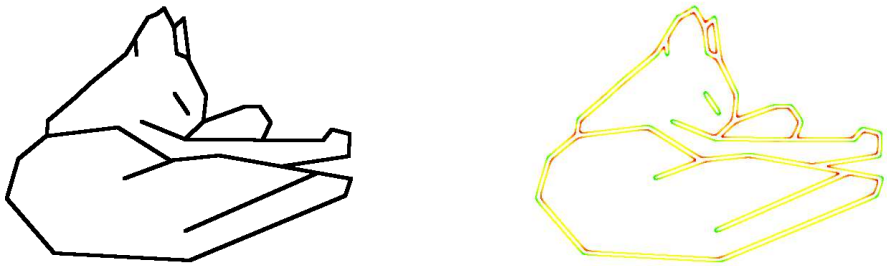
Image Curvature Microscope

MVA Lecture Notes

Adina Ciomaga, Lionel Moisan, Pascal Monasse,
Jean-Michel Morel

Introduction

Attneave's founding 1954 paper [**3**] on image perception anticipated the numerical analysis of digital pictures. He stated that in images: *"information is concentrated along contours (i.e., regions where color changes abruptly), and is further concentrated at those points on a contour at which its direction changes most rapidly (i.e., at angles or peaks of curvature)".*



(a). Attneave's cat (b). Curvature map.

FIGURE 1. Attneave's figure illustrating the prominent role of curvature peaks in image perception and its curvature map.

Shape analysis can be led to the study of Jordan curves which we shall call "elementary shapes". The many experiments where we display level lines of digital images make clear enough why a smoothing is necessary to restore their structure. These experiments also show that we can in no way assimilate these level lines with our common notion of shape as the silhouette of a physical object in full view. Indeed, in images of a natural environment, most observed objects are partially hidden (occluded) by other objects and often deformed by perspective. When we observe a level line we cannot be sure that it belongs to a single object; it may be composed of pieces of the boundaries of several objects that are occluding each other. Shape recognition technology has therefore focused on local methods, that is, methods that work even if a shape is not in full view or if the visible part is distorted. As a consequence, image analysis adopts the following principle: *Shape recognition must be based on local features of the shape's boundary, in this case local features of the Jordan curve, and not on its global features. If the boundary has some degree of smoothness, then these local features are based on the derivatives of the curve, namely the tangent vector, the curvature, and so on.* Many local recognition methods involve the "salient" points of a shape,

which are the points where the curvature is zero (inflection points) and points where the curvature has a maximum or minimum (the "corners" of the shape).

Yet, because of noise and aliasing effects, the direct computation of curvatures on a raw image is impossible and depends anyway on a smoothing scale. The fragment of scanned map in Figure 2 is exemplary, in its amount of ringing, aliasing, and JPEG artifacts. Such graphic images are satisfactorily restored with short time smoothing.

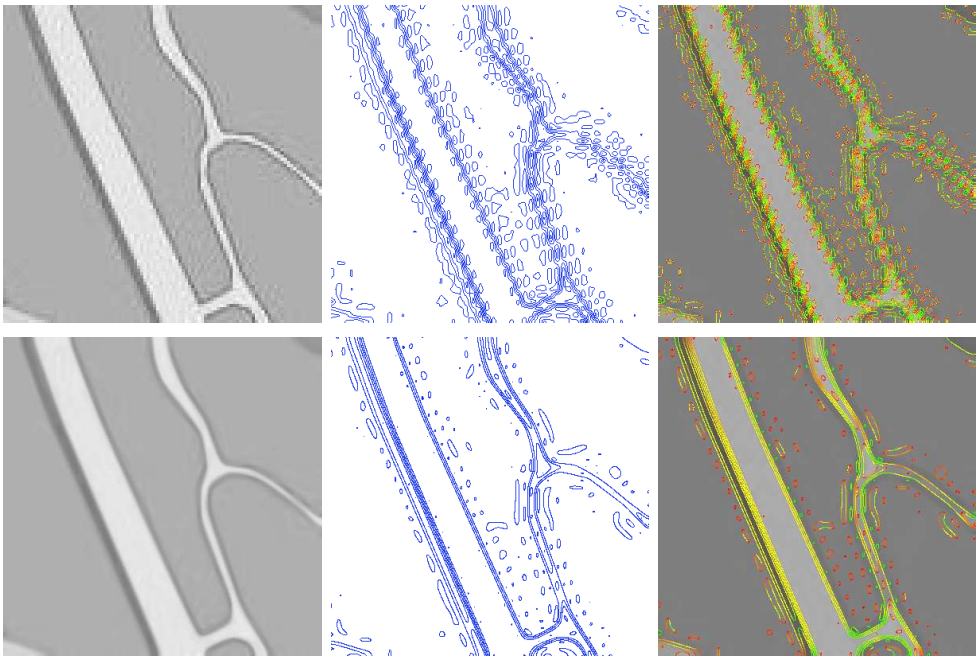


FIGURE 2. Top: Piece of map with roads, its corresponding level lines and curvatures. Bottom: smoothed image, smoothed level lines and curvature map after filtering.

This explains why, in one of the first serious attempts to cope with this numerical challenge, Asada and Brady [**2**] introduced the concept of *multiscale curvature*. They suggested to approximate contours by splines and to smooth them by a 1D heat equation. Their explicit goal was to implement Attneave's idea that shapes must be represented by curvature extrema. This paper led to increasingly sophisticated attempts to analyze planar shapes by their curvatures. A first difficulty is that, at fine scale, contours have high curvatures everywhere. Another problematic issue is the extraction of the contours on which the curvature could be computed. Contours obtained by "edge detection" are broken and plagued with spurious branches, which hinder the computation of any reliable curvature.

Clarifying the subject has required a fairly elaborate series of mathematical contributions. Grayson [**11**] proved that the intrinsic heat equation smooths Jordan curves and preserves their topology. The Osher-Sethian level set method [**19**] implements the motion by mean curvature of an embedded manifold by

applying the mean curvature PDE to its signed distance function. Evans-Spruck [**8**] and Chen-Giga-Goto [**7**] elaborated a viscosity solution theory for the scalar mean curvature motion. A mathematical link between the median filter and the motion by mean curvature was conjectured by Merriman, Bence and Osher [**16**] and later proved by several authors [**4**], [**9**], [**12**].

In parallel, Mackworth and Moktharian [**15**] proposed a fast numerical scheme to smooth a curve by the intrinsic heat equation. But their shape extraction algorithm was unconvincing. Caselles et al. realized the potential of using directly the image level lines instead of its edges. They proposed to perform contrast invariant image analysis directly on the set of level lines, or *topographic map* [**5**]. A fast algorithm computing the topographic map was developed by Monasse and Guichard in [**18**]. Sapiro and Tannenbaum [**21**] discovered the affine curve shortening and Alvarez et al. [**1**] the affine invariant and contrast invariant image smoothing. A remarkably fast and simple geometric algorithm for affine shortening was given by Moisan in [**17**].

The present work builds on the above mentioned contributions and it describes a complete image processing numerical chain. The chain starts from a digital image, proceeds to the level lines extraction and to their independent evolution by curve shortening or affine shortening. The chain ends up with an accurate visualization tool of image curvatures computed on the smoothed level lines. This, hopefully, advances Attneave's program and yields what we shall term a *curvature image microscope*. Indeed, the evolved level lines and image are not defined on the initial grid. The level lines have floating coordinates and the image can be reconstructed from them at any precision.

There is something slightly paradoxical in smoothing an image to see it better. Nevertheless, noise, JPEG artifacts, and aliasing (pixelization effects) will be shown to be nicely smoothed out by the subpixel curvature motion. As anticipated by Attneave, the level line evolution eliminates the erratic curvatures and yields a curvature more conform to our multiscale contour perception. Finally the level line visualization (after smoothing) reveals many hidden image details which can be zoomed in, thanks to the grid independent representation of the image by its level lines. The resulting algorithm is fast and can be tested on line[1].

[1]http://www.ipol.im/pub/algo/cmmm_image_curvature_microscope/

CHAPTER 1

Curves and Curvature Flows

This chapter contains the fundamentals of differential geometry that are used in the lecture notes. Our main aim is to define the curvature of a curve as the main contrast invariant differential operator we shall deal with in image and curve smoothing.

1. Tangent, normal, and curvature

We summarize in this section the concepts and results about smooth curves that are needed in this chapter and elsewhere in the notes. The curves we considered will always be plane curves.

Definition 1.1. *We call simple arc or Jordan arc the image* Γ *of a continuous one-to-one function* $\mathbf{x} : [0,1] \to \mathbb{R}^2$, $\mathbf{x}(t) = (x(t), y(t))$. *We say that* Γ *is a simple closed curve or Jordan curve if the mapping restricted to* $(0,1)$ *is one-to-one and if* $\mathbf{x}(0) = \mathbf{x}(1)$. *If* $\mathbf{x}$ *is continuously differentiable on* $[0,1]$, *we define the arc length of the segment of the curve between* $\mathbf{x}(t_0)$ *and* $\mathbf{x}(t)$ *by*

$$L(\mathbf{x}, t_0, t) = \int_{t_0}^{t} |\mathbf{x}'(\tau)| \, \mathrm{d}\tau = \int_{t_0}^{t} \sqrt{\mathbf{x}'(\tau) \cdot \mathbf{x}'(\tau)} \, \mathrm{d}\tau. \tag{1.1}$$

In particular, set

$$L(t) = L(\mathbf{x}, 0, t) = \int_{0}^{t} |\mathbf{x}'(\tau)| \, \mathrm{d}\tau = \int_{0}^{t} \sqrt{\mathbf{x}'(\tau) \cdot \mathbf{x}'(\tau)} \, \mathrm{d}\tau.$$

The curves we deal with will always be smooth. Now, we want the definition of "smoothness" to describe an intrinsic property of Γ rather than a property of some parameterization $\mathbf{x}(s)$ of Γ. If a function $\mathbf{x}$ representing Γ is C^1, then the function L in equation (1.1) has a derivative with respect to s,

$$L'(t) = |\mathbf{x}'(t)|$$

that is continuous. Nevertheless, the curve itself may not conform to our idea of being smooth, which at a minimum requires a tangent at every point $\mathbf{y} \in \Gamma$. For example, the motion of a point on the boundary of a unit disk as it rolls along the x-axis is described by $\mathbf{x}(t) = (t - \sin t, 1 - \cos t)$, which is a C^∞ function. Nevertheless, the curve has cusps at all multiples of 2π. The problem is that $\mathbf{x}'(2k\pi) = 0$.

Definition 1.2. *We say that a curve* Γ *admits an arc-length parameterization* $s \mapsto \mathbf{x}(s)$ *if the function* $\mathbf{x}$ *is* C^1 *and* $L'(s) = |\mathbf{x}'(s)| = 1$ *for all* s. *In case* Γ *is closed, we identify* $[0, l(\Gamma)]$ *algebraically with the circle group by adding elements*

of $[0, l(\Gamma)]$ *modulo* $l(\Gamma)$. *We say that* Γ *is* C^m, $m \in \mathbb{N}$, $m \geq 1$, *if the arc-length parameterization* $\mathbf{x}$ *is a* C^m *function.*

Exercise 1.1. *The aim of the exercise is to give a formula transforming a* C^1 *parameterization* $t \in [0,1] \to \mathbf{x}(t)$ *such that* $|\mathbf{x}'(t)| \neq 0$ *for all* t *into an arc-length parameterization. Notice that* $L : [0,1] \to [0, L(1)]$ *is increasing. Set, for* $s \in [0, L(1)]$, $\tilde{\mathbf{x}}(s) = \mathbf{x}(L^{-1}(s))$ *and check that* $\tilde{\mathbf{x}}$ *is an arc-length parameterization of the curve defined by* $\mathbf{x}$.

An arc-length parameterization is also called a *Euclidean parameterization*. If a Jordan curve has an arc-length parameterization $\mathbf{x}$, then the domain of definition of $\mathbf{x}$ on the real line must be an interval $[a, b]$, where $b - a$ is the length of Γ, which we denote by $l(\Gamma)$. In this case, we will always take $[0, l(\Gamma)]$ as the domain of definition of $\mathbf{x}$.

One can easily describe all Euclidean parameterizations of a Jordan curve.

Proposition 1.1. *Suppose that* Γ *is a* C^1 *Jordan curve with arc-length parameterization* $\mathbf{x} : [0, l(\Gamma)] \to \Gamma$. *Then any other arc-length parameterization* $\mathbf{y} : [0, l(\Gamma)] \to \Gamma$ *is of the form* $\mathbf{y}(s) = \mathbf{x}(s + \sigma)$ *or* $\mathbf{y}(s) = \mathbf{x}(-s + \sigma)$ *for some* $\sigma \in [0, l(\Gamma)]$.

Proof. Denote by C the interval $[0, l(\Gamma)]$, defined as an additive subgroup of $\mathbb{R}$ modulo $l(\Gamma)$. Let $\mathbf{x}$, $\mathbf{y} : C \mapsto \Gamma$ be two length preserving parameterizations of Γ. Then $f = \mathbf{x} \circ \mathbf{y}^{-1}$ is a length preserving bijection of C. Using the parameterization of C, this implies $f(s) = \pm s + \sigma$ for some $\sigma \in [0, l(\Gamma)]$ and the proof is easily concluded. □

Definition 1.3. *Assume that* Γ *is* C^2 *and let* $s \mapsto \mathbf{x}(s)$ *be an arc-length parameterization. The tangent vector* $\boldsymbol{\tau}$ *is defined as* $\boldsymbol{\tau}(s) = \mathbf{x}'(s)$. *The curvature vector of the curve* Γ *is defined by* $\boldsymbol{\kappa}(s) = \mathbf{x}''(s)$. *The normal vector* $\mathbf{n}(s)$ *is defined by* $\mathbf{n}(s) = \boldsymbol{\tau}^{\perp}$, *where* $(x, y)^{\perp} = (-y, x)$.

Proposition 1.2. *Let* Γ *be a* C^2 *Jordan curve, and let* $\mathbf{x}$ *and* $\mathbf{y}$ *by any two arc-length parameterizations of* Γ.

(*i*) *If* $\mathbf{x}(s) = \mathbf{y}(t)$, *then* $\mathbf{x}'(s) = \pm \mathbf{y}'(t)$.

(*ii*) *The vector* $\boldsymbol{\kappa}$ *is independent of the choice of arc-length parameterizations and it is orthogonal to* $\boldsymbol{\tau} = \mathbf{x}'$.

Proof. By Proposition 1.1, $\mathbf{y}(s) = \mathbf{x}(\pm s + \sigma)$ and (*i*) follows by differentiation. This is also geometrically obvious: $\mathbf{x}'(s)$ and $\mathbf{y}'(t)$ are unit vectors tangent to Γ at the same point. Thus, they either point in the same direction or they point in opposite directions.

Using any of the above representations and differentiating twice shows that $\mathbf{x}'' = \mathbf{y}''$. Since $\mathbf{x}' \cdot \mathbf{x}' = 1$, differentiating this expression shows that $\mathbf{x}'' \cdot \mathbf{x}' = 0$. Thus, $\mathbf{x}''$ and $\mathbf{x}'$ are orthogonal and $\mathbf{x}''$ and $\mathbf{x}'^{\perp}$ are collinear. □

It will be convenient to have a flexible notation for the curvature in the different contexts we will use it. This is the object of the next definition.

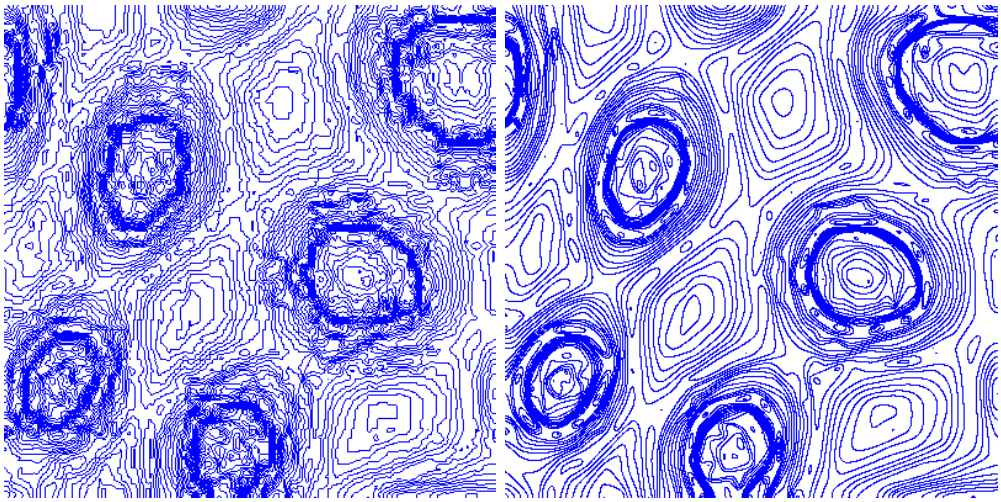


FIGURE 1. Simultaneous curve evolution by the intrinsic heat equation. The evolved curves are smooth for all times and oscillations reduce considerably. The level curves are included one into the other and conserve this property when applying the smoothing.

Definition 1.4 (and notation)**.** *Given a C^2 curve Γ, which is parameterized by length as $s \mapsto \mathbf{x}(s)$ and $\mathbf{x} = \mathbf{x}(s)$ a point of Γ, we denote in three equivalent ways the curvature of Γ at $\mathbf{x} = \mathbf{x}(s)$,*

$$\boldsymbol{\kappa}(\mathbf{x}) = \boldsymbol{\kappa}(\mathbf{x}(s)) = \boldsymbol{\kappa}(s) = \mathbf{x}''(s).$$

In the first notation, $\boldsymbol{\kappa}$ is the curvature of the curve Γ at a point $\mathbf{x}$ implicitly supposed to belong Γ. In the second notation a particular parameterization of Γ, $\mathbf{x}(s)$, is being used. In the third one, $\mathbf{x}$ is omitted.

The above notations create no ambiguity or contradiction, since by Proposition 1.2 the curvature is independent of the Euclidean parameterization.

2. Curve evolutions

Computational shape recognition methods often make the following two basic assumptions, neither of which is true in practice for the rough shape data:

(1) the shape is a smooth Jordan curve.
(2) the boundary has a small number of inflexion points and curvature extrema; this number can be made as small as desired by smoothing.

The fact that these conditions can be obtained by properly smoothing a C^1 Jordan curve was proven in 1986-87 by Gage and Hamilton [**10**] and Grayson [**11**]. They showed that it is possible to transform a C^1 Jordan curve into a C^∞ Jordan curve by using the so-called *intrinsic heat equation*.

For convenience, and unless it would cause ambiguity, we will not make a distinction between a Jordan curve Γ as a subset of the plane and a function

$s \mapsto \mathbf{x}(s)$ such that $\Gamma = \{\mathbf{x}(s)\}$. As we have already done, we will speak of the Jordan curve $\mathbf{x}$. Since we will be speaking of families of Jordan curves dependent on a parameter $t > 0$, we will most often denote these families by $\mathbf{x}(t, s)$, where the second variable is a parameterization of the Jordan curve. Thus, $\mathbf{x}(t, s)$ has three meanings: a family of Jordan curves, a family of functions that represent these curves, and a particular point on one of these curves. The notation s will be usually reserved to an arc-length parameter.

Definition 2.1. *Let $\mathbf{x}(t)$, $t > 0$, be a family of C^2 Jordan curves. We say that $\mathbf{x}(t)$ satisfies the intrinsic heat equation if*

$$\frac{\partial \mathbf{x}}{\partial t} = \boldsymbol{\kappa}(\mathbf{x}(t)). \tag{CS}$$

By this (nonlinear) evolution a curve instantly becomes smooth, shrinks asymptotically to a circle and develops no singularities or self-crossings. The proofs of these properties were given by Gage and Hamilton for convex Jordan curves [**10**] and later extended to embedded curves by Grayson [**11**].

Theorem 2.1 (Grayson)**.** *Let $\mathbf{x}_0$ be a C^1 Jordan curve. By using the intrinsic heat equation, it is possible to evolve $\mathbf{x}_0$ into a family of Jordan curves $\mathbf{x}(t, s)$ such that $\mathbf{x}(0, s) = \mathbf{x}_0(s)$ and such that for every $t > 0$, $\mathbf{x}(t, s)$ is C^∞ (actually analytical) and satisfies the equation (CS). Furthermore, for every $t > 0$, $\mathbf{x}(t, s)$ has only a finite number of inflection points and curvature extrema, and the number of these points does not increase with t. For every initial curve, there is a scale t_0 such that the curve $\mathbf{x}(t, s)$ is convex for $t \geq t_0$ and there is a scale t_1 such that the curve $\mathbf{x}(t, s)$ is a single point for $t \geq t_1$.*

A surprising variant of Curve Shortening is given by the Affine Shortening equation

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

It was introduced by Sapiro and Tannenbaum in [**21**], [**23**]. Angenent, Sapiro and Tannenbaum [**24**] gave the existence and uniqueness proofs for affine shortening and showed a result similar to Grayson's theorem:

Theorem 2.2 (Angenent, Sapiro, Tannenbaum)**.** *Let $\mathbf{x}_0$ be a C^2 Jordan curve. Then there is a unique classical solution $\mathbf{x}(t)$ of (AS). The curve eventually becomes convex and thereafter evolves towards an ellipse before collapsing.*

In computer vision the above equations are referred to as *curve scale spaces* or *shape scale spaces*. The term designates any process that smooths a Jordan curve and depends on a real parameter t, the scale. A shape scale space associates with an initial Jordan curve $\mathbf{x}(0, s) = \mathbf{x_0}(s)$ a family of smooth curves $\mathbf{x}(t, s)$. Curve shortening and affine shortening eliminate spurious details of the initial shape and retain simpler, more reliable versions of the shape. These smoothed shapes have finite codes in the sense of Attneave, since they have finitely many curvature extrema. A scale space is *causal* in the terminology of vision theory if it does not introduce new features. (New feature here means: a new extremum

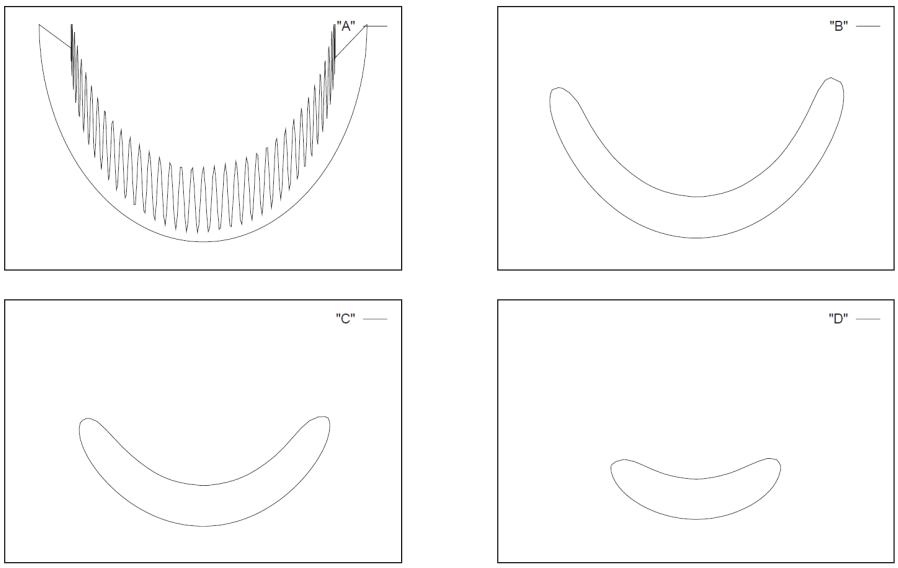


FIGURE 2. Curve evolution by the intrinsic heat equation. The evolved curve is smooth for all times, eventually becomes convex and shrinks to a point.

for some image differential operator). Thus, curve shortening and affine curve shortening define causal scale spaces. Indeed, the number of curvature extrema and inflexion points decreases by their application.

3. Algorithms on curves

3.1. Dynamic curve evolution. Mackworth and Mokhtarian proposed an algorithm consistent with curve shortening (CS). Instead of applying the linear heat equation for relatively long times, it applies to a plane curve the non-linear heat equation, by successively convolving the arc length parameterization $\mathbf{x}(\cdot, t)$ at time n with a Gaussian kernel G_h of standard deviation proportional to $h^{\frac{1}{2}}$.

Algorithm 1: Discrete Curve Shortening (CS)

Input: Polygon Γ_0, gaussian signal G
Output: Evolved polygon Γ_n, after n iterations
for *all* $i = \overline{0, n}$ **do**
    sample uniformly curve Γ_i ;
    convolve curve Γ_i with G.

The consistency of Algorithm 8 with (CS) is given by the (easy) Theorem 3.1.

Theorem 3.1. *Let* $\mathbf{x}$ *be a* C^2 *curve parameterized by its length parameter* $s \in [0, L]$. *Then*

$$G_h * \mathbf{x}(s) - \mathbf{x}(s) = ch\boldsymbol{\kappa}(\mathbf{x}(s)) + o(h). \tag{3.2}$$

where c *is a positive constant.*

PROOF. Using the definition of the convolution we write the left-hand side of the equation as

$$\begin{aligned} G_h * \mathbf{x}(s) - \mathbf{x}(s) &= \int G_h(s-\tau)\mathbf{x}(\tau)d\tau - \int G_h(\tau)\mathbf{x}(s)d\tau \\ &= \int \frac{1}{h^{1/2}} G(\frac{\tau}{h^{1/2}})[\mathbf{x}(s-\tau) - \mathbf{x}(s)]d\tau \\ &= \int G(\tau)[\mathbf{x}(s-h^{1/2}\tau) - \mathbf{x}(s)]. \end{aligned}$$

By Taylor's formula with Peano remainder

$$\mathbf{x}(s-h^{1/2}\tau) - \mathbf{x}(s) = -h^{1/2}\tau\frac{\partial \mathbf{x}}{\partial s}(s) + \frac{h}{2}\tau^2\frac{\partial^2 \mathbf{x}}{\partial s^2}(s) + \frac{h}{2}\alpha(s-h^{1/2}\tau),$$

with α a continuous function such that $\lim_{h\to 0}\alpha(s-h^{1/2}\tau) = 0$. Thus, we have

$$\begin{aligned} G_h * \mathbf{x}(s) - \mathbf{x}(s) &= h\frac{\partial^2 \mathbf{x}}{\partial s^2}(s)\frac{1}{2}\int G(\tau)\tau^2 + \frac{h}{2}\int \alpha(s-h^{1/2}\tau)d\tau \\ &= ch\boldsymbol{\kappa}(\mathbf{x}(s)) + o(h). \end{aligned}$$

□

3.2. Affine plane curve evolution. Several attempts to define an affine-invariant analysis for polygons are described in [**22**]. The 1/3 power law of planar motion perception and generation was related to affine invariance in [**20**]. Moisan [**17**] discovered an extremely fast and fully affine invariant geometric curve evolution consistent with affine shortening, which we summarize below. In the mathematical morphology terminology, this algorithm is an alternate filter, alternating an affine erosion and an affine dilation. This scheme is able to smooth all the level lines of an image (that is, several thousands of curves) in a couple of seconds.

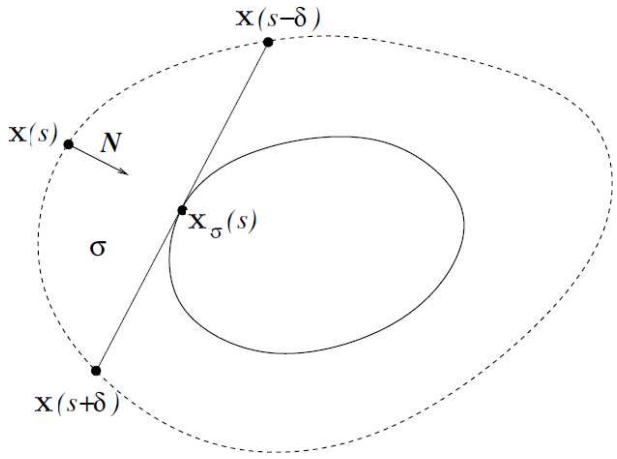
Algorithm 2: Discrete Affine Shortening (AS)

Input: Polygon Γ_0,
Output: Evolved polygon Γ_σ, at scale $\sigma^{2/3}$

1 break the curve into convex and concave parts ;
2 **for** *every convex/concave component* **do**
3 replace each component by the sequence of the middle points of each σ-chord such that one endpoint is a vertex of the polygonal curve;
4 concatenate the pieces of curves previously obtained.

Definition 3.1. *A σ-chord is a chord determined by two points of the curve $(\mathbf{x}(s-\delta), \mathbf{x}(s+\delta))$, where $\delta > 0$ is chosen in order that the area of the region enclosed by this chord and the piece of curve $\mathbf{x}\big|_{(s-\delta,s+\delta)}$ be equal to σ.*

The consistency of Algorithm 2 with affine shortening (AS) is given in Theorem 3.2.



Theorem 3.2. *Let* $\mathbf{x}$ *be a* C^2 *curve parameterized by its length parameter* $s \in [0, L]$ *and* $\sigma > 0$*. To each point of* $\mathbf{x}(s)$*, we associate* $\mathbf{x}_\sigma(s)$*, defined as the middle point of the* σ*-chord* $(\mathbf{x}(s-\delta), \mathbf{x}(s+\delta))$*, where* $\delta > 0$ *is chosen in order that the area of the region enclosed by this chord and the piece of curve* $\mathbf{x}\big|_{(s-\delta, s+\delta)}$ *be equal to* σ*. Then*

$$\mathbf{x}_\sigma(s) - \mathbf{x}(s) = c\sigma^{2/3}|\boldsymbol{\kappa}(\mathbf{x})|^{-2/3}\boldsymbol{\kappa}(\mathbf{x}) + o(\sigma^{2/3}) \text{ as } \sigma \to 0$$

where c *is a positive constant.*

Proof. We use the fact that if $\mathbf{x}(t, s)$ is a Jordan curve, then its (signed) area is given by Green's formula

$$\frac{1}{2}\oint [\mathbf{x}(\tau), \mathbf{x}'(\tau)]d\tau.$$

(we write $[u, v]$ for the determinant of two vectors u, v). In the case we consider, we have

$$\sigma = \frac{1}{2}A(s, \delta(s, \sigma)),$$

where

$$A(s, \delta) = \int_{s-\delta}^{s+\delta} [\mathbf{x}(\tau), \mathbf{x}'(\tau)]d\tau + [\mathbf{x}(s+\delta), \mathbf{x}(s-\delta) - \mathbf{x}(s+\delta)].$$

Taking the derivative with respect to δ yields

$$\begin{aligned}\frac{\partial A}{\partial \delta} &= [\mathbf{x}(s+\delta), \mathbf{x}'(s+\delta)] - [\mathbf{x}(s-\delta), \mathbf{x}'(s+\delta)] + \\ &\quad [\mathbf{x}'(s+\delta), \mathbf{x}(s-\delta)] + [\mathbf{x}(s+t), \mathbf{x}'(s-\delta)] \\ &= [\mathbf{x}(s+\delta) - \mathbf{x}(s-\delta), \mathbf{x}'(s+\delta) - \mathbf{x}'(s-\delta)].\end{aligned}$$

Using a Taylor's expansion on the right hand side we obtain that

$$\frac{\partial A}{\partial \delta} = [2\delta\mathbf{x}'(s) + o(\delta), 2\delta\mathbf{x}''(s) + o(\delta)] = 4\delta^2|\boldsymbol{\kappa}(s)| + o(\delta^2).$$

Integrating with respect to δ we further get

$$2\sigma = \frac{4}{3}\delta^3|\boldsymbol{\kappa}(s)| + o(\delta^3).$$

Therefore, whenever the curvature is not zero, we find

$$\delta(s;\sigma) = \big(\frac{3\sigma}{2|\boldsymbol{\kappa}(s)|}\big)^{1/3} + o(\sigma^{1/3}).$$

Finally we get

$$\begin{aligned} \mathbf{x}_\sigma(s) &= \frac{1}{2}(\mathbf{x}(s-\delta) + \mathbf{x}(s+\delta)) = \mathbf{x}(s) + \frac{\delta^2}{2}\mathbf{x}''(s) + o(\delta^2) \\ &= \mathbf{x}(s) + \frac{1}{2}(\frac{3}{2})^{2/3}\sigma^{2/3}|\boldsymbol{\kappa}|^{-2/3}\boldsymbol{\kappa}(s) + o(\sigma^{2/3}). \end{aligned}$$

□

CHAPTER 2

Image Representation in terms of Bilinear Level Lines

The topographic map provides a complete representation of an image. This representation is well suited for shape analysis and recognition, since it is based on the geometrical information of images, and can be embedded in a tree structure. However, since the level lines of digital images (zero order interpolates) suffer from pixelization effect, shapes cannot be accurately described. Higher order interpolates are then to be considered. In this chapter, the bilinear interpolation of gray level images is described. Then, following [**6**] a fast numerical method for extracting the topographic map is presented. We complete the work of Lissani et al. with a fast inverse algorithm, of image reconstruction from a family of level lines embedded in a tree structure.

1. The structure of the set of level lines

An image can be represented by its level sets. The next step, with a view toward shape analysis, is the representation of an image in terms of its level lines. We rely heavily on the implicit function theorem to develop this representation. We begin with a two-dimensional version. The statement here is just a slight variation on the implicit function theorem.

We denote by $\mathcal{F}$ the set of continuous functions defined on $\mathbb{R}^2$, tending to some constant at infinity. For a continuously differentiable function u we note its gradient $Du = (u_x, u_y)$.

Theorem 1.1. *Let $u \in \mathcal{F}$ be a C^1 function such that $Du(\mathbf{x}_0) \neq 0$ at some $\mathbf{x}_0 = (x_0, y_0)$. Let $\boldsymbol{i}$ denote the unit vector in the direction (u_x, u_y), let $\boldsymbol{j}$ denote the unit vector in the orthogonal direction $(-u_y, u_x)$, and write $\mathbf{x} = \mathbf{x}_0 + x\boldsymbol{i} + y\boldsymbol{j}$. Then there is a disk $D(\mathbf{x}_0, r)$ and a unique C^1 function φ, $\varphi : [-r, r] \to \mathbb{R}$, such that if $\mathbf{x} \in D(\mathbf{x}_0, r)$, then*

$$u(x, y) = u(\mathbf{x}_0) \quad \Longleftrightarrow \quad x = \varphi(y).$$

The following corollary is a global version of this local result.

Corollary 1.1. *Assume that $u \in \mathcal{F}$ is C^1 and let $u^{-1}(\lambda) = \{\mathbf{x} \mid u(\mathbf{x}) = \lambda\}$ for $\lambda \in \mathbb{R}$. If $\lambda \neq u(\infty)$ and $Du(\mathbf{x}) \neq 0$ for all $\mathbf{x} \in u^{-1}(\lambda)$, then $u^{-1}(\lambda)$ is a finite union of disjoint Jordan curves.*

Proof. From Theorem 1.1 we know that for each point $\mathbf{x} \in u^{-1}(\lambda)$ there is an open disk $D(\mathbf{x}, r(\mathbf{x}))$ such that $\overline{D}(\mathbf{x}, r(\mathbf{x})) \cap u^{-1}(\lambda)$ is a C^1 Jordan arc $\mathbf{x}(s)$ and we can take the endpoints of the arc on $\partial D(\mathbf{x}, r(\mathbf{x}))$. Since $\lambda \neq u(\infty)$, $u^{-1}(\lambda)$ is compact. Thus there is a finite number of points $\mathbf{x}_i$, $i = 1, \ldots, m$, such that

$u^{-1}(\lambda) \subset \bigcup_{i=1}^{m} D(\mathbf{x}_i, r(\mathbf{x}_i))$. This implies that $u^{-1}(\lambda)$ is a finite union of Jordan arcs which we can parameterize by length. The rest of the proof is very intuitive and is left to the reader. It consists of iteratively gluing the Jordan arcs until they close up into one or several Jordan curves. □

The next theorem is one of the few results that we are going to quote rather than prove, as we have done with the implicit function theorem.

Theorem 1.2 (Sard's theorem)**.** *Let $u \in \mathcal{F} \cap C^1$. Then for almost every λ in the range of u, the set $u^{-1}(\lambda)$ is nonsingular, which means that for all $\mathbf{x} \in u^{-1}(\lambda)$, $Du(\mathbf{x}) \neq 0$.*

As a direct consequence of Sard's Theorem and Corollary 1.1, we obtain:

Corollary 1.2. *Let $u \in \mathcal{F} \cap C^1$. Then for almost every λ in the range of u, the set $u^{-1}(\lambda)$ is the union of a finite set of disjoint simple closed C^1 curves.*

The sole purpose of the next proposition is to convince the reader that the level lines of a function provide a faithful representation of the function.

Proposition 1.1. *Let $u \in \mathcal{F} \cap C^1$. Then u can be reconstructed from the following data: the family of all of its level lines at nonsingular levels, the level of each level line being also kept.*

Proof. Let G be the closure of the union of the ranges of all level lines of u at nonsingular levels. If $\mathbf{x} \in G$, then there are points $\mathbf{x}_n$ belonging to level lines of some levels λ_n such that $\mathbf{x}_n \to \mathbf{x}$. As a consequence, $\lambda_n = u(\mathbf{x}_n) \to u(\mathbf{x})$. So we get back the value of $u(\mathbf{x})$.
Let now $\mathbf{x}$ belong to the open set G^c. Let us first prove that $Du(\mathbf{x}) = 0$. Assume by contradiction that $Du(\mathbf{x}) \neq 0$. By using the first order Taylor expansion of u around $\mathbf{x}$, one sees that for all $r > 0$ the connected range $u(B(\mathbf{x}, r))$ must contain some interval $(u(\mathbf{x}) - \alpha(r), u(\mathbf{x}) + \alpha(r))$ with $\alpha(r) \to 0$ as $r \to 0$. By Sard's theorem some of the values in this interval are nonsingular. Thus we can find nonsingular levels $\lambda_n \to u(\mathbf{x})$ and points $\mathbf{x}_n \to \mathbf{x}$ such that $u(\mathbf{x}_n) = \lambda_n$. This implies that $\mathbf{x} \in G$ and yields a contradiction.
Thus $Du(\mathbf{x}) = 0$ on G^c and u is therefore constant on each connected component A of G^c. The value of u is then uniquely determined by the value of u on the boundary of A. This value is known, since ∂A is contained in G. □

2. Bilinear level lines

In practice, we do not deal with real valued images, but with digital images. A digital image ud is a function defined in a rectangular grid, that takes values in a finite set, typically integer values between 0 and 255. One can think of ud as a regular sampling of an image u defined in a closed rectangle of R^2, whose grey levels were quantized, followed by a zero-order interpolation with a rectangular element.

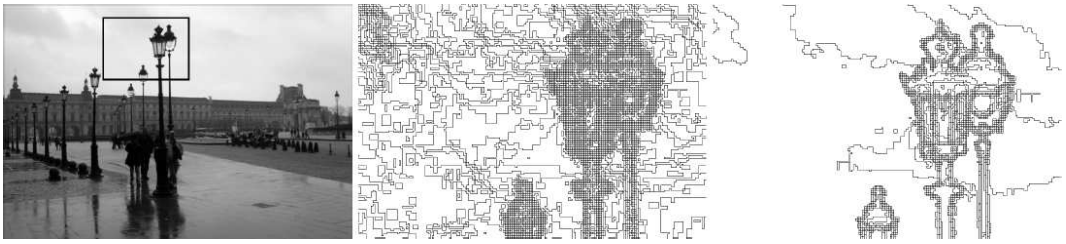


FIGURE 1. Level lines from a digital image (zero-order interpolation). Left: original image. Middle: all level lines for the small image inside the rectangle. Right: level lines for a gray level quantization step of 10. Level lines are restricted to lie in the original grid, and the majority of them provides no useful information from the perceptual viewpoint.

Each element of the grid is called a pixel. Digital images are piecewise constant functions, and as a consequence, level lines from digital images show pixelization effects, as shown in Figure 2.

Some consequences of this pixelization are:

- Useful invariant features such as inflexion points or curvature extrema, that are extensively used in shape recognition, cannot be computed on pixelized level lines,
- The accuracy of any measure based on the location of the level lines is limited by the pixel size,
- Level lines corresponding to different gray levels may have pieces in common (creating Tjunctions). This never happens when dealing with level lines of a continuous image.

Pixelization effect can be avoided by considering higher order (than zero-order) interpolations of digital images. Then, the level lines of the interpolated images can be computed. These level lines have some interesting properties:

- Level lines will be smoother than in the previous case,
- Subpixel accuracy can be achieved when measuring level lines, since they are not restricted to the grid of the digital image,
- Level lines from different gray levels never touch each other, since the considered images are now continuous functions.

Among the possible interpolations, the bilinearly interpolation presents two advantages: it is the most local of continuous interpolations, and it preserves the order between the gray levels of the image.

2.1. Bilinear interpolation. The bilinear interpolate of a digital image ud, denoted by $\tilde{u}$, can be obtained as the convolution of ud (considered as a network of Dirac masses concentrated at the centers of pixels) with the function $\phi(x; y) = \varphi(x)\varphi(y)$, where

$$\varphi(x) = \max(1 - |x|; 0)$$

FIGURE 2. Left: level lines from a digital image (zero-order interpolation). The quantization step for the gray levels is 10, starting from 10. The pixelization effect creates artificial T-junctions, and do not allow to compute useful features such as inflexion points or curvature extrema. Middle: level lines from the piecewise bilinear interpolated image. The quantization step for the gray levels is 10, starting from 10. Pixelization effect has been reduced, but some pixelized regions can still be seen. Right: level lines from the piecewise bilinear interpolated image, with a gray level quantization step of 10, starting at gray level 0:5. These level lines do not suffer from pixelization effects.

As $\varphi \geq 0$, bilinear interpolation is a convolution with a nonnegative kernel and hence an increasing operator. Since $\varphi(x) < \varphi(0) = 1$, the extrema of $\tilde{u}$ are all located at points on the discrete grid. More precisely, all regional extrema of $\tilde{u}$ contain at least a local extremum in the original grid.

The general form of a bilinear function is

$$u(x, y) = axy + bx + cy + d.$$

For each set of four adjacent pixels (which will be called a *Qpixel* from now on), a bilinear function can be determined: parameters a, b, c and d are fixed by the gray levels of the four pixels $(i, j); (i+1, j); (i, j+1); (i+1, j+1)$. More precisely

$$\begin{aligned} u(x, y) &= (j + 1 - y)\big((i + 1 - x)u(i, j) + (x - i)u(i, j + 1)\big) + \\ &\quad (y - j)\big((i + 1 - x)u(i + 1, j) + (x - i)u(i + 1, j + 1)\big). \end{aligned}$$

The bilinear interpolate of a *Qpixel* is defined only inside the rectangle delimited by the Qedgels, which are the segments between adjacent pixels centers in the *Qpixel*. The bilinear interpolation of a digital image is the concatenation of bilinear interpolates of its Qpixels. Continuity of the gray levels between contiguous *Qpixels* is guaranteed by the properties of the bilinear interpolation, but higher continuities (e.g., of the gradient) are not preserved at Qedgels.

2.2. Bilinear level lines. In this subsection, some basic results concerning level lines of a bilinear interpolated image are presented. These results will be used for the extraction algorithm. The equation for a level line at level λ of the bilinear interpolated image inside a dual pixel can be written

$$a(x - x_s)(y - y_s) + (\lambda_s - \lambda) = 0$$

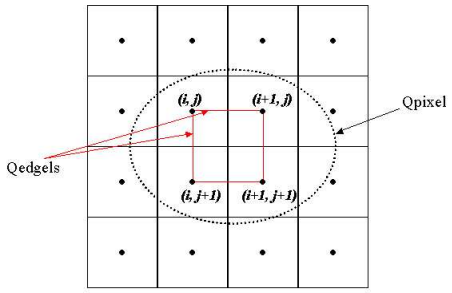


FIGURE 3. Definition of Qpixel and Qedgels.

or

$$axy + bx + cy + (d - \lambda) = 0.$$

In the first case, level lines are pieces of hyperbola, of asymptotes $x = x_s, y = y_s$. When $\lambda = \lambda_s$ the level line consists of two orthogonal straight lines crossing at the saddle point (x_s, y_s), provided this point is inside the dual pixel. In the second case, level lines are straight lines. This may lead to visual pixelization effects, for instance when level lines pass through the center of a pixel and follow a dual-edge. This phenomenon will be attenuated by taking for λ only half-integer values (given that the digital image has integer values).

In the following, some basic results concerning level lines of a bilinear interpolated image are presented. These results will be used for the extraction algorithm.

Proposition 2.1. *(Properties of level lines of bilinear interpolates)*

- *Except a finite set $\lambda_1, \ldots, \lambda_n$ of levels, the level set $u(x, y) = \lambda$ is the union of a finite number of Jordan curves;*
- *All level lines are continuous and piecewise C^1.*

The above structure is quite simple, but it does not describe what happens at the critical levels λ_i, which can have a rather complicated form (see Figure 2.2). For the sake of precision, we choose to avoid these levels, by setting for example a quantization step at half integer coordinates.

2.3. The inclusion tree. One can decompose an interpolated image into its level lines at predefined levels. A fast algorithm, the Fast Level Set Transform (FLST) performing the decomposition into a tree of shapes, is described in [**6**] and [**18**]. The image is parsed into a set of parametric Jordan curves. This set is ordered in a tree structure, induced by the geometrical inclusion. We say that a curve Γ^{λ_1} is a *child* of the curve Γ^{λ_2} and we denote

$$\Gamma^{\lambda_1} \prec \Gamma^{\lambda_2}$$

if its interior is included in the interior of the latter. In addition, each curve has an assigned tag ± 1 according to whether it is the boundary of a connected component of a lower level set ($\mathrm{sgn}(\Gamma^\lambda) = -1$) or upper level set ($\mathrm{sgn}(\Gamma^\lambda) = +1$).

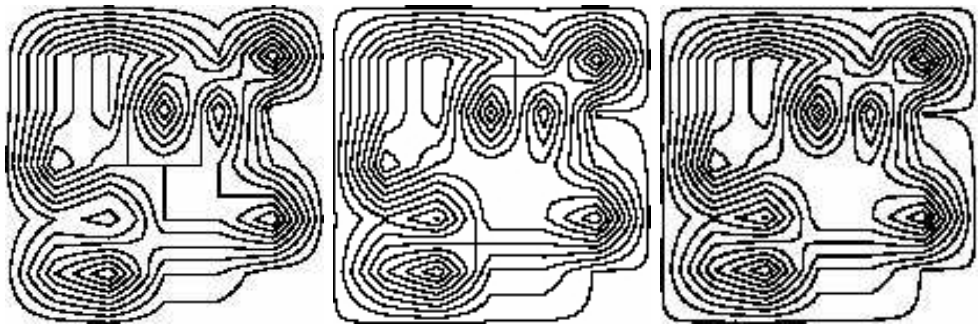


FIGURE 4. Gray levels for a piecewise bilinearly interpolated image. Three different sets of level lines were computed. Left: gray levels from 10 to 100 with step 10. Observe how some of the level lines (the ones at gray level 70) follow the Qedgels, producing an effect similar to pixelization. Middle: level lines were computed at gray levels different from those of the original image but we get 90^o crossings between level lines due to the presence of a saddle point. Nevertheless, these saddle points will always appear inside the Qpixels and the curves never go along the grid of the digital image. Right: gray level 11 to gray level 91 with quantization step 10. Pixelization effect no longer arises since level lines are computed at gray levels different from those of the original image.

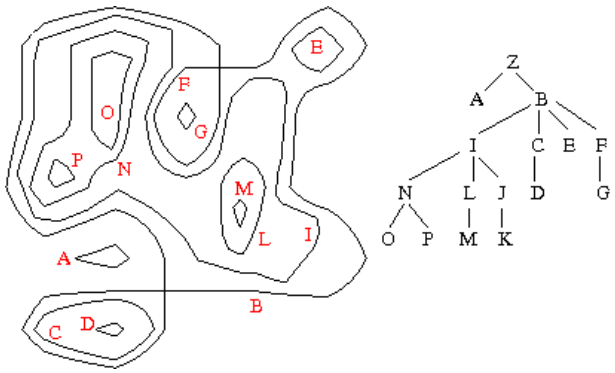


FIGURE 5. Tree of bilinear level lines, including some saddle regions.

For each gray level $\lambda \in \mathbb{N} + 1/2$ there corresponds a finite set F^λ of level lines $\{\Gamma_0^{\lambda,i}\}_{i \in F^\lambda}$. Each level line $\Gamma_0^{\lambda,i}$ is stored as a set of ordered points $\{P_k(x_k, y_k)\}_{1 \le k \le N}$ with N depending on Γ, leaving the level line interior on the left hand side. The real numbers x_k and y_k are the floating point coordinates of the vertex number k of the polygon Γ

Thus, the tree of level lines is given by a finite set of *tagged* polygonal lines, indexed by half-integer gray values

$$\mathcal{T}_0 = \{\Gamma_0^{\lambda,i}; i \in F^\lambda, \lambda \in \mathbb{N} + 1/2\}. \tag{2.3}$$

An inclusion tree of bilinear level lines is displayed in Figure 5.

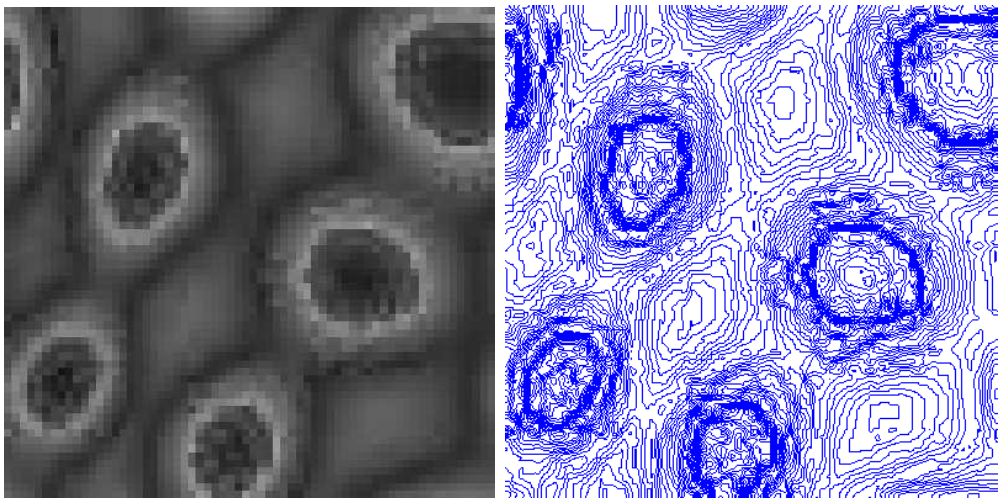


FIGURE 6. Extraction of level lines from a bilinear interpolated image, with quantization step 16. Singular levels are avoided. Nevertheless, level lines are highly oscillatory, especially along transversal lines. For this reason, before computing any features on these level lines, a previous smoothing is necessary.

2.4. Direct Algorithm: extraction of level lines from a bilinear interpolated image. We describe a simple algorithm for the extraction of level lines of a bilinear interpolated image, but which requires a quantization avoiding initial levels of the image. The algorithm is in two phases: first extract the bilinear level lines then order them in a tree. The extraction of a level line relies on these observations:

(1) the level line is not a regional extremum, so that an orientation can be chosen, for example leaving the upper level at the left of the level line;
(2) the level line meets Qedgels at non endpoints, so that when we enter a Qpixel by a Qedgel, we can compute from which other Qedgel to exit.

Considering we get into the Qpixel through some Qedgel, the exit Qedgel is in front, on the left or on the right depending on the value of with respect to levels at both other corners of the Qpixel. We may store for the line only its list of intersections with Qedgels.

To avoid extracting several times the same level line, we put markers at all intersection points of extracted level lines with Qedgels. The algorithm starts by considering all Qedgels at the boundary of the image. If a level of interest is between the levels at endpoints of the Qedgel, we follow the level line, marking intersection points with Qedgels. We then close the level line by the shortest path along the image boundary. Finally we do the same for interior Qedgels. For them, we detect closure by checking the marker at exit Qedgels.

To build the tree structure of the extracted level lines, we consider all intersection points of level lines with vertical edgels and order them. While scanning a column of Qedgels, we are in the interior domain of a level line if we have crossed

it an odd number of times. If we meet a level line that has no parent yet, its parent has to be the last level line we are in. For the root, we add the boundary of the image as a level line to the list L at the beginning.

Let L be the list of level lines. To describe the intersection y of a level curve Γ_{id}, having index id in the list L, with Qedgel at column $i + 0.5$, we use the triplet (i, y, id). The main idea used in the ordering algorithm is to look for the innermost shapes that we denote generically by Γ. The arrow $\Gamma \leftarrow \tilde{\Gamma}$shorlty says that we set the curve Γ to be now $\tilde{\Gamma}$. And finally, we denote the parent of a level line by $Parent(\Gamma)$.

Algorithm 3: Ordering the level lines in a tree.

Input: List L of bilinear level lines Γ
Output: Fill tree structure

```
Collect all (i, y, id) in array V;
Order V lexicographically by key (i, y);
Γ ← ∅;
for all (i, y, id) ∈ V do if Γ = Γ_id then
    Γ ← Parent(Γ);
    else if Γ ≠ ∅ and Parent(Γ_id) = ∅ then
        set Γ_id as a child of Γ;
        Γ ← Γ_id.
```

This algorithm relies on the fact that the quantization is chosen so that each level line crosses Qedgels, but does not contain any. For this, it is sufficient that it avoids the levels at the centers of pixels (the initial data). In particular, regional extrema of the image cannot be extracted by the algorithm.

2.5. Inverse algorithm: image reconstruction from a set of level lines. The algorithm described in this section performs an exact image reconstruction from a topographic map, i.e. from an arbitrary family of Jordan curves organized in a tree structure with respect to geometrical inclusion.

The reconstruction starts from a topographic map, namely a family of discrete level lines (typically obtained after (affine) curve shortening) $\{\Gamma^{\lambda,i}\}_{i\in F^\lambda,\lambda\in\Lambda}$ organized in an inclusion tree structure. This tree is walked down (parent before children) and the interior of the current level line is filled in with its level λ. Using that order, each level line interior is painted before its descendants, ensuring that its private pixels are at the correct level while non-private pixels get painted over by the children. This yields an exact reconstruction for any digital image u_d from its level lines at half-integer levels:

Theorem 2.1. *Let $\mathcal{T} = \{\Gamma^{\lambda,i}; i \in F^\lambda, \lambda \in \mathbb{N} + 1/2\}$ be the tree of bilinear level lines associated to u_d. For every x let λ be such that $x \in Int(\Gamma^\lambda)$ and $\forall \Gamma^{\tilde{\lambda}} \prec \Gamma^\lambda$,*

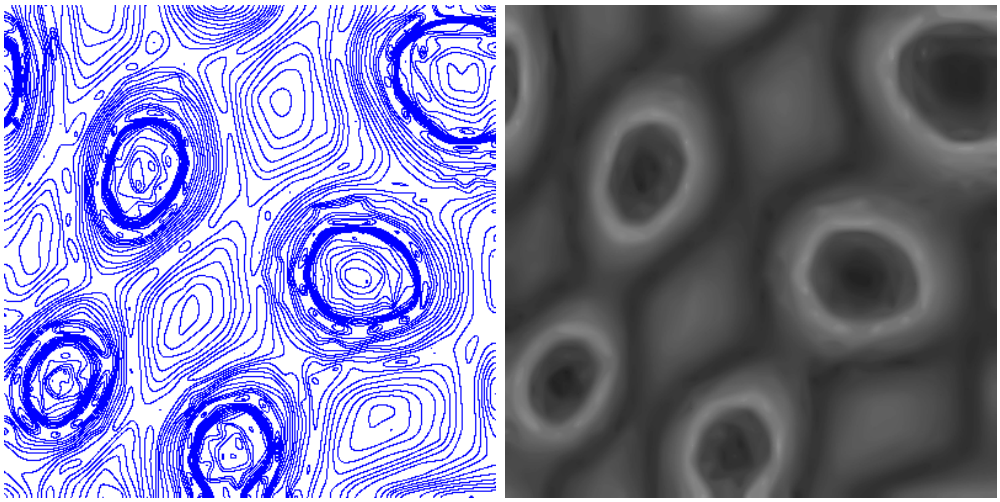


FIGURE 7. Reconstruction of the image from a family of level lines. Left: level lines are displayed with quantization step 16 and have been previously smoothed by the intrinsic heat equation. Right: image reconstructed from the family of all level lines, except for the ones corresponding to singular gray levels.

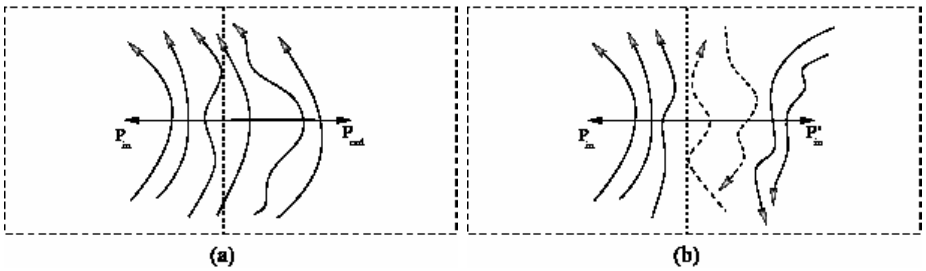


FIGURE 8. The level line $2D$ inclusion topology is reflected in the $1D$ ordering of their intersections with the dual edges.

$x \notin Int(\Gamma^{\tilde{\lambda}})$ and define

$$\tilde{u}_d(x) = \begin{cases} \lambda - 1/2, & \text{if } sgn(\Gamma^{\lambda}) = -1 \\ \lambda + 1/2, & \text{if } sgn(\Gamma^{\lambda}) = +1 \end{cases}$$

Then $u_d \equiv \tilde{u}_d$.

A closed curve Γ is stored as a set of ordered points $\{P_k(x_k, y_k)\}_{1 \leq k \leq N}$ with N depending on Γ. The real numbers x_k and y_k are the floating point coordinates of the vertex number k of the polygon Γ. We need to fill in all pixels with integral coordinates (j, i) inside the polygon. To avoid any ambiguity, the algorithm secures that y_k is never an integer by translating when necessary Γ by a tiny amount ε vertically or horizontally, at the price of a minor numerical uncertainty in the reconstructed image. The filling in of each curve is performed by a fast ray casting algorithm described below.

2.5.1. *Polygon intersections with the grid.* The goal of Algorithm 4, which is a preliminary to the filling algorithm, is to find the intersections of the polygonal level line with all horizontal lines $y = i$. For any given i the intersection is in fact the intersection of a segment $[P_k P_{k+1}]$ of the polygon with the line $y = i$. These intersections are ordered by their abscissae so that

$$x_1^i \leq x_2^i \leq \cdots \leq x_p^i,$$

where p is even because Γ is a closed curve. This gives a simple and fast decision rule: a pixel (j, i) is surrounded by the polygon if and only if j is within an odd interval $[x_{2k+1}^i, x_{2k+2}^i]$.

Algorithm 4: Intersections of a polygon Γ with the grid

Input: Vertices $P_k(x_k, y_k)$ of polygon Γ
Output: For each i, the ordered list L^i of points of Γ on the line of equation $y = i$

```
for all i do L^i ← ∅;
for all segments [P_k P_{k+1}] do
    for i ∈ [y_k, y_{k+1}] ∩ ℕ do
        (x, i) ← [P_k P_{k+1}] ∩ {y = i};
        Insert x in L^i;
for all i do sort list L^i
```

2.5.2. *Filling the interior.* Line by line all odd intervals on L^i are enumerated and filled in with level $\lambda \pm 1/2$ at all pixels with ordinate i whose abscissa is inside such an interval, as shown in Algorithm 5.

Algorithm 5: Filling polygon Γ

Input: Sorted lists L^i of intersections of Γ with lines $\{y = i\}$, level λ
Output: Pixels inside polygon Γ are at level $\lambda \pm 1/2$, pixels outside unchanged

```
for all i do
    for all x_{2k+1}^i ∈ L^i do
        for j ∈ ℕ ∩ [x_{2k+1}^i, x_{2k+2}^i] do
            pixel (j, i) ← λ ± 1/2
```

Due to the inclusion principle it is possible to go from the $2D$ topology of the level lines to the $1D$ topology on a dual edge and conversely. Suppose that two or more level lines belonging to different gray levels intersect a dual edge, leaving the same data points outside and inside: denote them P_{in} and P_{out} (Figure 8(a)). Then the restored gray value at P_{out} is the gray value associated to the largest shape ordered by inclusion which leaves the pixel outside, whereas P_{in} belongs to the smallest shape that includes the pixel. If curves with different orientation cross the same dual edge it is enough to update the gray value at P_{in}. This

conforms to our choice of filling the interiors of the lines in the order given by the level line inclusion tree.

CHAPTER 3

Junction: Image Curvature Microscope

Whenever we talk about curvatures in a digital image, we actually refer to the curvatures of the level lines associated to the image. Yet, most curvature computation algorithms are based on finite difference schemes (FDS). In this case, the curvature depends on the gray values of the neighbor pixels and consequently, high oscillations along transverse level lines appear.

We show in the following how curvatures can be accurately estimated by a direct computation on level lines after their independent smoothing. The sub-pixel algorithm yields a microscopic visualization of the curvature map revealing many image details, and getting rid of aliasing effects. This "curvature microscope" showing curvatures in false colors runs on line on any image proposed by users at http://www.ipol.im/pub/algo/cmmm_image_curvature_microscope/

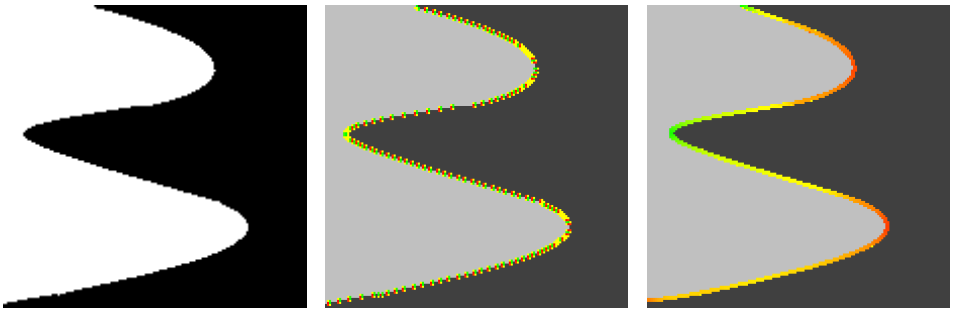
1. The Curvature Microscope

For the sake of precision, we suggest that curvatures should be computed directly on level lines and not on a discrete grid. A polygonal line approximation followed by uniform and fine sampling allows one to compute reliable curvatures, but only after level line smoothing. This smoothing is necessary because the initial level lines present oscillations due to the initial aliasing and to the interpolation itself. Thus curvatures wouldn't correspond to our visual perception. But, more fundamentally, the perception of curvature is and must be multiscale. The striking difference between an FDS result and an LLS result is displayed in Figure 1. With LLS, the curvature is computed at each vertex of each level line. A curvature image is then created by associating to each dual pixel an average of all curvatures computed in it.

1.1. Discrete curvature for a polygonal line. We recall that each level line is stored as a set of ordered points

$$\Gamma = \{P_i(x_i, y_i)\}_{i=0..n}, \text{ with } P_0 = P_n.$$

The simplest discrete scalar curvature k_i computed at each vertex P_i is obtained by taking the triple (P_{i-1}, P_i, P_{i+1}) and computing k_i as the inverse of the circumscribed radius R_i of this triangle. Set $\overrightarrow{u_i} = \overrightarrow{P_{i-1}P_i}$ and its length $u_i = (u_i^1, u_i^2) = |\overrightarrow{P_{i-1}P_i}|$, respectively $\overrightarrow{v_i} = \overrightarrow{P_{i-1}P_{i+1}}$, with the corresponding length $v_i = (v_i^1, v_i^2)| = \overrightarrow{P_{i-1}P_{i+1}}|$. Then



(a) (b) (c)

FIGURE 1. The curvature color display rule. Zero curvatures are displayed in yellow, positive curvatures are shown in a gradation from yellow to red, and negatives from yellow to green. The initial image (a) had its curvatures computed in two different ways: by an FDS formula (b), and directly on level lines, after their independent smoothing (c). In the first case the curvature presents oscillations, whereas the second result is coherent with our perception.

Lemma 1.1. *The curvature at vertex P_i is given by*

$$k_i = 2\frac{u_i^1 u_{i+1}^2 - u_i^2 u_{i+1}^1}{u_i u_{i+1} v_i}. \tag{1.4}$$

1.2. Curvature map. The algorithm computing the curvature map of any digital image is based on level lines shortening. The image level lines at given quantization levels are first extracted, then uniformly sampled with fine sub-pixel step, and smoothed by affine or curve shortening. Curvatures are then computed at each vertex of each level curve and associated to the dual pixels containing the vertex. A curvature image is eventually created by attributing to each dual pixel the average of all curvatures computed in it.

Algorithm 6: Curvature map

Input: Original Image u_0.
Output: Curvatures u_0 at scale t: $u(\cdot, t)$.

```
Extract the tree of level lines {Γ_0^{λ,i}}_{i∈F_λ;λ};
Sample uniformly each level line Γ_0^{λ,i}
for Level line Γ_0^{λ,i} do
    Γ_t^{λ,i} = Curve Shortening Flow (Γ_0^{λ,i});
for Γ_t^{λ,i} = {P_i(x_i, y_i)}_{i=0..n} do
    k_i = 1/R_i;
for each dual pixel do
    k = mean(k_{i_1}, k_{i_2}, ..., k_{i_m}).
```

Topological curvatures and scalar curvatures can be computed as well. Indeed, the information encoded in the tree enables the computation of signed

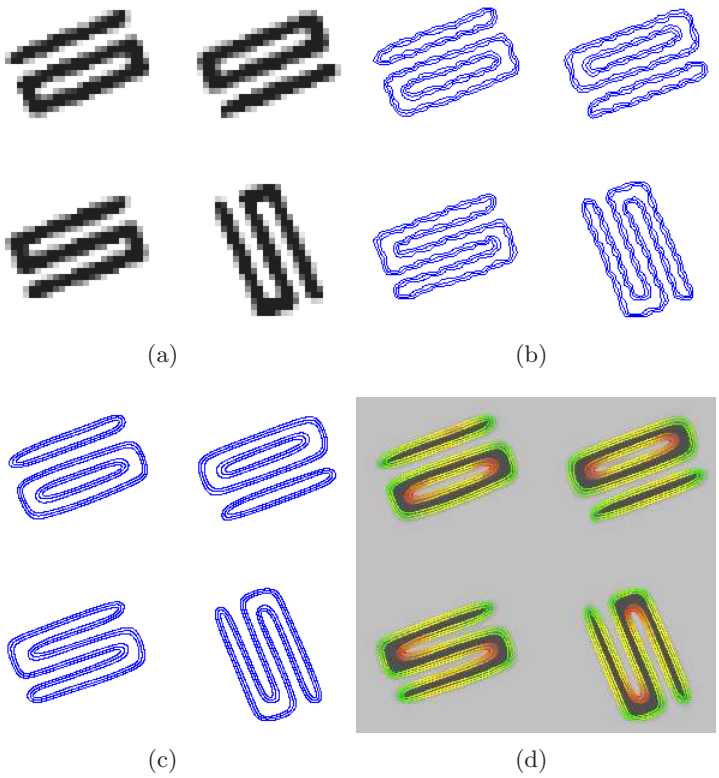
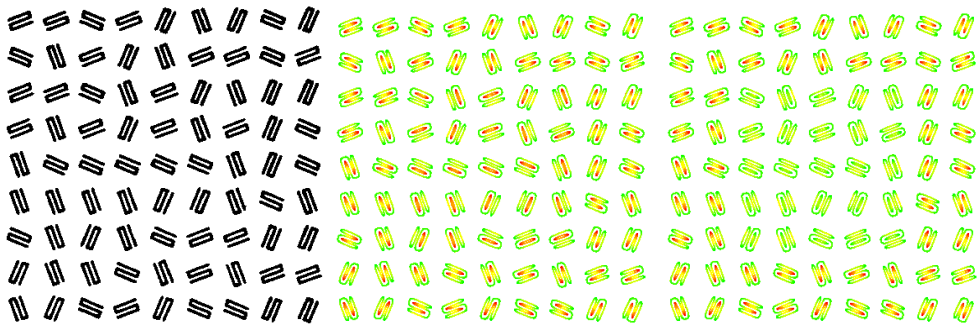


FIGURE 2. The curvature map numerical chain: (a) original image, (b) level lines, uniformly sampled, (c) evolved level lines, (d) curvature image.

curvatures, where the sign is either given by the gradient ascent, or by the topological orientation of the curve. In the first case, the curvature changes sign when the grey scale is reverted. Thus, a black disk and a white disk on grey background have opposite curvatures. The topological curvature is instead invariant to contrast changes. But it is nonlocal, since its sign depends on the global curve topology and not on the local curve shape. Figure 3 illustrates the difference on a famous Julesz texture discrimination experiment. On the left image, a pre-attentively undiscriminate texture pair. The "10" in random orientations surround a square made of "S". The signed curvature is identical for both shapes. The topological curvature (right) changes because the "0" have an interior circle missing in the "S". This proves that our perception does not compute the topological curvature. If it did, we would discriminate the two textures.

1.3. Curvature Microscope. By performing a scaled zoom on the considered image one can expect to have one level line passing through each dual pixel, and thus to observe more and more exactly the curvatures at microscopic scale. *The fact that all level lines are polygons with real coordinates allows one to zoom in the image at an arbitrary resolution.* This is necessary to explore visually the



(a) (b) (c)

FIGURE 3. (a) Original image, Julesz pair of undiscriminate textures (b). Signed curvatures, no discrimination (c). Topological curvatures: probably not computed in our perception, it would discriminate the texture pair.

intricacy of the local image structure. Hence the name of *curvature microscope* given to the final visualization.

Since the curve shortening is only defined for closed curves, a rule is needed for the level lines finishing on the image border. One could close these lines by joining their endpoints by (e.g.) a geodesic on the image boundary. But such junctions would create strong curvatures at the meeting points of the level lines with the image frame. To avoid this phenomenon a standard extrapolation is performed by flipping the image left and right, up and down and extending it in that way by a wide band.

For better rendering, the curvature map is printed over the smoothed image and the latter is attenuated (its gray values are concentrated around 128). Curvature values shade from red to green as follows: positive curvatures scale from red down to yellow; negative ones go down from yellow to green. Thus yellow means a small curvature. The image curvature microscope is a complex visualization tool dealing with three scale space parameters

(1) the zooming factor;
(2) the quantization step of the level lines;
(3) the renormalized smoothing scale (the scale l at which a circle of radius $r = l$ vanishes).

These parameters vary according to the total variation and the gradient amplitude of the image and therefore cannot be *a priori* fixed for any type of image. However, the zooming factor is proportional to the renormalized smoothing scale. The quantization step can be fixed once for all.

2. The curvature Gallery

After processing the pixelized level lines become accurate curves with subpixel control points, whose curvature can be faithfully computed. Thus the whole

FIGURE 4. Image curvature microscope. (a) the original image, 2X zoom and 4X zoom of the up-right corner; (b) curvature map computed on the original level lines with a quantization step $s = 36$; (c) curvature map computed on shortened level lines. The left column permits to observe the curvature densities. A zoom is necessary to observe the single curvatures. The middle column and right column focus more and more on shape and texture details.

chain can be viewed as a numerical preprocessing before further numerical analysis and feature extraction. Indeed, after processing, the curvature extrema are easily detected. But there is also a strong interest in the direct visualization of the level lines and of the microscopic curvature map of an image. The following

gallery on a variety of image details illustrates the recovery of shapes freed from their aliasing, JPEG, and noise artifacts.

2.1. Attneave's cat. Short time smoothing reveals useful invariant features (curvature extrema, inflection points, angles and junctions). Therefore, as pointed out by Attneave, objects are represented with great economy and striking fidelity by marking the points at which their contours change direction maximally. In Figure 5, the head of the Attneave cat is scanned and processed by LLAS. Before filtering, the curvature values reflect essentially the pixel staircases: Positive and negative curvatures in red and green alternate along contours. A visual inspection shows that, after LLAS, the level curves can be easily segmented into concave and convex parts, separated by flat parts (in yellow).

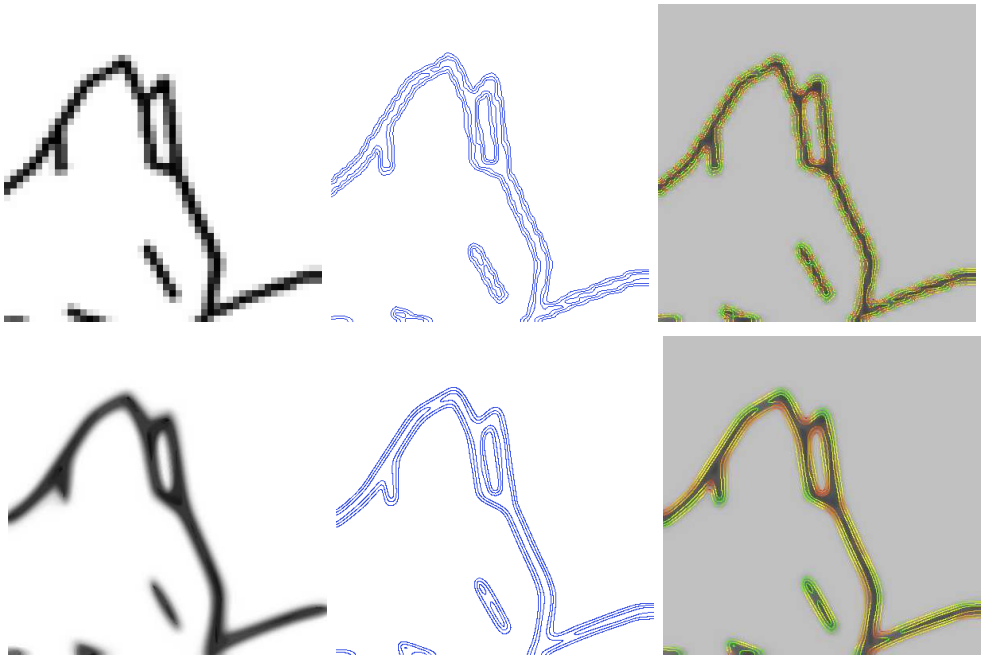


FIGURE 5. Zoom on the Attneave cat, its corresponding level lines and curvatures. smoothed image, affine smoothed level lines and curvature map after filtering.

2.2. Geometric shapes. The same improvements can be demonstrated on the geometric drawings of Figures 6, 7 and 8. A straight oblique line appears serrated because of its pixel representation. Thus the right angle that it forms with another line is simply lost in clutter: there are locally right angles everywhere. When a curve stops onto another curve, T-junctions or Y-junctions are created. In such cases, our perception tends to interpret the interrupted curve as the boundary of some object undergoing occlusion. In the image on the left of Figure 7, which is a typical Kanizsa experiment demonstrating our layered perception, one tends to see a grey rectangle on top of a black polygon. The

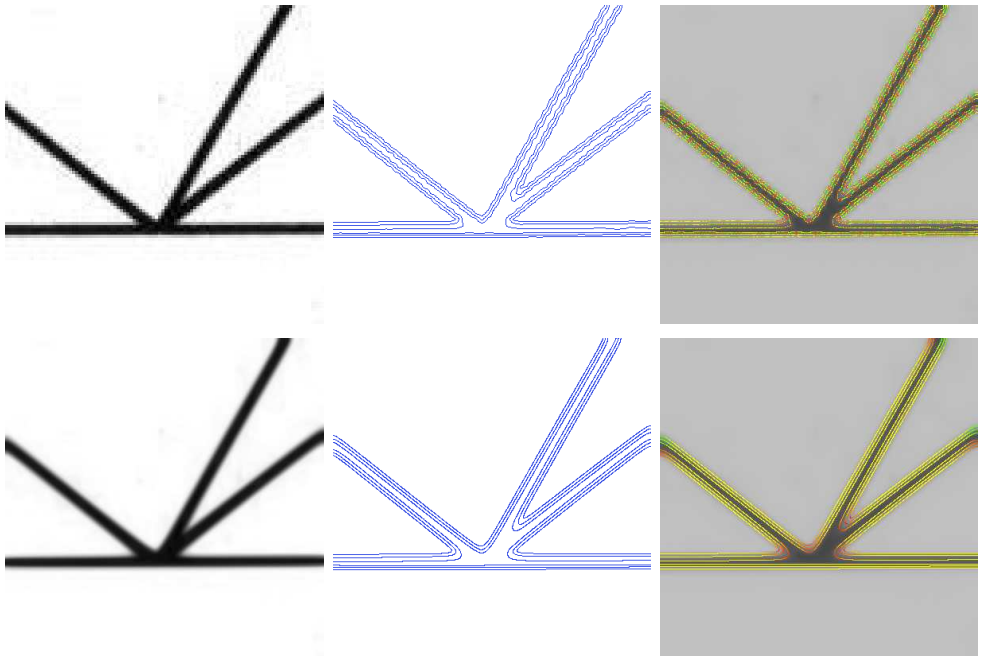


FIGURE 6. Top: original image, extracted level lines and corresponding curvatures. Bottom: smoothed image and curvature map after filtering.

T-junctions creating this layered illusion can be detected by their adjacent positive and negative curvatures. Note that a short time smoothing is necessary to extract these meaningful curvatures from the clutter of oscillating curvatures due to the staircase effect. Another series of typical experiments was dedicated by Kanizsa to the transparency illusion, by which, in presence of X-junctions, our perceptions infers the presence of two objects on top of each other, the upper one being transparent. For instance the left image of Figure 8 is spontaneously described by viewers as a grey transparent disk in front of a black wedge. Kanizsa [**14**] pointed out the paradox of such a description, which sees two objects where there are in fact four regions with different grey levels. The local configuration responsible for the transparency illusion is the X-junction, seen as the apparent crossing of the boundaries of the disk and of the black wedge. As illustrated after applying LLAS to the figure, X junctions can be detected as a particular configuration of adjacent negative, positive, and zero curvatures.

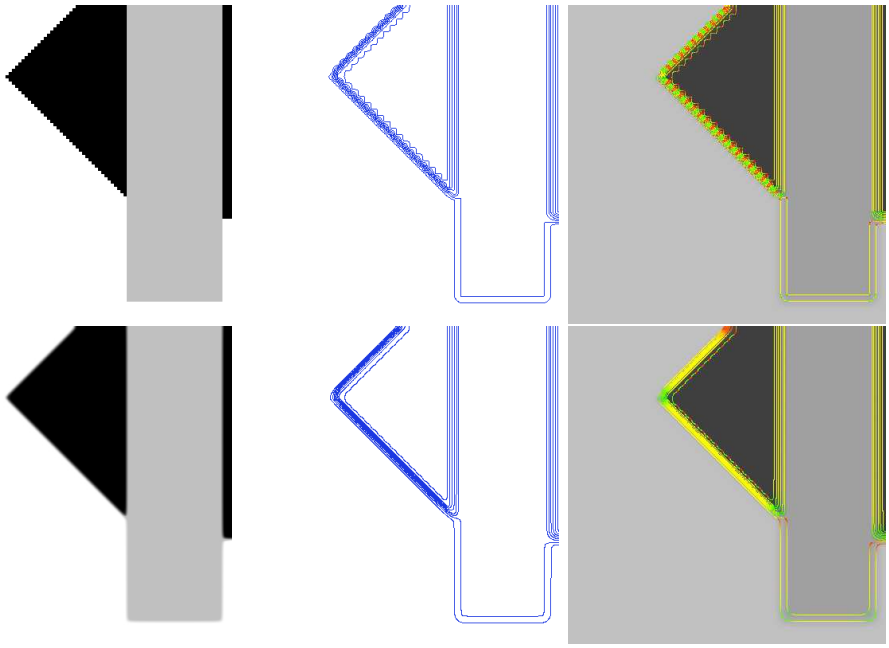


FIGURE 7. Original image, non-filtered curvatures, smoothed level lines by LLAS and curvature map.

2.3. Graphics and aliasing. Aliasing due to pixelization is common in scanned documents. As illustrated by all experiments, LLAS can be used for a graphic quality improvement smoothing contours. This is actually done at the cost of smoothing out corners and junctions, but this smoothing is necessary to single them out as the stable peaks of curvature. All in all, in most zoomed-in figures the improvement is manifest, starting with the laughing mouse of Figure 9. A decent recovery is possible even with badly pixelized shapes such as the one reproduced in Figure 10. This drawing is not perfectly restored because of the fattening effect at junctions, but it definitely improves on the original, and opens the way to a geometric analysis that would be impossible on the original. But the example in Figure 11 demands the impossible. Although some undulating curves still may be figured out by an intelligent viewer, the figure locally is nothing but a checkerboard at pixel size. Thus the curvature motion removes all squares, black and white, and creates a huge fattening effect.

2.4. Pre-attentively undiscriminable textons. Julesz conjectured in his second texture perception theory [**13**] that two different textures cannot be pre-attentively discriminated if they have the same texton density. For instance the Julesz patterns in Figure 12 are different, but have the same "texton densities", namely the same number of bars, corners, and terminators. After filtering, the microscopic curvature map will permit to compute a density of positive, negative and zero curvatures.

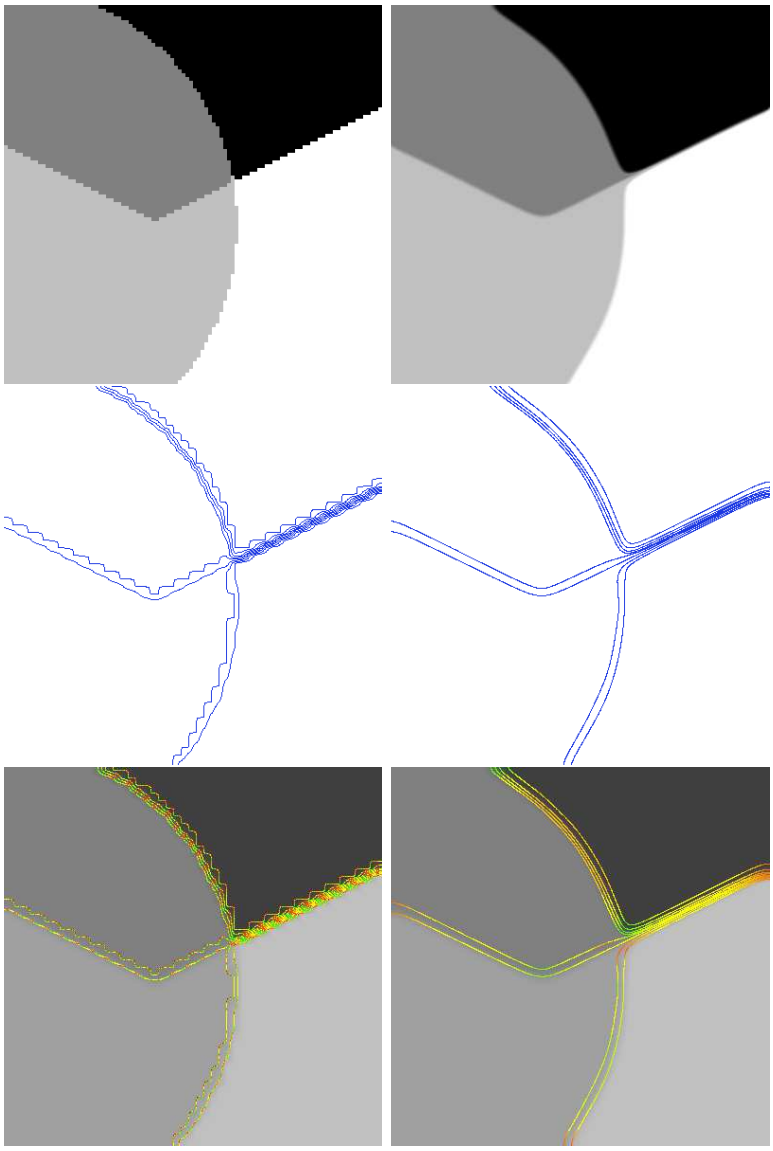


FIGURE 8. Left: original image, bilinear level lines and the curvature map before filtering. Right: smothed image, affine smoothed level lines and curvature map after filtering.

2.5. Bacteria morphologies. Bacteria shapes are determined by the bacterial cell wall and cytoskeleton. The curvature is an intrinsic geometrical descriptor, useful for shape discrimination. In Figure 13 we display 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. In microbiology, many tasks involve the counting of geometrically simple objets. An accurate curvature filter permits to make curvature statistics.

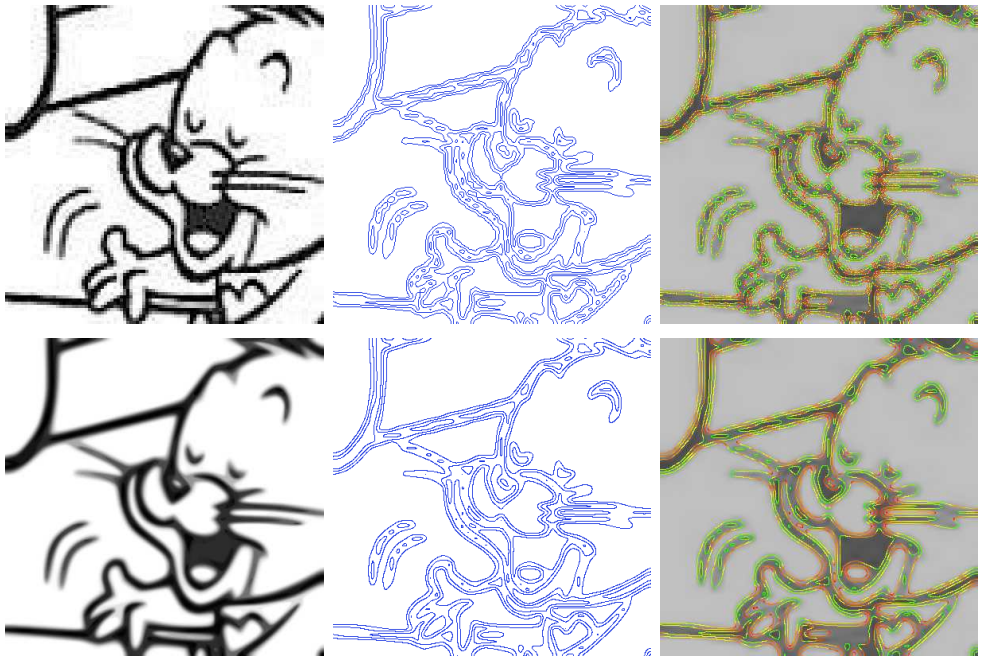


FIGURE 9. Top: original image, its corresponding level lines and curvatures. Bottom: smoothed image, affine smoothed level lines and curvature map after filtering.

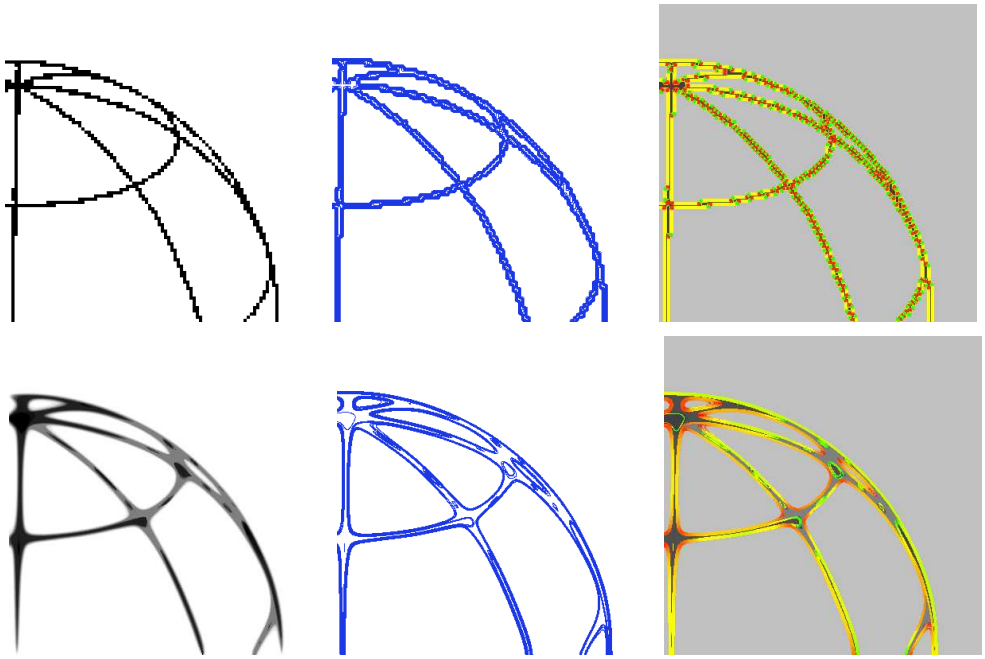


FIGURE 10. Top: original image, its corresponding level lines and curvatures. Bottom: smoothed image, affine smoothed level lines and curvature map after filtering.

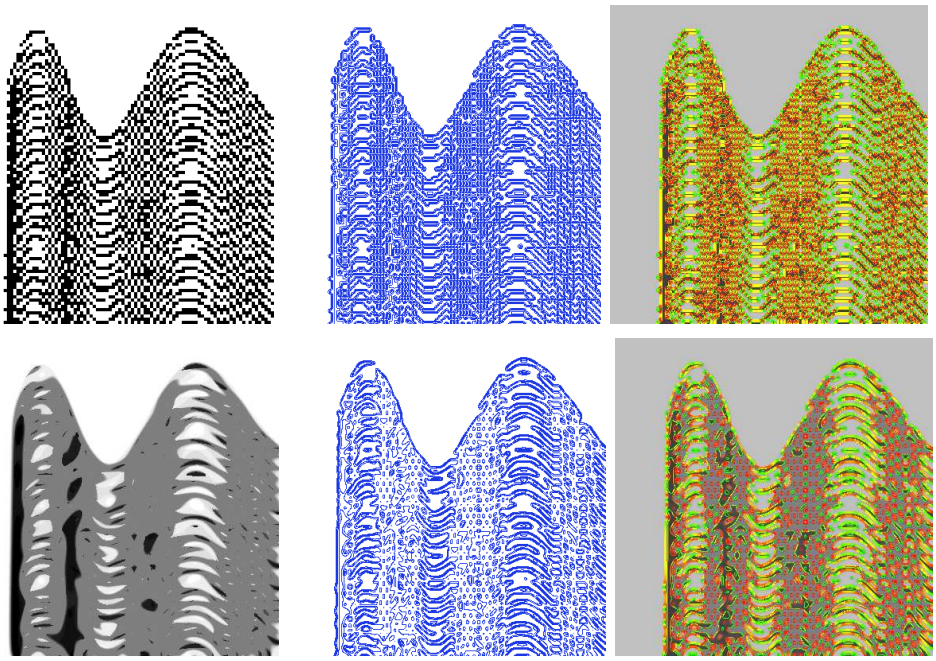


FIGURE 11. Top: original image, its corresponding level lines and curvatures. Bottom: smoothed image, affine smoothed level lines and curvature map after filtering.

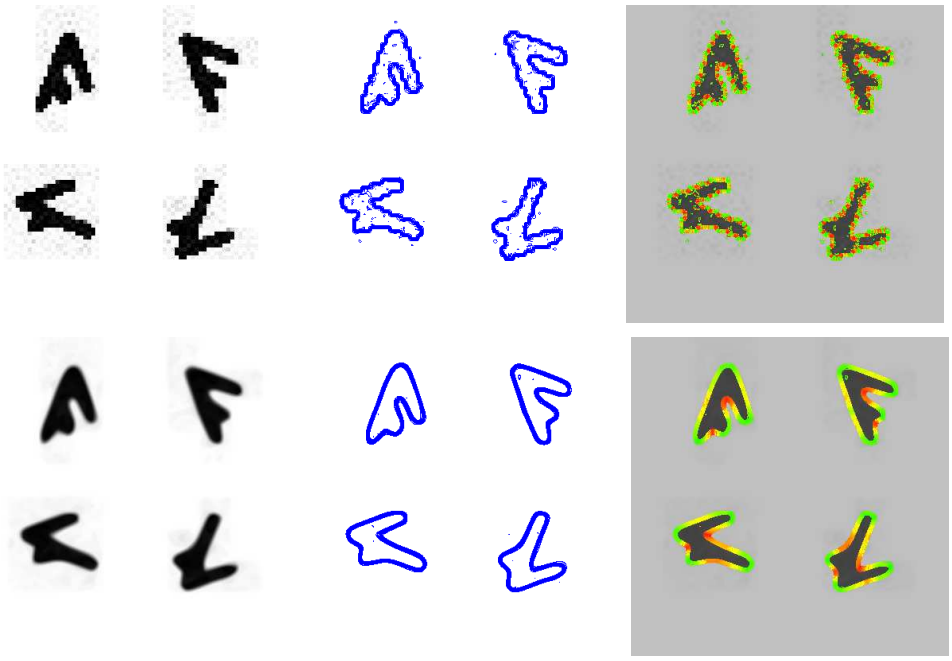


FIGURE 12. Top: original image, its corresponding level lines and curvatures. Bottom: smoothed image, affine smoothed level lines and curvature map after filtering.

2.6. Topography. Digital elevation models represent ground surface topography. Gray levels indicate ground elevation (lightest shades for highest elevations) and therefore the image level lines are true level lines. As can be seen in

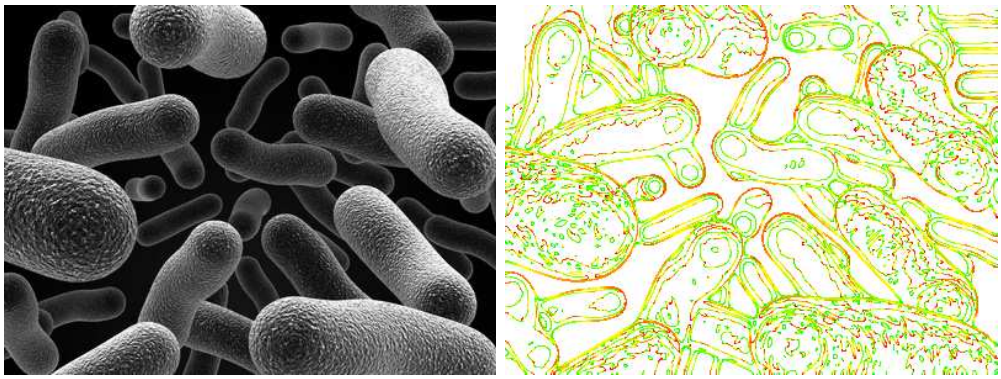


FIGURE 13. (a). Original image (b). Curvature Map

Figure 14, the set of level lines of a digital image is a natural representation of the shape contents, because it provides topological information invariant to contrast changes. The bilinear interpolation is the most local of continuous interpolations preserving the order between the gray levels of the image. Because the interpolation is continuous, level lines with different gray levels never touch. However, they are concatenations of pieces of hyperbolae and straight segments and hence present oscillations along transverse contours. A short time smoothing reduces the oscillations and straightens up the edges. The remaining curvature extrema after filtering become relevant as geometric shape descriptors.

2.7. Textures. The experiments of Figures 15 and 16 illustrate the potential use of LLAS to restore the image micro-geometry and to facilitate the identification of smoothly varying shapes in a texture.

2.8. Paintings. Even on details of paintings, this geometric analysis can be relevant. As already mentioned, the smoothed can be used for noise reduction and picture restoration. In Figure 17 the desaliasing successfully restores the paint strokes and improves for example the perception of the pearls and of their shadows. Leonardo's portrait of Mona Lisa is remarkable for its *sfumato* technique of soft shaded modeling. The stylistic motifs are reflected in the fact that level lines fall widely apart like if it were a very blurry image. The experiment of Figure 18 demonstrates the amazing sparsity of visual information in the Mona Lisa. It is only by a few level lines, falling widely apart, and with very smooth corners, that all nuances of the Mona Lisa face are suggested.

2.9. Text processing. The same good effects are observable with pixelized written text. After the application of LLAS the image in Figure 19 retrieve a curvature signature that is obviously usable for handwriting recognition. To that aim the *causality* of the process is essential: no creation of new levels and no creation of new curvatures.

2.10. Fingerprints restoration and discrimination. Minutiae such as cores, bifurcations and ridge endings characterize uniquely fingerprints. Their detection requires a careful smoothing, particularly to avoid a spurious diffusion

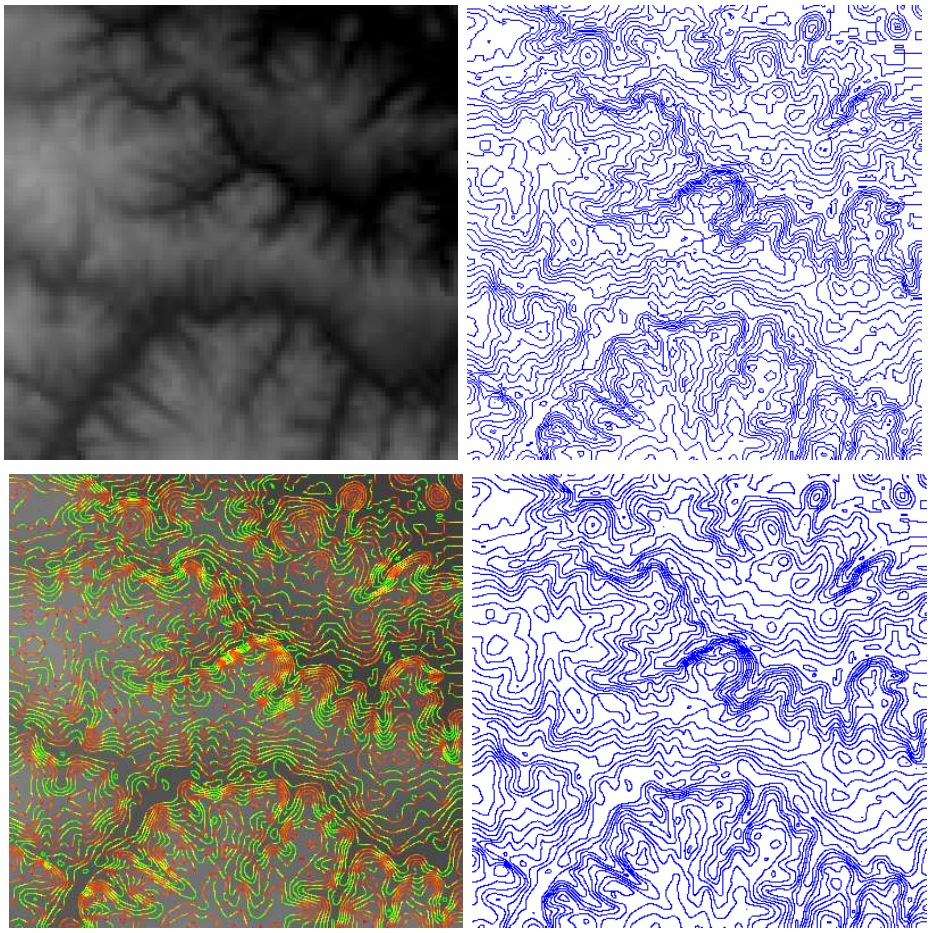


FIGURE 14. Digital elevation map, its corresponding level lines (for once a real topographic map), the affine smoothed level lines and their curvature map.

mixing the ridges. The main objective of smoothing is to sieve the curvature extrema. Indeed, many are present everywhere on the ridge borders before smoothing. LLAS removes these ridge border oscillations and provides a smooth version of the fingerprint on which the curvature map locates its characteristic points. Figure 20 shows the results and compares LLAS to finite difference scheme and to the level set method. Observe that by performing pixel evolutions, the ridge endings shrink fast, and the islands and crossovers diffuse. The subpixel smoothing instead tears apart ridges and emphasizes crossovers.

3. Conclusion

The first outcome of the Level lines Shortening algorithm is the evolved image, which presents some sort of denoising, simplification, and desaliasing. But the main outcome is an accurate curvature estimate on all level lines. As a visualization tool, the fact that all level lines are polygons with real coordinates allows to zoom in the image at an arbitrary resolution. This is necessary to explore

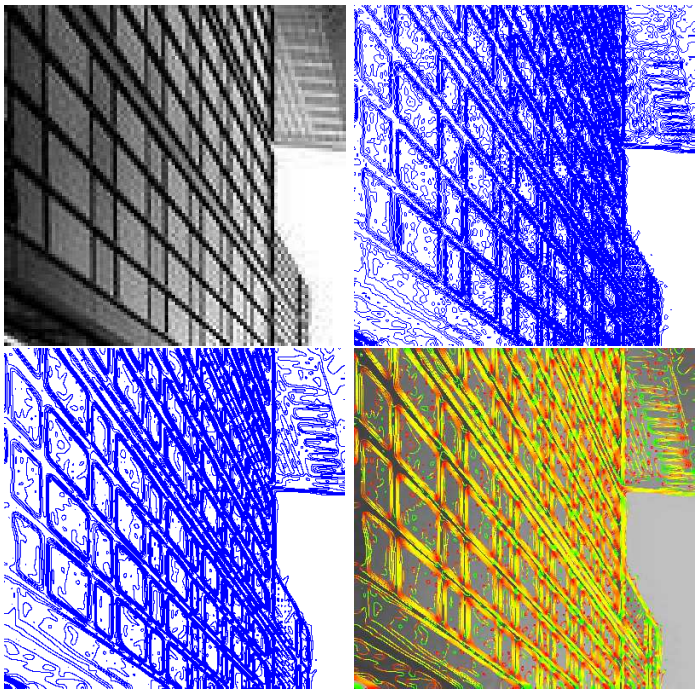


FIGURE 15. Original image, extracted level lines, affine smoothed level lines and curvature map.

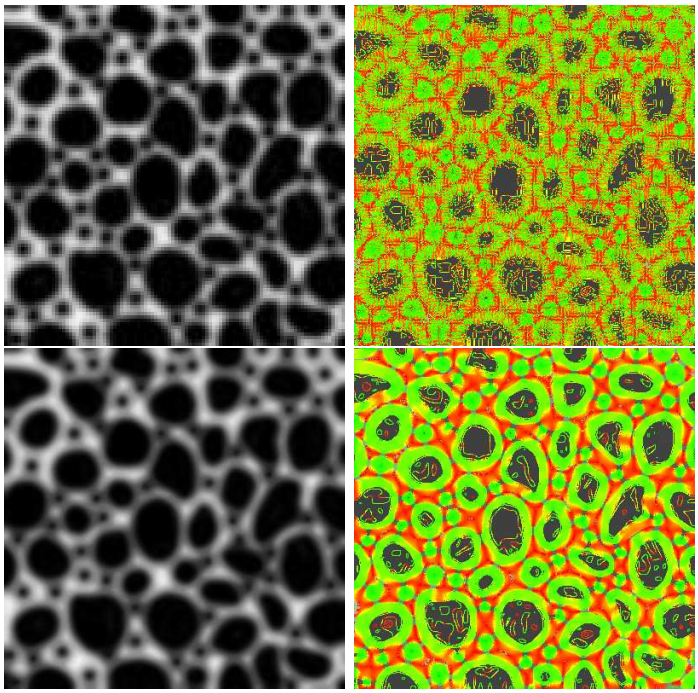


FIGURE 16. Original image, extracted level lines, affine smoothed level lines and curvature map

visually the intricacy of the local image structure. Hence the name of *curvature microscope* given to the final visualization.

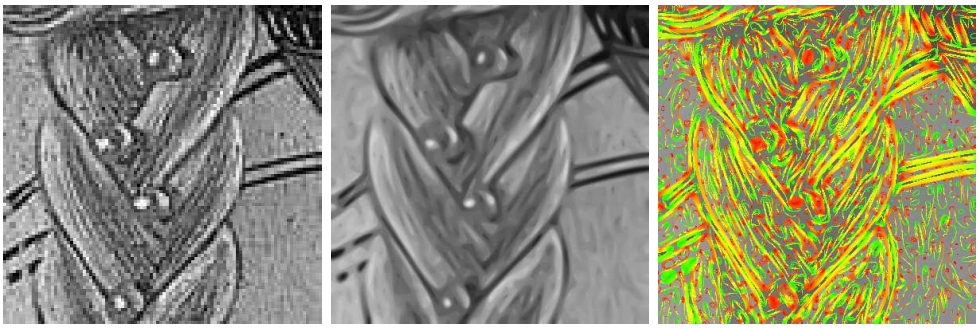


FIGURE 17. Original photo-painting, smoothed image and curvature map after filtering.

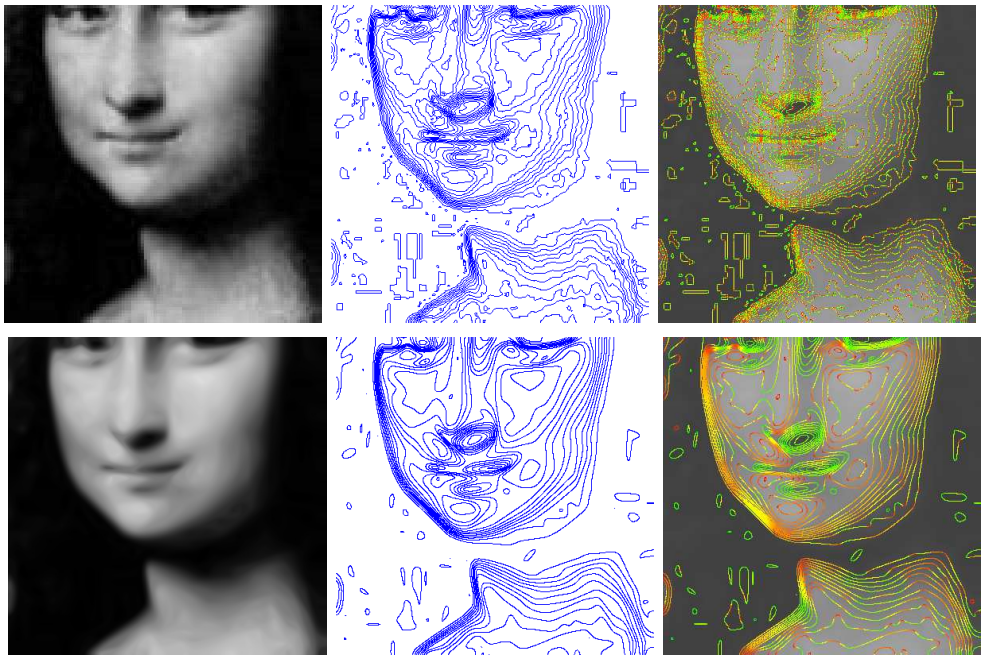


FIGURE 18. Extraction with zoom of *Mona Lisa* photograph, its corresponding level lines and curvatures. LLQS evolution, affine smoothed level lines and curvature map after filtering.

FIGURE 19. Left: original image, bilinear level lines and the curvature map before filtering. Right: smothed image, affine smoothed level lines and curvature map after filtering.

FIGURE 20. Original fingerprint, smoothed fingerprint and its curvature map after filtering.

CHAPTER 4

IPOL internships

1. Bilinear Tree of Level Lines. Direct Extraction Algorithm

Shape analysis can be led to the study of Jordan curves which we shall call "elementary shapes". The many experiments where we display level lines of digital images make clear enough why a smoothing is necessary to restore their structure. These experiments also show that we can in no way assimilate these level lines with our common notion of shape as the silhouette of a physical object in full view. Indeed, in images of a natural environment, most observed objects are partially hidden (occluded) by other objects and often deformed by perspective. When we observe a level line we cannot be sure that it belongs to a single object; it may be composed of pieces of the boundaries of several objects that are occluding each other. Shape recognition technology has therefore focused on local methods, that is, methods that work even if a shape is not in full view or if the visible part is distorted.

The topographic map provides a complete representation of an image. This representation is well suited for shape analysis and recognition, since it is based on the geometrical information of images, and can be embedded in a tree structure.

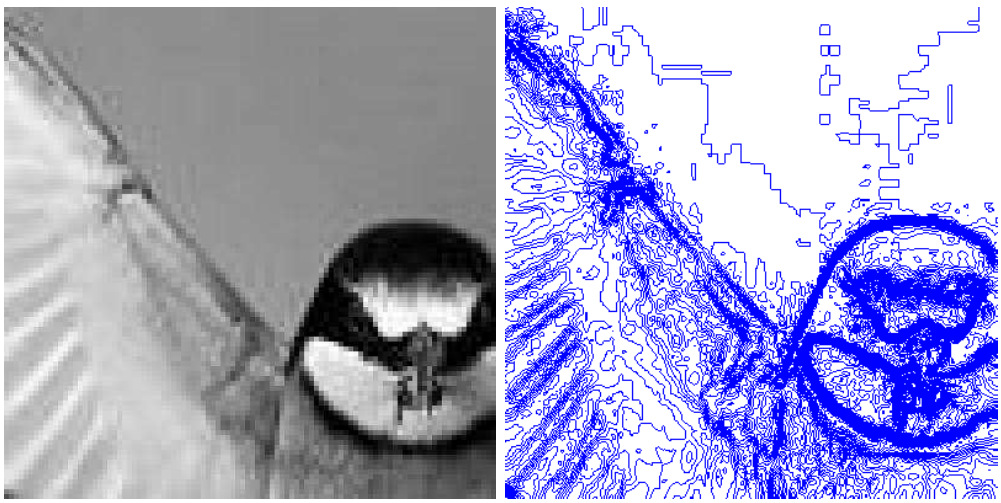


FIGURE 1. Extraction of level lines from a bilinear interpolated image, with quantization step 16. Singular levels are avoided. Nevertheless, level lines are highly oscillatory, especially along transversal lines. For this reason, before computing any features on these level lines, a previous smoothing is necessary.

Following [**6**] we propose the implementation of **a fast numerical method for extracting the topographic map and the vizualization of level lines**. The algorithm should be based on the Direct Decomposition Algorithm, described in Chaper 2.

To build the tree structure of the extracted level lines, we consider all intersection points of level lines with vertical edgels and order them. While scanning a column of Qedgels, we are in the interior domain of a level line if we have crossed it an odd number of times. If we meet a level line that has no parent yet, its parent has to be the last level line we are in. For the root, we add the boundary of the image as a level line to the list L at the beginning.

Let L be the list of level lines. To describe the intersection y of a level curve Γ_{id}, having index id in the list L, with Qedgel at column $i + 0.5$, we use the triplet (i, y, id). The main idea used in the ordering algorithm is to look for the innermost shapes that we denote generically by Γ. The arrow $\Gamma \leftarrow \tilde{\Gamma}$shorlty says that we set the curve Γ to be now $\tilde{\Gamma}$. And finally, we denote the parent of a level line by $Parent(\Gamma)$.

Algorithm 7: Ordering the level lines in a tree.

Input: List L of bilinear level lines Γ

Output: Fill tree structure

```
1 Collect all (i, y, id) in array V;
2 Order V lexicographically by key (i, y);
3 Γ ← ∅;
4 for all (i, y, id) ∈ V do if Γ = Γ_id then
5     Γ ← Parent(Γ);
6     else if Γ ≠ ∅ and Parent(Γ_id) = ∅ then
7         set Γ_id as a child of Γ;
8         Γ ← Γ_id.
```

This algorithm relies on the fact that the quantization is chosen so that each level line crosses Qedgels, but does not contain any. For this, it is sufficient that it avoids the levels at the centers of pixels (the initial data). In particular, regional extrema of the image cannot be extracted by the algorithm.

2. Curve evolution by a power of curvature. Simultaneous erosion and dilation

Shape recognition technology has therefore focused on local methods, that is, methods that work even if a shape is not in full view or if the visible part is distorted. As a consequence, image analysis adopts the following principle: *Shape recognition must be based on local features of the shape's boundary, in this case local features of the Jordan curve, and not on its global features. If the boundary has some degree of smoothness, then these local features are based on the derivatives of the curve, namely the tangent vector, the curvature, and so on.* Many local recognition methods involve the "salient" points of a shape, which are the points where the curvature is zero (inflection points) and points where the curvature has a maximum or minimum (the "corners" of the shape). But any feature computation requires a careful smoothing, in order to get rid of, or reduce at least, any undesired artifacts.

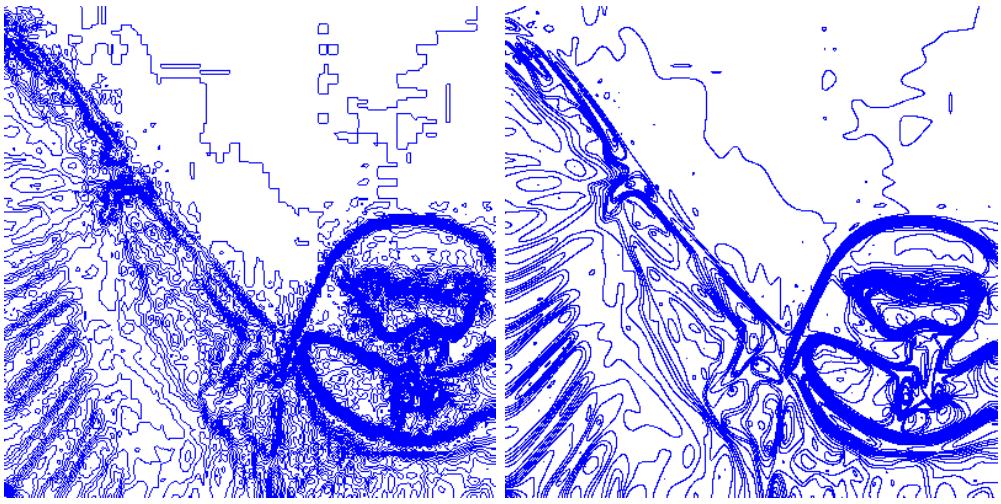


FIGURE 2. Simultaneous curve evolution by the intrinsic heat equation. The evolved curves are smooth for all times and oscillations reduce considerably. The level curves are included one into the other and conserve this property when applying the smoothing.

We propose as an internship **an analysis of the famous Mackworth and Mokhtarian algorithm consistent with curve shortening (CS) and variants of it for a power of curvature**.

We remind that this algorithm applies to a plane curve the non-linear heat equation, by successively convolving the arc length parameterization $\mathbf{x}(\cdot, t)$ at time n with a Gaussian kernel G_h of standard deviation proportional to $h^{\frac{1}{2}}$.

In practice, a level line is perceived as a signal. The short time convolution with a gaussian reduces to a discrete convolution given by

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m].$$

Algorithm 8: Discrete Curve Shortening (CS)

Input: Polygon Γ_0, gaussian signal G
Output: Evolved polygon Γ_n, after n iterations
for *all* $i = \overline{0,n}$ **do**
    sample uniformly curve Γ_i ;
    convolve curve Γ_i with G.

In many situations, discrete convolutions can be converted to circular convolutions so that fast transforms with a convolution property can be used to implement the computation.

$$(f * g_N)[n] \equiv \sum_{m=0}^{N-1} \left(\sum_{k=-\infty}^{\infty} f[m + kN] \right) g_N[n - m].$$

In particular, when applying the convolution the number of samples of the original signal (level line) doesn't change. One can then find out that $f[n]$ evolves into $(f * g)[n]$ and therefore have (the approximation) of the normal direction $\boldsymbol{n} = (f * g)[n] - f[n]$, as well as the amplitude of the curvature.

Taking advantage of this property, it can be easily obtained an evolution by a power of curvature (in general, sufficiently small), just by re-projecting the evolved point $(f * g)[n]$ in the normal direction, at the corresponding distance.

When the power of the curvature goes to zero, we obtain the so called erosion-dialation filter, which is described by the following equation

$$\frac{\partial \mathbf{x}}{\partial t} = |\boldsymbol{\kappa}|^{-1} \boldsymbol{\kappa}(\mathbf{x}(t)). \tag{GCS}$$

This equation solves the famous problem never fixed in Mathematical Morphology: the basic equations are erosions and dilations. Erosions correspond to the minus sign (the case where $\boldsymbol{\kappa}$ points towards the interior of the curve) in the above equation, and dilation to the plus sign in the above equation. Therefore the above equation is new and solves the long dilemma unsatisfactorily solved by the so called *alternate filters*: should we make a tiny erosion first, then a tiny dilation, and go on doing this with bigger and bigger erosions-dilations, to attain a final joint "erosion-dilation"? No, this is only a trick which somehow describes the algorithm which will be much better implemented with the above equation.

3. Semi-Implicit Level Set Methods for Curve and Surface Motions

The level set method, developed by Osher and Sethian in [3] has had large impact on computational methods for interface motion and is now being used by engineers, physicists and mathematicians for a wide variety of problems in image processing or surfaces reconstruction.

We suggest as an IPOL project **an analysis of the semi-implicit method proposed by P. Semereka in [1] for simulating surface and curve diffusion by mean curvature motion (even for a larger class of curvature flows)**.

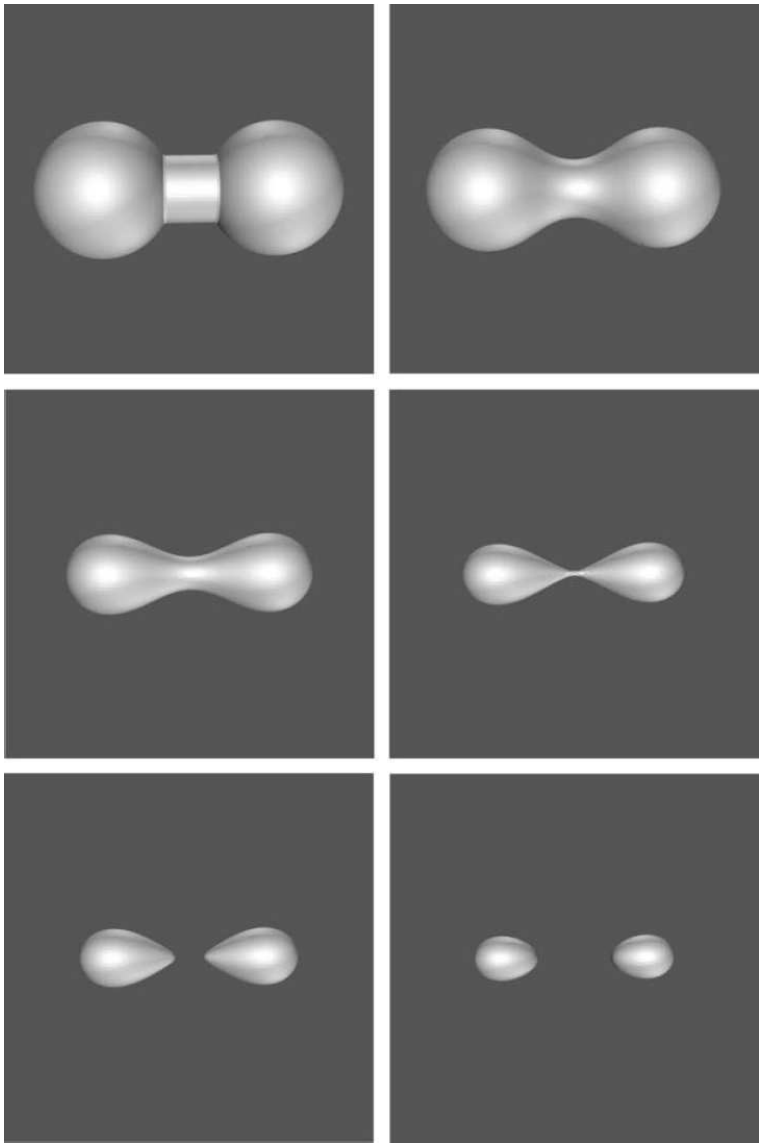


FIGURE 3. Motion by mean curvature in 3D.

In the level set method, the interface Γ is represented implicitly as the zero level set of a continuous function which we will denote as u; therefore we write

$$\Gamma(t) = \{x; u(x,t) = 0\}.$$

We wish to consider the motion of interface whose normal speed is equal to the mean curvature, that is the PDE formulation for the intrinsic heat equation

$$u_t = curv(u)|Du| \tag{3.5}$$

The semi-implicit algorithm for curvature flow is based the formula,

$$curv(u)|Du| = \Delta u - N(u)$$

where

$$\Delta u = u_{xx} + u_{yy} \text{ and } N(u) = \frac{u_x^2 u_{xx} + 2u_x u_y u_{xy} + u_y^2 u_{yy}}{u_x^2 + u_y^2}.$$

Based on this observation we discretize equation (3.5) in time as follows:

$$u_{n+1} = u_n + dt\Delta u_{n+1} - dtN(u_n) \tag{3.6}$$

where dt is the time step.

This discretization is first order operator splitting where one first takes one step of forward Euler on the nonlinear term followed by one step of backward Euler on the linear term. The equation is spatially discretized by expressing Δu and $N(u)$ explicitly in terms of first and second order partial derivatives and then discretizing using center differences.

$$\begin{aligned} u_x &= \frac{u(i+1,j) - u(i-1,j)}{2dx} \\ u_{xx} &= \frac{u(i-1,j) - 2u(i,j) + u(i+1,j)}{dx^2} \\ u_{xy} &= \frac{u(i-1,j-1) + u(i+1,j+1) - u(i-1,j+1) + -u(i+1,j-1)}{dxdy} \end{aligned}$$

In order to implement the scheme given by 3.6 we must invert the operator $I - dt\Delta_h$. This is done using a FFT (Fast Fourier Trasnform), where Δ_h is the standard discrete form of the Laplacean based on center differencing.

References

[1] Smereka, Peter, Semi-Implicit Level Set Methods for Curvature and Surface Diffusion Motion, Journal of Scientific Computing 19 (1-3): 439-456, 2003;

[2] J.A. Sethian, Level Set Methods and Fast Marching Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision, and Materials Sciences, Cambridge University Press, Cambridge, 1999.

[3] Stanley Osher , James A. Sethian, Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations, Journal of Computational Physics, v.79 n.1, p.12-49, Nov. 1988

Bibliography

[1] L. Alvarez, F. Guichard, P.-L. Lions, and J.-M. Morel. Axioms and fundamental equations of image processing. *Arch. Rational Mech. Anal.*, 123(3):199–257, 1993.

[2] H. Asada and M. Brady. The curvature primal sketch. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(1):2–14, 1986.

[3] F. Attneave. Some informational aspects of visual perception. *Psychological review*, 61(3):183–193, 1954.

[4] Guy Barles and Christine Georgelin. A simple proof of convergence for an approximation scheme for computing motions by mean curvature. *SIAM J. Numer. Anal.*, 32(2):484–500, 1995.

[5] V. Caselles, B. Coll, and J.-M. Morel. A Kanizsa program. *Progress in Nonlinear Differential Equations and their Applications*, 25:35–55, 1996.

[6] V. Caselles and P. Monasse. *Geometric Description of Images as Topographic Maps*, volume 1984 of *Lecture Notes in Mathematics*. Springer, 2010.

[7] Yun Gang Chen, Yoshikazu Giga, and Shun'ichi Goto. Uniqueness and existence of viscosity solutions of generalized mean curvature flow equations. *J. Differential Geom.*, 33(3):749–786, 1991.

[8] L. C. Evans and J. Spruck. Motion of level sets by mean curvature. I. *J. Differential Geom.*, 33(3):635–681, 1991.

[9] Lawrence C. Evans. Convergence of an algorithm for mean curvature motion. *Indiana Univ. Math. J.*, 42(2):533–557, 1993.

[10] M. Gage and R. S. Hamilton. The heat equation shrinking convex plane curves. *J. Differential Geom.*, 23(1):69–96, 1986.

[11] Matthew A. Grayson. The heat equation shrinks embedded plane curves to round points. *J. Differential Geom.*, 26(2):285–314, 1987.

[12] H. Ishii. A generalization of the Bence, Merriman and Osher algorithm for motion by mean curvature. In *Curvature flows and related topics (Levico, 1994)*, volume 5 of *GAKUTO Internat. Ser. Math. Sci. Appl.*, pages 111–127. Gakkōtosho, Tokyo, 1995.

[13] B. Julesz. Textons, the elements of texture perception, and their interactions. *Nature*, 290(5802):91–97, 1981.

[14] G. Kanizsa. *Organization in Vision: Essays on Gestalt Perception*. Praeger, 1979.

[15] A. Mackworth and Mockhtarian F. Scale-based description and recognition of planar curves and two-dimensional shapes. *IEEE Trans. Pattern Analysis and Machine Intell.*, 8(1):34–43, 1986.

[16] B; Merriman, J. Bence, and S. Osher. Diffusion generated motion by mean curvature. *J. E. Taylor, Editor, Computational Crystal Growers Workshop*, pages 73–83, 1992.

[17] Lionel Moisan. Affine plane curve evolution: a fully consistent scheme. *IEEE Trans. Image Process.*, 7(3):411–420, 1998.

[18] P. Monasse and F. Guichard. Fast computation of a contrast invariant image representation. *IEEE Trans. on Image Proc*, 9:860–872, 1998.

[19] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. *J. Comput. Phys.*, 79(1):12–49, 1988.

[20] F. Pollick and G. Sapiro. Constant affine velocity predicts the 1/3 power law of planar motion perception and generation. *Vision Research*, 37(3):347–353, 1997.

[21] G. Sapiro and Tannenbaum A. Affine invariant scale space. *IJCV*, 11(1):25–44, 1993.

[22] Guillermo Sapiro, Albert Cohen, and Alfred M. Bruckstein. A subdivision scheme for continuous-scale B-splines and affine-invariant progressive smoothing. *J. Math. Imaging Vision*, 7(1):23–40, 1997.

[23] Guillermo Sapiro and Allen Tannenbaum. On invariant curve evolution and image analysis. *Indiana Univ. Math. J.*, 42(3):985–1009, 1993.

[24] Guillermo Sapiro and Allen Tannenbaum. On affine plane curve evolution. *J. Funct. Anal.*, 119(1):79–120, 1994.