LEVEL LINES SHORTENING YIELDS AN IMAGE CURVATURE MICROSCOPE

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

[1] CMLA, ENS Cachan, CNRS, UniverSud, 61 avenue du Président Wilson,F -94230, Cachan, France
[2] IMAGINE - LIGM/Université Paris-Est, 19 rue Alfred Nobel, 77455, Marne-la-Vallée, France

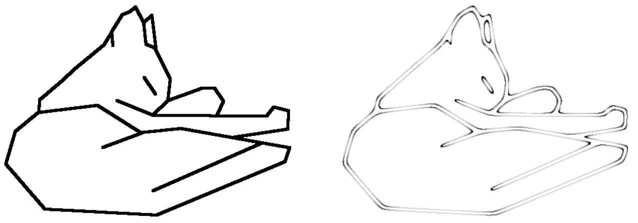
ABSTRACT

This paper presents an image processing algorithm simulating a *sub-pixel* evolution of an image by mean curvature motion or by affine curvature motion. The sub-pixel algorithm computes the image curvature directly on the smoothed level lines, and yields a microscopic visualization of the curvature map revealing many image details, and getting rid of aliasing effects. This "curvature microscope" showing curvatures in false colors runs on line on any image proposed by users at http://www.ipol.im/pub/algo/cmmm_image_curvature_microscope/.

Index Terms— Partial differential equations, scientific visualization, image shape analysis, image reconstruction curvature scale space

1. INTRODUCTION

Attneave's founding 1954 paper [1] on image perception anticipated the numerical analysis of digital pictures by stating 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, a direct computation of curvature on a raw image is impossible. We show in this paper how curvatures can be accurately estimated by a direct computation on level lines after their independent smoothing, as illustrated in Fig. 1.



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

Fig. 1. Attneave's figure illustrating the prominent role of curvature peaks in image perception and its corresponding curvature map computed by Level Lines Shortening.

Asada and Brady [2] introduced the concept of "multiscale curvature". In their paper the contours are approximated by splines and smoothed 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 represent planar shapes from their curvatures, and to compute curvature correctly. One of the difficulties is the extraction of the contours on which the curvature should be computed.

The subject was clarified by several mathematical contributions. Grayson [3] proved that the intrinsic heat equation shrinks and smooths Jordan curves and preserves their topology. The Osher-Sethian level set method [4] implemented the motion by mean curvature of a manifold by applying a new PDE (the scalar mean curvature motion) to the distance function to the manifold.

In parallel Mackworth and Moktharian [5] proposed a fast numerical scheme for curve shortening. But their shape extraction algorithm was unconvincing. Caselles et al. realized the potential of using directly level lines instead of edges and proposed to perform image analysis directly on the topographic map [6]. A fast algorithm computing the topographic map was proposed by Monasse and Guichard [7]. Sapiro and Tannenbaum [8] discovered the affine curve shortening and Alvarez et al. [9] the affine invariant and contrast invariant image smoothing. A remarkably fast and simple geometric algorithm for affine shortening was published by [10]. It has been used for shape identification algorithms in the works of Cao *et al.* [11].

The present paper builds on the above mentioned contributions and describes a complete image processing numerical chain starting from a digital image and ending with an accurate computation and visualization tool of its curvatures. This chain, which had been outlined in [12], is now completed with a subpixel image reconstruction from an arbitrary tree of level lines. It, hopefully, advances Attneave's program and yields anyway what we shall term an *image curvature microscope*. The algorithm can be tested on line and many examples of all kinds are provided.[1]

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

2. LEVEL LINES SHORTENING

2.1. Curvature evolutions

Let $u(\mathbf{x})$ be the continuous image, which is assumed to be at least C^2 in a neighborhood of a point $\mathbf{x_0} \in \mathbb{R}^2$, and assume that its gradient is not null, $Du(\mathbf{x_0}) \neq 0$. The scalar curvature of u at $\mathbf{x_0}$, denoted by $\mathrm{curv}(u)(\mathbf{x_0})$, is defined by

$$\mathrm{curv}(u)(\mathbf{x_0}) = \frac{u_{xx}u_y^2 - 2u_{xy}u_xu_y + u_{yy}u_x^2}{(u_x^2 + u_y^2)^{3/2}}(\mathbf{x_0}). \qquad (1)$$

This scalar curvature at $\mathbf{x_0}$ is linked to the curvature $\kappa(\mathbf{x_0})$ of the level line passing by. Recall that the curvature of a C^2 curve $\mathbf{x}(s)$ parameterized by a length parameter s is defined by $\kappa(\mathbf{x}) := \mathbf{x}''(s)$. The link between the curvature of an image level line $\kappa(\mathbf{x})$ and the scalar curvature $\mathrm{curv}(u)(\mathbf{x})$ at nonsingular points is given in the next formula. Denote by $\mathbf{x} = \mathbf{x}(s)$ the level line of u passing by $\mathbf{x}_0$. Then

$$\kappa(\mathbf{x_0}) = -\mathrm{curv}(u)(\mathbf{x_0})\frac{Du}{|Du|}(\mathbf{x_0}). \qquad (2)$$

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 is necessary. This introduces a new parameter, the smoothing *scale*, denoted hereafter by t. Hence the notion of *curvature scale space* which is associated with curve or image evolutions. The smoothing can be therefore alternatively performed by *curve shortening*,

$$\frac{\partial \mathbf{x}}{\partial t} = \kappa(\mathbf{x}) \qquad (CS) \qquad (3)$$

or by *mean curvature motion*

$$\frac{\partial u}{\partial t} = |Du| curv(u). \qquad (MCM) \qquad (4)$$

As shown by Grayson [3], this (nonlinear) evolution can be applied to any piecewise C^1 curve. The curve instantly becomes smooth and shrinks asymptotically to a circle.

Based on this principle, the *level lines shortening* (LLS) algorithm gives a subpixel evolution by curvature driven flows of both level lines and images. It starts by extracting all level lines of a digital image, with a number of levels sufficient to grant an <u>exact</u> reconstruction of the initial image. Then it simulates a subpixel evolution of the image by moving independently <u>all</u> 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 in Fig. 2. This setting is based on a topological structure, the inclusion tree structure of the level lines as a full and non-redundant representation of an image (see [13]), and on a topological property, namely the monotonicity with respect to inclusion of the curve shortening evolution. The inclusion of the level lines is 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 tree each region surrounded by each level line. At each step the interior of the current level line is filled in with its own level, and these levels are updated when passing to its descendants.

$$\begin{tikzcd} u_0(\cdot) \arrow[r, "level\ lines\ extraction"] \arrow[d, dotted, "MCM/ACM"'] & \{\Sigma_0^{\lambda,i}\}_{\lambda,i} \arrow[d, "CS/AS"] \\ u(\cdot,t) & \{\Sigma_t^{\lambda,i}\}_{\lambda,i} \arrow[l, "reconstruction"'] \end{tikzcd}$$

Fig. 2. Level Lines Shortening scheme.

Denote by $\mathcal{BL}(\Omega)$ the space of bilinear interpolations of digital images defined on a grid of Ω. We prove in a companion paper the next theorem which justifies the algorithm:

Theorem 2.1 *Let $u_0 \in \mathcal{BL}(\Omega)$. Then its Level Lines Shortening evolution $u(\mathbf{x}, t) = LLS(t)u_0(\mathbf{x})$ is a (viscosity) solution for the curvature PDE*

$$\begin{cases} \frac{\partial u}{\partial t} = curv(u)|Du|, & \textit{in } \mathbb{R}^2 \times [0,\infty) \\ u(\cdot, 0) = u_0, & \textit{on } \mathbb{R}^2. \end{cases} \qquad (5)$$

(A similar result holds for $\frac{\partial u}{\partial t} = curv(u)^{\frac{1}{3}}|Du|$, the affine curvature motion (ACM).)

2.2. Image Reconstruction from a set of level lines

The algorithm described here performs an accurate 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. It starts with 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 in preorder (parent before children) and the interior of the current level line filled in with its level λ. Using that order, each level line is painted before its descendants, ensuring that its private pixels (that is, pixels of its interior and not in the interior of any child) are at the correct level while non-private pixels get painted over by the children.

Each closed curve Σ is stored as a polygon defined by its ordered vertices $\{P_k(x_k, y_k)\}_{1\leq k\leq N}$. N depends on Σ, and x_k and y_k are floating point coordinates. We need to fill in all pixels with integral coordinates (j, i) inside the polygon. The filling in of each curve is performed by a fast ray casting algorithm described below.

Polygon intersections with the grid. The goal of this algorithm, which is a preliminary to the filling algorithm, is to find the intersections of a polygonal level line Σ with all horizontal lines $y = i$. For any given i, these are given by the intersections x_l^i, $l = 1 \dots p$ of the segments $[P_k P_{k+1}]$ with the line $y = i$. These intersections are ordered by their abscissae so that $x_1^i \leq x_2^i \leq \cdots \leq x_p^i$. Translating if necessary Σ by a tiny amount ε vertically, the algorithm secures that no y_k is an integer. Therefore p is even, because Σ is a Jordan 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 $[x_{2l+1}^i, x_{2l+2}^i]$.

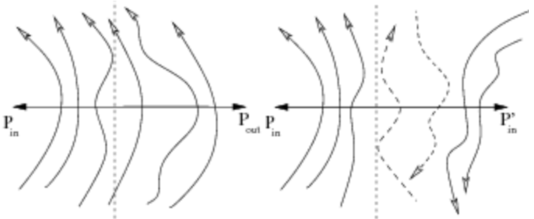


Fig. 3. The $1D$ topology of *level lines* corresponding to different gray values on a dual edge.

Filling the interior. Line by line all odd intervals are enumerated and pixels with ordinate i whose abscissa is inside such an interval are filled in with level λ. 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. As shown in Fig. 3, 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}. 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 tree of inclusions.

3. IMAGE CURVATURE MICROSCOPE

Whenever we talk about curvatures in a digital image, we actually refer to the curvatures of the level lines associated to the image. Most curvature computation algorithms are based on finite difference schemes (FDS) for formula (1). In this case, the curvature depends on the gray values of the neighbor pixels and consequently, high oscillations along transverse level lines appear.

For the sake of precision, we suggest that curvatures should be computed directly on level lines and not on a discrete grid. For a given digital image, one can extract the level lines at given quantization levels. Each level line is stored as a set of ordered points $\Sigma = \{P_i(x_i, y_i)\}_{i=0..n}$, with $P_0 = P_n$. A polygonal line approximation followed by uniform and fine sampling allows to compute realiable curvatures, but only after level line smoothing. The reason for smoothing is that bilinear level lines present oscillations due to the grid, which is an interpolation artefact and thus curvatures wouldn't correspond to our visual perception. In this setting, we give meaning to the curvature of a polygonal line, in terms of pointwise behavior at each vertex. The simplest way is to consider at each point P_i a discrete curvature k_i, by taking the triple (P_{i-1}, P_i, P_{i+1}) and compute k_i as the inverse of the circumscribed radius R_i of this triangle. Set $\overrightarrow{P_{i-1}P_i} := \overrightarrow{u_i} = (u_i^1, u_i^2)$ and its length $u_i = |\overrightarrow{P_{i-1}P_i}|$, respectively $\overrightarrow{v_i} = \overrightarrow{P_{i-1}P_{i+1}} = (v_i^1, v_i^2)$, with $v_i = |\overrightarrow{P_{i-1}P_{i+1}}|$.

Lemma 3.1 *The curvature at vertex P_i is given by*

$$k_i = 2\frac{u_i^1 u_{i+1}^2 - u_i^2 u_{i+1}^1}{u_i u_{i+1} v_i}. \tag{6}$$

The computed curvature is then attributed to the pixel surrounding P_i. A curvature image is created by associating to each dual pixel (a pixel of vertices at integer coordinates) an average of all curvatures computed in it.

To summarize, the curvature map is computed as follows:

1. extract the tree of level lines at given quantization levels;
2. perform uniform, fine sampling of the level lines;
3. short time curve shortening to each level line;
4. compute discrete curvatures;
5. register at each dual pixel the average of all discrete curvatures computed in and create thus the curvature image.

4. NUMERICAL RESULTS

A simple binary shape has its curvatures computed in two different ways: by a FDS for formula (1) and by smoothing level lines and displaying with high resolution the corresponding curvature map. In the first case the curvature presents oscillations, whereas the second result is coherent with our perception of curvature. It is zero on straight edges, constant on arcs of circles and high at bent parts, such as corners of shapes. An example is presented in Fig. 4, where zero curvatures are displayed in gray, positive ones shade from gray to white and negatives from gray to black.

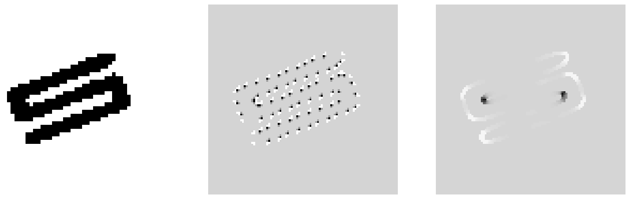


Fig. 4. Curvatures computed with a FDS and by LLS.

As illustrated in Fig. 5, LLS can be used for a graphic quality improvement of contours. Performing a scaled zoom on the image one can expect to have one level line passing through each dual pixel, and thus to observe more and more exactly the level lines and curvatures at microscopic scale.

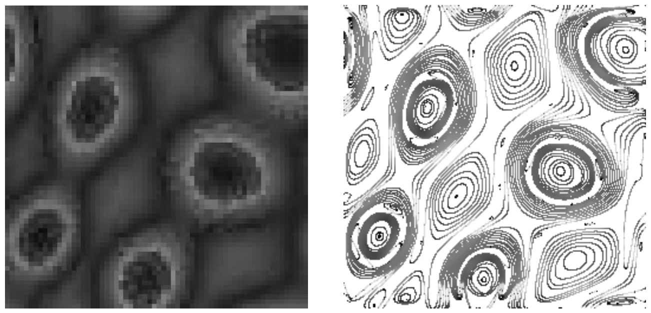


Fig. 5. Scaled zoom on a texture. Curvatures in absolute value are displayed along the level lines.

We display in Fig. 6 an original image, its level lines and the corresponding curvature map before level lines filtering, respectively its LLS evolution, smoothed level lines and the curvature map. Observe that LLS removes JPEG artefacts and performs a nice geometric restoration.

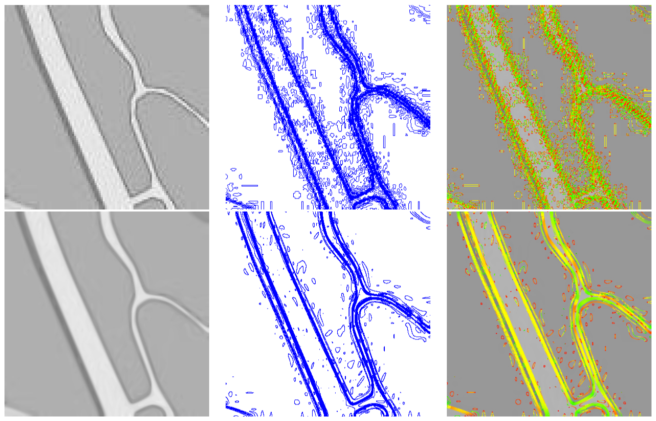
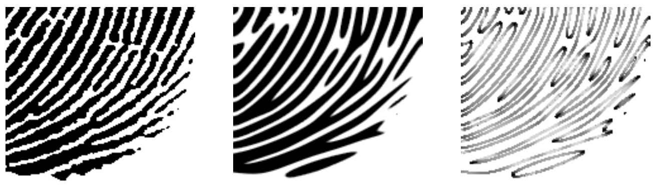


Fig. 6. Piece of a map, its corresponding level lines and curvatures before (top) and after (botom) LLS filtering.

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. As shown in Fig. 7, LLS removes the ridge border oscillations and provides a smooth version of the fingerprint on which the curvature map locates its characteristic points .



(a). Fingerprint (b). LLS evolution (c). Curvatures

Fig. 7. LLS tears apart ridges and emphasizes crossovers.

5. CONCLUSION

The first outcome of 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.

6. REFERENCES

[1] F. Attneave, "Some informational aspects of visual perception," *Psychological review*, vol. 61, no. 3, pp. 183–193, 1954.

[2] H. Asada and M. Brady, "The curvature primal sketch," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 1, pp. 2–14, January 1986.

[3] Matthew A. Grayson, "The heat equation shrinks embedded plane curves to round points," *J. Differential Geom.*, vol. 26, no. 2, pp. 285–314, 1987.

[4] S. Osher and J. Sethian, "Fronts propagating with curvature-dependent speed: Algorithms based on the Hamilton–Jacobi formulation," *J. Comput. Phys.*, vol. 79, no. 1, pp. 12–49, 1988.

[5] A. K. Mackworth and F. Mokhtarian, "A theory of multiscale, curvature-based shape representation for planar curves," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 14, pp. 789–805, 1992.

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

[7] P. Monasse and F. Guichard, "Fast computation of a contrast invariant image representation," *IEEE Transactions on Image Processing*, vol. 9, no. 5, pp. 860–872, 2000.

[8] G. Sapiro and A. Tannenbaum, "Affine invariant scale space," *International Journal of Computer Vision*, vol. 11, no. 1, pp. 25–44, 1993.

[9] L. Alvarez, F. Guichard, P.-L. Lions, and J.-M. Morel, "Axioms and fundamental equations of image processing," *Arch. Rational Mech. Anal.*, vol. 16, no. 9, pp. 200–257, 1993.

[10] L. Moisan, "Affine plane curve evolution: A fully consistent scheme," *IEEE Transactions on Image Processing*, vol. 7, no. 3, pp. 411–420, 1998.

[11] F. Cao, J.L. Lisani, J.M. Morel, P. Musé, and F. Sur, *A Theory of Shape Identification*, Springer Verlag, 2008.

[12] G. Koepfler and L. Moisan, "Geometric multiscale representation of numerical images," in *2nd Internat. Conf. on Scale Space Theories in Comp. Vision*. 1999, vol. 1682 of *Lecture Notes in Computer Science*, pp. 339–350, Springer.

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