, the computation of the curvature map, only levels multiples of the quantization step q are used, so we should not need to smooth *all* level lines. The reason for the extraneous calculation is to reconstruct the image smoothed at scale t with the same quantization level as I_0, i.e., 1. This image is shown as background of the curvature map.

3. *Computation of curvatures.* For each curve $\Sigma_t^{\lambda,i} = \{P_j(x_j, y_j)\}_{j=1}^N$ and at each vertex P_j we compute a discrete curvature κ_j, following formula (10), and then list k_j as a curvature belonging to the pixel containing the vertex. As noted earlier, in (10) we actually take the *opposite* of the determinant to account for the anti-trigonometric orientation of pixel coordinates.

4. *Creation of the curvature map.* A curvature image is eventually created by attributing to each pixel a curvature value. In fact, several level lines can pass through a pixel $Q = (i, j) \in \mathcal{R} \times \mathbb{N}^2$ and thus multiple curvatures $\{\kappa_{i_1}, \kappa_{i_2}, \ldots, \kappa_{i_m}\}$ are associated to it. We select as curvature for pixel Q the median value of all the listed curvatures. Finally, for the visualization of the curvature map, we need to saturate the curvature at 1, assuming that a curvature radius greater than 1 is either due to pixelization or to a fast motion during the smoothing process. In the color coding scheme representing the curvatures, this prevents a few extreme values from "flattening" the dynamic range of the values.

The *Image Curvature Microscope* numerical pipeline is summarized in Algorithm 1.

Some results are illustrated in Figure 5. We display the original image I_0 (top-left), its level lines $\Sigma_0^{\lambda,i}$ at levels $\lambda \in 0.5 + 5\mathbb{N}$ (top-center) and the curvature map before smoothing the level lines (top-right), which is practically unreadable, due to pixelization effects. Below we display the smoothed level lines $\Sigma_t^{\lambda,i}$ at levels $\lambda \in 0.5 + 5\mathbb{N}$ (bottom-center), the image reconstructed from the entire family of smoothed level lines (bottom-left) and the curvature map at scale t, superposed over the smoothed image (bottom-right). Note how the curvatures are now conformal to our visual perception.

3.2 Quantitative Evaluation

To measure the accuracy of the curvature estimate, one can perform the following experiment:

- Generate a synthetic image of a black disc with a radius r pixels on white background[2];
- Smooth the level lines at scale t;
- Measure the curvatures, whose ground truth is $1/r$.

Figure 6 shows the box plots of the measurements for various values of r and different smoothing scales t. Given an image of a circle with curvature $\kappa = 1/r$, we display for each scale t a summary of the curvatures measured on the whole image. The median of the measured curvatures is quite accurate when the smoothing scale is $t = 2$, but their variation is more important than at larger scales.

[2]The image is generated using ImageMagick (http://imagemagick.org/) with the command:
`convert -size 512x512 xc:white -fill black -stroke black -draw "circle 256,256 256,356" circ.png`

Algorithm 1: The *Image Curvature Microscope* pipeline

Input: Original image $I_0 = (i,j)_{i,j=1}^{w,h}$, quantization step q, scale t.

Output: Curvatures of I_0 at scale t: $K(\cdot,t) = (\kappa(i,j))_{i,j=1}^{w,h}$.

Enlarge the image I_0 by 20 pixels on each side, filling with mirror effect
Put constant gray value 0 on the boundary of enlarged image
foreach *level* $\lambda \in (0.5 + q\mathbb{N}) \cap [0, 255]$ **do**
    extract the bilinear level lines $\{\Sigma_0^{\lambda,i}\}_{i \in I_\lambda}$
    sample uniformly each polygonal level line $\Sigma_0^{\lambda,i}$ while preserving its orientation
    shift by ± 0.5 the level λ of each $\Sigma_0^{\lambda,i}$ depending on orientation of the curve
end
foreach *level line* $\Sigma_0^{\lambda,i}$ **do**
    smooth the level line $\Sigma_0^{\lambda,i}$ up to scale t and get a curve $\Sigma_t^{\lambda,i}$ with the same orientation
end
foreach *level line* $\Sigma_t^{\lambda,i} = \{P_j(x_j, y_j)\}_{j=0\ldots N}$ *with* $N \geq 10$, $\lambda \in 0.5 + q\mathbb{N}$ **do**
    compute the opposite of the discrete curvature κ_j at each vertex P_j by (10)
end
foreach *pixel* $Q = (i,j) \in \mathcal{R} \cap \mathbb{N}^2$ **do**
    register the median value of all discrete curvatures inside $[i, i+1] \times [j, j+1]$: $\kappa(i,j) = \max(-1, \min(1, \text{median}(\kappa_{i_1}, \kappa_{i_2}, \ldots, \kappa_{i_m})))$.
end

However, at larger scales, even if the variation is less important, there is a bias, more noticeable at large curvatures (0.1), which can be easily explained: the motion of the level lines increases with their curvature during the smoothing process, thus a perfect circle moves faster as its radius diminishes. Therefore the measured radius after smoothing is more homogeneous, but we are measuring the curvature of a modified level line. Of course this bias can be corrected, as one disposes of explicit functions $R(0) \to R(t)$ associating an initial radius $R(0)$ to its affine curvature evolution at time t. Thus, once observed $R(t)$ and knowing t, one can infer $R(0)$. This correction is only perfect when the initial curve is a circle.

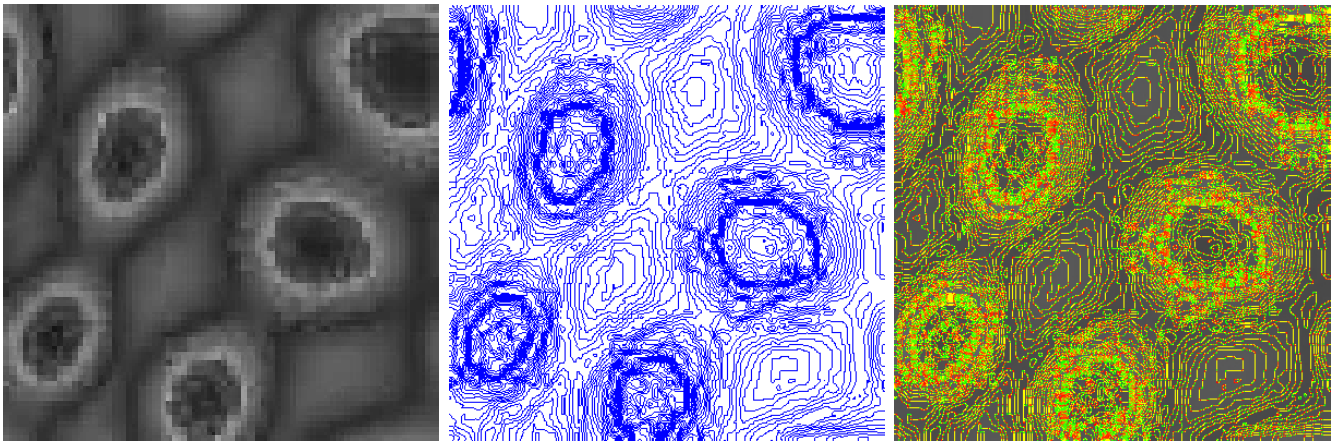
3.3 Curvature Near Saddle Points

As pointed out earlier in the paper, not all level lines are Jordan curves. In particular, level lines corresponding to saddle points (x_s, y_s) consist, locally around the saddle point, of two orthogonal straight lines crossing at (x_s, y_s). To better understand how curvatures are computed, we consider the following example. Let I be an image corresponding to the graph of the function $f(x,y) = x^2 - y^2$, as displayed in Figure 7. In this case, the level lines of the image coincide with the level lines of the function itself. The saddle point is the origin $(0,0)$ and the level lines corresponding to the level 0 are the two diagonals. Since they present sharp angles at the origin, qualitatively the curvature is infinite there. Indeed, this is confirmed by applying (7) in a neighborhood of the origin

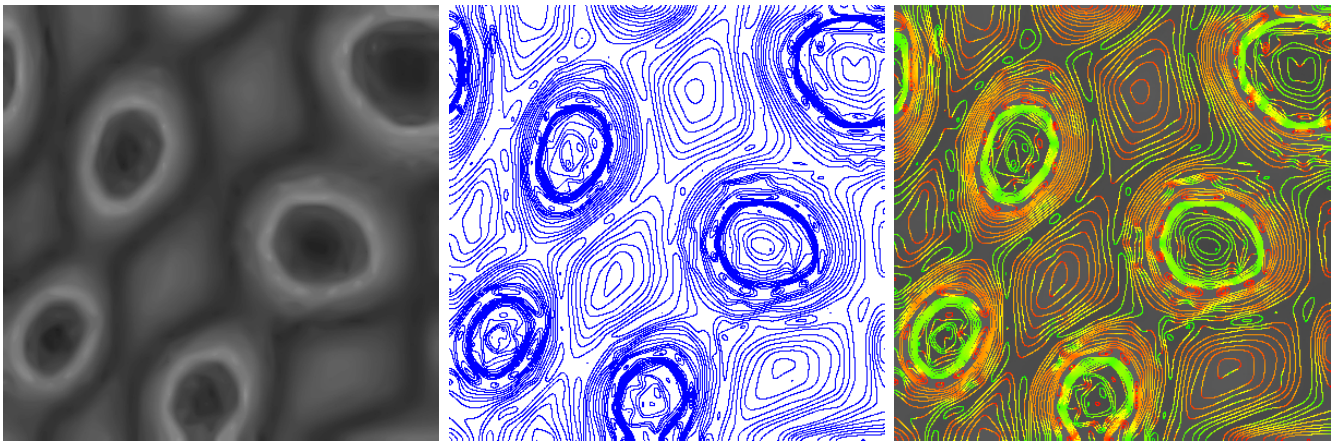
$$f_x = 2x, \qquad f_y = -2y, \qquad f_{xx} = 2, \qquad f_{xy} = 0, \qquad f_{yy} = -2,$$

so that

$$\text{curv}(f)(x,y) = \frac{y^2 - x^2}{(y^2 + x^2)^{3/2}}.$$



(a) Original image (left), its corresponding level lines (center) and curvature map before smoothing (right).



(b) Smoothed image (left) reconstructed from independently smoothed level lines (center) and curvature map after smoothing, computed directly on level lines (right).

Figure 5: The *Image Curvature Microscope* numerical pipeline: from the original image (top-left), the family of all its level lines is extracted at quantized non-degenerate levels (top-center). Each level line is evolved independently (bottom-center) and the curvature map is computed directly on the smoothed level lines (bottom-right). The top-right image shows the confuse and noisy result we obtain, without smoothing the level lines. The bottom-left image is the image reconstructed from the smoothed level lines.

This ratio has no limit when (x, y) goes to $(0, 0)$, as witnessed by looking at particular directions $y = kx$, $k \in \mathbb{R}$

$$\mathrm{curv}(f)(x, kx) = \frac{k^2 - 1}{(k^2 + 1)^{3/2}} \frac{1}{|x|}.$$

When x goes to 0, this expression stays constant and converges to 0 if $k = \pm 1$, goes to $+\infty$ if $|k| > 1$ and to $-\infty$ if $|k| < 1$. Indeed, while running the algorithm on the image I generated by f, one observes in Figure 7 that curvatures computed around the origin have higher and higher values (in absolute value) on each side of the asymptotes, colliding at the origin.

4 Examples

The image curvature microscope is exemplified below on several categories of images. It is a complex visualization tool, displaying the level lines of an input image before and after smoothing, as well as the curvature map printed over the smoothed image. The visualization depends on three parameters: the zoom factor, the quantization step q for the level lines, and the smoothing scale t. These tuning

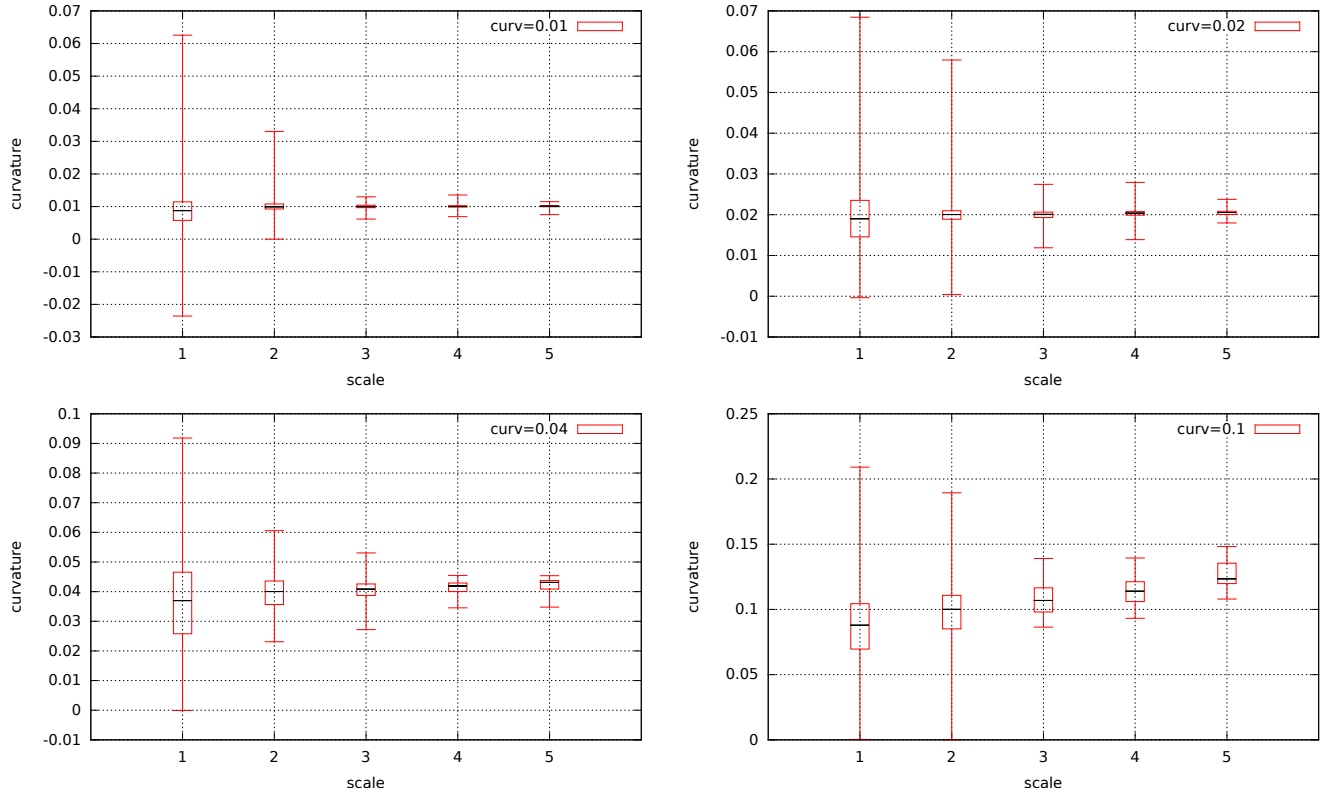


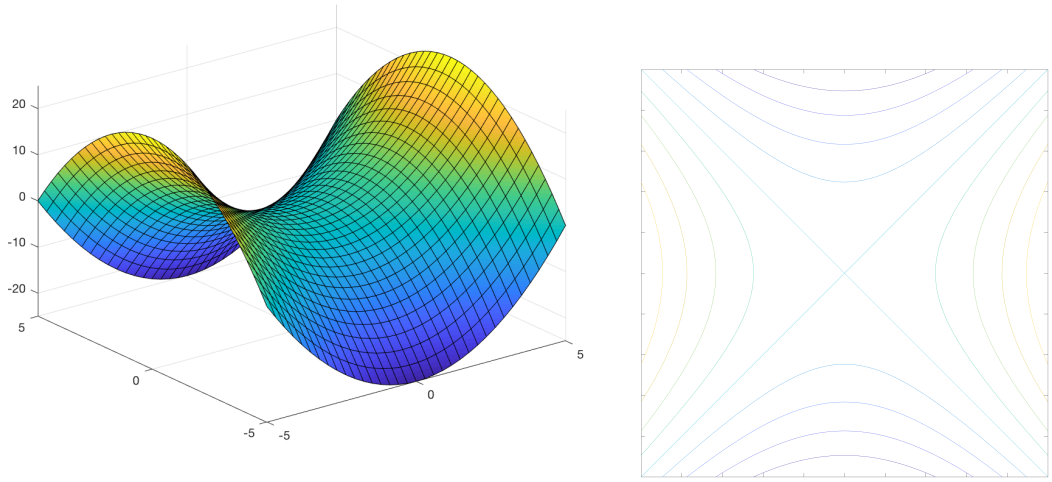
Figure 6: Statitics of measured curvatures on synthetic images of a disk for various ground truth values $\kappa = 1/r$, as a function of the smoothing scale. From left to right and top to bottom: $\kappa = 0.01, 0.02, 0.04, 0.1$ ($r = 100, 50, 25, 10$ pixels). Each box shows the minimum, maximum, first and third quartiles, and the median value of the curvatures measured on the whole image.

parameters vary according to the total variation of the image and should be chosen by the user accordingly. The hue encodes the curvature, with yellow meaning zero curvature, and shades of green and red meaning positive and negatives curvatures respectively. In the experiments, color images are first converted to grayscale intensity values before extraction of the level lines.

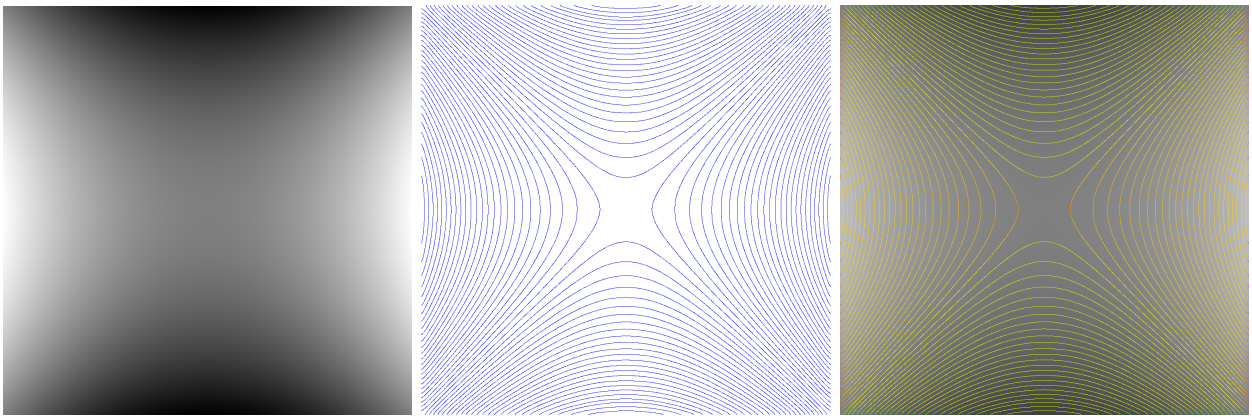
4.1 Binary Images

We first compute the curvature map on binary images. This type of images presents strong aliasing effects on slanted lines. In order to observe the curvatures, one has to choose a rather high smoothing scale to avoid measuring spurious curvatures. The drawback is that some parts with high curvature (see the base of the cat's ear in Figure 8) evolve too much. The good point is that the cat's upper back is classified as straight, as it should. For a drawing with regular curves, we can recover successfully the curvatures (Figure 9), but again the necessary smoothing modifies some small and highly curved details, like the eyes and the eyebrow.

The effect of the smoothing scale on a binary image is demonstrated in Figure 10. This is a binary image where each pixel was zoomed in by a factor 4×4. Without smoothing, horizontal and vertical portions of the level lines get curvature 0, while strong curvatures due to pixelization are detected. This effect remains somewhat visible at scale $t = 4$, but is no longer apparent at scale $t = 6$. Notice the smooth consistent curvatures along the curves.



(a) Graph of the function $f(x, y) = x^2 - y^2$ and its level lines.



(b) Image corresponding to the graph of the function above (left), its smoothed level lines at quantized levels $0.5 + 5\mathbb{N}$ (middle) and curvature map.

Figure 7: Curvature computation around the saddle point: curvatures have higher and higher values, colliding as they approach the origin. For visualization only, curvature values are thresholded at ± 1.

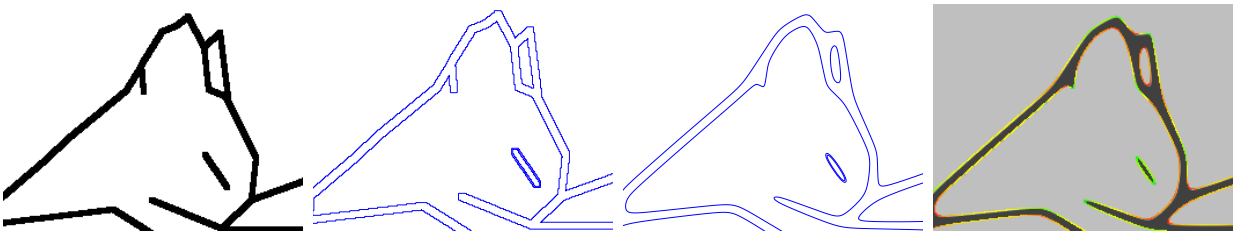


Figure 8: Detail of Attneave's cat (a binary image, see Figure 1), level lines (at quantization $q = 200$) before and after smoothing (scale $t = 5$) and curvature map.

4.1.1 Julesz's Textons

The famous textures of B. Julesz [18] are built so as to be difficult to discriminate, because they have the same density of space orientations, corners, terminators, and bar lengths. This remains true also for curvatures in Figure 11, where shapes `S` and `10` cannot be discriminated pre-attentively.

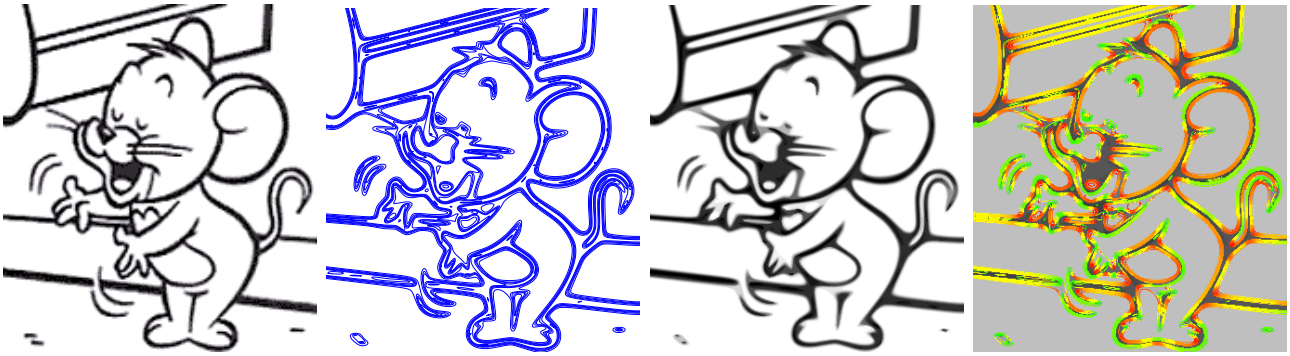


Figure 9: A drawing, its smoothed level lines (quantization $q = 50$), smoothed image (scale $t = 2$), and the curvature map.

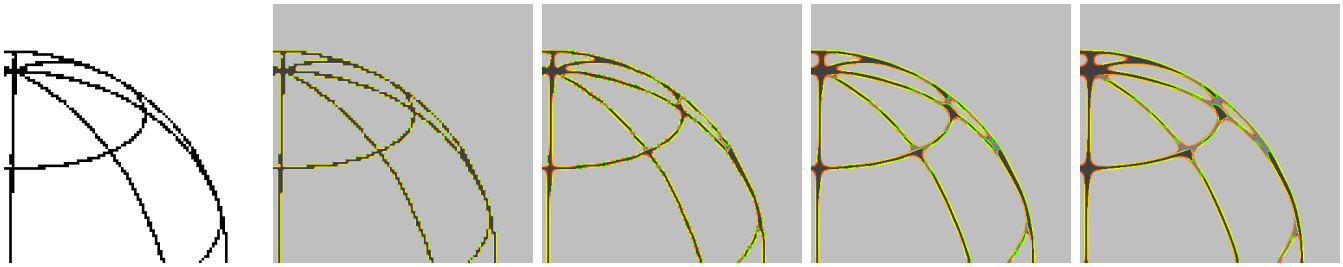


Figure 10: A $\times 4$ zoomed-in binary image, and curvature maps (quantization $q = 50$) at different scales: $t = 0, 2, 4, 6$.

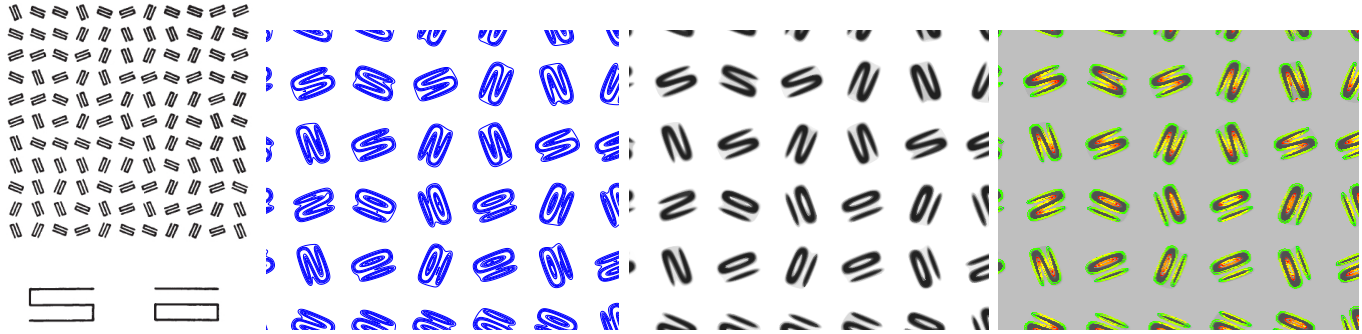


Figure 11: A texton picture by Julesz: original image, part of its smoothed level lines (quantization $q = 50$, scale $t = 2$), recontructed image and curvatures.

4.1.2 Textures With Regular Patterns

Noise on a regular pattern (Figure 12) is gracefully handled: curvature on repeated features is the same. Notice that a few level lines are on their way to early disparition through affine shortening at a nearby scale. One is too short to give rise to curvature while the other still has the required 10 vertices to be considered, creating high curvatures.

4.1.3 Fingerprints

The curvature map of a scanned fingerprint (Figure 13) could be of particular interest for minutiae detection. Notice still that low contrast edges in the black structures can create long level lines that do not disappear through affine shortening, and are visible in the curvature map. An easy fix for this problem is to put a threshold corresponding to the minimal number of lines passing through a pixel in order to display a curvature: it is implicitely set to 1 in the code, meaning that a curvature is computed at each pixel through which a level line passes. Increasing this threshold amounts to ignore low contrast level lines.

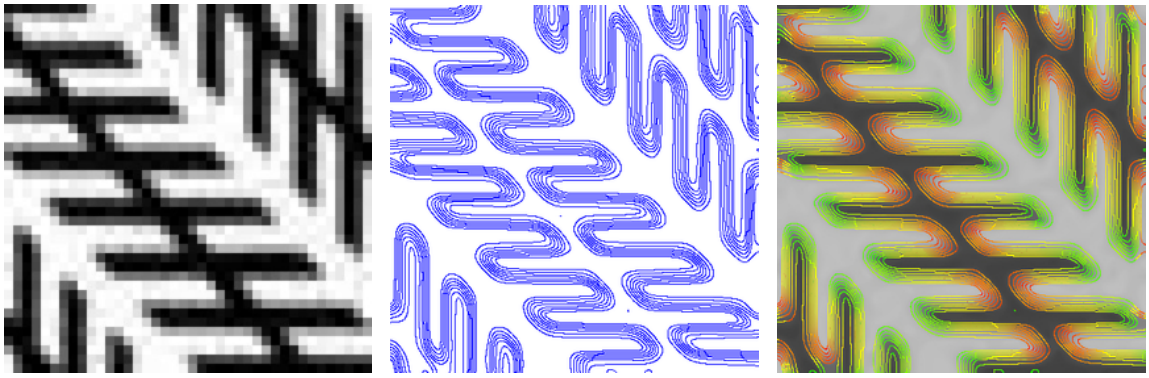


Figure 12: Texture image, smoothed level lines and curvature map.

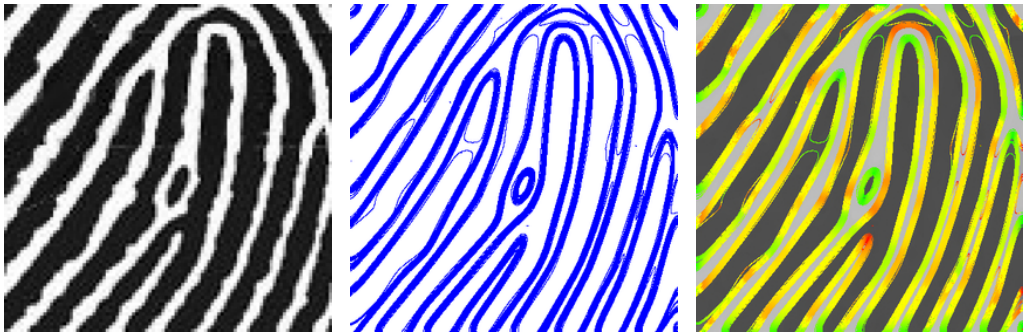


Figure 13: Original image of fingerprint, smoothed level lines and curvature map.

4.2 Paintings

Experiments on photographed paintings are illustrated below. Notice the elaborate shadows around the eyes and lips of Mona Lisa in Figure 14, as well as the subtle transition from the cheek and forefront to the hair, typical of Leonardo's famous *sfumato* technique. This is in contrast to the strong edges in an earlier portrait by van der Weyden.

4.3 Nature Photographs

The complex level lines of carpels and stamens of a flower can be significantly simplified by the smoothing process, as witnessed by Figure 15.

Figure 16 shows that sufficient smoothing in the curvature microscope can give good results for the curvature map despite the bad quality input image, due to excessive JPEG compression. In particular, note the constant curvature in green on the round head top of the bird.

In Figure 17, the slightly contrasted ridges of the finger are not detected as full level lines because of the interplay with the light reflexions. However, this does not prevent from computing correctly the curvatures on the insect.

4.4 Medical Imaging

Basic morphological differences between bacteria rely on forms and their associations. The shapes are determined by the bacterial cell wall and cytoskeleton. This provides information about the ability of bacteria to acquire nutrients, attach to surfaces or swim through liquids. The curvature is an intrinsic geometrical descriptor, useful for shape discrimination. Figure 18 displays bacterial morphologies and the corresponding curvature map. Bacteria porosities are characterized by strong curvature oscillations, whereas the borders of bacterial shapes present smooth curvature variations.

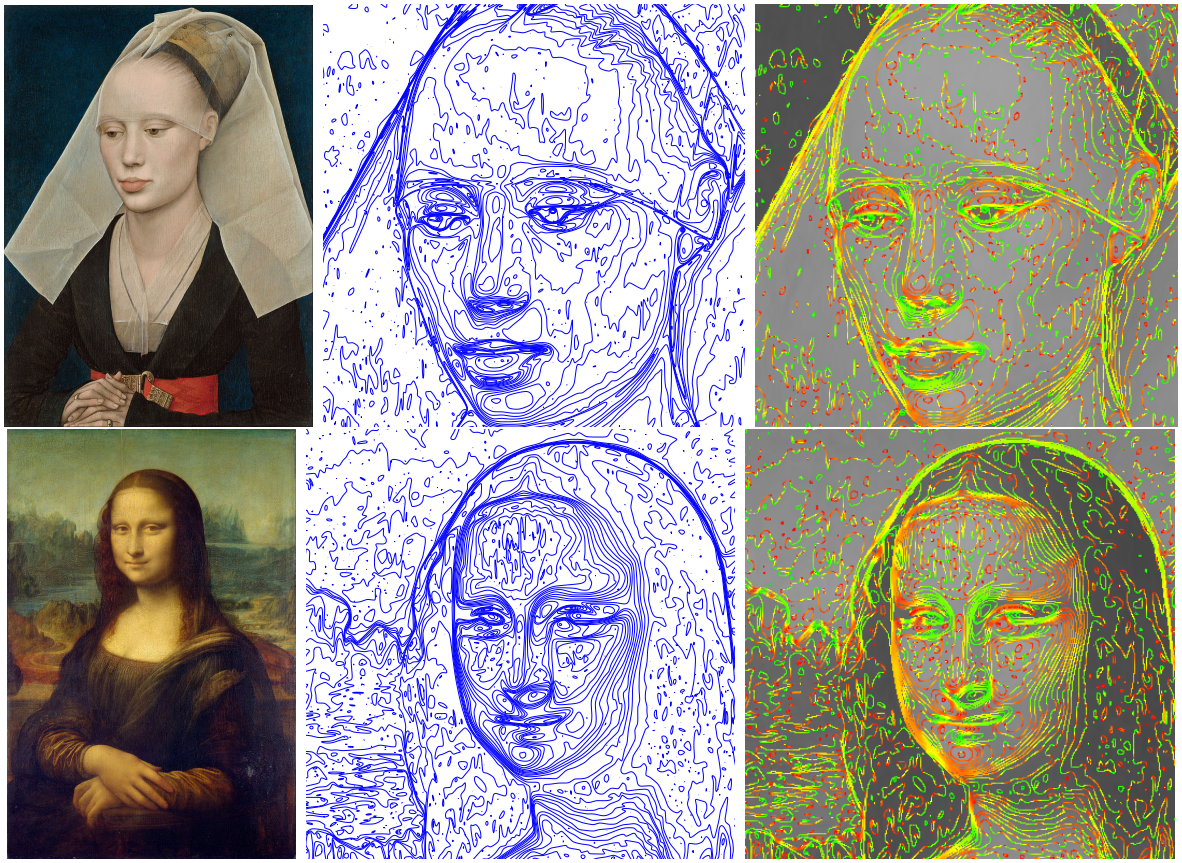


Figure 14: Experiments on paintings: Rogier van der Weyden's painting *Portrait of a lady* (c. 1460) and Leonardo da Vinci's *Mona Lisa* (1503). The level lines are displayed with a quantization $q = 10$ and smoothed at scale $t = 3$.

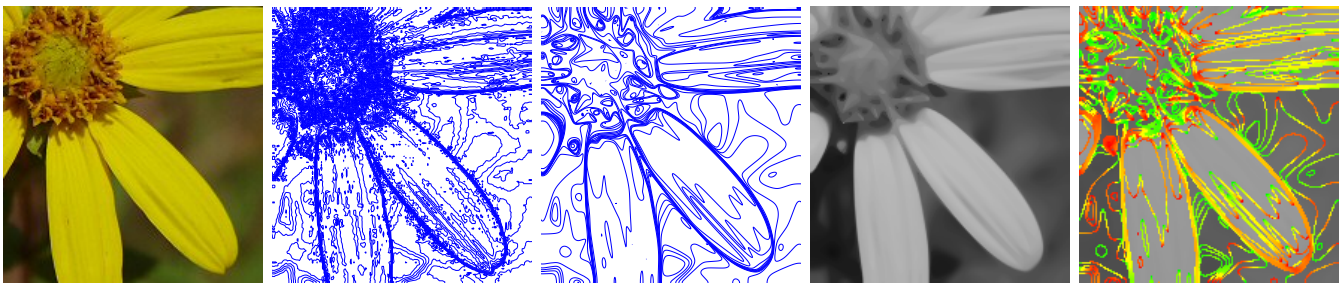


Figure 15: Original image, initial level lines (quantization $q = 10$) and smoothed level lines (scale $t = 3$), smoothed image, and curvature map.

Another example of a microscope image of bacteria is Figure 19. Notice how a little amount of smoothing can homogeneize textures.

4.5 Topography

Finally, we return to the *origin* of level lines with a topographic image. The gray level is proportional to the elevation, with lighter shades indicating higher terrain. We can recover the level lines as in a topographic map and compute their curvature, as in Figure 20.

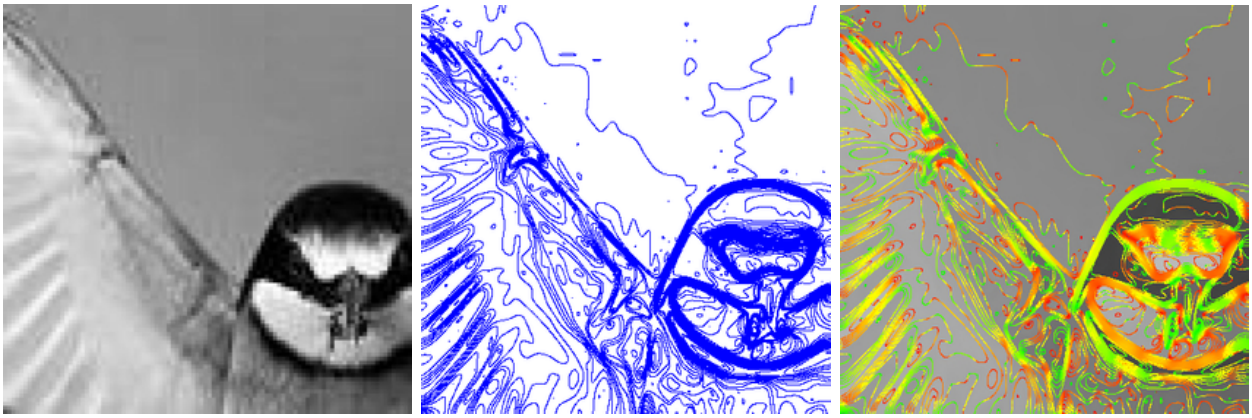


Figure 16: Original image (notice the bad quality due to image compression), smoothed level lines and curvature map.

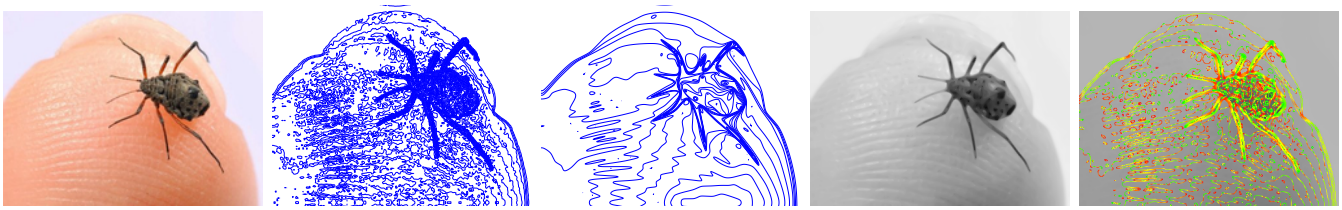


Figure 17: Original image, initial and smoothed level lines (quantization $q = 10$, scale $t = 1$), smoothed image, and curvature map.

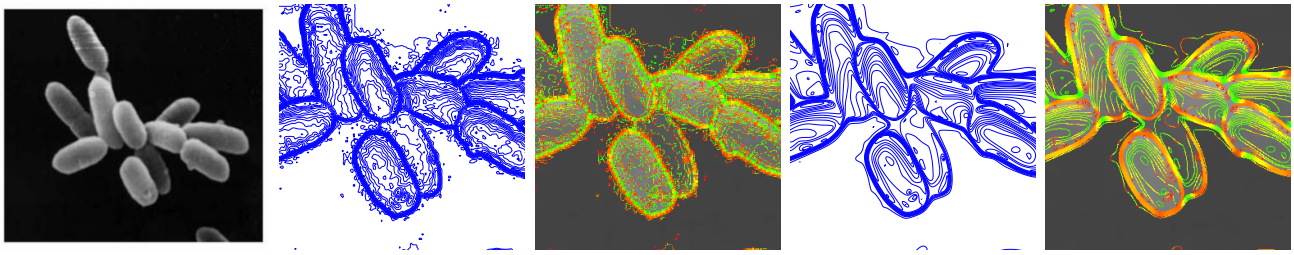


Figure 18: Bacteria image, initial level lines (quantization $q = 10$) and curvature map without smoothing. The same after smoothing at scale $t = 2$.

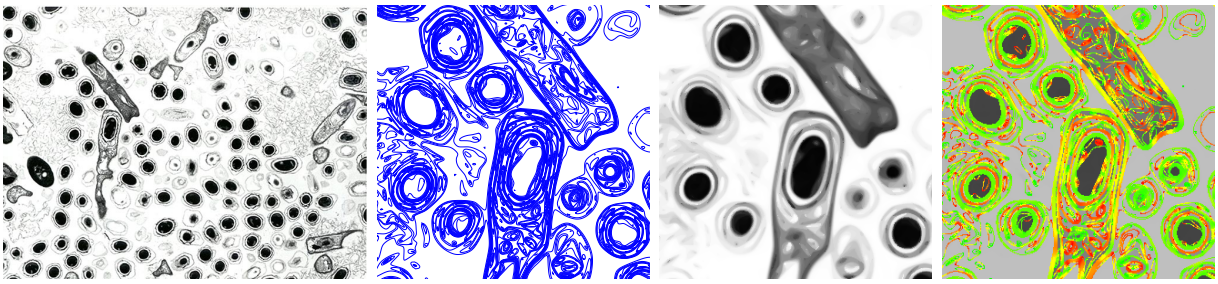


Figure 19: Bacteria microscope image, smoothed level lines (quantization $q = 10$), smoothed image (scale $t = 2$), and curvature map.

5 Conclusion

We presented a method to compute the curvature of an image at any user-chosen scale, hence the name "curvature microscope". The demonstration associated to this article uses a $\times 2$ zoom factor for the resulting bitmap image, sufficient for visualization, but the code can support any zoom factor. Despite the indirect, and at first view overly complex, scheme to compute the curvature, via the level lines and their affine erosion, the algorithm runs at comfortable speed, as witnessed by the online

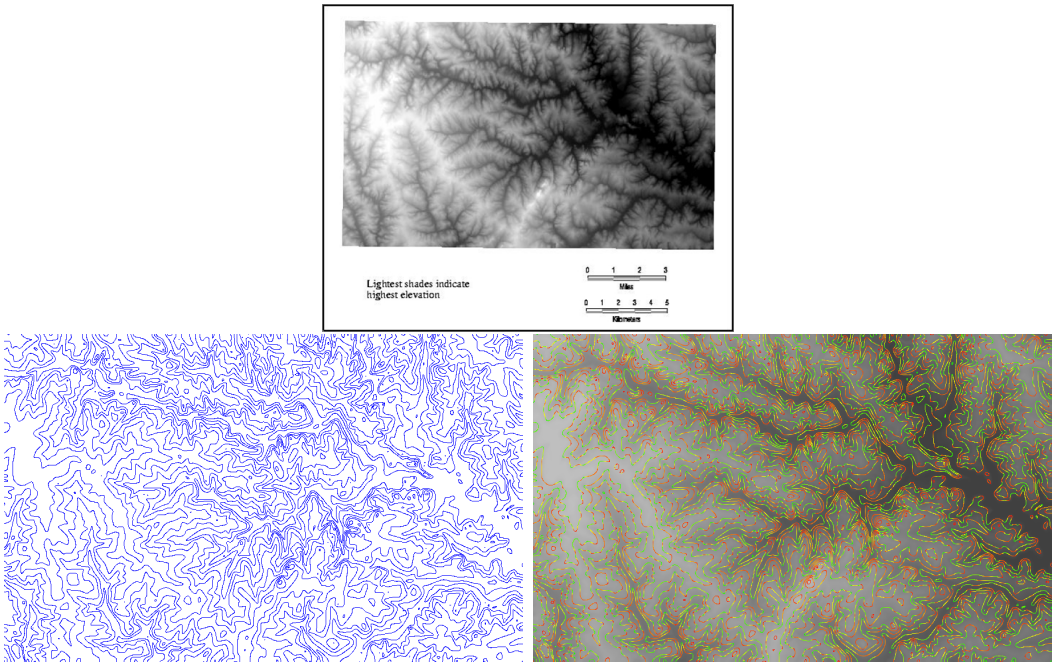


Figure 20: A topographic image, smoothed level lines ($q = 20$, $t = 1$), and curvature map.

demonstration. This is due to the fact that the smoothing step, the slower part of the pipeline, is computed in parallel. This complexity is somewhat unavoidable to obtain an accurate evaluation of the curvature.

Image Credits

Images by the authors except:

 ©American Psychological Association [4]
 ©Best Coloring Pages for Kids
 ©Nature Publishing Group
 public domain[3]
 public domain[4]
 ©Mobile Toones
 ©Rothamsted Research Center
 public domain[5]
 ©Getty Images
 ©Blaise Mibeck

References

[1] L. Alvarez, F. Guichard, P.-L. Lions, and J.-M. Morel, *Axioms and fundamental equations of image processing*, Arch. Rational Mech. Anal., 123 (1993), pp. 199–257. http://dx.doi.org/10.1007/BF00375127.

[2] L. Alvarez and J.-M. Morel, *Formalization and computational aspects of image analysis*, Acta Numerica, 3 (1994), pp. 1–59. http://dx.doi.org/10.1017/S0962492900002415.

[3] https://en.wikipedia.org/wiki/Rogier_van_der_Weyden#/media/File:Rogier_van_der_Weyden_-_Portrait_of_a_Lady_-_Google_Art_Project.jpg

[4] https://en.wikipedia.org/wiki/Mona_Lisa#/media/File:Mona_Lisa,_by_Leonardo_da_Vinci,_from_C2RMF_retouched.jpg

[5] https://en.wikipedia.org/wiki/Archaea#/media/File:Halobacteria.jpg

[3] H. Asada and M. Brady, *The curvature primal sketch*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 8 (1986), pp. 2–14. http://dx.doi.org/10.1109/TPAMI.1986.4767747.

[4] F. Attneave, *Some informational aspects of visual perception.*, Psychological Review, 61 (1954), pp. 183–193. http://psycnet.apa.org/doi/10.1037/h0054663.

[5] C. Ballester, V. Caselles, and P. Monasse, *The tree of shapes of an image*, ESAIM: Control, Optimisation and Calculus of Variations, 9 (2003), pp. 1–18. http://dx.doi.org/10.1051/cocv:2002069.

[6] F. Cao and L. Moisan, *Geometric computation of curvature driven plane curve evolutions*, SIAM Journal on Numerical Analysis, 39 (2002), pp. 624–646. http://dx.doi.org/10.1137/S0036142999363863.

[7] V. Caselles, B. Coll, and J.-M. Morel, *A Kanizsa programme*, in Variational Methods for Discontinuous Structures, Springer, 1994, pp. 35–55. http://dx.doi.org/10.1007/978-3-0348-9244-5_4.

[8] V. Caselles, E. Meinhardt, and P. Monasse, *Constructing the tree of shapes of an image by fusion of the trees of connected components of upper and lower level sets*, Positivity, Birkhäuser, 12 (2008), pp. 55–73. http://dx.doi.org/10.1007/s11117-007-2150-2.

[9] V. Caselles and P. Monasse, *Geometric Description of Images as Topographic Maps*, Springer, 2010. ISBN: 978-3-642-04610-0. http://dx.doi.org/10.1007/978-3-642-04611-7.

[10] Y.G. Chen, Y. Giga, and S. Goto, *Uniqueness and existence of viscosity solutions of generalized mean curvature flow equations*, Journal of Differential Geometry, 33 (1991), pp. 749–786.

[11] A. Ciomaga, P. Monasse, and J.-M. Morel, *Level lines shortening yields an image curvature microscope*, in Proceedings of ICIP 2010, September 26-29, Hong Kong, China, 2010, pp. 4129–4132. http://dx.doi.org/10.1109/ICIP.2010.5649850.

[12] Adina Ciomaga and Jean-Michel Morel, *A proof of equivalence between level lines shortening and curvature motion in image processing*, SIAM J. Math. Anal., 45 (2013), pp. 1047–1067. http://dx.doi.org/10.1137/11082347X.

[13] M.G. Crandall and P.-L. Lions, *Convergent difference schemes for nonlinear parabolic equations and mean curvature motion*, Numer. Math., 75 (1996), pp. 17–41. http://dx.doi.org/10.1007/s002110050228.

[14] L. C. Evans and J. Spruck, *Motion of level sets by mean curvature. I*, Journal of Differential Geometry, 33 (1991), pp. 635–681.

[15] M. Gage and R. S. Hamilton, *The heat equation shrinking convex plane curves*, Journal of Differential Geometry, 23 (1986), pp. 69–96.

[16] M. A. Grayson, *The heat equation shrinks embedded plane curves to round points*, Journal of Differential Geometry, 26 (1987), pp. 285–314.

[17] F. Guichard, J.-M. Morel, and R. Ryan, *Contrast invariant image analysis and PDEs*, preprint, 2001.

[18] B. Julesz, *Textons, the elements of texture perception, and their interactions*, Nature, 290 (1981), pp. 91–97. http://dx.doi.org/10.1038/290091a0.

[19] D. Marr, *The low-level symbolic representation of intensity changes in an image*, tech. report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1974.

[20] K. Mikula and D. Ševčovič, *Evolution of plane curves driven by a nonlinear function of curvature and anisotropy*, SIAM Journal on Applied Mathematics, 61 (2001), pp. 1473–1501 (electronic). http://dx.doi.org/10.1137/S0036139999359288.

[21] F. Mockhtarian and A. Mackworth, *A theory of multiscale, curvature-based shape representation for planar curves*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 14 (1992), pp. 789–805. http://dx.doi.org/10.1109/34.149591.

[22] L. Moisan, *Affine plane curve evolution: a fully consistent scheme*, IEEE Transactions on Image Processing, 7 (1998), pp. 411–420. http://dx.doi.org/10.1109/83.661191.

[23] P. Monasse and F. Guichard, *Fast computation of a contrast invariant image representation*, IEEE Transactions on Image Processing, 9 (2000), pp. 860–872. http://dx.doi.org/10.1109/83.841532.

[24] M. Mondelli and A. Ciomaga, *Finite difference schemes for MCM and AMSS*, Image Processing On Line, 1 (2011). http://dx.doi.org/10.5201/ipol.2011.cm_fds.

[25] S. Osher and J. A. Sethian, *Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations*, Journal of Computational Physics, 79 (1988), pp. 12–49. http://dx.doi.org/10.1016/0021-9991(88)90002-2.

[26] G. Sapiro and A. Tannenbaum, *Affine invariant scale space*, International Journal of Computer Vision, 11 (1993), pp. 25–44. http://dx.doi.org/10.1007/BF01420591.

[27] A. Witkin, *Scale-space filtering: A new approach to multi-scale description*, Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing, 9 (1984), pp. 1019–1021. http://dx.doi.org/10.1109/ICASSP.1984.1172729.