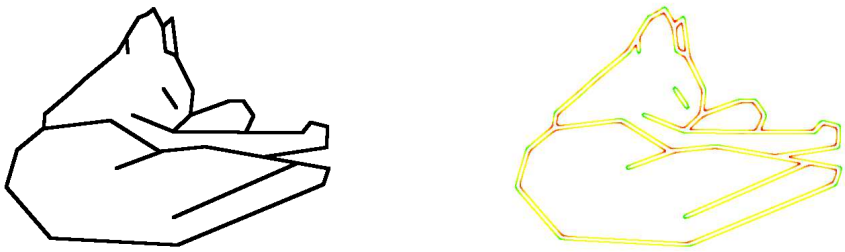
AN IMAGE CURVATURE MICROSCOPE *

A. CIOMAGA †, P. MONASSE ‡, AND J.M. MOREL †

Abstract. This paper presents a review, analysis and comparison of numerical methods implementing the curvature motion and the affine curvature motion for 2D images, shapes, and curves. These *curvature scale spaces* allow, in principle, to compute an accurate multiscale curvature in digital images. The fastest and most invariant of them can be used in a complete image processing chain. This numerical chain simulates the accurate *sub-pixel* evolution of an image by mean curvature motion or by affine invariant curvature motion. To do so, it lets all the level lines of the image evolve by curvature shortening (of affine shortening), computes the image curvature directly on the smoothed level lines, and reconstructs the evolved image and its curvatures in an intrinsic, grid-independent representation. The paper describes a careful implementation of this chain, and analyzes its effects on many examples. The microscopic visualization of an image curvature map reveals after processing many image details. This image process improves graphic images, gets rid of compression and aliasing effects. It also gives an accurate tool to explore the validity of Attneave's and Julesz theories on shape perception and texture discrimination. The "curvature microscope" runs on line for any image at `http://www.ipol.im/pub/algo/cmmm_image_curvature_microscope/`.

Key words. partial differential equations, mean curvature motion, affine curvature motion, image shape analysis, image reconstruction, curvature scale spaces, scientific visualisation, curvature scale space, topographic map, level lines.

1. Introduction. Attneave's founding 1954 paper [4] 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)"*. 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.



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

FIG. 1.1. *Attneave's figure illustrating the prominent role of curvature peaks in image perception and its curvature map computed by level lines shortening.*

*Research partially financed by the MISS project of Centre National d'Etudes Spatiales, the Office of Naval research under grant N00014-97-1-0839 and by the European Research Council, advanced grant "Twelve labours".

†CMLA, ENS Cachan, 61 Avenue du Président Wilson, F-94230 Cachan, (`ciomaga@cmla.ens-cachan.fr`), (`morel@cmla.ens-cachan`)

‡Université Paris-Est, LIGM/IMAGINE, 19 rue Alfred Nobel, 77455 Marne-la-Vallée, (`monasse@imagine.enpc.fr`)

This explains why, in one of the first serious attempts to cope with this numerical challenge, Asada and Brady [3] 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 [18] proved that the intrinsic heat equation smooths Jordan curves and preserves their topology. The Osher-Sethian level set method [39] implements the motion by mean curvature of an embedded manifold by applying the mean curvature PDE to its signed distance function. Evans-Spruck [15] and Chen-Giga-Goto [11] 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 [32] and later proved by several authors [6], [16], [21].

In parallel, Mackworth and Moktharian [29] 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* [9]. A fast algorithm computing the topographic map was developed by Monasse and Guichard in [36]. Sapiro and Tannenbaum [41] 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 [35].

The present paper starts with a review of the main classes of curvature algorithms, focusing on isotropic curvature equations and on the two curvature powers that are relevant for image analysis, namely 1 and 1/3. Then, building on the above mentioned contributions, 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. The results herein were announced in [12].

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]. The above described numerical chain was outlined

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

in [25] and also in [28] where shape recognition algorithms were explored. The chain will be completed here with a subpixel image reconstruction from an arbitrary tree of level lines, which results in a powerful visualization tool.

There are several definitions of curvature and of multiscale curvature, and we shall detail them before entering into the discussion of how to compute them. The next two sections review and compare the various types of numerical analysis for curvature motion, and clarify the links between them. Section §4 describes in detail the Level Lines Shortening (LLS) and its variant the Level Lines Affine Shortening (LLAS). The Image Curvature Microscope is described in section §5 where many numerical comparisons are performed. Section §6 is devoted to illustrative experiments on a choice of image parts containing contours, shapes and textures.

2. Curvature Scale Spaces.

2.1. Curvatures. Digital images are given in discrete sampled form on a rectangle Ω but the underlying continuous substratum is assumed to be C^∞ and interpolated as such on Ω. By Sard's theorem and by the implicit function theorem for almost every level λ, the iso-level set $u^{(-1)}(\lambda)$ is a finite union of disjoint smooth Jordan curves. These Jordan curves are called the level lines of u and coincide with the topological boundaries of upper and lower level sets.

Assume in the following that u is at least C^2 in a neighborhood of a point $\mathbf{x_0} \in \Omega$ and that its gradient is not null, $Du(\mathbf{x_0}) \neq 0$. Then the scalar curvature of u at $\mathbf{x_0}$, denoted by $\mathrm{curv}(u)(\mathbf{x_0})$, is the real number defined by

$$\mathrm{curv}(u)(\mathbf{x_0}) = \frac{u_{xx}u_y^2 - 2u_{xy}u_xu_y + u_{yy}u_x^2}{(u_x^2+u_y^2)^{3/2}}(\mathbf{x_0}). \tag{2.1}$$

This scalar curvature at $\mathbf{x_0}$ is linked to the vectorial curvature of the level line passing by $\mathbf{x_0}$. The vectorial curvature of a C^2 curve $\mathbf{x(s)}$ parameterized by a length parameter s (so that $|\mathbf{x}'(s)| = 1$) is defined by

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

The link between the vectorial curvature of an image level line $\kappa(\mathbf{x})$ and the scalar curvature $\mathrm{curv}(u)(\mathbf{x})$ at nonsingular points is given by the next formula. Denote by $\mathbf{x} = \mathbf{x}(s)$ the level line of u passing by $\mathbf{x}_0$. Then

$$\kappa(\mathbf{x_0}) = -\mathrm{curv}(u)(\mathbf{x_0}).\frac{Du}{|Du|}(\mathbf{x_0}). \tag{2.2}$$

This relation already suggests that the curvature can be computed in two quite different ways: either as the curvature of a level line extracted from the image and parameterized by length, or as a 2D differential operator. In both cases, a previous smoothing (of the level line, of the level sets) is necessary, which introduces a new parameter, the smoothing *scale*. Hence the notion of *curvature scale space* which will be associated with curve or image evolutions.

2.2. Curve evolutions. Curve smoothing by the *heat equation* was one of the first versions of curve analysis proposed by Mackworth and Mokhtarian in [29]. Smoothing a curve by separately smoothing the coordinate functions seems reasonable, yet the evolved curve may develop self-crossings and singularities. This model

error was corrected in [30] by the same authors. Instead of applying the heat equation for relatively long times, they proposed an evolution by *Curve Shortening (CS)* (also called *intrinsic heat equation*)

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

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 [17] and later extended to embedded curves by Grayson [18].

The Affine Shortening equation (AS)

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

is a surprising variant of curve shortening introduced by Sapiro and Tannenbaum in [41], [43]. Angenent, Sapiro and Tannenbaum [44] gave the existence and uniqueness proofs for affine shortening and showed a result similar to Grayson's theorem: a shape 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 $\mathbf{x}(s,0) = \mathbf{x_0}(s)$ a family of smooth curves $\mathbf{x}(s,t)$. 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 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.

2.3. Image evolutions. Alvarez *et al.* [1] characterized axiomatically all image multiscale theories, and gave explicit formulae for the partial differential equations generated by scale spaces. They showed that *causal, local scale spaces* are governed by PDEs and that under sound stability conditions for the scale space, the PDE's have unique viscosity solutions. In particular all causal, local, isometric and contrast invariant scale spaces are given by curvature evolution equations:

$$\frac{\partial u}{\partial t} = |Du|G(curv(u), t).$$

The mean curvature equation (MCM)

$$\frac{\partial u}{\partial t} = |Du|curv(u)$$

is the simplest equation in this class for which existence, and uniqueness of viscosity solutions can be proved [15], [11]. Planar shape recognition algorithms should ideally be projective invariant, namely invariant to all planar homographies. The affine curvature evolution (ACM)

$$\frac{\partial u}{\partial t} = |Du|curv(u)^{1/3}$$

has a more restrictive form of projective invariance: it commutes with all planar affine maps with determinant 1. It is therefore preferable to the scalar curvature motion, and is definitely the most invariant image smoothing algorithm. Indeed, like the curvature motion, it is invariant to any continuous increasing contrast change $u \to g(u)$.

A consequence of the contrast invariance for both mentioned equations is that, at least formally, *an image evolves by scalar mean curvature motion (resp. affine curvature motion) if and only if its level lines evolve by curvature shortening (resp. affine shortening).* This fact can be checked by elementary differential calculus under the assumption that the scalar solution $u(\mathbf{x}, t)$ is smooth [20]. Yet, precisely, the curvature evolution does not yield a C^2 function in time and space. Thus, the above equivalence is a bit trickier and is proved in [13].

3. Curvature algorithms. All sound shape smoothing algorithms in the computer vision literature perform a curvature or an affine curvature shortening. But the numerical variety of the underlying numerical algorithms is worth noticing. This section discusses their history, implementation, advantages and drawbacks. There are three kinds of initial data for the algorithm: digital curves, digital sets, or digital images. We shall examine each in turn.

3.1. Algorithms on curves.

3.1.1. Dynamic curve evolution. As mentioned before, 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 Σ_0, gaussian signal G
Output: Evolved polygon Σ_n, after n iterations
for *all* $i = \overline{0, n}$ **do**
    sample uniformly curve Σ_i ;
    convolve curve Σ_i with G.

The consistency of Algorithm 1 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*

$$G_h * \mathbf{x}(s) - \mathbf{x}(s) = ch\ \kappa(\mathbf{x}(s)) + o(h). \tag{3.1}$$

where c *is a positive constant.*

3.1.2. Affine plane curve evolution. Several attempts to define an affine-invariant analysis for polygons are described in [42]. The 1/3 power law of planar motion perception and generation was related to affine invariance in [40]. Moisan [35] 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.

Algorithm 2: Discrete Affine Shortening (AS)

Input: Polygon Σ_0,
Output: Evolved polygon Σ_σ, at scale $\sigma^{2/3}$

break the curve into convex and concave parts ;
for *every convex/concave component* **do**
    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;
concatenate the pieces of curves previously obtained.

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 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* $\mathbf{x}\big|_{(s-\delta, s+\delta)}$ *is equal to* σ. *Then*

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

where c *is a positive constant.*

Lisani and al. [27] and later Musé and al. [37] have used the affine curve evolution scheme for shape recognition and image comparison algorithms. We have limited ourselves to numerical schemes that are extremely fast, being linear or, in the case of Moisan's scheme, super-linear in time and unconditionally stable.

Algorithm 3: Backward Euler Method for intrinsic heat equation

Input: Polygon Σ_0,
Output: Evolved polygon Σ_σ, parametrized by x_i at time $t_i = i\tau$

for *each* $i = 0,n$ **do**
    find x_{i+1} by a semi-implicit finite difference scheme of the type

$$\frac{x_{i+1} - x_i}{\tau} = f_i(k_i, x_{i+1})$$

    with f_i a nonlinearity depending on the curvature k_i of the curve x_i and the natural parameterization of the curve itself.

There is, however, a rich literature on numerical schemes for anisotropic curvature motions occurring (e.g.) in crystalline formation. These motions can depend on other powers of the curvature than the relevant ones for image processing (1 and 1/3) and have a spatial anisotropy. Mikula and Ševčovič have given theoretical and numerical methods for such more general curvature motions [33], [34]. They use implicit methods for curve evolution, which are very accurate but too slow to be performed on all image level lines. Their schemes are able to cope with almost arbitrarily high or low powers of the curvature, and they display an accurate asymptotic behavior.

Cao and Moisan [8] have also proposed "morphological" schemes for the motion of curves by arbitrary powers of the curvature. They are described in detail in the book by Frédéric Cao [7], which also contains a thorough numerical and mathematical

analysis. For more general image PDE's performing nonlinear diffusion, finite volume methods have been proposed with remarkable results in [26].

3.2. Algorithms on sets. Koenderink and van Doorn defined a *shape* in R^N as any closed subset X of R^N [24]. They proposed to simulate the shape multiscale perception by applying the heat equation to the characteristic function of the shape, or, in other terms, to convolve it with Gaussians with increasing variance. Of course, the solution $G_t * \mathbf{1}_X$ is not a characteristic function and therefore the authors defined the evolved shape at scale t to be

$$X_t = \{\mathbf{x} \mid u(\mathbf{x}, t) \geq 1/2\}.$$

The very same process was suggested in Attneave [4]: "The perceived *contour* of a cat (...) is the resultant of an orthogonal averaging process in which texture is eliminated or smoothed out almost entirely, somewhat as if a photograph of the object were blurred and then printed on high-contrast paper (...)" Similar to the heat equation for curve evolution, the method presents two inconveniences: the possible fusion of shapes which are too close, and the development of new singularities, which occur precisely at the times where two disjoint shapes coalesce.

The improvement of *dynamic shape* analysis is due to Merriman, Bence, and Osher who discovered and heuristically argued in [32] that the convolution of the indicator function of a shape with a Gaussian followed by a threshold at 1/2 simulated the mean-curvature motion.

Algorithm 4: Merriman-Bence-Osher Algorithm (threshold dynamic shape)

Input: initial shape X_0
Output: Evolved shape X_n at scale nh
for *i=0,n-1* **do**
  convolve the characteristic function of the shape X_i with G_h, where h is small;
  define $X_{i+1} = \{\mathbf{x} \mid G_h * \mathbf{1}_{X_i} \geq 1/2\}$.

The consistency of their arguments was checked by Barles and Georgelin [6] and Evans [16]. In addition they showed that iterated median filters converge asymptotically to the *Mean Curvature Motion*

$$u_t = |Du| curv(u).$$

An extension of this result to all iterated weighted median filters was given by Ishii in [21]. Algorithm 4 of Merriman, Bence and Osher is nothing but an *iterated median filter* applied to a binary image. The main problem of discrete median filters is their grid dependence which make them blind to small curvatures. For instance a black disk with radius 9 does not move if the discrete gaussian has a 2 pixels standard deviation. It is observed that the iterated process stops after a few iterations, making it an inaccurate scheme for MCM.

3.3. Algorithms on images.

3.3.1. Median filters and threshold dynamics. Weighted median filters are defined by

$$\mathrm{Med}_k u(\mathbf{x}) = \inf_{B \in \mathcal{B}} \sup_{\mathbf{y} \in \mathbf{x}+B} u(\mathbf{y}). \tag{3.2}$$

where k is a radial density distribution and $\mathcal{B} = \{B \mid \int_B k(\mathbf{x})d\mathbf{x} = 1/2 \int k(\mathbf{x})d\mathbf{x}\}$, $\mathcal{B}$ being formed of measurable sets. The discrete implementation of the median filter is almost trivial.

Algorithm 5: Iterated Median Filter Algorithm

Input: initial image $u(\mathbf{x})$
Output: evolved image $\text{Med}_k u(\mathbf{x})$
for *every point* **x do**
  consider the points **y** in a discrete neighborhood of **x**;
  compute the weight of **y** as the integral of k over the pixel of center **y**;
  take the weighted median value of the discrete neighborhood.

Algorithm 5 is fast but, like the dynamic shape, it is blind to small curvatures. Indeed, this algorithm applied on binary images is nothing but the dynamic shape, with the kernel k instead of a Gaussian [20]. The link with the curvature motion is obtained by scaling the convolution, exactly as in the Merriman-Bence-Osher dynamic shape algorithm. Define the scaled median by $(\text{Med}_k)_h = \text{Med}_{k_h}$, where $k_h(\mathbf{x}) := \frac{1}{h^2}k(\frac{\mathbf{x}}{h})$. Then:

THEOREM 3.3. *[20] If $u : \mathbb{R}^2 \to \mathbb{R}$ is C^2, then there is a constant c_k depending only on the kernel k such that*

1. *on every compact set $K \subset \{\mathbf{x} \mid Du(\mathbf{x}) \neq 0\}$,*

$$\text{Med}_{k_h} u(\mathbf{x}) - u(\mathbf{x}) = c_k |Du(\mathbf{x})| \text{curv}(u)(\mathbf{x}) h^2 + O(\mathbf{x}, h^3),$$

   where $|O(\mathbf{x}, h^3)| \leq C_K h^3$ for some constant C_K that depends only on u, k and K;
2. *on every compact set K in $\mathbb{R}^2$,*

$$|\text{Med}_{k_h} u(\mathbf{x}) - u(\mathbf{x})| \leq C_K h^2$$

   where the constant C_K depends only on u, k and K.

In short, by the above theorem the iterated median filter is in theory an implementation of the mean curvature motion but, when applied on a digital image, it stops prematurely because of the blindness of the grid to small curvatures. Adam Oberman's work [38] is intimately related to median filters and to the threshold dynamics. The novelty of his paper consists in the radial lattice displacement of the neighborhood points, as well as its 3D variant. Accordingly, he gives a clever consistency proof. The scheme is obviously monotone and therefore the convergence is guaranteed by the general approximation results for viscosity solutions given by Barles and Souganidis in [5].

3.3.2. Finite difference schemes. There are antecedent papers proposing clever schemes for the mean curvature motion or (more rarely) for the affine curvature motion. But, to the best of our knowledge, these papers do not attempt to compute the curvatures of the image. The present paper defends the thesis that the curvature is a 1D operator, computable only on the level lines themselves, and only after the adequate smoothing has been applied to each level line.

FDSs either create oscillations or, if they are adequately regularized to avoid oscillations, cause a strong and spurious diffusion. As illustrated later on (see Figure 5.5)

even the tiniest diffusion or oscillation created by an FDS brings up spurious curvatures with erratic value and sign. To emphasize this diffusion-sharpness dilemma, we shall make a comparison between Guichard's scheme, which attempts to be the least diffusive and the most isotropic, the Crandall-Lions scheme [14], which is monotone but diffusive, and subsequently the standard finite difference scheme that discretizes formula (1).

An efficient finite difference scheme (FDS) implementation of the scalar curvature motions was proposed by Alvarez and Guichard and is described in [19] and [2]. The 3×3 scheme takes advantage of the diffusive interpretation of the mean curvature, which can be expressed as the second derivative of u in the direction orthogonal to the gradient

$$|Du| curv(u) = u_{\xi\xi},$$

where $\xi = Du^{\perp}/|Du|$. A straightforward variant of the FDS applies to the affine curvature motion. The FDS is optimized to be as isotropic as possible and as close as possible to satisfy the maximum principle. It improves on the dynamic shape by computing correctly small curvatures, but it cannot properly handle the contrast invariance of the curvature equation. It creates new grey levels and blurs edges. Spurious diffusions occur around image extrema.

The finite difference scheme described by Crandall and Lions in [14] is monotone, consistent and stable; therefore its convergence is guaranteed by general approximation results (Barles and Souganidis [5]). Denote by u^n the discrete approximation of the solution at iteration n. Then the Crandall-Lions discrete curvature flow V is defined by

$$Vu^{n+1}(\rho z) = u^n(x) + dt \sum_{i=1}^{N} \frac{u^n(\rho z + ha(Du^n(\rho z))e_i) + u^n(\rho z - ha(Du^n(\rho z))e_i) - 2u^n(\rho z)}{h^2}$$

for all $z \in Z^N$, where $\{e^i\}_{i=1,N}$ is the standard basis of R^N and

$$a(p) = I - \frac{p \otimes p}{|p|^2}.$$

There are for this scheme three delicate issues.

1. The above formula does not fully discretize on a fixed grid $\mathcal{G}_\rho$ because the term $a((Du^n)(\rho z))e_i$ is not a displacement on the grid. Thus, this is an adaptive stencil. One must involve for every grid function u^n its continuous piecewise linear interpolation $\tilde{u}^n$. Hence, the previous formula must be updated as

$$Vu^{n+1}(\rho z) = \tilde{u}^n(x) + dt \sum_{i=1}^{N} \frac{\tilde{u}^n(\rho z + ha(Du^n(\rho z))e_i) + \tilde{u}^n(\rho z - ha(Du^n(\rho z))e_i) - 2u^n(\rho z)}{h^2}$$

   Note that in a second part of the paper the authors show that *the comparison principle cannot be guaranteed for finite difference schemes with a fixed stencil,* even for the linear case, when the matrix a has constant coefficients. This means that a centered differences approximation dealing with fixed stencils can create spurious oscillations and therefore parasitic curvatures. The

scheme introduced by Guichard is a sort of intermediate solution: it estimates the gradient direction ξ by evaluating numerically $u_{\xi\xi} = curv(u)|Du|$ with a quasi-linear scheme, based on a 3×3 stencil. The scheme is quasi-linear because the coefficients of the linear combination are functions of the angle θ that the gradient direction makes with the horizontal axis.

2. The (updated) discretization scheme is not monotone. *To ensure monotonicity one has to add a diffusion term.* Thus the discretization scheme becomes

$$Tu^n = Vu^n + \alpha\Delta u^n,$$

where

$$\Delta u^n = dt \sum_{i=1}^{N} \frac{u^n(\rho z + \rho e_i) + u^n(\rho z + \rho e_i) - 2u^n(\rho z)}{\rho^2}$$

Even though asymptotically we still have consistency of this monotone scheme, numerically *this scheme introduces a 2D diffusion.*

3. The projection $a(p)$ has two unpleasant features: it is degenerate (in the sense that aa^T has zero eigenvalues) and has a singularity at $p = 0$. The first drawback can be easily handled by the choice of parameters such that the approximation scheme produces bounds on the perimeter. For the second one, one should replace $a(p)$ by

$$a_\varepsilon(p) = I - \frac{p \otimes p}{|p|^2 + \varepsilon}.$$

However, when the gradient is small, its direction becomes substantially random, being driven by rounding errors and noise. This is why Guichard et al. suggest to replace in this case the mean curvature diffusion by a half Laplacian, instead of using $a_\varepsilon(p)$.

Accordingly, several relations must be satisfied among the parameters to ensure convergence of the scheme to the continuous solution when $dt \to 0$. There is no convergence result for Guichard's scheme. Again, a small perturbation with a linear diffusion term should handle the problem, but would also render the scheme very diffusive in practice.

One can devise an FDS for the curvature motion which is monotone and consistent. However, all such schemes cause, even after few iterations, a non asked diffusion that creates new levels and spurious curvatures. As illustrated by Figures 5.4 and 5.5 and their comments, computing a curvature by FDS is simply disastrous, even on an image smoothed by LLS.

A different approach was given by Peter Smereka introduced in [46], where he uses semi-implicit methods to derive fast implementation of the level set evolution PDEs for curvature motions. The semi-implicit algorithm for the mean curvature flow is based on the formula

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

where u is parameterized so that it remains close to the distance function (thus $N(u)$ is small and $|Du| \approx 1$). The discretization takes one step of forward Euler on the nonlinear term followed by one step of backward Euler on the linear term.

3.3.3. Level set extension, superposition principle, stack filters. FDSs for image curvature motions do not commute with increasing contrast changes. Yet, a full contrast invariance can be restored on any numerical scheme by coupling two techniques: the *superposition principle*, and the Osher-Sethian *level set extension*. The idea of the level set extension [39] is to treat a given curve as the zero level line of a signed distance to the curve. More generally a set, understood as a shape, is identified with its characteristic function. After applying the FDS to this function the evolved set can be obtained as the 1/2 upper level set of the evolved function. Thus, the level set extension is a generalization of the threshold dynamic.

By the level set extension, a curve evolution is made in two steps:

(1). apply a finite difference scheme to the characteristic function of the shape bounded by the curve;

(2). take the level line of the result corresponding to the level 1/2 .

Since the image level lines are boundaries of the image upper level sets, it is natural to apply directly the level set extension to all upper level sets. This processes implicitly all level lines of each level. After evolution of all upper level sets, an image is reconstructed by *superposition principle.* The superposition principle and its link to the contrast invariance property come from mathematical morphology [45], [31]. If all upper level sets of a given image have been processed independently by an inclusion preserving scheme, then there is a single image having for level sets the evolved level sets.

Any process that decomposes the image into the *stack* of its level sets and then reconstructs the processed image from the stack of its processed level sets is called a *stack filter*. The only requirement to make a stack filter with any numerical scheme is its monotonicity. Indeed, the inclusion of upper level sets in each other must be preserved. *Every stack filter is contrast invariant:* the upper (resp. lower) level sets $\mathcal{X}_\lambda u_0 := \{\mathbf{x}, u_0(\mathbf{x}) \geq \lambda\}$ (resp. $\leq \lambda$) of an image u_0 are invariant to increasing contrast changes.

In short a stack filter consists of:

(a). extracting all image upper level sets,

(b). processing each of them by a (monotonic) set operator (e.g. the FDS) and

(c). reconstructing the evolved image by "superposition".

Thus Algorithm 6 is a contrast invariant curvature evolution. For example (see [19]) the image median filter is the stack filter of the threshold dynamics. It makes sense to apply the superposition principle strategy to FDSs because they are not contrast invariant, being diffusive and creating spurious level lines.

Algorithm 6: Stack Filter

Input: initial image $u(\mathbf{x})$

Output: evolved image $u(\mathbf{x}, t)$

for *each* $\lambda \in [0, 255]$*, in increasing order* **do**

  let $v_\lambda(\mathbf{x})$ be the characteristic function of $\mathcal{X}_\lambda u_0 := \{\mathbf{x}, u_0(\mathbf{x}) \geq \lambda\}$;

  apply to v_λ an FDS-scheme until scale t; this yields the images $w_\lambda(t, .)$;

  set $u(\mathbf{x}, t) = \max\{\lambda \mid w_\lambda(t, \mathbf{x}) \geq 1/2\}$ at each point $(t, \mathbf{x})$.

3.4. Discussion. One can devise an FDS for the curvature motion which is monotone and consistent. However, all such schemes cause, even after few iterations, a non asked diffusion that creates new levels and spurious curvatures. As illustrated

by Figures 5.4 and 5.5 and their comments, computing a curvature by FDS is simply disastrous, even on an image smoothed by LLS. On the other hand, FDSs lack invariance and structural properties that curvature motions possess, namely:

- monotonicity: they lead to slightly oscillatory solutions, unless adequately regularized;
- contrast invariance: FDSs create new grey levels and blurs edges, leading to spurious diffusions around image extrema.
- Euclidean or affine invariance: FDSs are grid dependent.

We have just seen that the full contrast invariance can be restored by the stack filters. But are stack filters based on FDS a sufficient solution? We will see that they are not. By evolving sets sampled on a fixed grid, boundaries either jump by a positive integer number of pixels, or they don't move at all. If follows that these numerical motions are quantized, and in particular blind to small curvatures. As we shall see in the experiments, they fall short of matching the human perception precision. The experiments will show that human fine perception of curvatures is better explained by sampling all image level lines at fine sub-pixel resolution and by smoothing them very accurately.

4. Level Lines Shortening. In this section, we set forth a continuous, grid independent evolution of a digital image by curvature motion, that will jump over all hurdles listed above. Yet to do so the process must be almost ludicrously sophisticated. The last sections will have to prove that the effort was worth it.

This image processing algorithm, that we shall call *Level Lines Shortening* (LLS) or *Level Lines Affine Shortening* (LLAS), first extracts all level lines of a digital image, with a number of levels sufficient to grant an exact reconstruction of the initial image. Then the algorithm simulates an image evolution by moving independently and simultaneously all of its level lines by curve shortening (CS) (resp. affine curve shortening (AS)). The evolved image is eventually reconstructed from its evolved level lines. Thus the algorithm realizes the commutative diagram:

$$\begin{array}{ccc} u_0(\cdot) & \xrightarrow{\textit{level lines extraction}} & \{\Sigma_0^{\lambda,i}\}_{\lambda,i} \\ \Big\downarrow{\scriptstyle MCM/ACM=LLS/LLAS} & & \Big\downarrow{\scriptstyle CS/AS} \\ u(\cdot,t) & \xleftarrow{\textit{reconstruction}} & \{\Sigma_t^{\lambda,i}\}_{\lambda,i} \end{array}$$

We prove in a companion paper [13] that the image reconstructed from the evolved level lines is a viscosity solution of the mean curvature motion (MCM) (resp. affine curvature motion (ACM)) provided that the level lines at almost all levels evolve by curve shortening (resp. affine shortening). The initial image will be considered as an element of the space $\mathcal{BL}(\Omega)$ of the digital images on a rectangle Ω interpolated by bilinear interpolation. We state the theorem in this exact numerical framework, used in the algorithm.

THEOREM 4.1. *Let $u_0 \in \mathcal{BL}(\Omega)$. Then the Level Lines Shortening evolution of the function u_0,*

$$u(\mathbf{x},t) = LLS(t)u_0(\mathbf{x}), \forall \mathbf{x} \in \mathbb{R}^2, \forall t \in [0,\infty)$$

is a viscosity solution for the mean curvature PDE, with the initial data u_0

$$\begin{cases} u_t = curv(u)|Du|, & \text{in } \mathbb{R}^2 \times [0,\infty) \\ u(\cdot,0) = u_0, & \text{on } \mathbb{R}^2. \end{cases}$$

A similar result holds for LLAS and the affine curvature PDE, (ACM) with initial condition u_0*.*

The level lines (affine) shortening chain LL(A)S, described in Algorithm 7, is based on a topological structure, the inclusion tree of level lines as a full and non-redundant representation of an image, and on a topological property, the monotonicity of curve shortening with respect to inclusion. The hierarchy of the level lines is therefore maintained while performing the smoothing. Thus, the reconstruction can start with the largest level line, namely the frame of the image, and continue by filling from top to bottom in this inclusion the interior of each level line. At each step the lamina bounded by the current level line is filled in with its own level, and these levels are updated when passing to its descendants.

Algorithm 7: Level Lines Shortening (LL(A)S) Algorithm

Input: Original Image u_0.
Output: The LL(A)S evolution of u_0 at scale t: $u(\cdot,t)$.

```
Extract the tree of level lines {Σ_0^{λ,i}}_{i∈F_λ,λ};
for Level line Σ_0^{λ,i} do
    Σ_t^{λ,i} = Discrete Curve/Affine Shortening of (Σ_0^{λ,i});
for Evolved Level line Σ_t^{λ,i} do
    fill the interior of level line Σ_t^{λ,i}.
```

LL(A)S is illustrated in Figure 4.1. We perform and display each step of the numerical chain. The level lines were extracted at half-integer gray values and were chosen with a quantization step $q = 4$. The Moisan affine plane curve evolution [35] is then applied independently to all level lines, at *renormalized scale* $s = 4$. This scale is chosen so that a circle with radius $r = 4$ (where the unit is given by the length of a pixel edge) disappears at scale $s = 4$. This normalization by the result is numerically important for comparing numerical schemes with very different settings. A new image which has exactly these curves as level lines is finally reconstructed. The result is by Theorem 4.1 an affine invariant curvature motion (ACM) of the original image. The rest of this section is devoted to several crucial details of LL(A)S regarding the level lines extraction, their evolution, and the reconstruction algorithm.

4.1. Level Lines extraction. The simplest image interpolation that preserves its continuity is the bilinear interpolation on the dual pixels. (A dual pixel is any square whose vertices are centers of contiguous pixels). The bilinear interpolation in the dual pixel is written in the form

$$u_0(x,y) = axy + bx + cy + d$$

where the parameters a, b, c, d are computed from the values taken at the four vertices of the dual pixel, which are normal pixel values. The bilinear interpolated image is the concatenation of the bilinear interpolations on all dual pixels; it is continuous, but its gradient may present discontinuities.

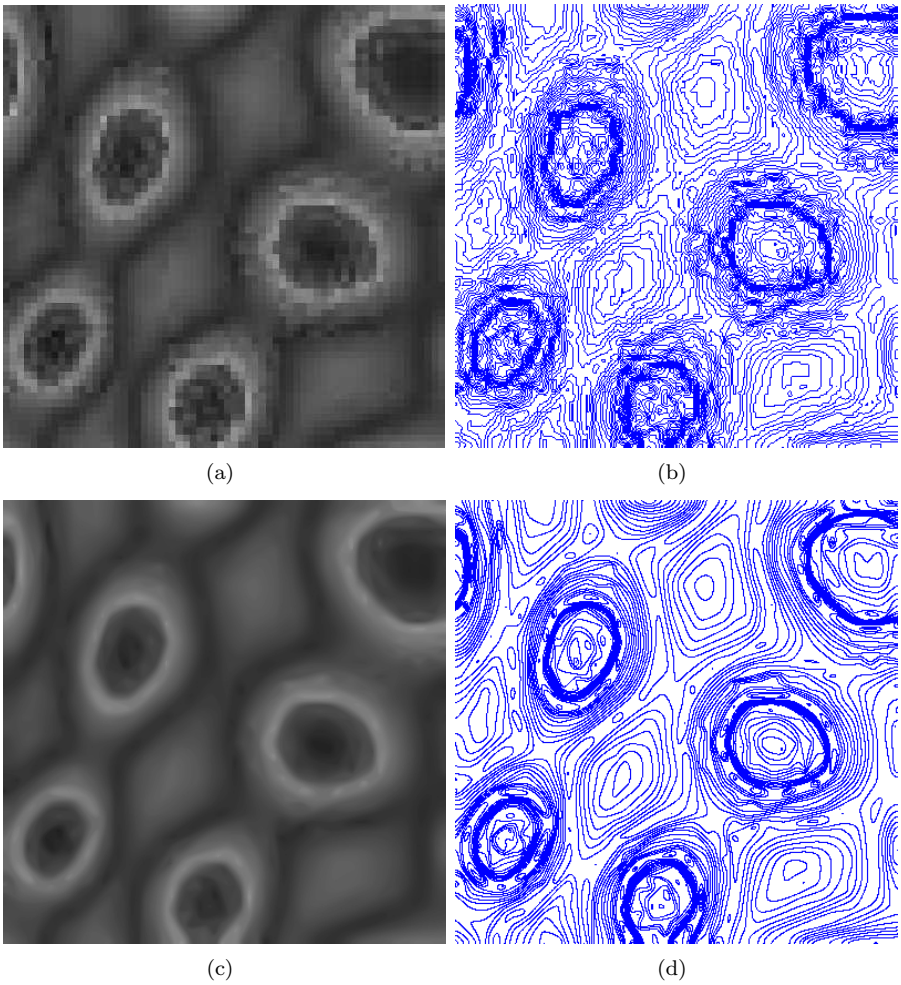


FIG. 4.1. *Illustration of the LL(A)S numerical chain. (a) Original image. (b) Bilinear level lines extraction. (d) Simultaneous and independent smoothing of level lines by affine shortening. (c) Image reconstructed from shortened level lines.*

4.1.1. Bilinear level lines. The equation for a level line at level λ of the bilinear interpolated image inside a dual pixel can be written either

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

or

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

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 (see Figure 4.2). This phenomenon will be attenuated by taking for λ only half-integer values (given that the digital image has integer values).

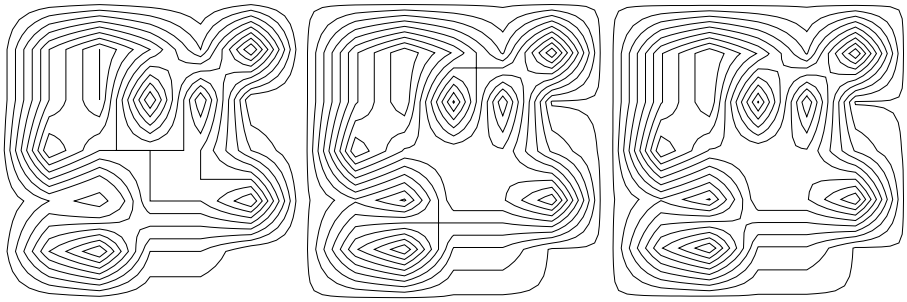


FIG. 4.2. *Gray levels for a piecewise bilinearly interpolated image. Three different sets of level lines were computed. Left: gray levels from 10 to 100 with step 10. Observe how some of the level lines (the ones at gray level 70) follow the Qedgels, producing an effect similar to pixelization. Middle: 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.*

4.1.2. 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 [10] and [36]. 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 Σ^{λ_1} is a *child* of the curve Σ^{λ_2} and we denote $\Sigma^{\lambda_1} \prec \Sigma^{\lambda_2}$ 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 ($\mathrm{sgn}(\Sigma^\lambda) = -1$) or upper level set ($\mathrm{sgn}(\Sigma^\lambda) = +1$).

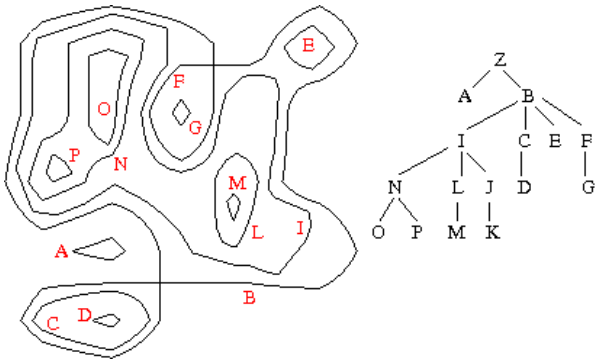


FIG. 4.3. *Tree of bilinear level lines.*

For each gray level $\lambda \in \mathbb{N} + 1/2$ there corresponds a finite set F^λ of level lines $\{\Sigma_0^{\lambda,i}\}_{i \in F^\lambda}$. Each level line $\Sigma_0^{\lambda,i}$ is stored as a set of ordered points leaving the level line interior on the left hand side. Thus, the tree of level lines is given by a finite set of *tagged* polygonal lines, indexed by half-integer gray values

$$\mathcal{T}_0 = \{\Sigma_0^{\lambda,i}; i \in F^\lambda, \lambda \in \mathbb{N} + 1/2\}. \tag{4.1}$$

An inclusion tree of bilinear level lines is displayed in Figure 4.3.

4.2. Independent evolution of all level lines. As already described, the affine shortening is numerically defined as an alternate filter of affine erosion and affine dilation. Up to re-sampling issues, the scheme is monotonous with respect to geometrical inclusion and therefore the tree structure of the level lines is preserved. It consists of a finite set of Jordan curves, denoted by

$$\mathcal{T}_n = \{\Sigma_n^{\lambda,i}; i \in F^\lambda, \lambda \in \mathbb{N} + 1/2\}. \tag{4.2}$$

Figure 4.1.(d) displays the affine evolution of the level lines appearing in Figure 4.1.(b). As can be observed, the curves become smoother, oscillations due to the grid reduce, and curves with small perimeter vanish. The inclusion tree structure is clearly preserved under the affine shortening evolution. The same chain applies to the curve shortening by the Mackworth-Mokhtarian scheme. With a fine enough curve sampling it is consistent with the curve shortening and therefore also numerically monotone.

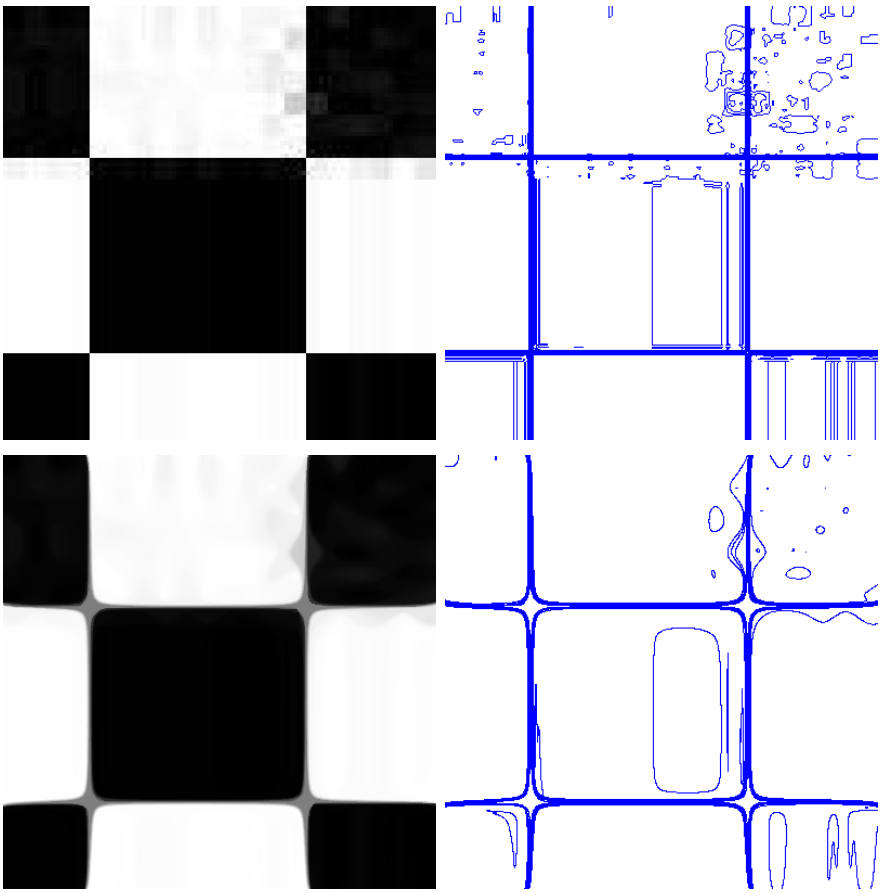


FIG. 4.4. *Fattening effect. From top to botom: the original image and its extracted level lines with quantization step* $s = 16$*, their independent evolution by affine shortening at renormalized scale* $l = 8$ *and the image reconstructed from the evolved level lines. Observe that distinct, touching level lines tear apart and non-empty interiors appear at the saddle points. At these points four squares, two black, two white, meet initially by their corners. After evolution, fattened grey regions are liberated by the retraction of the square level lines surrounding the black and the white squares.*

4.2.1. The fattening effect. Bilinear level lines can present self-intersections at image saddle points. In that case, LLS develops a non-empty interior, meaning that two distinct touching curves instantly tear apart, and that the space liberated becomes a flat region with a grey level equal to the level line value. Thus the level line "fattens". This effect is easily explained by considering the classic evolution of the level curves just above and just below the saddle curve. Let Σ^μ be a level line passing through a saddle point (see Figure 4.4). Assume that the curve is the limit of level lines from above and from below. More precisely, suppose (e.g.) that for slightly higher levels λ the interiors of level lines Σ^λ include the interior of Σ^μ and that for smaller levels each connected component of the interior of Σ^μ contains the interiors of Σ^λ (this is always the case if u_0 is a bilinear interpolation). Then the exterior curve will evolve as the limit of the level lines surrounding it from the outside and simultaneously all interior (touching) Jordan curves will evolve as the monotone limits of interior level lines. Consequently, they will tear apart from the exterior curves, thus liberating a flat region.

4.3. 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) $\{\Sigma^{\lambda,i}\}_{i\in F^\lambda,\lambda\in\Lambda}$ 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 4.2. *Let $\mathcal{T} = \{\Sigma^{\lambda,i}; i \in F^\lambda, \lambda \in \mathbb{N} + 1/2\}$ be the tree of bilinear level lines associated to u_d. For every x let λ be such that $x \in Int(\Sigma^\lambda)$ and $\forall \Sigma^{\tilde{\lambda}} \prec \Sigma^\lambda$, $x \notin Int(\Sigma^{\tilde{\lambda}})$ and define*

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

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

A closed curve Σ is stored as a set of ordered points $\{P_k(x_k, y_k)\}_{1\le k\le N}$ with N depending on Σ. The real numbers x_k and y_k are the floating point coordinates of the vertex number k of the polygon Σ. 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 Σ 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.

4.3.1. Polygon intersections with the grid. The goal of Algorithm 8, 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 abscissas so that $x_1^i \leq x_2^i \leq \cdots \leq x_p^i$, where p is even because Σ 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 8: Intersections of a polygon Σ with the grid

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

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

4.3.2. 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 9.

Algorithm 9: Filling polygon Σ

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

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

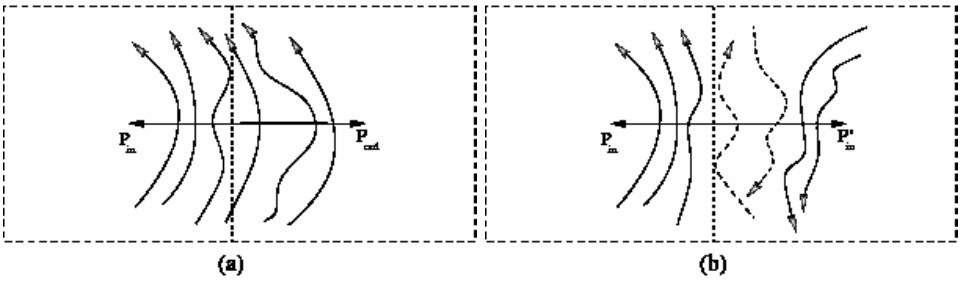


FIG. 4.5. *The level line* $2D$ *inclusion topology is reflected in the* $1D$ *ordering of their intersections with the dual edges.*

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 4.5(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.

Numerical examples of image reconstruction from the tree of evolved level lines are displayed in Figure 4.1 and Figure 4.4.

4.4. Numerical properties of the LLS numerical chain.

4.4.1. Fixed point property. *The filling algorithm* itself is a stand alone *image reconstruction* method, working for every family of curves endowed with levels and a tree inclusion structure. To check its consistency, it is enough to take any digital image u_0, to extract its level lines at quantized gray levels, with quantization step $s = 1$ but without applying any evolution. Then the digital image is reconstructed exactly from its level line tree by the filling algorithm.

4.4.2. Local comparison principle and regularity. The order preserving property or *inclusion principle* is the main structural requirement of a level line evolution algorithm. It basically prevents the crossing of two different level curves and therefore permits the construction of a unique image having a prescribed set of level lines. Some level lines may present multiple crossings at saddle points, in which case the level lines shortening develops a non-empty interior. The phenomenon is due to an instantaneous tearing apart of two distinct, touching curves.

Any level curve with self-contact points develops a non-empty interior by CS (curve shortening), which implies the formation of a flat area (see Figure 4.6.) We compare in the following the LLS algorithms with the FDSs for mean curvature given by Guichard et al. in [19]. The finite difference schemes (FDS) tested here have been optimized by its authors to ensure a maximal rotation invariance and stability. Nonetheless, they ensure neither monotonicity nor full contrast invariance. They create new grey levels and blurs out the edges. It is true that the full contrast invariance of an FDS is restored by its stack filter. Nevertheless, spurious diffusions occur around the image extrema and at *T-junctions* or *X-junctions*. At saddle points both algorithms create new extrema, and therefore spurious level lines. LLS solves this issue, by separately evolving the level lines and then reconstructing the image.

Figure 4.6 compares various implementations of the mean curvature motion on a checkerboard image (a) with calibration of the numerical scales. The left images of each pair show the evolutions of the image by the various implementations of mean curvature motion, while the right ones display a zoom at the X-junction and the corresponding level lines. The iterated median filter (a) instantaneously stops and leaves the checkerboard invariant. This may look fine, but it is not consistent with curvature motion. LLS (b) is performed with a 1D gaussian kernel of standard deviation $\sigma = 2$. The level lines are encoded with a $p = 5$ points per pixel precision and displayed with a $s = 4$ quantization step, starting at an offset $o = 96$. The figure next shows the effect of FDS (c) at normalized scale $l = 3$, the FDS stack filter (d) at normalized scale $l = 3$, and finally the LLS evolution with the same normalized scale. At $X-junctions$, both FDS and the FDS stack filter create spurious diffusions, while LLS doesn't. With LLS a grey region develops at the junction, because level lines corresponding to different gray levels instantly tear apart. This is not necessarily gratifying perceptually, but it is mathematically consistent.

Figures 4.7 and 4.8 put in evidence the failures of FDSs on a binary fingerprint. Up to some critical scale, FDS stack filters restore the correct topology, but in case of fast

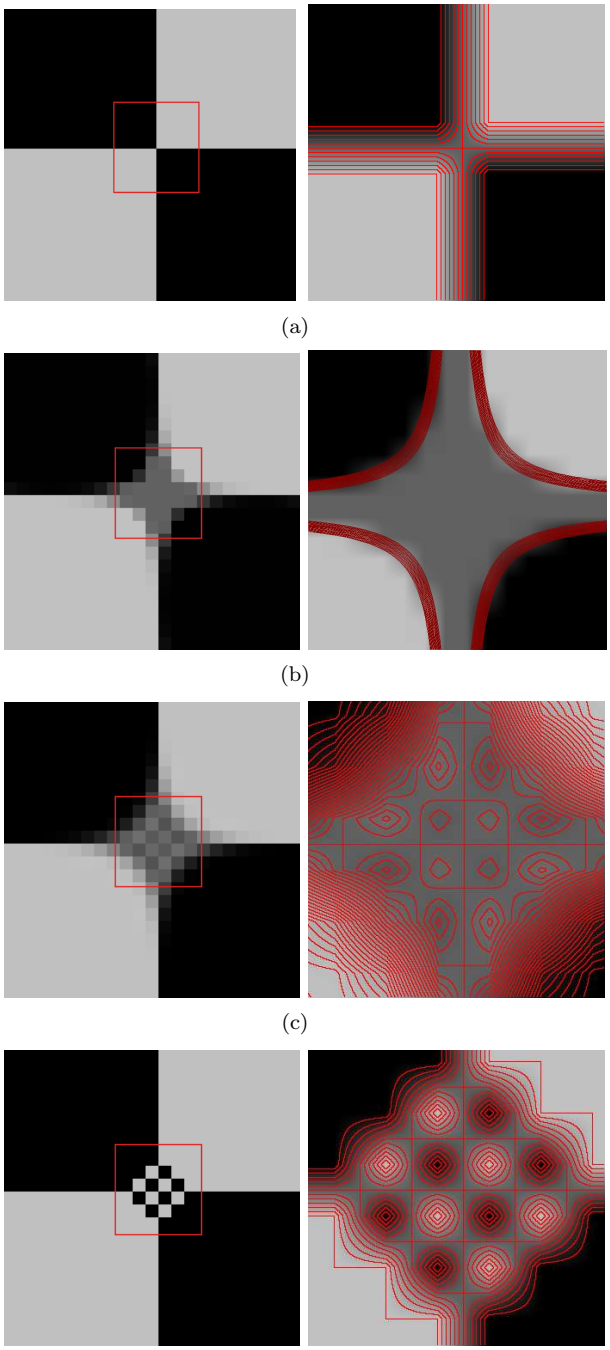


FIG. 4.6. *The four pairs present various implementations of the mean curvature motion on a checkerboard image (left column) and zooms at an X-junction, with its level lines overprinted on the image (right column). From top to bottom : (a). original image (the zoom is by bilinear interpolation), (b). Level lines shortening, (c). Finite difference scheme, (d) FDS stack filter. Only LLS does not create new extrema.*

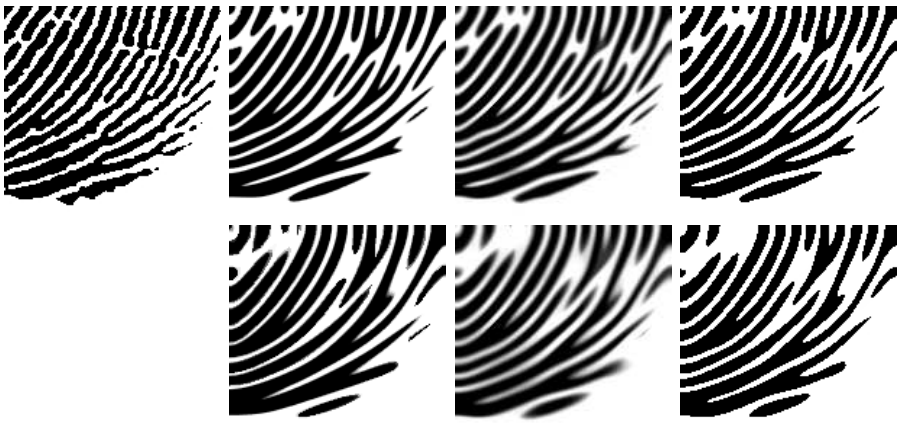


FIG. 4.7. *Various implementations of the affine curvature motion. the original image is displayed alone in the left column. In the other columns, from left to right: LLAS, FDS and FDS-stack filter at renormalized scales* $l = 4$ *(top) and* $l = 8$ *(bottom). In the case of the affine curvature scale space, the gradient amplitude keeps down ridge diffusion, unlike the mean curvature scale space. In general the affine smoothing performs better than the curvature motion (compare with Figure 4.8).*

FIG. 4.8. *Various implementations of the mean curvature motion. Are compared (from left to right): LLS, FDS, the FDS-stack filter and the median filter at renormalized scales* $l = 4$ *(top) and* $l = 8$ *(bottom). Up to some critical scale, the stack filters restore the correct topology, but in case of fast diffusions they break off as well. Observe that spurious diffusion mixing the ridges occurs in all cases except LLS, which tears apart ridges and emphasizes crossovers. A comparison with Figure 4.7 shows that the affine curvature schemes perform all better than their analogous curvature schemes.*

diffusions they break off as well. Oscillating ridges with high gradient amplitudes make it difficult to keep separated the various connected components during the smoothing process. A visual comparison of these two figures proves abundantly that the affine curvature is a much better shape preserver than the curvature motion.

4.5. JPEG artifacts reduction on color images. The prevailing JPEG 1992 image coding format aims at compressing images while maintaining acceptable image

quality. This is achieved by dividing the image in 8×8 pixels blocs and applying a discrete cosine transform (DCT) on the partitioned image. The resulting coefficients are quantized. In particular the less significant coefficients are set to zero. This process causes several types of artifacts such as Gibbs oscillations, staircase noise along curving edges, and checkerboard patterns (which are nothing but cosine functions). The phenomenon is illustrated in Figure 4.9.

LL(A)S seems to be a useful postprocessing technique for JPEG artifact reduction. In color images LLS is applied independently to each color channels.

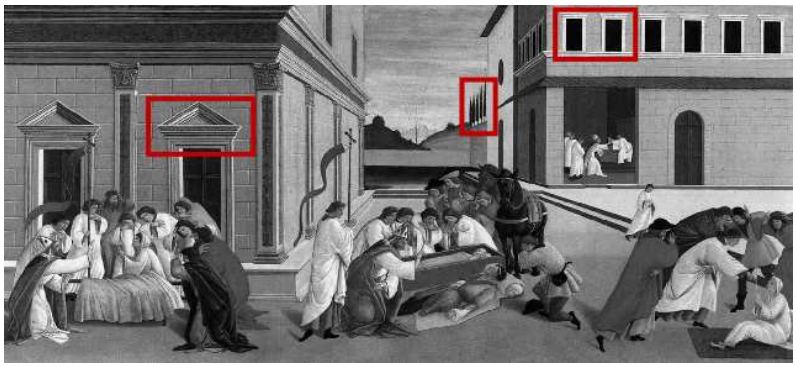
(a) (b)

FIG. 4.9. *(a). Original image, suffering of JPEG artifacts such as Gibbs oscillations, staircase noise along curving edges and checkerboarding. (b). LLAS is applied separately to each RGB channel. Although diffusions occur at junctions, LLAS considerably reduces these artifacts.*

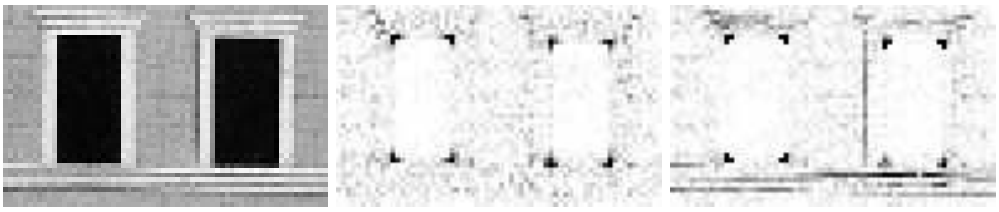
4.5.1. Accurate mean curvature evolution. The main goal of the implementation is to obtain and move level lines with arbitrarily high sub-pixel precision. Indeed, level lines are encoded as polygons whose vertices have double precision coordinates. Moving simultaneously level lines extracted with high sample precision allows straight level lines with high gradient to stand still with LLS, whereas they are diffused by FDS, even in its stack variant.

The phenomenon is shown in Figure 4.10 on a photograph of one of Botticelli's paintings. We display the original image and three different zooms of different parts of the image: the window frames (b) the trees in the background (c) and finely textured bricks (d). We illustrate the original details (left column) and the differences in absolute value between the original and its evolutions by LLS (middle column) and by the FDS stack filter (right column). The FDS stack filter was applied at normalized scale $l = 2$ and the LLS evolution at an equivalent normalized scale. For LLS the level lines were quantized at half integer levels with a step $s = 1$ and extracted with a precision of $p = 5$ points per pixel. Even though the curvature is zero, the FDS stack filter lets level lines with high gradient evolve, while with LLS straight lines stand still.

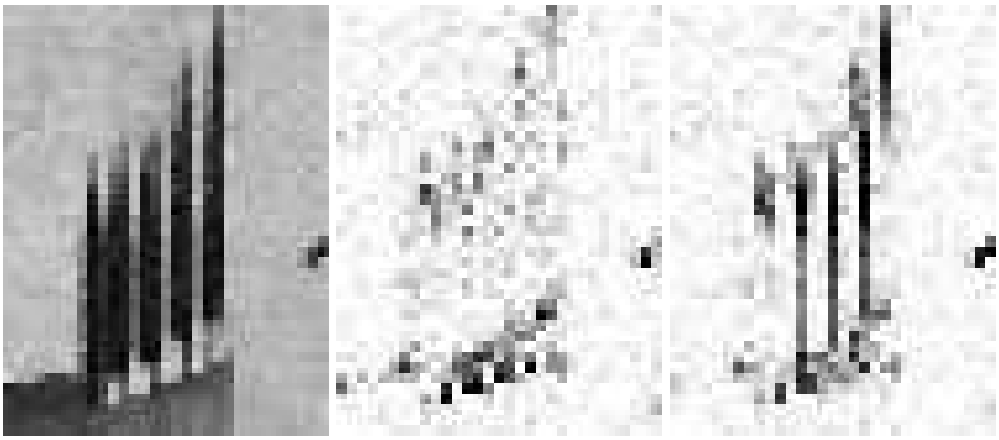
5. The Curvature Microscope. Whenever we talk about curvatures in a digital image, we actually refer to the curvatures of the level lines associated to the image. Yet, most curvature computation algorithms are based on finite difference



(a)



(b)



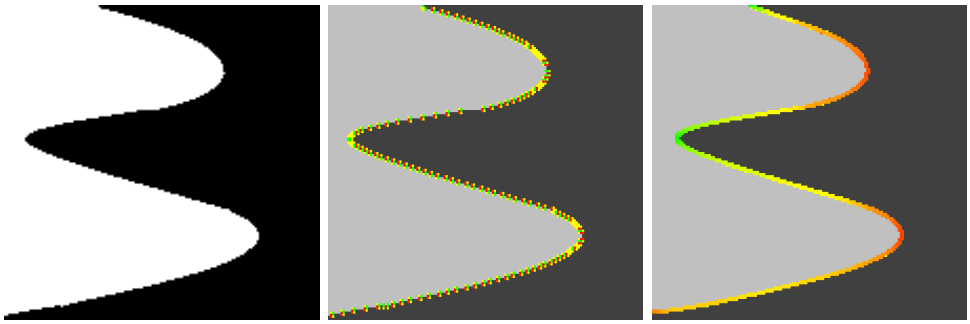
(c)

(d)

FIG. 4.10. *Zooms on three highlighted details in the painting "Three miracles of St. Zenobius" by Sandro Botticelli (Left). Middle column: the differences in absolute value between the original image and the evolutions by level line shortening. Right: same result, after applying the FDS stack filter. Even though the curvature is zero, the FDS stack filter lets level lines with high gradient evolve, while with LLS straight lines stand still.*

schemes (FDS) with formula (2.1). But with FDSs, the curvature depends on the gray values of a whole neighborhood. Consequently, high oscillations along transverse level lines do appear.

For the sake of precision, 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 5.1.



(a) (b) (c)

FIG. 5.1. *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 by formula (2.1) (b), and by LLS (c). In the first case the curvature presents oscillations, whereas the second result is coherent with our perception.*

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.

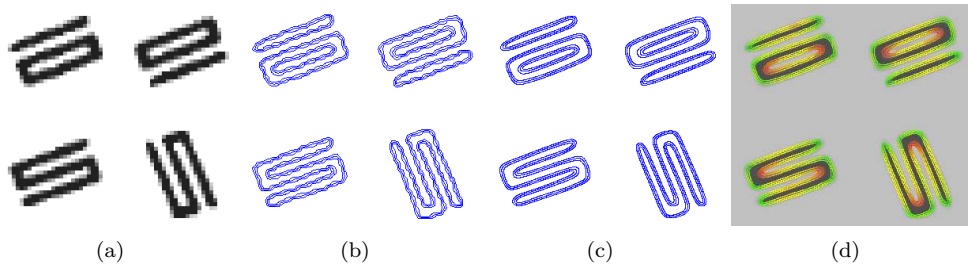
5.1. Discrete curvature for a polygonal line.. We recall that each level line is stored as a set of ordered points

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

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 $\overrightarrow{u_i} = (u_i^1, u_i^2) = \overrightarrow{P_{i-1}P_i}$ and its length $u_i = |\overrightarrow{P_{i-1}P_i}|$, respectively $\overrightarrow{v_i} = (v_i^1, v_i^2) = \overrightarrow{P_{i-1}P_{i+1}}$, with the corresponding length $v_i = |\overrightarrow{P_{i-1}P_{i+1}}|$. Then

LEMMA 5.1. *The curvature at vertex P_i is given by*

$$k_i = 2\frac{u_i^1 u_{i+1}^2 - u_i^2 u_{i+1}^1}{u_i u_{i+1} v_i}. \tag{5.1}$$



(a) (b) (c) (d)

FIG. 5.2. *The curvature map numerical chain: (a) original image, (b) level lines, uniformly sampled, (c) evolved level lines, (d) curvature image.*

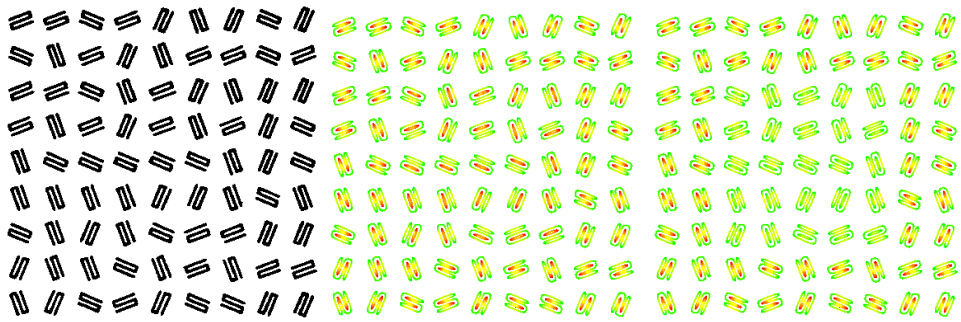
5.2. Curvature map. The algorithm computing the curvature map of any digital image is based on LLS. The image level lines at given quantization levels are first extracted, then uniformly sampled with fine sub-pixel step, and smoothed by affine or curve shortening. Curvatures are then computed at each 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 10: Curvature map

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

Extract the tree of level lines $\{\Sigma_0^{\lambda,i}\}_{i \in F_\lambda; \lambda}$;
Sample uniformly each level line $\Sigma_0^{\lambda,i}$
for *Level line* $\Sigma_0^{\lambda,i}$ **do**
    $\Sigma_t^{\lambda,i}$= Curve Shortening Flow $(\Sigma_0^{\lambda,i})$;
for $\Sigma_t^{\lambda,i} = \{P_i(x_i, y_i)\}_{i=0..n}$ **do**
    $k_i = 1/R_i$;
for *each dual pixel* **do**
    $k = \text{mean}(k_{i_1}, k_{i_2}, ..., k_{i_m})$.

Topological curvatures and scalar curvatures can be computed as well. Indeed, the information encoded in the tree enables the computation of signed 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. Indeed, $curv(-u) = -curv(u)$. 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 5.3 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 middle image shows the scalar curvature defined by formula (2.1). This 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.



(a) (b) (c)

FIG. 5.3. *(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.*

5.3. Comparison with FDSes. One could object that the above shortcomings of FDSs can be fixed by interpolating the image on a very thin grid. All schemes being consistent eventually should give similar results. The comparison would be fair because the LLS method actually starts with a subpixel level line description, equivalent to sampling on a finer grid. Nevertheless, curvature computation by FDS wouldn't work satisfactorily, even after a finer interpolation, and even if the smoothing has been done by LLS, which has the advantage of being less diffusive.

To prove this, we run an FDS and LLS on a very simple geometric image, as displayed in Figure 5.4. The FDS was the scheme implemented in [17], which creates minimal smoothing. The LLS was the affine level line motion, whose algorithm can now be tested online. The Level Lines Affine Shortening Algorithm was applied to the original image at the normalized scale $l = 2$. In Figure 5.4 the curvature map is estimated by a direct computation on the shortened level lines and compared to the curvature map computed on the smoothed image by the FDS (approximating the directional derivative $u_{\xi\xi}$ with a 3×3 scheme).

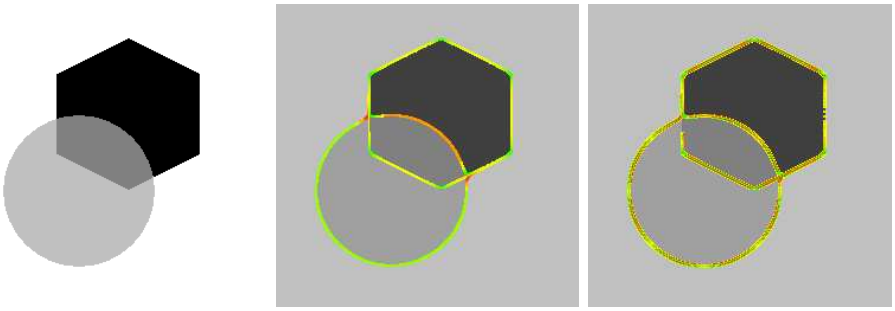


FIG. 5.4. *From left to right: the original image, the curvature map estimated by a direct computation on the shortened level lines, the curvature map computed by FDS on the LLAS image at normalized scale* $l = 2$.

On the other hand an FDS with grid refinement was tested in Figure 5.5. The image was zoomed in by a factor $Z = 2$ using bilinear interpolation. The finite difference scheme for the affine curvature motion was run and the curvatures computed by FDS, before and after smoothing.

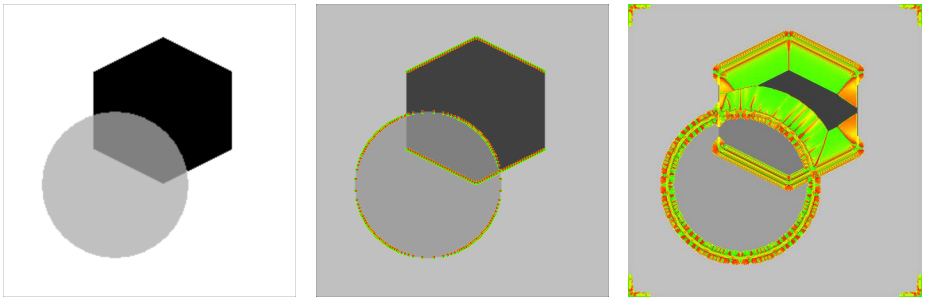


FIG. 5.5. *From left to right: the original image sampled on a thinner grid using a bilinear zoom by a factor $Z = 2$, the curvature map before and after FDS filtering at renormalized scale $l = 4$, computed with a 3×3 stencil.*

As can be seen by comparing Figures 5.4 and 5.5, the difference between both *smoothed images* can be perceived by the human eye as a slight blur with the FDS, due to the diffusion term in the FDS. Note however (by comparing the lower -right images in both figures) that a large diffusion occurs after several iterations of the finite differences scheme. Of more concern, however, is the fact that wrong curvatures appear everywhere with the FDS: see how red and green color alternate on all boundaries. With LLS, in contrast, the computed curvatures are coherent with the geometry.

5.4. 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 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.

FIG. 5.6. *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 at normalized scales* $l = 1$*,* $l = 2$*, and* $l = 4$ *(the zoom factor must be equal to the normalized smoothing scale). 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.*

6. The curvature Gallery. After processing the pixelized level lines become accurate curves with sub-pixel control points, whose curvature can be faithfully computed. Thus the whole chain can be viewed as a numerical preprocessing before further numerical analysis and feature extraction. 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.

6.1. 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 6.1, 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).

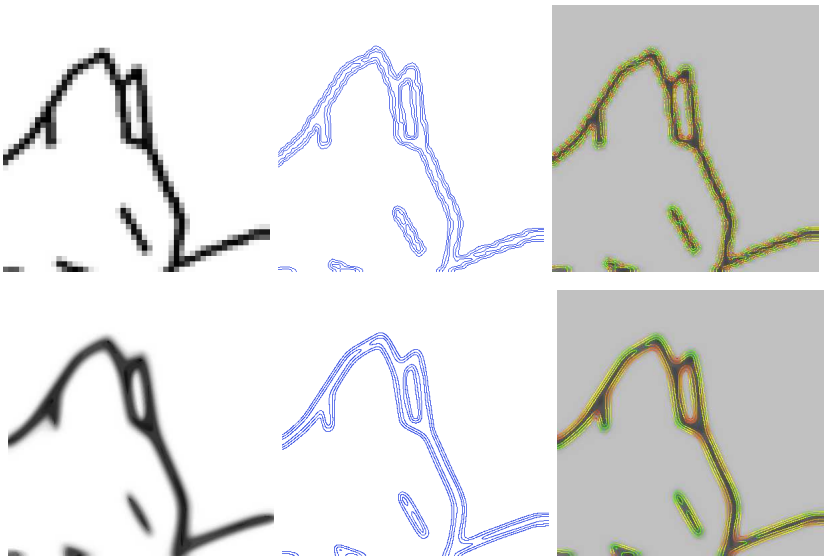


FIG. 6.1. *Part of Attneave cat, its corresponding level lines and curvatures. LLAS evolution, smoothed level lines and curvature map after filtering.*

6.2. Geometric shapes. The same improvements can be demonstrated on the geometric drawings of Figures 6.2, 6.3 and 6.4. 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.

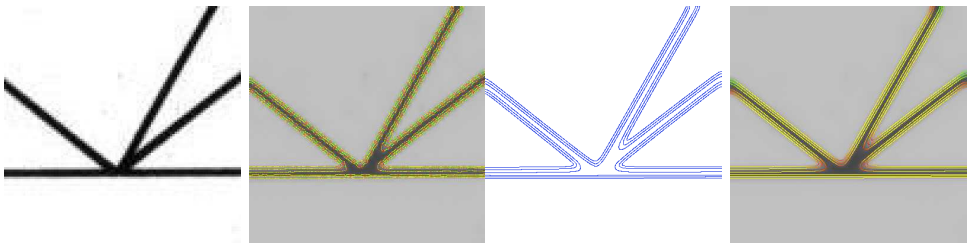


FIG. 6.2. *Image and corresponding curvatures before and after LLAS filtering.*

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 6.3, which

is a typical Kanizsa experiment demonstrating our layered perception, one tends to see a grey rectangle on top of a black polygon. The 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.

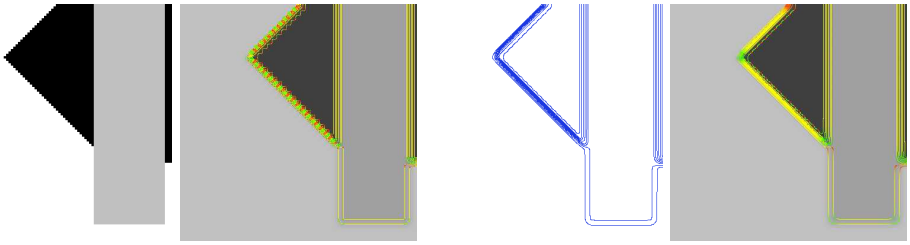


FIG. 6.3. *Original image, non-filtered curvatures, smoothed level lines by LLAS and curvature map after filtering.*

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 6.4 is spontaneously described by viewers as a grey transparent disk in front of a black wedge. Kanizsa [23] 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.

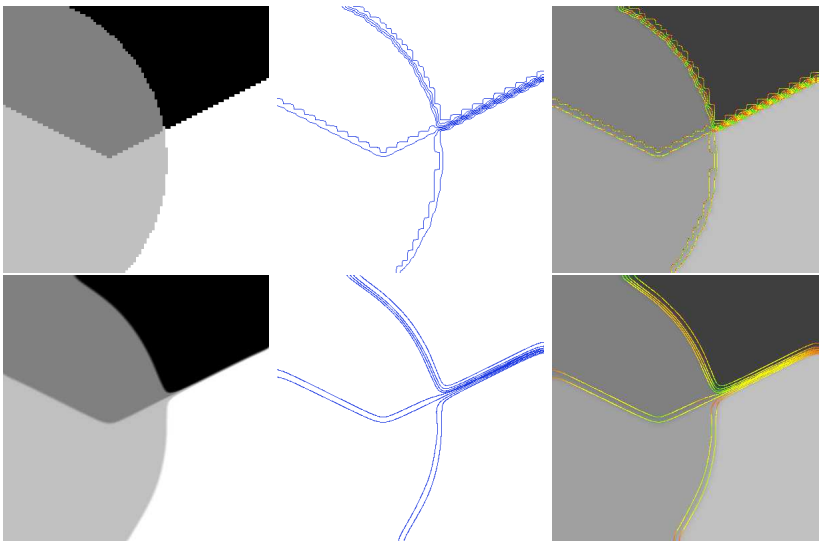


FIG. 6.4. *Top: original image, extracted bilinear level lines, non-filtered curvatures. Bottom: LLAS image, smoothed level lines and curvature map.*

6.3. 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 6.5.

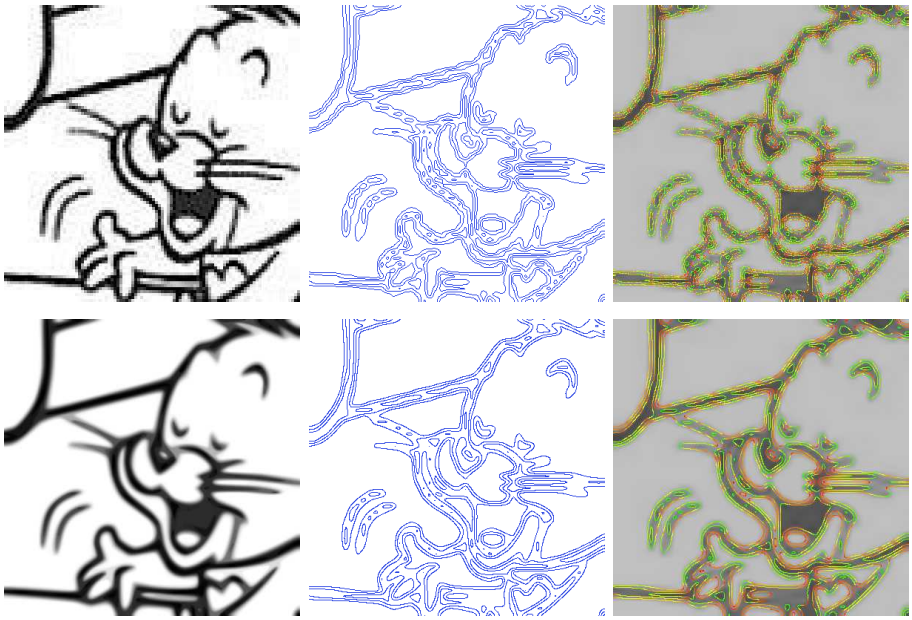


FIG. 6.5. *Top: original image, its corresponding level lines and curvatures. Bottom: LLAS evolution, smoothed level lines and curvature map after filtering.*

A decent recovery is possible even with badly pixelized shapes such as the one reproduced in Figure 6.6. 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.

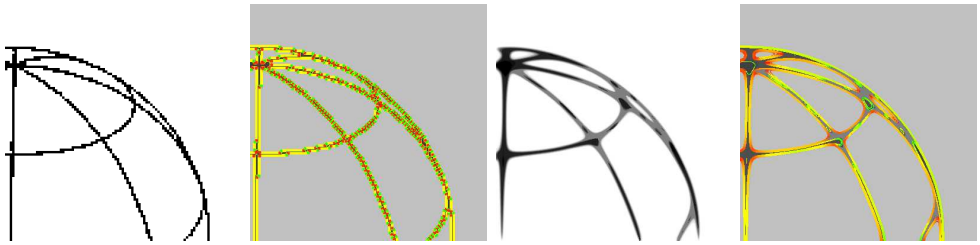


FIG. 6.6. *Top: original image, its corresponding level lines and curvatures. Bottom: LLAS evolution, smoothed level lines and curvature map after filtering.*

But the example in Figure 6.7 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.

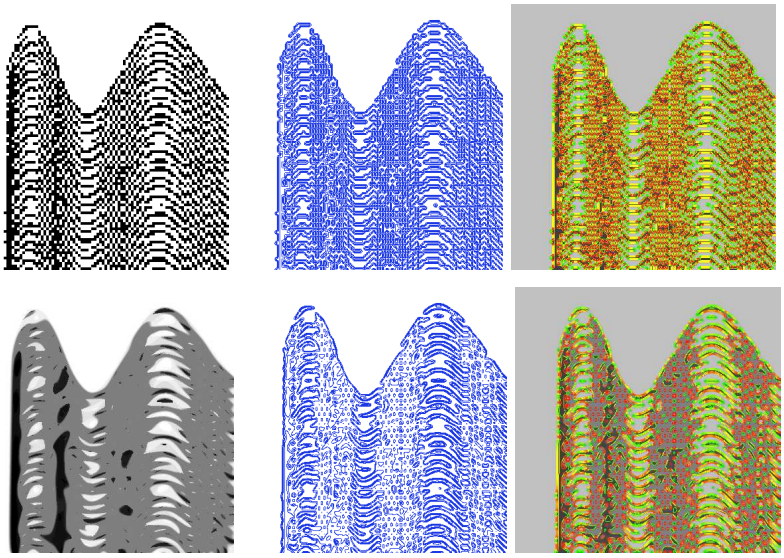


FIG. 6.7. *Top: original image, its corresponding bilinear level lines and curvatures. Bottom: LLAS evolution, smoothed level lines and curvature map after filtering.*

6.4. Pre-attentively undiscriminable textons. Julesz conjectured in his second texture perception theory [22] that two different textures cannot be pre-attentively discriminated if they have the same texton density. For instance the Julesz patterns in Figure 6.8 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.

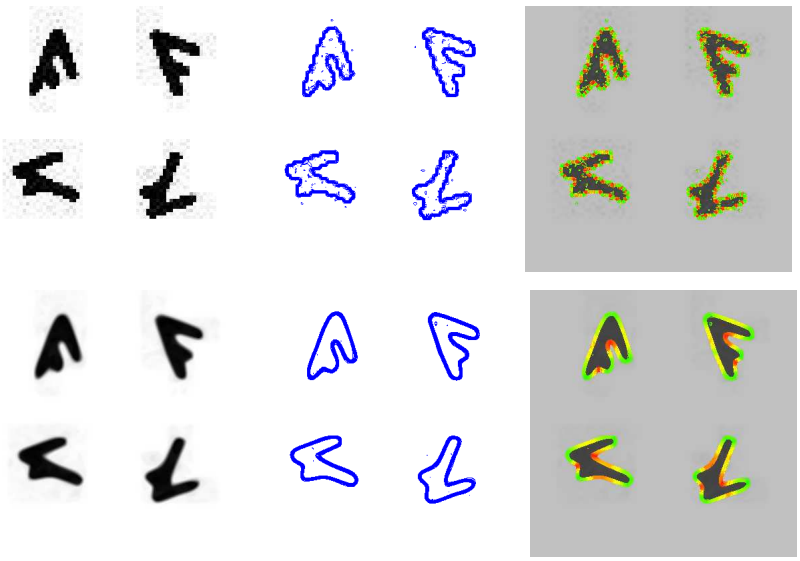
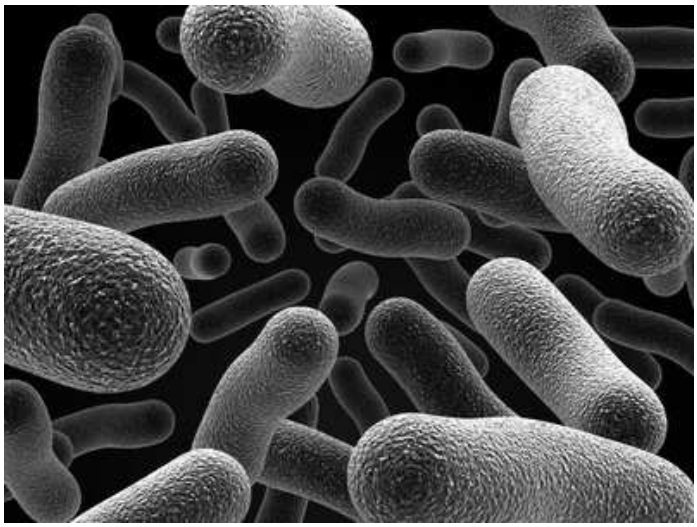
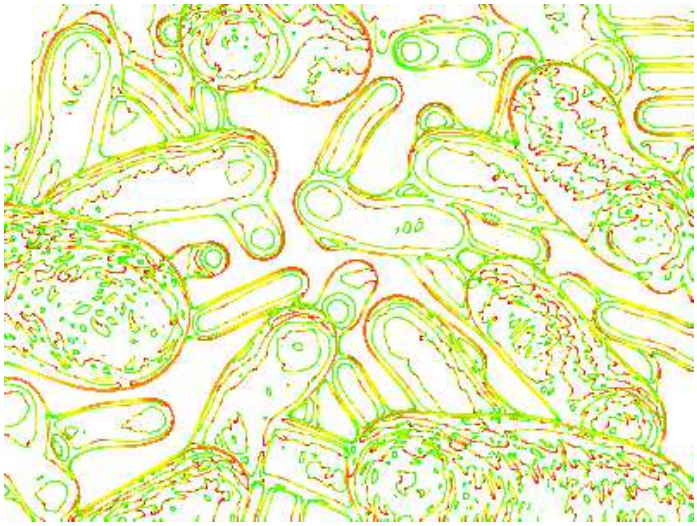


FIG. 6.8. *Top: original image, its corresponding level lines and curvatures. Bottom: LLAS evolution, smoothed level lines and curvature map after filtering.*

6.5. 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 6.9 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.



(a)



(b)

FIG. 6.9. *(a). Original image (b). Curvature Map*

6.6. 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 6.10, 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.

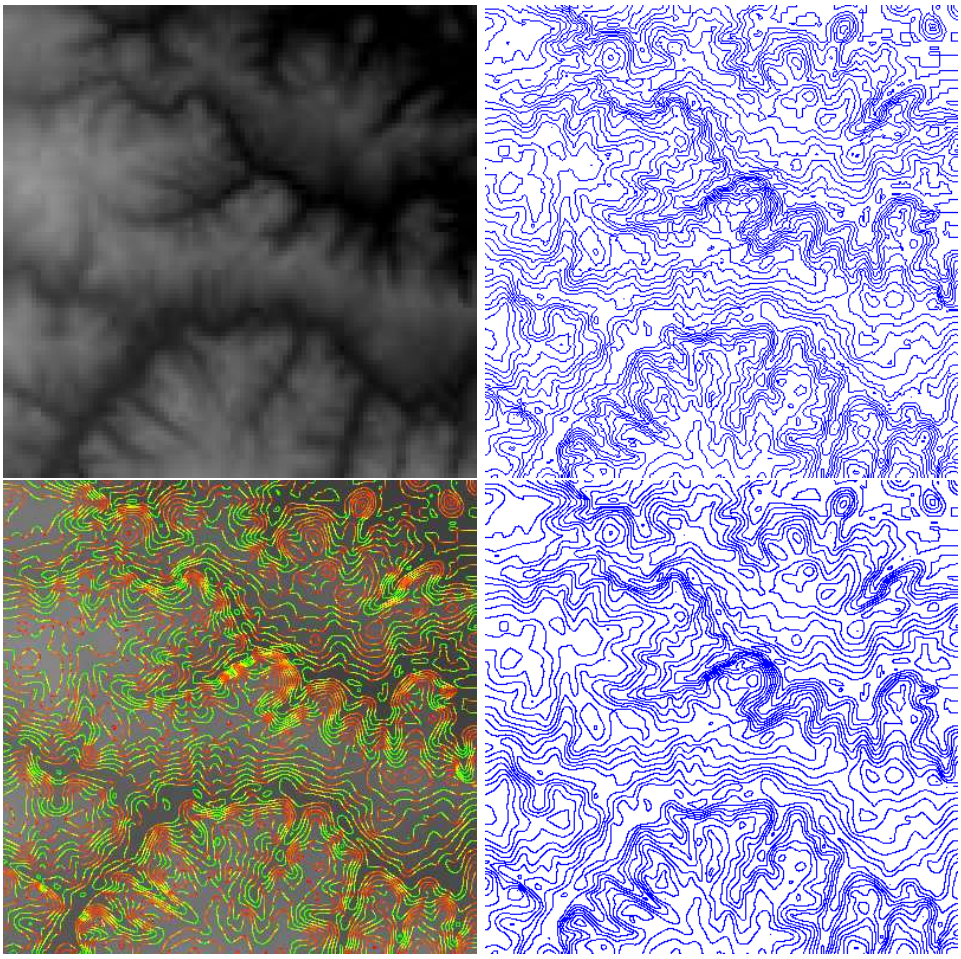


FIG. 6.10. *Digital elevation map, its corresponding level lines (for once a real topographic map), the affine smoothed level lines and their curvature map.*

The fragment of scanned map in Figure 6.11 is exemplary, in its amount of ringing, aliasing, and JPEG artifacts. Such graphic images are satisfactorily restored with short time affine smoothing. The essential ingredient in restoring graphic image, is to remove the lines distortion without creating new level lines. This requirement is respected to the letter by LL(A)S, which only smooths out existing level lines.

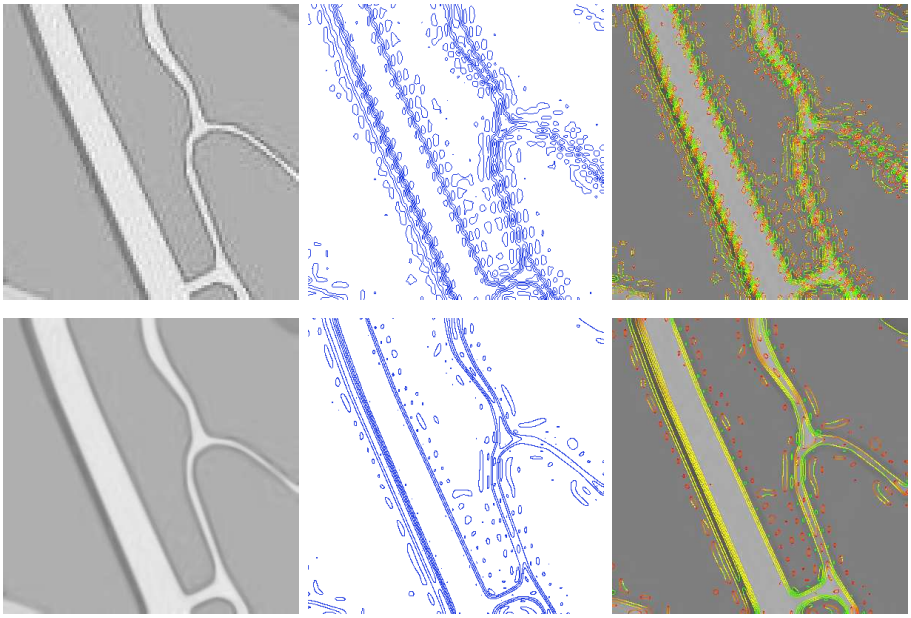


FIG. 6.11. *Top: Piece of map with roads, its corresponding level lines and curvature map before filtering; smoothed image, shortened level lines and curvature map after filtering.*

6.7. Textures. The experiments of Figures 6.12 and 6.13 illustrate the potential use of LLAS to restore the image micro-geometry and to facilitate the identification of smoothly varying shapes in a texture.

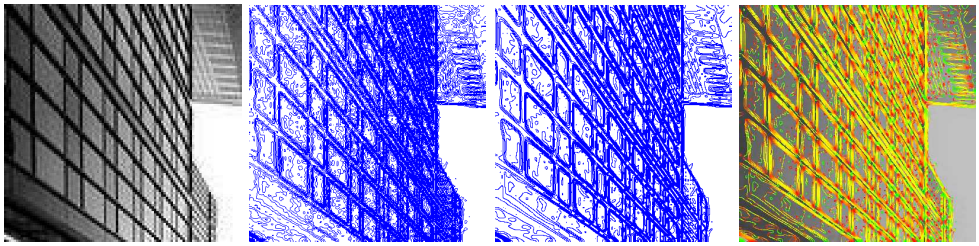


FIG. 6.12. *Original image, extracted level lines, smoothed level lines and curvature map.*

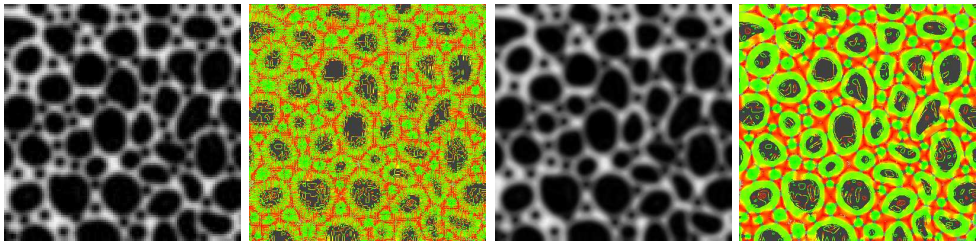


FIG. 6.13. *Original image and its curvature map, filterd image and its curvature map.*

6.8. Paintings. Even on details of paintings, this geometric analysis can be relevant. As already mentioned, the LLAS evolution can be used for noise reduction and picture restoration. In Figure 6.14 the desaliasing successfully restores the paint strokes and improves for example the perception of the pearls and of their shadows.

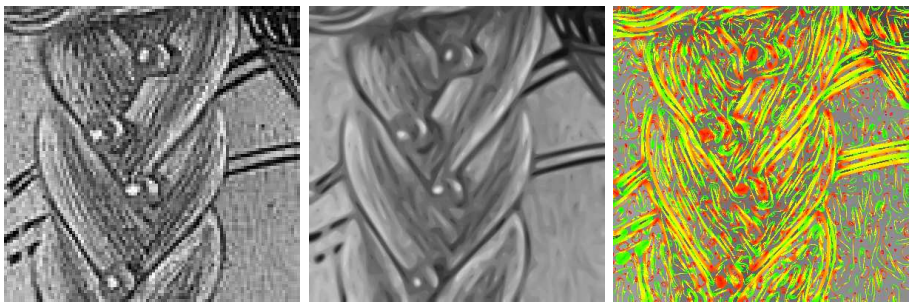


FIG. 6.14. *Original photo-painting, LLAS evolution and curvature map after filtering.*

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 6.15 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.

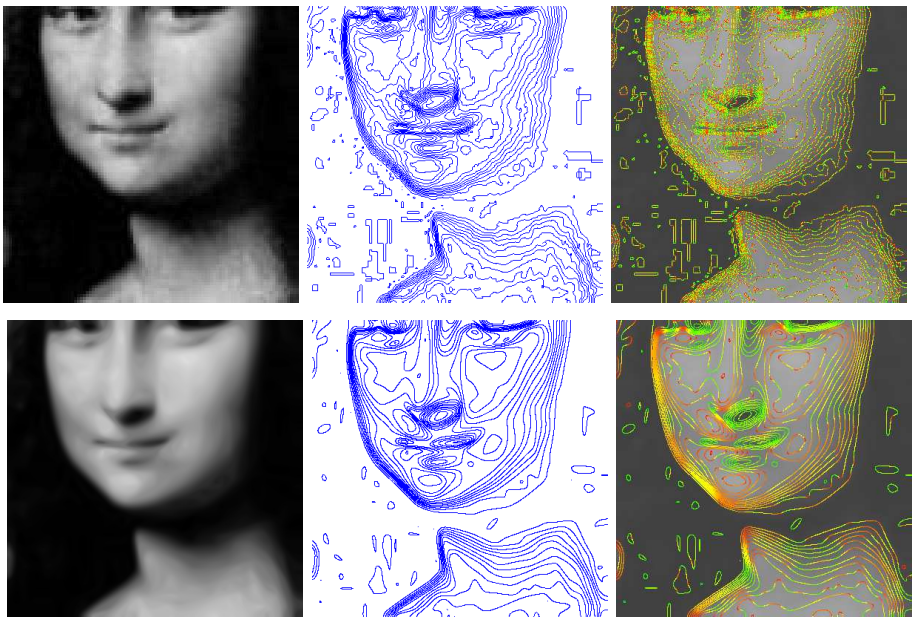


FIG. 6.15. *Extraction with zoom of* Mona Lisa *photograph, its corresponding level lines and curvatures. LLAS evolution, affine smoothed level lines and curvature map after filtering.*

6.9. Text processing. The same good effects are observable with pixelized written text. After the application of LLAS the image in Figure **??** 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.

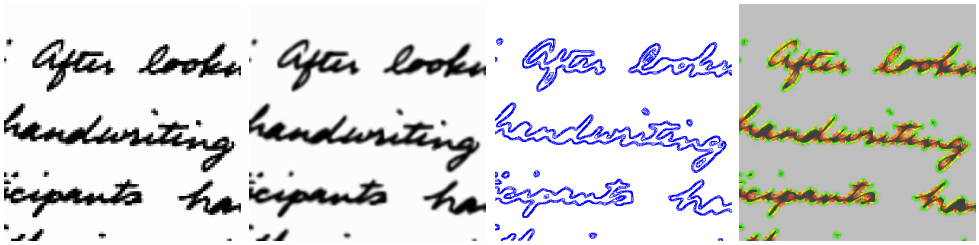


FIG. 6.16. *Original image and its corresponding LLAS evolution, smoothed level lines and curvature map after filtering.*

6.10. 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 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. Observe in Figure 6.17 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.

FIG. 6.17. *Original fingerprint, Level Lines Affine Shortening and its Curvature map.*

7. Conclusion. Full contrast invariance can be restored by the stack filters based on finite difference schemes but they are not sufficient at any scale. Numerical motions based on pixel approximation are quantized, and in particular blind to small curvatures. This drawback was overcome by evolving independently the level curves of the image and by reconstructing from them a new image which has exactly these level lines.

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

There is no chance whatsoever of extending the approach proposed here to 3D images. In 3D the level surfaces evolving by curvature motion can generate singularities. They cannot be efficiently parameterized. The state of the art is therefore to use the Osher-Sethian level set method. Thanks to Grayson's theorem, the 2D case has a very peculiar structure which has been taken advantage of.

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(3):199–257, 1993.

[2] L. Alvarez and J.-M. Morel. Formalization and computational aspects of image analysis. *Acta Numerica*, pages 1–59, 1999.

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

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

[5] G. Barles and P. M. Souganidis. Convergence of approximation schemes for fully nonlinear second order equations. *Asymptotic Analysis*, 4:271–283, 1991.

[6] 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.

[7] F. Cao. *Geometric Curve Evolution and Image Processing*. Number 1805 in Lecture Notes in Mathematics. Springer Verlag, February 2003.

[8] F. Cao and L. Moisan. Geometric computation of curvature driven plane curve evolutions. *SIAM Journal on Numerical Analysis*, pages 624–646, 2002.

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

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

[11] 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.

[12] A. Ciomaga, P. Monasse, and J.-M. Morel. Level lines shortening yields an image curvature microscope. In *Proceedings of the 17th IEEE International Conference on Image Processing, Hong Kong, HK*, p. 4129 - 4132, 2010.

[13] A. Ciomaga and J.M. Morel A proof of equivalence between level lines shortening and curvature motion in image processing. submitted.

[14] Michael G. Crandall and Pierre-Louis Lions. Convergent difference schemes for nonlinear parabolic equations and mean curvature motion. *Numer. Math.*, 75(1):17–41, 1996.

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

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

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

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

[19] F. Guichard and J.-M. Morel. Partial differential equations and image iterative filtering. In I. S. Duff et al., editor, *The State of the Art in Numerical Analysis*, Inst. Math. Appl. Conf. Ser., New Ser. 63, pages 525–562. Institute of Mathematics and its Applications (IMA), Oxford Univ. Press, 1997.

[20] F. Guichard and J.-M. Morel. Image iterative smoothing and P.D.E.'s. Book in preparation, 2000.

[21] 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.

[22] B. Julesz. Textons, the elements of texture perception, and their interactions. *Nature*, 290(5802):91–97, 1981.
[23] G. Kanizsa. *Organization in Vision: Essays on Gestalt Perception*. Praeger, 1979.
[24] J. J. Koenderink and A. J. van Doorn. Dynamic shape. *Biological Cybernetics*, 53:383–396, 1986.
[25] G. Koepfler and L. Moisan. Geometric multiscale representation of numerical images. In *Proc. of the Second International Conference on Scale Space Theories in Computer Vision*, volume 1682 of *Lecture Notes in Computer Science*, pages 339–350, Corfu, Greece, 1999. Springer.
[26] Z. Krivá and K. Mikula. An adaptive finite volume scheme for solving nonlinear diffusion equations in image processing. *Journal of Visual Communication and Image Representation*, 13(1-2):22–35, 2002.
[27] J.-L. Lisani, L. Moisan, P. Monasse, and J.-M. Morel. Affine invariant mathematical morphology applied to a generic shape recognition algorithm. In *Proceedings of the International Symposium of Mathematical Morphology, Palo Alto, CA*, 2000.
[28] J.L. Lisani, L. Moisan, P. Monasse, and J.M. Morel. On the theory of planar shape. *SIAM Multiscale Modeling and Simulation*, 1(1):1–24, 2003.
[29] 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.
[30] A. Mackworth and Mockhtarian F. A theory of multiscale, curvature-based shape representation for planar curves. *IEEE Trans. Pattern Analysis and Machine Intell.*, 14:789–805, 1992.
[31] P. Maragos and R. W. Schafer. Morphological filters. Part II: Their relations to median, order-statistic, and stack filters. *IEEE Trans. Acoust. Speech Signal Process.*, 35:1170–1184, 1987.
[32] 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.
[33] Karol Mikula and Daniel Ševčovič. Solution of nonlinearly curvature driven evolution of plane curves. *Appl. Numer. Math.*, 31(2):191–207, 1999.
[34] Karol Mikula and Daniel Ševčovič. Evolution of plane curves driven by a nonlinear function of curvature and anisotropy. *SIAM J. Appl. Math.*, 61(5):1473–1501 (electronic), 2001.
[35] Lionel Moisan. Affine plane curve evolution: a fully consistent scheme. *IEEE Trans. Image Process.*, 7(3):411–420, 1998.
[36] P. Monasse and F. Guichard. Fast computation of a contrast invariant image representation. *IEEE Trans. on Image Proc*, 9:860–872, 1998.
[37] P. Musé, F. Sur, F. Cao, Gousseau Y., and Morel J.-M. An a contrario decision method for shape element recognition. *International Journal of Computer Vision*, 69:295–315, 2006.
[38] Adam M. Oberman. Convergent difference schemes for degenerate elliptic and parabolic equations: Hamilton-Jacobi equations and free boundary problems. *SIAM J. Numer. Anal.*, 44(2):879–895 (electronic), 2006.
[39] 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.
[40] 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.
[41] G. Sapiro and Tannenbaum A. Affine invariant scale space. *IJCV*, 11(1):25–44, 1993.
[42] 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.
[43] Guillermo Sapiro and Allen Tannenbaum. On invariant curve evolution and image analysis. *Indiana Univ. Math. J.*, 42(3):985–1009, 1993.
[44] Guillermo Sapiro and Allen Tannenbaum. On affine plane curve evolution. *J. Funct. Anal.*, 119(1):79–120, 1994.
[45] J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, San Diego, CA, 1982.
[46] Peter Smereka. Semi-implicit level set methods for curvature and surface diffusion motion. *J. Sci. Comput.*, 19(1-3):439–456, 2003. Special issue in honor of the sixtieth birthday of Stanley Osher.