Geometric Multiscale Representation of Numerical Images

Georges Koepfler[1,2] and Lionel Moisan[1]

[1] Centre de Mathématiques et de Leurs Applications (CNRS UMR 8536),
Ecole Normale Supérieure de Cachan,
61 avenue du président Wilson, 94235 Cachan cedex, France
[2] Université René Descartes
UFR de Mathématiques et Informatique, équipe PRISME,
Centre Universitaire des Saints-Pères,
45 rue des Saints-Pères, 75270 PARIS cedex 06, France
koepfler@cmla.ens-cachan.fr moisan@cmla.ens-cachan.fr

Abstract. We explain how a discrete grey level image can be numerically translated into a completely pixel independent geometric structure made of oriented curves with grey levels attached to them. For that purpose, we prove that the Affine Morphological Scale Space of an image can be geometrically computed using a level set decomposition/reconstruction and a well adapted curve evolution scheme. Such an algorithm appears to be much more accurate than classical pixel-based ones, and allows continuous deformations of the original image.

1 Introduction

If a mathematician had to examine recent evolutions of image analysis, he would certainly notice a growing interest for geometric techniques, relying on the computation of differential operators like orientation, curvature, ... or on the analysis of more global objects like level curves. Of course Fourier or wavelet analysis are still very efficient for image compression for example, but in order to analyze larger scales geometric approaches seem to be more relevant. At large scales a real-world image can hardly be considered –like a sound signal– as a superimposition of waves (or wavelets), since the main formation process relies on occlusion, which is highly nonlinear. This is not without some mathematical consequences : in this context, images are more likely to be represented in a geometrical space like $BV(\mathbb{R}^2)$, the space of functions on $\mathbb{R}^2$ with bounded variation, than in the more classical $L^2(\mathbb{R}^2)$ space. From a practical point of view, the question of the numerical geometric representation of an image certainly deserves to be investigated, since images have been described so far by arrays of numbers (or wavelet/DCT coefficients for compressed images). It is likely that in the future alternative geometric descriptions will be commonly used, relying on some level-set/texture decomposition like the one proposed in [7].

In this paper, we show how it is possible to compute numerically a completely geometric and multiscale representation of an image, for which the notion of

pixel disappears (though it can be recovered). Our algorithm is a fully geometrical implementation of the so-called Affine Morphological Scale Space (AMSS, see [1]), described in Sect. 2. Due to its contrast invariance, this scale space is equivalent to the affine curve shortening process described in [14], for which a fully geometrical algorithm has been recently proposed in [10]. A simplified version of this scheme, described in Sect. 3, allows to process all level curves of an image with a high precision in a couple of minutes. In association with level set decomposition and reconstruction algorithms, described in Sect. 4, we thus compute the AMSS of an image with much more accuracy than classical scalar schemes, as is shown in Sect. 5. Another interest of this method is that it yields a contrast-invariant multiscale geometric representation of the image that provides a framework for geometry based analyses and processing. We illustrate this in Sect. 6 by applying our algorithm to image deformation.

2 The Affine Morphological Scale Space

A natural way of extracting the geometry of an image consists in the level set decomposition inherited from Mathematical Morphology. Given an image u viewed as an intensity map from $\mathbb{R}^2$ to $\mathbb{R}$, one can define the (upper) level sets of u by

$$\chi_\lambda(u) = \{\mathbf{x} \in \mathbb{R}^2;\ u(\mathbf{x}) \geqslant \lambda\}.$$

This collection of planar sets is equivalent to the function u itself since one has the reconstruction formula

$$u(\mathbf{x}) = \sup\{\lambda;\ \mathbf{x} \in \chi_\lambda\}.$$

The main interest of this representation is its invariance under contrast changes : if g is an increasing map from $\mathbb{R}$ to $\mathbb{R}$ (i.e. a contrast change), then one has

$$\chi_{g(\lambda)}(g(u)) = \chi_\lambda(u).$$

Hence, the collection of all level sets of an image does not depend a priori on the global lightning conditions of this image, and is thus an interesting geometrical representation.

Now, because an image generally contains details of different sizes, the notion of scale-space has been introduced. It consists in representing an original image $u_0(\cdot)$ by a collection of images $(u(\cdot,t))_{t\geqslant 0}$ which are simplified versions of u_0 such that $u(\cdot,0) = u_0(\cdot)$ and, with increasing scale t, the u_t represent more and more coarser versions of u_0. There are plenty of possibilities for such representations, but it is possible to reduce them by demanding strong invariance properties from the operator T_t which transforms $u_0(\cdot)$ into $u(\cdot,t)$. In particular, it is possible to enforce the level set decomposition evoked above to be compatible with the scale-space representation, in the sense that the λ-level set of $u(\cdot,t)$ only depends on the λ-level set of u_0. If one asks, in addition, for other properties like regularity, semi-group structure, and Euclidean Invariance (i.e. translation and rotation

invariance), then according to [1] one reduces the possibilities to the choice of a nondecreasing continuous function F governing the scale space given by

$$\frac{\partial u}{\partial t} = |Du|\, F\left(\mathrm{div}\frac{Du}{|Du|}\right), \tag{1}$$

where Du represents the spatial gradient of u. In this paper we have chosen the particular case of the Affine Morphological Scale Space, given by $F(s) = s^{1/3}$, for mainly two reasons :

First, this is the only case which yields an additional invariance property called Affine Invariance :

$$T_t(u_0 \circ \phi) = (T_t u_0) \circ \phi \quad \text{for any} \quad \phi(\mathbf{x}) = A\mathbf{x} + b, \;\; A \in SL(\mathbb{R}^2), \;\; b \in \mathbb{R}^2. \tag{2}$$

This property allows to perform affine-invariant shape recognition under occlusions (see [5]) and to compute local affine-invariant features like affine curvature for example.

Second, there exists a fully consistent geometric scheme (see [10]) for solving the level curve evolution induced by the AMSS,

$$\frac{\partial \mathbf{C}}{\partial t} = \kappa^{1/3}\, \mathbf{N}. \tag{3}$$

Here $\mathbf{C}$ is any point of a level curve, κ the local curvature and $\mathbf{N}$ the normal vector at this point. In particular, this scheme guarantees that the inclusion of any two sets is preserved by the evolution (inclusion principle). In the simplified version described in Sect. 3, it is very fast (linear complexity) and robust, as only areas and middle points are computed.

3 A Fast Geometric Scheme

The numerical implementation of the affine scale space of a curve given by (3) can be realized in several ways. For our purpose, an ideal algorithm should satisfy, up to a given computer precision, the following properties :

P1: preserve inclusion, which is necessary for level set reconstruction;
P2: be affine invariant, since the scale-space is;
P3: have linear complexity, so that all level curves of an image can be processed with a high precision in a reasonable time.

Of course, algorithms based on scalar formulations (see [15]) are not relevant here, since our goal is precisely to get rid of pixel based representations. In any case, such algorithms satisfy neither P1 nor P2, and are not computationally efficient (in terms of time and memory) if an accurate precision is needed (e.g. 100 points per original pixel). The purpose of this paper is to present a scheme

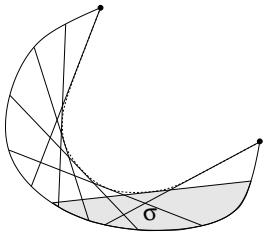

Fig. 1. Affine erosion (- - -) of a convex curve (—).

opposite to Sethian's formulation [1], since we want to solve a scalar (contrast-invariant) evolution equation with a geometric algorithm. We cannot either use local point evolution schemes (naive ones or more refined ones like in [9]) since they do not guarantee P1 (because they rely on a local estimation of the curvature that is not directly connected with a global property like inclusion), and for the same reason P2 would be uncertain (and might depend on the discretization). This is the reason why we started from the geometric scheme described in [10], for it satisfies P1 and P2. This scheme is based on a simple operator called *affine erosion*, which we define now.

Consider a (non necessarily closed) convex parameterized curve $C : [a, b] \mapsto \mathbb{R}^2$ and an area parameter σ. Define a σ-chord set of C as the region with area σ that is enclosed by a segment $[C(s_1)C(s_2)]$ and the corresponding piece of curve $C([s_1, s_2])$. Then, the σ-affine erosion of C, written $E_\sigma(C)$, can be defined as the segment-side boundary of the union of these σ-chord sets (see Fig. 1). Without going into details (which might be found in [10] and [11]), we briefly recall two main results about the affine erosion :

First, the evoked boundary is essentially obtained by the middle points of the segments $[C(s_1)C(s_2)]$ defining the σ-chord sets (minus some "ghosts parts" that do not appear in general, and plus two end-segments if the curve is not closed)

Second, the infinitesimal iteration of such an operator asymptotically yields the affine scale space of the initial curve C_0 (with fixed endpoints if C_0 is not closed), one has

$$(E_\sigma)^n (C_0) \to C(\cdot, t)) \quad \text{as} \quad n \to +\infty,\ \sigma \to 0 \quad \text{and} \quad \frac{1}{2}\left(\frac{3}{2}\right)^{\frac{2}{3}} n\sigma^{2/3} \to t\,,$$

where $C(\cdot, t)$ is defined from C_0 by (3).

The scheme presented in [10] relies on a more general definition of the affine erosion that also applies to non-convex curves. Compared to the convex case, the computations are more complex since the computation of curve intersections may be needed, which requires careful programming and causes the complexity

[1] The main interest of scalar formulations is that they naturally handle topological changes of the level sets : it has however been proved in [2] that no such changes occur for the affine scale space.

of the algorithm to become quadratic in the number of vertices of the polygonal curve to be processed. This is the reason why, as suggested in [10], we have chosen an alternative scheme based on the separate treatment of each convex part. Given a possibly non-convex and non-necessarily closed polygonal curve, we iterate the following three-step process :

1. Break the curve into convex components (by "cutting" inflection segments at their middle point), and compute the minimum value σ_{min} of the area of any non-closed component.
2. Define $\sigma_{real} = \min(\sigma_{min}, \sigma)$ and apply a discrete affine erosion of area σ_{real} to each convex component.
3. Concatenate the obtained pieces of curves in order to obtain a new (possibly non-convex) curve.

This approach yields a good approximation of the exact affine-erosion of C. In fact, the only difference may occur near inflection points of C, but in practice for reasonable values of σ both evolutions give the same result (see [11]). The main advantage of this simplified algorithm is that it is very fast (it has linear complexity) and it is very robust, since each evolution is obtained by middle points of segments whose endpoints lie on the original curve and whose selection only relies on an area computation.

4 The Complete Algorithm

Our geometric multiscale representation algorithm for numerical images is made out of the following 5 steps.

Step 1: decomposition.

The level set extraction is done currently in a straightforward way. Using 4-connectedness, we extract, for each grey value which appears in the image, the corresponding level sets (upper or lower) and keep the oriented border such that the level set lies on the left of the curve. We thus obtain a set of curves which are either closed or start and end on the image border. For each curve we keep the associated grey level, so that we have a representation that is completely equivalent to the initial image. It is important to notice that no interpolation is made to extract these level curves : the pixels are simply considered as adjacent squares.

Step 2: symmetrization.

Then, in order to get Neumann boundary conditions for the affine scale space, we have the possibility to symmetrize the level lines which start and end on the border, which guarantees that the curve will remain orthogonal to the image border during the evolution. Curves with end points on the same side are reflected once, curves with end points on adjacent sides are reflected twice, thus yielding closed curves. Finally curves with end points on opposite sides are reflected once at each side (they should, theoretically, be reflected infinitely many times, but in practice once is enough). Without this symmetrization the endpoints of the non-closed level curves would remain fixed.

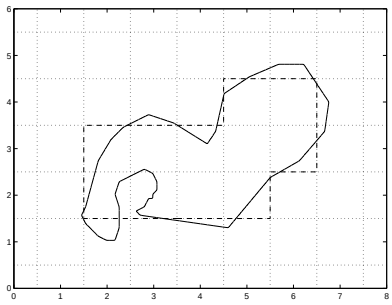

Fig. 2. Example of rasterization of a floating-point curve (—) into a pixel-separating integer curve (- - -).

Step 3 : AMSS.
At this stage, we process all level curves with the geometric implementation of the affine scale space described in Sect. 3. This involves two parameters : the scale of evolution t and the precision ε at which curves are computed. We normalize ε such that $1/\varepsilon$ corresponds to the number of points that will be used to describe a one-pixel-length curve.

Step 4: geometric transformation and/or computation.
Once achieved steps 1 to 3, we have a smooth geometric description of the image that allows to perform any geometric transformation and/or computation. We shall give an example of image deformation in section 6, but there are many other possibilities of geometric processing. For example, one can simply remove level sets that are too small, or too oscillatory (see [12]), or satisfying any geometric criterion that can be estimated on a smooth curve.

Step 5: rasterization and reconstruction.
After transforming the level lines of the initial image, we need to reconstructed the corresponding image. This is done by filling in the level sets and using the grey level information associated with each level line. This step is more complex than the extraction of level lines, since now the level curves are made of points with non-integer coordinates, and thus we have to decide whether a pixel is inside the set or not. We first rasterize the curves by taking care to respect the inclusion principle, using an adaptation of Bresenham's algorithm [2], so that a pixel belongs to the side of the curve where more than half of its area is. We added a special treatment for very near points (sub-pixel). Our implementation allows a good approximation of the real curves. Only some very small, sub-pixel, details of the eroded curve might be lost during the rasterization, such as curves enclosing an area smaller than one pixel.

[2] Bresenham's algorithm is a well known algorithm from computer graphics which allows to join two points which have floating-point coordinates and are distant by more than a couple of pixels (see [3] and any computer graphics book or internet site).

Complexity of the Algorithm

The complexity of the different steps of the algorithm depends on the following parameters : N, the number of pixels contained in the original image (typically 10^6) ; G, the number of grey-levels contained in the original image (typically 256) ; ε, the precision (number of points per pixel) at which the level curves need to be computed (typically from 1/100 to 1/2) ; t, the scale at which the level curves are smoothed, and N', the number of pixels contained in the reconstructed image. Table 1 gives upper bounds of time and memory complexity for the steps described above.

	computation time	memory
decomposition	$N \times G$	$N \times G$
affine scale space	$N \times G \times t/\varepsilon$	$N \times G/\varepsilon$
rasterization	$N \times G/\varepsilon$	$N \times G/\varepsilon$
reconstruction	$N' \times G$	$N' \times G$

Table 1. Complexity of the proposed algorithm

Notice that the upper bound of $N \times G$ points to describe all level lines of G is very rough. For the classical Lena image (see Fig 5, one has $N = 256^2$, $G = 238$, and the decomposition yields in 10 seconds about 48000 curves and 1.1 million points. Then the affine scale space computation for $\varepsilon = 1/2$ takes 2.5 minutes and yields 13000 curves and 0.78 million points. The final rasterization and reconstruction takes 10 seconds.

5 Comparison with Scalar Schemes

The purpose of this section is to compare the geometric algorithm we proposed with explicit scalar schemes based on the iteration of a process like

$$u^{n+1} = u^n + |Du^n| \left(\mathrm{div} \frac{Du^n}{|Du^n|} \right)^{1/3},$$

where Du^n and $\mathrm{div}\frac{Du^n}{|Du^n|}$ are computed using finite differences on a 3x3 neighborhood of the current pixel (see [6]) or using a non-local estimation of the image derivatives obtain by a Gaussian convolution (see [13]). Such a scheme is strongly limited by the grid : the localization of the level curves is known up to a precision of the order of the pixel size (even when interpolation is used), and affine-invariance could only be achieved at large scales (but even rotation-invariance is difficult to ensure at all scales, as noticed in [13]). Another striking side effect of such scalar schemes is that they need to produce artificial diffusion in the gradient direction. In other terms, a scalar scheme cannot be contrast invariant, and in practice new grey levels (and consequently new level curves) are

created. The reason is the following : for a purely contrast-invariant algorithm defined on a fixed grid, a point of a level curve either does not move or moves by at least one pixel. This constraint causes small (i.e. large scale) curvatures to be treated as zero, and is for that reason incompatible with a curvature-driven evolution like AMSS.

These effects are illustrated on Fig. 3. We have chosen an image of size 100×70 which has been magnified 10 times and subsampled to only 10 20-spaced grey levels, see the first line. The left column presents the image, the right column the level sets. In the second line we show the result of our algorithm, no level sets (i.e. grey levels) are created, the level lines are smoothed. The last line presents the result of a classical scalar algorithm. As expected, the scalar scheme produces artificial diffusion which causes a multiplication of the level lines. This can be seen in the left half of the level lines image where 5-spaced level lines are represented, in the right part 20-spaced level lines are represented which should be the only present. One can also remark some anisotropy in that side effect diffusion : it is more attenuated along the directions aligned with the grid (i.e. horizontal or vertical directions), which enforces the visual perception of this grid.

6 Applications

6.1 Visualization of Level Curves

The level sets of an image are generally so irregular that only a few of them can be visualized at the same time. Extracting and processing independently each level curve of an image produces an interesting tool to visualize clearly the level lines of a given image, as illustrated in Fig. 5.
Here, we can see all 4-spaced level lines of the Lena image thanks to the smooth geometric representation provided by the geometric Affine Morphological Scale Space. Such a superimposition shows interesting information about the geometric structure of the image.

6.2 Image Deformation

In this part, we show how our algorithm can be used to apply a geometric transform to an image. In the experiments that follow, projective or affine transforms are used, but more complex geometric transform will work as well. Let $u(i,j)$ be a given, discrete image, how can one define an approximation (or interpolation) $\tilde{u}$ of u that allows to build the transformed image

$$v(i,j) = \tilde{u}\left(\frac{ai+bj+c}{di+ej+1}, \frac{fi+gj+h}{di+ej+1}\right),$$

where a, b, c, d, e, f, g, h are given coefficients (that may vary) ? One can distinguish between two kinds of methods. The first are continuous (and explicit) methods, for which a continuous model for $\tilde{u}$ is explicitly given and computed

from u once and for all. The second are discrete (and implicit) methods, for which $\tilde{u}$ is implicitly defined and must be estimated for each discrete grid.

For example, zero order interpolation defined by $\tilde{u}(i,j) = u([i+1/2],[j+1/2])$, where $[x]$ represent the integer part of x, or bilinear interpolation and higher order generalizations are explicit methods. On the opposite, image interpolation/approximation algorithms based on the minimization of a certain error between discrete images (like the Total Variation used in [8]) are implicit methods. In fact, this distinction is rather formal if the practical criterion is not "how is $\tilde{u}$ defined ?", but "how much time does it take to compute $\tilde{u}$?". For example, Fourier representation is an explicit method, but for non-Euclidean transformations it is computationally expensive. Indeed if N is the number of pixels of the original image u, it requires N operations to compute the value of $\tilde{u}$ in a given point. From that point of view, our representation is a compromise between computation time (once the level lines have been extracted and smoothed, the deformation and the reconstruction processes are fast) and accuracy (the geometry of the level sets is precisely known). We do not affirm that the Affine Morphological Scale Space yields the best image approximation : it is geometrically better than bilinear interpolation (for which pixelization effects remain), but less accurate than sophisticated image interpolation algorithms like [8]. However, we proved that it can be precisely computed in a reasonable time and then allowing any kind of geometric deformation.

We compared deformations yielded by our method, zero and bilinear interpolation on two images :

On the simple binary image (left in Fig. 4), we applied an affine deformation using three different methods. A bilinear interpolation (left part of middle image) and a zero-order interpolation (right part of middle image). Our geometric representation described in this paper gives the right image. Contrary to classical methods, a geometric curve shortening quickly provides a good compromise between pixelization effects, accuracy and diffusion effects.

In Fig. 6 we present a satellite image from which we have simulated a projective view (from right to left as indicated by the black trapezoid). Fig. 7 left shows the results with zero-order interpolation (left part) and bilinear interpolation (right part). Our algorithm, using the geometric implementation of the affine morphological scale space gives the result shown in Fig. 7 right.

7 Conclusion

In this paper, we described how the Affine Morphological Scale Space of an image can be implemented in a geometric manner. Compared to classical scalar schemes, the main advantages are a much higher accuracy both in terms of image definition and in terms of fidelity to the scale space properties (contrast-invariance and affine-invariance). The algorithm needs a large amount of memory but it still is rather fast, and the representation it induces also allows very fast geometric image deformations and contrast changes.

Our method relies on a level-set decomposition/reconstruction and on a particular geometric algorithm for affine curve shortening, but it could be generalized to other curve evolutions, as similar geometric algorithm for general curvature-driven curve evolutions are likely to appear soon. Another generalization could be made by using some image interpolation for the extraction of the level-sets : however, in this case the representation will generally no be contrast-invariant any more. A more geometric extension of the algorithm relying on the interpolation of new level lines using the Absolute Minimizing Lipschitz Extension (see [4]) could also be investigated for visualization tasks.

References

1. L. Alvarez, F. Guichard, P.L. Lions, J.M. Morel, "Axioms and fundamental equations of image processing", *Archives for Rational Mechanics* 123, pp. 199-257, 1993.
2. S. Angenent, G. , A. Tannenbaum, "On the affine heat equation for nonconvex curves", preprint.
3. J. E. Bresenham, "Algorithm for computer control of a digital plotter", *IBM Syst. J.* 4:1, pp. 25-30, 1965.
4. V. Caselles, J.-M. Morel "An Axiomatic Approach to Image Interpolation", *IEEE Transactions On Image Processing*, vol. 7:3, pp. 376-386, march 1998.
5. T. Cohignac, "Reconnaissance de formes planes", PhD dissertation, Ceremade, 1994.
6. T. Cohignac, F. Eve, F. Guichard, J.-M. Morel, "Numerical analysis of the fundamental equation of image processing", preprint Ceremade, 1992.
7. J. Froment, "A Functional Analysis Model for Natural Images Permitting Structured Compression", preprint CMLA, 1998.
8. F. Guichard, F. Malgouyres, "Total Variation Based Interpolation", *Proceedings of Eusipco'98*, vol. 3, pp.1741-1744.
9. K. Mikula, "Solution of nonlinear curvature driven evolution of plane convex curves", *Applied Numerical Mathematics*, vol. 23, pp. 347-360, 1997.
10. L. Moisan, "Affine Plane Curve Evolution : a Fully Consistent Scheme", *IEEE Transactions On Image Processing*, vol. 7:3, pp. 411-420, march 1998.
11. L. Moisan, "Traitement numérique d'images et de films : équations aux dérivées partielles préservant forme et relief", PhD dissertation, Ceremade, 1997.
12. P. Monasse, F. Guichard, "Fast computation of a contrast-invariant image representation", preprint CMLA, 1998.
13. W.J. Niessen, B. M. ter Haar Romeny, M. A. Viergever, "Numerical Analysis of Geometry-Driven Diffusion Equations", in *Geometry-Driven Diffusion in Computer Vision*, Bart M. ter Haar Romeny Ed., Kluwer Acad. Pub., 1994.
14. G. Sapiro, A. Tannenbaum, "Affine invariant scale-space", *Int. J. Comp. Vision*, vol. 11, pp. 25-44, 1993.
15. J.A. Sethian, *Level Set Methods*, Cambridge University Press, 1996.

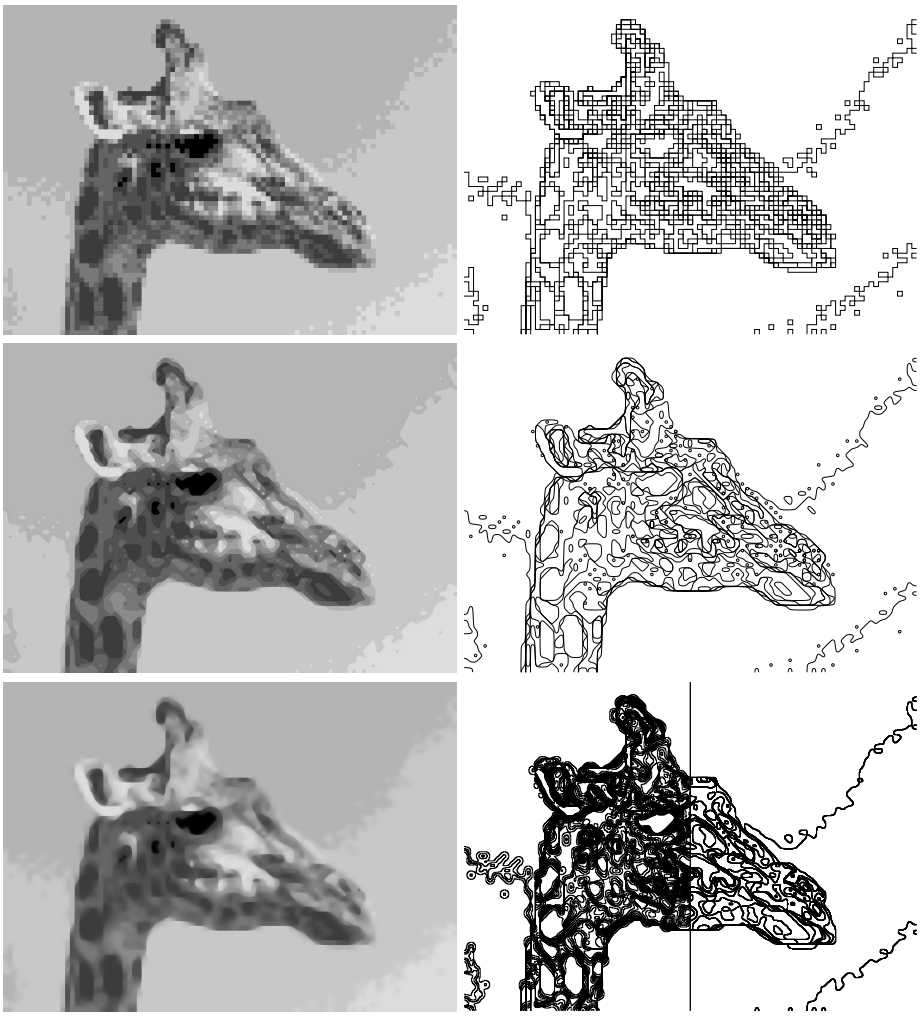

Fig. 3. AMSS of giraffe, original top left, level lines on the right.

Fig. 4. Affine transform of an ellipse image.

Fig. 5. The Lena image superimposed with its smoothed level lines.

Fig. 6. Original satellite image.

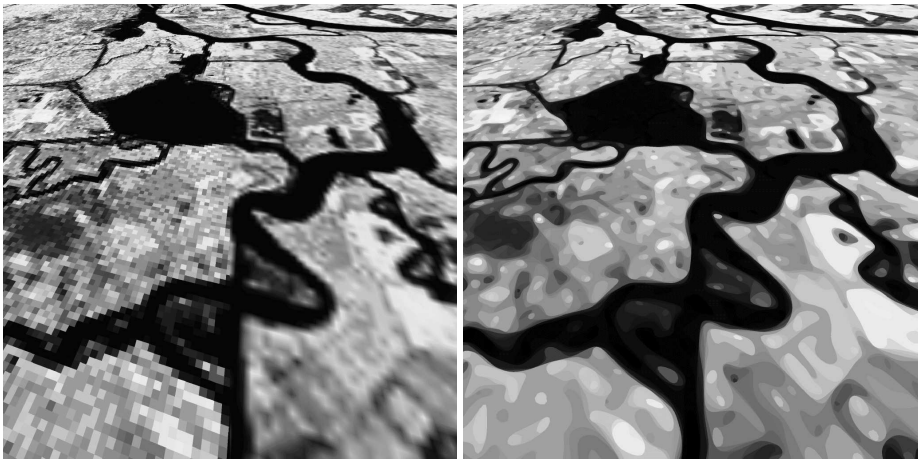

Fig. 7. Projective view of satellite image.