Visualizing level lines and curvature in images

Pascal Monasse[1]
(joint work with Adina Ciomaga[2] and Jean-Michel Morel[2])

[1] IMAGINE/LIGM, École des Ponts ParisTech/Univ. Paris Est, France
[2] CMLA, ENS de Cachan, France

IPAM, Computational Photography and Intelligent Cameras
Febuary 4-6, 2015

Plan

Introduction: What is an image?

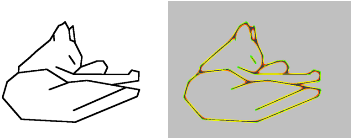
Computer scientist's point of view:
array of pixel values: 0=black, 255=white

58	61	67	58	52
77	56	62	54	62
71	66	63	44	53
78	60	37	55	62
82	86	51	31	70

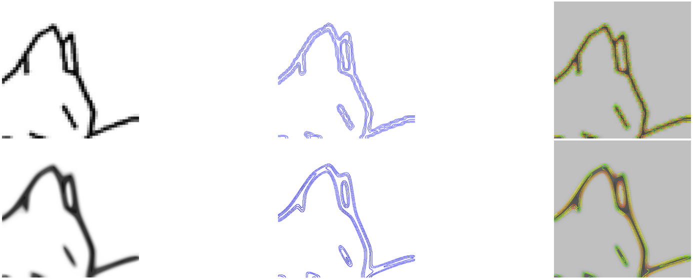
Mathematician's point of view:
$u : \mathbb{R}^2 \to \mathbb{R}$ (but only samples are known). Which regularity?

Introduction: description of a shape by its curvatures

Idea: a shape is well described by its extrema of curvature:

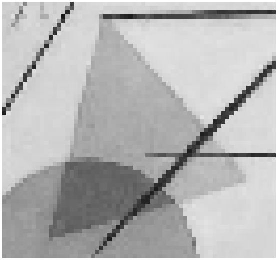


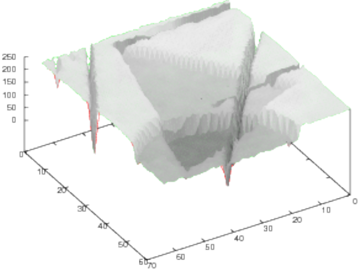
Question: how to compute these curvatures in the presence of pixelization?
Answer: We need to smooth the lines

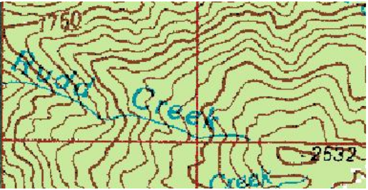


Introduction: images, contrast, level lines

Introduction: topographic map







Introduction: importance of invariance

- There is a necessity to smooth the image
- The curvature is a Euclidean invariant, so our smoothing process should not disturb that invariance
- Commutation of the smoothing process with a rotation is a nice property and will be satisfied
- Actually, we will get even stronger invariance: commutation with special affine transforms

Plan

Fundamental geometric flows in image processing

Theorem (Alvarez, Guichard, Lions, Morel 93)

Causal, local, Euclidean and contrast covariant scale-spaces are all governed by a curvature equation of type:

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

Two of these are the most interesting for image processing:

- the simplest one, the mean curvature motion:

$$\frac{\partial u}{\partial t} = |Du| \, \mathrm{curv}(u) \tag{MCM}$$

- the unique special affine covariant one, the affine curvature motion:

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

Curvatures

We can define the curvature in two different ways:

- If u is C^2 and $Du(x_0) \neq 0$, the scalar curvature at x_0 is
$$\mathrm{curv}(u)(x_0) = \frac{u_{xx} u_y^2 - 2 u_{xy} u_x u_y + u_{yy} u_x^2}{(u_x^2 + u_y^2)^{3/2}}(x_0) \tag{1}$$
- If $x(s)$ is a C^2 curve parameterized by length s ($|x'(s)| = 1$), the vector curvature at $x_0 = x(s_0)$ is
$$\kappa(x_0) = x''(s_0) \tag{2}$$

Link: denote by $x(s)$ the level line of u passing by x_0 ($u(x(s)) = u(x_0)$, $x(s_0) = x_0$), then

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

This suggests two ways to compute curvature: 2D differential operator (1) or curvature of level line (2).

Curve shortening

[Mackworth Mokhtarian 92]: The curvature motion $\frac{\partial x}{\partial t} = \kappa(x)$ (CS) can be implemented by applying the heat equation to each coordinate independently:

Theorem (Grayson 87)

If $x(s, 0)$ is a C^2 Jordan curve, then applying the intrinsic heat equation:

- *For $t > 0$, $x(s, t)$ is C^∞ and satisfies (CS).*
- *For $t > 0$, $x(., t)$ has a finite and non-increasing number of inflection points and curvature extrema.*
- *For $t \geq t_o$, $x(., t)$ is convex, and for $t \geq t_1$, $x(., t)$ is a point.*

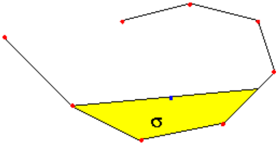
Algorithm: Input: polygon P. Output: smoothed polygon P. Iteratively:

- Sample P uniformly by length.
- Convolve coordinates of P by Gaussian kernel G_σ.

Affine shortening

Algorithm [Moisan 98]

- Break P into convex and concave parts
- Replace each part by the middle points of σ-chords originating from vertices of P.
- Concatenate the pieces of curves



Properties:

- Fast and simple algorithm (but numerically delicate)
- Special affine covariance (only inflection points, areas and middle points are involved)

Plan

Algorithms on sets

Algorithm [Koenderink and van Doorn 86]
Input: a closed subset X of $\mathbb{R}^N$

- Compute $u(x,t) = G_t \star \mathbf{1}_X(x)$
- Threshold at $1/2$: $X_t = [u(.,t) \geq 1/2]$.

Problem: fusion of shapes that are close.

An improvement is the threshold dynamic shape:
Algorithm [Merriman Bence Osher 92]
Iteratively:

- Convolve $\mathbf{1}_X$ with G_σ
- Threshold at $1/2$: $X \leftarrow [G_\sigma \star \mathbf{1}_X \geq 1/2]$

This is a Gaussian-weighted median filter applied to a binary image.

Median filters

Algorithm: Iterated median filter. Iteratively, for every point x:

- Gather points y in a discrete neighborhood of x
- Put at x the median value of the discrete neighborhood

$$\begin{array}{|c|c|c|}\hline 52 & 49 & 50 \\ \hline 47 & 20 & 53 \\ \hline 51 & 48 & 50 \\ \hline \end{array} \rightarrow 20,47,48,49,50,50,51,52,53 \rightarrow \begin{array}{|c|c|c|}\hline 52 & 49 & 50 \\ \hline 47 & 50 & 53 \\ \hline 51 & 48 & 50 \\ \hline \end{array}$$

Problem: the algorithm is blind to small curvatures, due to its discrete nature.

Theorem (Ishii 95)

Iterated weighted median filters on images converge to the MCM.

Finite difference schemes

- [Guichard Morel 97] uses the second derivative in the direction orthogonal to the gradient: $|Du|\,\mathrm{curv}(u) = u_{\xi\xi}$ with $\xi = Du^\perp/|Du|$

$$(u_{\xi\xi})_{ij} = \begin{pmatrix} \lambda_3 & \lambda_2 & \lambda_4 \\ \lambda_1 & -4\lambda_0 & \lambda_1 \\ \lambda_4 & \lambda_2 & \lambda_3 \end{pmatrix} (\theta) \star u_{i,j} \text{ with } Du = |Du| \begin{pmatrix} \cos\theta \\ \\ sin\theta \end{pmatrix}$$

  and all $\lambda_i = 1/2$ when $|D_u| < \epsilon$ (half Laplacian)

- [Crandall Lions 96] Explicit scheme

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

  with (e_i) basis of $\mathbb{R}^N$ and $a(p) = I - \frac{pp^T}{|p|^2+\epsilon}$

Stack filters

Finite difference schemes do not commute with contrast changes. To recover the contrast covariance, we can "stack" the results on level sets:

Algorithm [Stack filter]

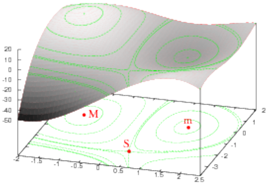
- Extract upper level sets $X_\lambda = [u \geq \lambda]$
- Apply FDS to $\mathbf{1}_{X_\lambda}$
- Set X'_λ by threshold at $1/2$ of the results
- Reconstruct by superposition: $u(x) \leftarrow \max\{\lambda : x \in X'_\lambda\}$

Anyway, other problems of FDSs persist, including lack of:

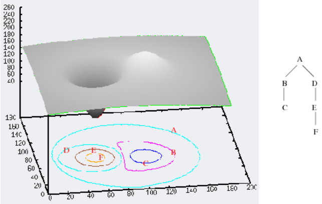
- Monotonicity, we always have slightly oscillatory solutions
- Euclidean (or affine) covariance, since they are grid dependent

Plan

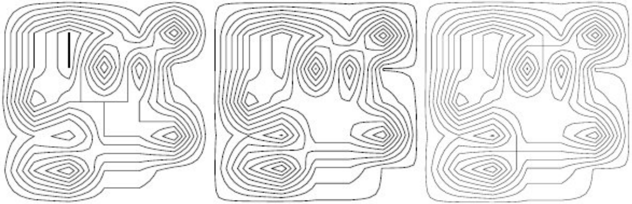
Inclusion tree



Morse functions and critical points



Tree of inclusion of Jordan curves



Level lines at different levels:

Left: pixels still visible. Center: Jordan curves. Right: saddle points

Bilinear interpolated images

- Convolution of a lattice of Dirac masses by separable triangle function

$$\varphi(x)\varphi(y) \text{ with } \varphi(x) = (1 - |x|)^+$$

- Between a square $ABCD$ of data values, we have the equation

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

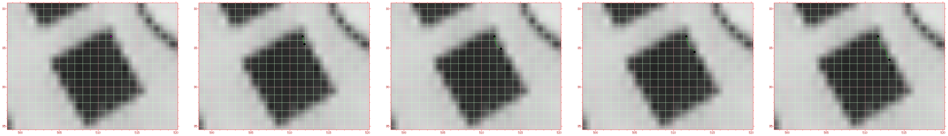
- Level line is intersection of the square with hyperbola

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

- If $\lambda_s \in [\min_{P=ABCD} u(P), \max_{P=ABCD} u(P)]$, we have a saddle point and the piece of level line at λ_s is two orthogonal segments

Tree extraction algorithm

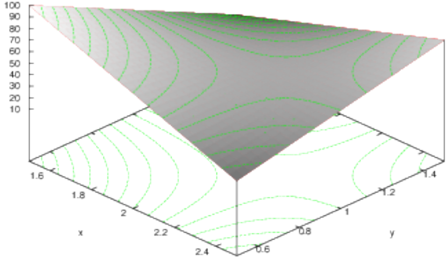
- Choose a number of levels, avoiding:
  1. Initial values
  2. Saddle values
- Two steps: follow level lines, then recover inclusion structure
- Step 1: Follow the level line inside square: given entry point, find exit point and sample the curve in between

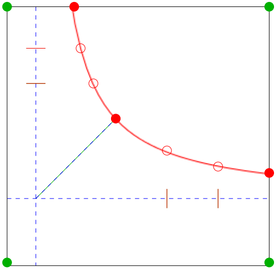


- Store intersections with horizontal edges
- Step 2: Order intersection points at each horizontal line
- Set inclusion by parity argument:
  1. Odd: get inside
  2. Even: go outside

Sampling in a dual pixel

- We can write
$$f(x,y) = a(x - x_S)(y - y_S) + \lambda_S \quad (a = u_{00} + u_{11} - u_{01} - u_{10}).$$
When $a \neq 0$, level lines are equilateral hyperbola branches.
- Maximum curvature point is at $|x - x_S| = |y - y_S|$, we add it if inside the dual pixel.
- We can sample the hyperbola branch by writing $y(x)$ (if $|y'| \leq 1$) or $x(y)$ (if $|x'| \leq 1$) and sampling uniformly along x or y.



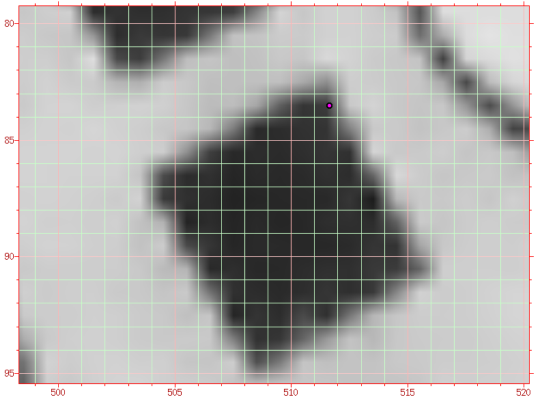


Following the level line

- Between two adjacent pixels, linear variation, thus a single point at level λ
- We follow the level line from dual pixel to dual pixel.

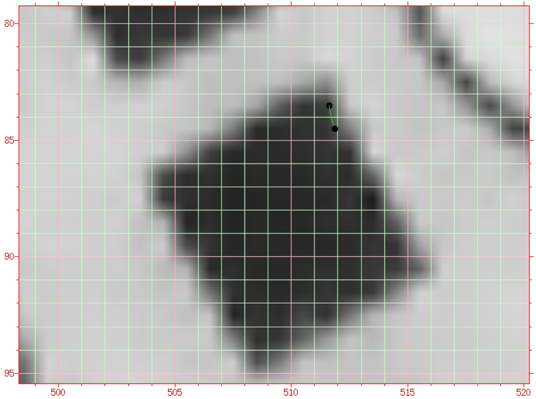
Following the level line

- Between two adjacent pixels, linear variation, thus a single point at level λ
- We follow the level line from dual pixel to dual pixel.



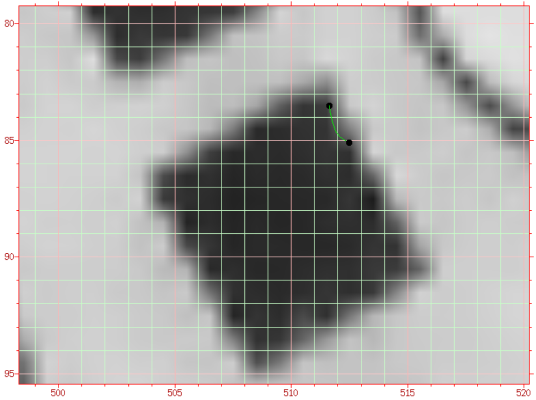
Following the level line

- Between two adjacent pixels, linear variation, thus a single point at level λ
- We follow the level line from dual pixel to dual pixel.



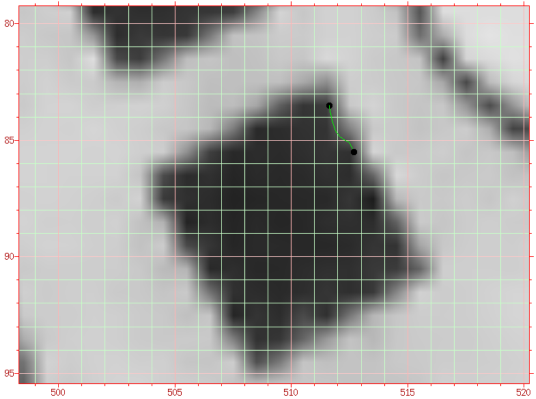
Following the level line

- Between two adjacent pixels, linear variation, thus a single point at level λ
- We follow the level line from dual pixel to dual pixel.



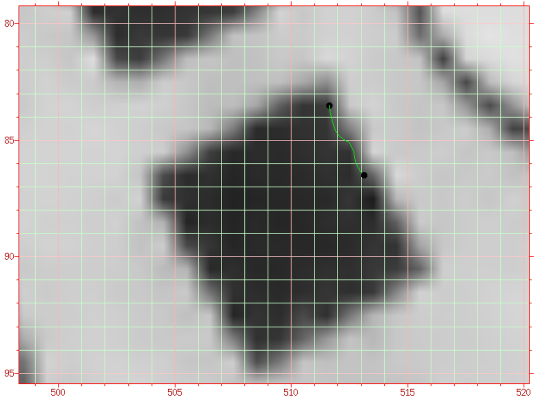
Following the level line

- Between two adjacent pixels, linear variation, thus a single point at level λ
- We follow the level line from dual pixel to dual pixel.



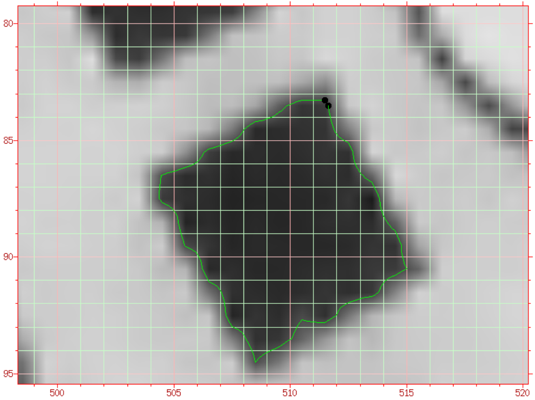
Following the level line

- Between two adjacent pixels, linear variation, thus a single point at level λ
- We follow the level line from dual pixel to dual pixel.



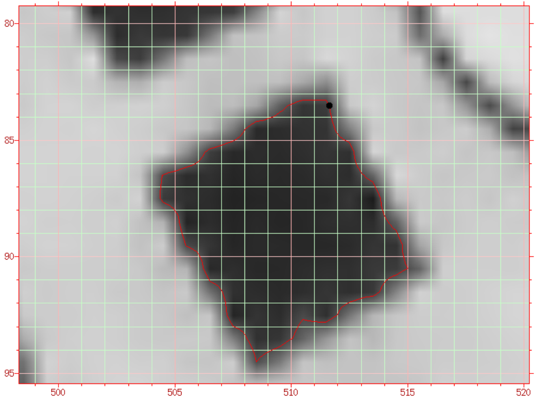
Following the level line

- Between two adjacent pixels, linear variation, thus a single point at level λ
- We follow the level line from dual pixel to dual pixel.



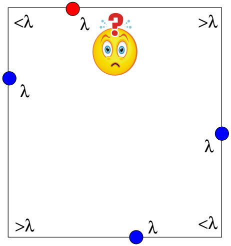
Following the level line

- Between two adjacent pixels, linear variation, thus a single point at level λ
- We follow the level line from dual pixel to dual pixel.



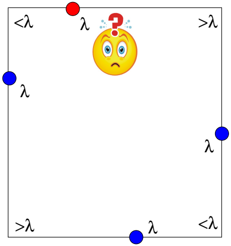
Particular cases

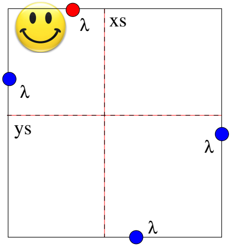
- Saddle point inside the dual pixel



Particular cases

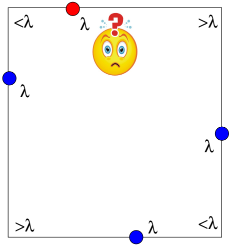
- Saddle point inside the dual pixel

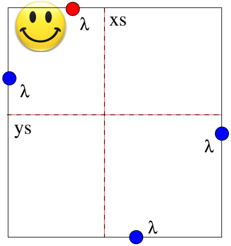




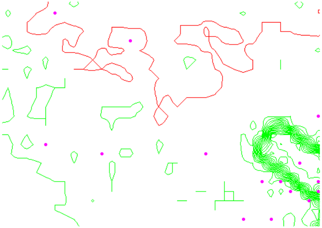
Particular cases

- Saddle point inside the dual pixel

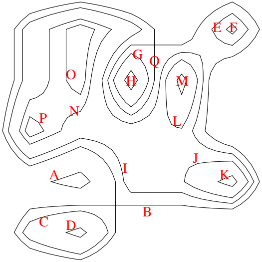


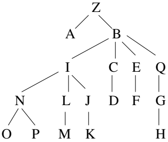


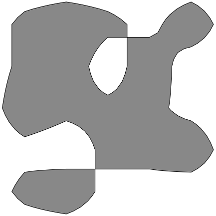
- Initial image level

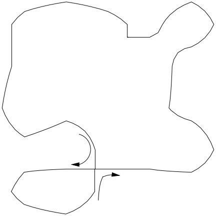


Defect at saddle point









B goes through two saddle points: by a local decision, it is not possible to treat both according to formal definition.

Reconstruction from the tree

- Walk of the tree in preorder, and for each node representing a level line do (noting λ its level):
- Two steps: intersection with horizontal lines, then filling
- Step 1: similar to step 2 of extraction, intersection with each horizontal line and ordering inside each horizontal line, $(x_1^i \dots x_{2N_i}^i)$
- Step 2: for all i and all $k = 0, \dots, N_i - 1$ do
- for all $j \in \mathbb{N} \cap [x_{2k+1}^i, x_{2k+2}^i]$:

$$\text{pixel } (j, i) \leftarrow \lambda$$

Geometric scheme curvature motion

Proceed in 3 steps:

1. Extract tree of bilinear level lines
2. Let each level line evolve by curve shortening or affine shortening
3. Reconstruct image from the tree of (shortened) level lines

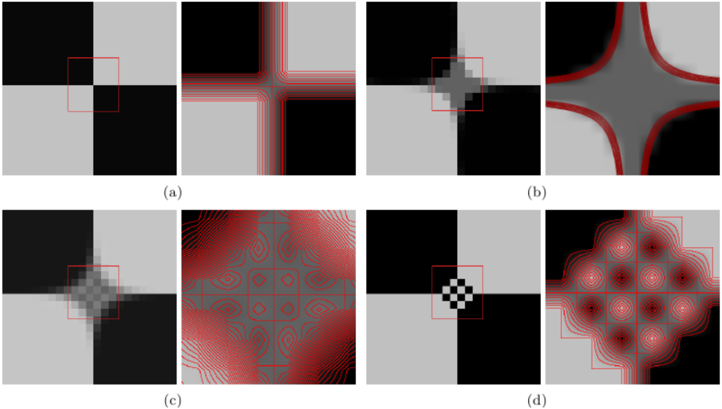
$$
\begin{array}{ccc}
u_0(\cdot) & \xrightarrow{\quad level\ lines\ extraction \quad} & \{\Sigma_0^{\lambda,i}\}_{\lambda,i} \\
MCM/ACM \Big\downarrow & & \Big\downarrow CS/AS \\
u(\cdot,t) & \xleftarrow{\quad reconstruction \quad} & \{\Sigma_t^{\lambda,i}\}_{\lambda,i}
\end{array}
$$

Curvature map

- Each level line is represented by a polygon.
- For each vertex P_i of the polygon, compute curvature as the inverse of the radius of the circumscribed circle of $P_{i-1}P_iP_{i+1}$
- Inside a pixel, average the curvatures of level lines passing through

Plan

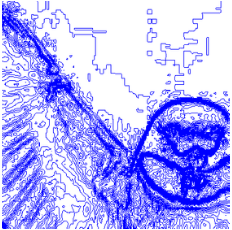
Fattening effect



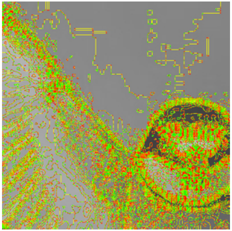
(a) Original and interpolation
(b) Level lines shortening
(c) Finite difference scheme
(d) Same with stack filter

JPEG artifacts

Original image

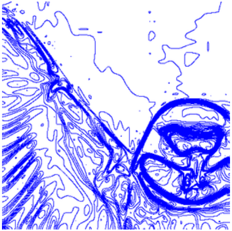


Original level lines

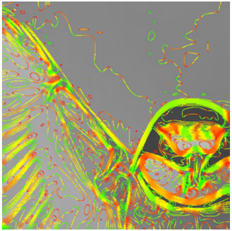


Curvature map

LLS image

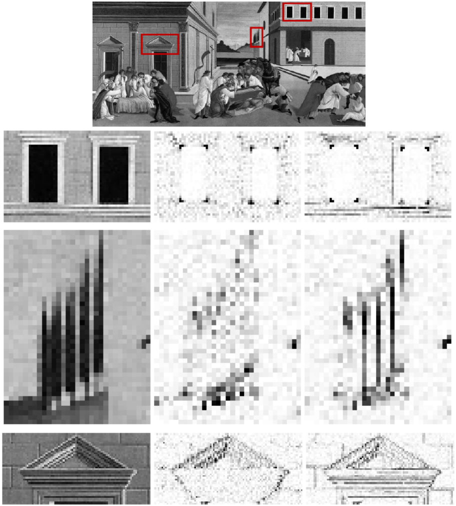


Shortened LL



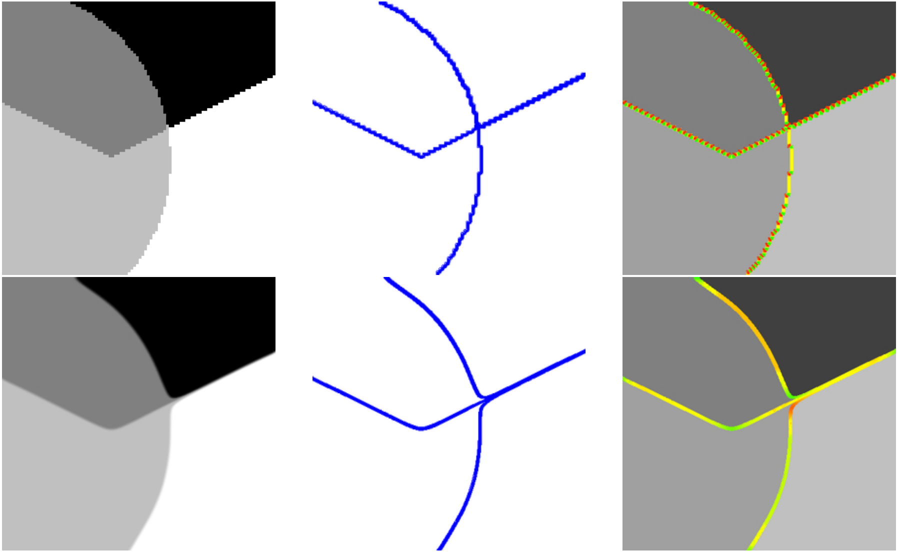
After filtering

Accurate mean curvature evolution

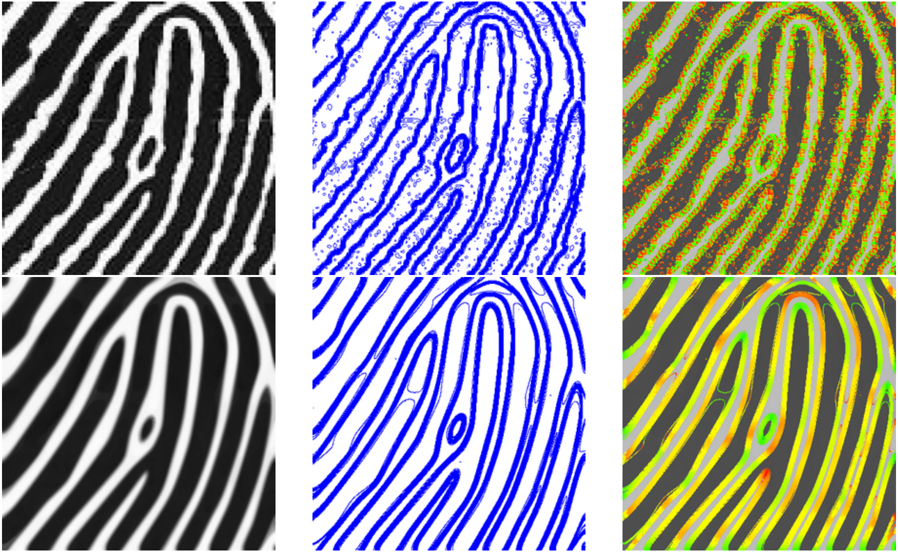


Left: detail of painting. Center: diff with LLS. Right: diff with stack FDS

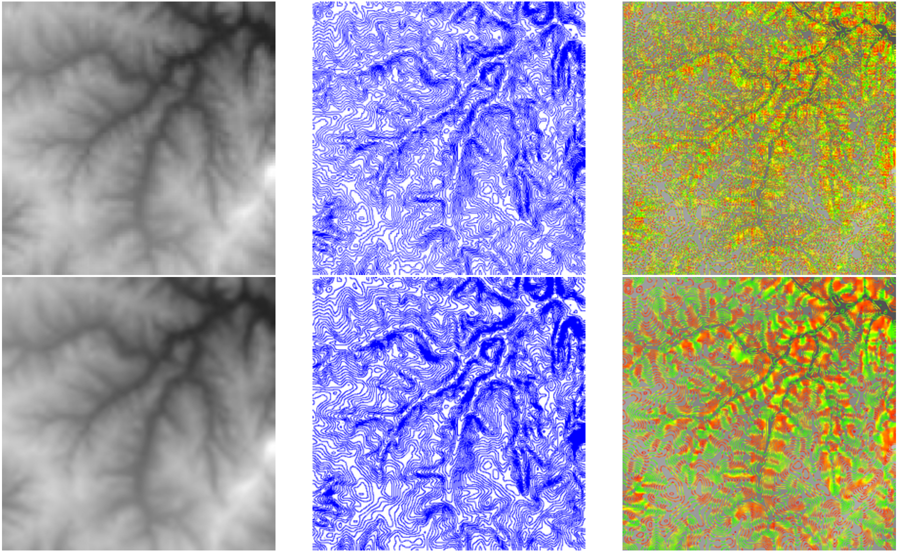
Junctions



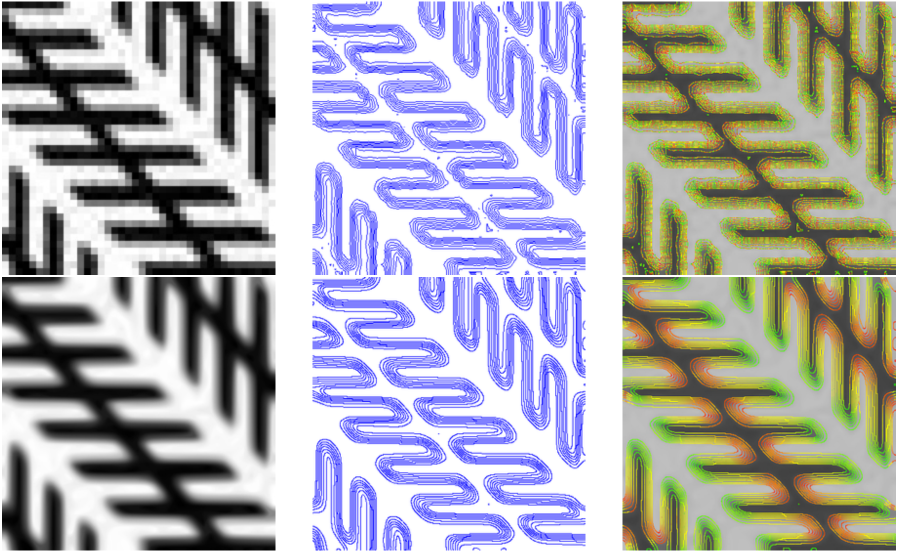
Curvature microscope



Topographic map of a digital elevation model



Textures



Conclusion

- The best scheme for motion by curvature of images proceeds in 3 steps:
  1. Decompose the image into its level lines
  2. Smooth each line independently
  3. Reconstruct from shortened level lines
- Such a scheme satisfies the covariance requirements (geometric and contrast)
- This is necessary to estimate reliably the curvature and avoid pixel artifacts
- This can be used as a microscope on the image: look at fine structures at any scale

Further reading

- On inclusion tree and applications: Vicent Caselles and PM, *Geometric description of images as topographic maps* (Springer Lecture Notes in Mathematics) 2010
- FDSs in IPOL:
  `http://www.ipol.im/pub/algo/cm_fds_mcm_amss/`
- `http://dev.ipol.im/~monasse/ipol_demo/cmmm_image_curvature_microscope/`