In R. P. Barneva, V. E. Brimkov, G. Nordo (eds.), Combinatorial Image Analysis (IWCIA 2022), Lecture Notes in Computer Science, vol. 13348, pp. 270–287, Springer, Cham, 2023. — The final publication is available at `link.springer.com`.

Curvature-Based Denoising of Vector-Valued Images

Christian Gapp and Martin Welk

UMIT TIROL – Private University for Health Sciences, Medical Informatics and Technology, Eduard-Wallnöfer-Zentrum 1, 6060 Hall in Tirol, Austria
`christian.gapp@umit-tirol.at, martin.welk@umit-tirol.at`
ORCID ID: 0000-0002-4520-298X (CG), 0000-0002-6268-7050 (MW)

Abstract. Salient visual information in images is often concentrated on contours or on regions where edges or curves change their direction abruptly. It is therefore of utmost importance in the processing of images to preserve this kind of information. Recently, a curvature-based denoising method has been proposed which first transforms an image into a level-line tree, then smoothes the level lines, and finally reassembles the image from those. Curvature information generated in this approach has also potential for further applications in image analysis.
Focusing on denoising, we transfer curvature-based smoothing to vector-valued images. We replace level lines by pseudo-level lines (integral curves of the vector field of directions of least vectorial contrast) and design a robust algorithm for their extraction from a vector-valued image. In this context we also propose a modification of the level line extraction from grey-scale images for better rotational invariance. Since intensities along pseudo-level lines are not constant, our method stores this information along the pseudo-level lines, and performs an appropriate smoothing on intensities. Finally we adapt the reconstruction process.
We present experiments on grey-scale and colour images to validate our proposed modification of the original grey-scale method as well as our new vector-valued curvature-based denoising method.

Keywords: Denoising · Curvature · Affine morphological scale space · Pseudo-level lines

1 Introduction

Due to the ubiquity of noise of various sources across image formation processes, denoising continues to be a fundamental task of image processing. The purpose of denoising is to remove noise while at the same time preserving as much as possible the image features needed for further processing of images by humans or computers. Telling apart noise from the relevant features is challenging, such that denoising methods inevitably interfere with image features along with removing noise. Together with the great variability in both noise sources and features that need to be preserved depending on application context, this constitutes a major

Fig. 1: Morphological Cat, constructed by taking 38 points of maximum curvature from the level lines, that represent the contours, and then connecting them with straight lines. From [4].

reason why even after decades of research there is no universal denoising method that suits all kinds of applications. For example, methods that directly smooth the source image by minimising an energy functional struggle with preserving the contrast and sharpness of contours.

A long-standing observation [4] is that salient information in images, especially for human observers, is concentrated along contours as well as feature points like angles or curvature extrema, compare the example in Fig. 1. This has inspired researchers to design denoising methods that specifically focus on this kind of features.

In [5] it is suggested to denoise images by extracting first the curvature image, similar to Fig. 1, then denoising the curvature image, and obtaining the final denoised image by reconstruction from the modified curvature image. It turns out that curvature images are less affected by additive noise n, leading to a better separation of salient image information and noise in the process.

Despite promising results, the method from [5] has so far only been studied for grey-scale images. Our aim in this paper is to extend the approach to vector-valued, such as (RGB) colour images. To this end, several obstacles need to be overcome.

First, the concept of level sets as such is suitable for grey-scale images only, and needs to be replaced by a suitable generalisation in the case of vector-valued images. To this end, we resort to the concept from [6] in which lines of minimal colour/vector contrast are proposed as level lines; for clarity, we will denote these as *pseudo-level lines*. Adapting the level-line extraction procedure from [19] to pseudo-level lines is the first component of our proposed method.

Second, as image intensities along pseudo-level lines are not constant as they are along level lines in grey-scale images, richer intensity information must accompany an extracted pseudo-level line. In the smoothing step, it is therefore necessary to not just smooth the pseudo-level line curves (which can be done essentially by the same affine morphological scale space as for grey-scale images) but also to take care of the intensity information. This is the second component of our proposed method.

Third, we need to provide a way to (approximately) reconstruct the original image from pseudo-level line information, which again is more complex than in the case of grey-scale images.

Our contributions. Our main contribution is the extension of the curvature-based smoothing algorithm from grey-scale to vector-valued images, which relies on the concept of pseudo-level lines as a replacement for level lines. We spell out the necessary adaptations of, and additions to the algorithm step by step. Moreover, we introduce a modified choice for pixel neighbourhoods for the sake of reducing directional bias, which can also be used beneficially in the base algorithm for grey-scale images.

Structure of the paper. In Section 2 we recall the curvature-based denoising method for grey-scale images from literature, and introduce our modification of pixel neighbourhoods. Our extension of the method to vector-valued images is developed in Section 3. Section 4 is devoted to the experimental demonstration of the techniques. A summary and outlook in Section 5 conclude the paper.

2 Curvature-Based Denoising of Scalar-Valued Images

Let us recall first the curvature-based denoising method for grey-scale images. We largely follow [7] but introduce a small modification of the pixel neighbourhoods that helps to avoid directional bias.

2.1 Level Line Tree

Level lines in a (space-continuous) grey-scale image are lines of constant intensity. They are closed curves (where curves ending in the image boundary can be closed by suitable boundary segments), and different level lines cannot cross each other; of two level lines, either one encircles the other, or both lie apart. Any finite set of level lines is therefore naturally organised in a Level Line Tree (LLTree), with the image boundary as its root. This intuition also carries over to discrete images, with the only caveat that segments of different discrete level lines can coincide, but still a strict tree-order is established by inclusion and exclusion.

A level line is a closed list of edge elements (edgels). An edge element (edgel) is given by a pair of neighbouring pixels, with the understanding that the space-continuous curve represented by the level line passes between these pixels. One of the pixels making up an edgel, the immediate interior pixel (IIP), is inside the private region (pregion) of the level line. The other one represents the immediate exterior pixel (IEP) outside this region. The sequences of immediate interior pixels (IIPs) and immediate exterior pixels (IEPs) in the list of edgels progress from pixels to neighbouring pixels; repetitions (i.e. subsequent edgels sharing their IIPs or IEPs) are allowed. Closedness of the level line means that the first and last entry of the list of edgels are identical. The pregion is represented by a list of pixels. As for grey-scale images a private value (pvalue) can easily be defined as either being the maximum or minimum intensity of all IIPs along the level line, the pregion contains all pixels inside the region with intensity equal to the pvalue. Considering the connectedness graph of an image, i.e. the graph whose vertices are the pixels, and edges connect exactly those pixels which are

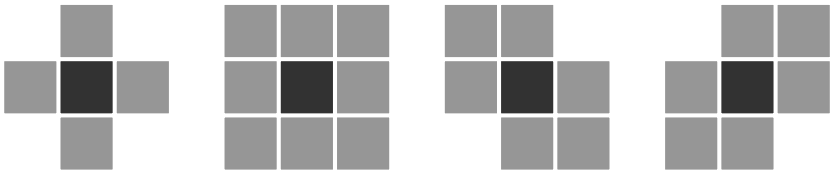


Fig. 2: Neighbourhood types as proposed in [19]; dark centre pixels are shown surrounded by neighbours in grey. (a) 4-neighbourhood, (b) 8-neighbourhood, (c) 6-neighbourhood of type 1, (d) 6-neighbourhood of type 2.

neighbours (thus, edges represent edgels), one sees that a level line represents a cut of this graph. The pregion of the level line corresponds to one of the connected components into which the connectedness graph is split by the cut.

Defining a curve as a list of pixels, a level line is associated with two curves: the curve-immediate-interior-pixel (curveIIP) that contains all IIPs, and the curve-immediate-exterior-pixel (curveIEP) of all IEPs.

To establish the LLTree, each line is stored in a node together with its pvalue and pregion, and an unordered list of references to their children (child nodes). The LLTree contains all these nodes from the uppermost parent node down to the bottom child node. Here the first node, the uppermost parent, always has a level line that expresses the border of an image. A child of the parent describes a region inside the parent's region. Two or more children are called siblings. They are in a common list of references and on the same level in the LLTree.

Note that a parent and a child node can have common IIPs and IEPs. Two siblings can only have common IEPs.

Neighbourhoods. The set of edgels available in the extraction process depends on a choice of neighbourhoods, see Fig. 2. Admissible edgels are always the pairs (p, q_i) of a centre pixel p as IIP and one of its n neighbours q_i $(i = 0, \ldots, n-1$ where n is 4, 8 or 6) as IEP. Each of these choices, however, comes with a downside. Using 8-neighbourhoods, Fig. 2b, on the pixel set of the given image implies a non-planar connectedness graph due to the intersection of diagonal edgels. However, representing level lines, thus cuts of the connectedness graph, by *sequences* of edgels, actually relies on the assumption that the connectedness graph is planar.

For the further discussion we remark that the level line extraction process will be designed in a way that it proceeds from edgel to edgel via the meshes of the connectedness graph. Whenever a level line enters one mesh of the connectedness graph via an edgel, one has to determine by which edgel it leaves the mesh. This will be particularly easy if the meshes of the connectedness graph are triangles: In this case, the exit edge is always adjacent to the entrance edge of each mesh. It only takes to choose between these to adjacent edges by keeping fixed either

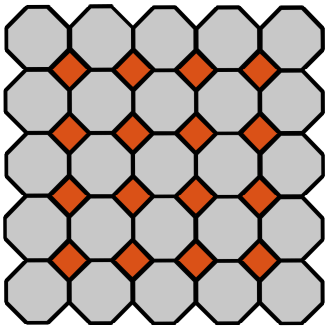


Fig. 3: Image region Ω containing 5×5 original pixels (grey), extended by dummy pixels (orange). Note that original pixels have 8-connectedness, whereas dummy pixels have 4-connectedness.

the IIP or the IEP. Meshes circumscribed by more than three edges need more complicated case distinctions.

Returning to neighbourhood choices, the 4-neighbourhood, Fig. 2a, leads to a connectedness graph with quadrilateral meshes, which is therefore unfavourable for the level line extraction process. For this reason, the 6-neighbourhoods from Fig. 2c and Fig. 2d have been proposed; they yield planar graphs with triangular cells that are a perfect fit for the extraction algorithm. Unfortunately, this comes at the cost of sacrificing symmetry by preferring one diagonal direction over the other. Thereby they introduce a directional bias which is generally unfavourable in image processing; indeed, it leads to visible artifacts in the smoothed images, cf. Fig. 12 in Section 4 where diagonal streaks in the direction of the preferred diagonals are clearly visible.

We therefore favour an alternative approach. We insert dummy pixels located at the common corners of four adjacent pixels of the original image grid, see Fig. 3. By bilinear interpolation, each dummy pixel is assigned the average of the intensities of the four surrounding original pixels as its intensity value. The neighbourhood relation within this extended set of pixels, and thereby the connectedness graph, is defined as follows. Each original pixel has eight neighbours: the four original pixels which are located next to it in vertical and horizontal direction, and the four dummy pixels next to it. In contrast, each dummy pixel has only the four original pixels next to it as neighbours. In Fig. 3 this is visualised by showing original pixels as octagons but dummy pixels as squares. Pixels are considered neighbours if and only if they have a common border in this representation. With this definition, the connectedness graph is planar and consists entirely of triangular meshes. Each mesh is made up by two original pixels and one dummy pixel. Thus, the graph meets the needs of the extraction algorithm, while retaining all symmetries of the regular pixel grid.

Level line extraction. A new level line always begins with a start edgel α. Let the actual edgel be $\mathrm{e} = (p, q)$. Then the $\mathit{NextPixel}(p, q)$ operation returns the pixel r that either is the next IIP or IEP, see Fig. 4.

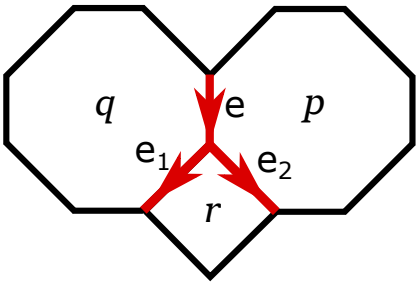


Fig. 4: Two choices for the next edgel: e_1 or e_2. Edgel $e = (p, q)$ is already in the boundary. In case r is inside the region, the next edgel is $e_1 = (r, q)$. Otherwise r becomes IEP and thus the next edgel is $e_2 = (p, r)$.

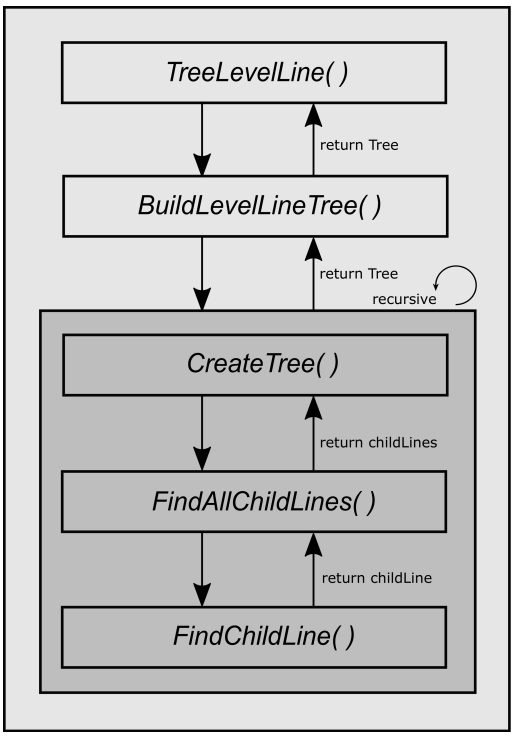


Fig. 5: Routine *TreeLevelLine* for scalar valued images.

As soon as the start edgel α is reached again, the level line is closed, and further pixels are investigated. These are either pushed to the pregion or represent IIPs of a child's start edgel, meaning a child line is passing through.

The overall algorithm to build the LLTree is visualised in Fig. 5. In the recursive part (*CreateTree*), all children are successively added to the parent currently processed within the subroutine *FindAllChildLines*.

Within *FindChildLine*, edgels are added successively until the start edgel is reached again. The edgel (p, q) is either followed by (r, q) or (p, r), depending on whether r (returned from *NextPixel*(p, q)) is declared to be an IIP or IEP. In grey-scale images this decision is made using the pvalue as threshold. Let us assume the pvalue v of the current level line is less than its parent's pvalue. Then

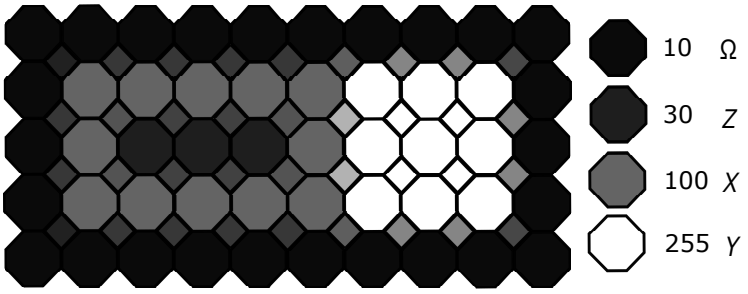


Fig. 6: Example grey-scale image (10×5) with dummy pixels inserted. Ω, X, Y, Z are the four regions with identical intensities each.

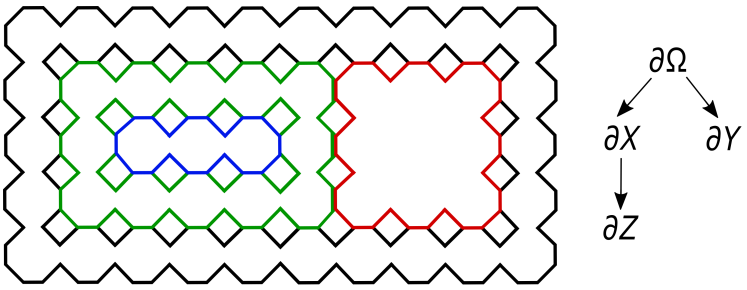


Fig. 7: Level lines in Fig. 6. $\partial\Omega$ (black) is the parent of ∂X (green) and ∂Y (red). ∂Z (blue) is a child of ∂X. ∂X and ∂Y are siblings.

pixels with intensity lower than v are IIPs, and the others IEPs. In the case v is higher than the parent's pvalue, only pixels with intensity greater than v are IIPs.

2.2 Level Line Shortening

Discrete curvature of level lines. Let $\boldsymbol{\Gamma} = \{\boldsymbol{x}(s) : s \in [0, L], \|\boldsymbol{x}'(s)\| = 1\}$ be a sufficiently smooth (C^2) curve in arc-length parametrisation. The second derivative $\boldsymbol{x}''(s)$ then always points in normal direction, i.e. [12]

$$\boldsymbol{x}''(s) = \kappa(s)\,\boldsymbol{n}(s) \tag{1}$$

with some function $\kappa(s)$ which is called *curvature* of $\boldsymbol{\Gamma}$, and $\boldsymbol{n}(s)$ denoting the unit normal vector $\boldsymbol{n}(s) \perp \boldsymbol{x}'(s)$. Assuming that the moving frame $(\boldsymbol{x}'(s), \boldsymbol{n}(s))$ is positively oriented, $\kappa(s) > 0$ indicates that the curve is locally bent in mathematically positive sense whereas $\kappa(s) < 0$ indicates it turns in the mathematically negative sense. The definition of κ via arc-length parametrisation is transferred to curves in arbitrary parametrisation by reparametrisation, making $\kappa(s) \equiv \kappa(\boldsymbol{x})$ in fact dependent only on the shape but not the parametrisation of the curve.

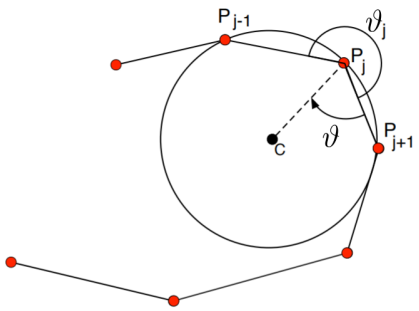


Fig. 8: Definitions and discretisation of the level line to display the computation of the discrete curvature at the vertex $\overline{P_{j-1}P_jP_{j+1}}$, from [7]. The curvature is $\kappa(P_j) = 1/r_j$, with $r_j = \|P_jC\|$.

In fact, for $\kappa(\boldsymbol{x}) \neq 0$, $1/|\kappa(\boldsymbol{x})|$ is the radius of a best-fit (osculating) circle to $\boldsymbol{\Gamma}$ at $\boldsymbol{x}$.

For any non-singular point $\boldsymbol{x}$ of a sufficiently smooth (C^2) image $u : \mathbb{R}^2 \to \mathbb{R}$, the curvature $\kappa(\boldsymbol{x})$ of the level line of u passing through $\boldsymbol{x}$ can be computed as [7,17]

$$\kappa(\boldsymbol{x}) = \frac{u_{xx}u_y^2 - 2u_{xy}u_xu_y + u_{yy}u_x^2}{(u_x^2 + u_y^2)^{3/2}}(\boldsymbol{x}) \; . \tag{2}$$

Regarding the sign of κ, level lines are understood here to be oriented such that the normalised local image gradient vector $\nabla u/\|\nabla u\|$ points to the right of the level line, i.e. $\nabla u/\|\nabla u\| = -\boldsymbol{n}$ locally.

As pointed out by Mondelli and Ciomaga [16], direct implementation of (2) by finite difference scheme (FDS) models as done in [2] and [9] suffers from numerous artifacts. Therefore it is preferable to compute instead the curvature directly on the level lines.

To this end, let $\boldsymbol{\Gamma}$ be a closed discrete curve denoted as $\boldsymbol{\Gamma} = \{P_j(x_j, y_j)\}$, with $j \in \{0, \ldots, N\}$ and $P_0 = P_N$. We assume that $\boldsymbol{\Gamma}$ approximates a (space-continuous) level line; in fact, we will use for $\boldsymbol{\Gamma}$ the curveIIP of a discrete level line. Building on the relation between curvature and osculating circles, the curvature $\kappa(P_j)$ of the discrete curve at P_j can be defined as

$$\kappa(P_j) := \pm 1/r_j \; , \tag{3}$$

with the radius $r_j = \|P_jC\|$ computed with the three points P_{j-1}, P_j, P_{j+1} (see also Fig. 8) and the sign consistent with (1), compare Fig. 8.

As noted in [7], the discrete curvature at point P_j is

$$\kappa(P_j) = \frac{-2\,\sin(\vartheta_j)}{\|P_{j-1}P_{j+1}\|} = \frac{-2\,\det\,(P_jP_{j-1} \quad P_jP_{j+1})}{\|P_{j-1}P_j\|\,\|P_jP_{j+1}\|\,\|P_{j-1}P_{j+1}\|} \; , \tag{4}$$

with

$$\det\,(P_jP_{j-1} \quad P_jP_{j+1}) := \det \begin{pmatrix} x_{j-1} - x_j & x_{j+1} - x_j \\ y_{j-1} - y_j & y_{j+1} - y_j \end{pmatrix} \; . \tag{5}$$

AMSS – Affine morphological scale space. Affine morphological scale space is a curvature-driven process that preserves invariance properties such as monotonicity, morphology and affine invariance [1,15].

The affine scale space can be interpreted as an intrinsic heat equation [13]. Let $\sigma \mapsto \boldsymbol{x}(t,\sigma)$ be a Jordan arc (or curve) for each scale t. Then, in any neighbourhood without an inflection point, the affine scale space

$$\frac{\partial \boldsymbol{x}}{\partial t} = \kappa(\boldsymbol{x})^{1/3}\, \boldsymbol{n}(\boldsymbol{x}) \tag{6}$$

is equivalent to the intrinsic heat equation $\partial \boldsymbol{x}/\partial t = \partial^2 \boldsymbol{x}/\partial \sigma^2$ with parametrisation σ (affine length) [13,18].

To implement (6) the geometric scheme proposed by Moisan [15] is used. The latter equation can be interpreted as an alternating filter, switching between affine erosion and dilation in dependence of the scale space parameter σ. This can be realised working with affine erosion on the individual convex and concave parts of the discrete curve $\boldsymbol{\Gamma}$. Therefore first the inflection points $E_i = P_{j(i)}$ must be detected. After the resampling process, where points are added/removed to get good smoothing results, each convex component

$$C_i = (E_i = P_{j(i)}, P_{j(i)+1}, P_{j(i)+2}, \ldots, P_{j(i+1)} = E_{i+1}) \tag{7}$$

is processed by affine erosion resulting in an envelope of σ-chords

$$C_i^\sigma = (E_i, P^\sigma_{j(i)}, P^\sigma_{j(i)+1}, P^\sigma_{j(i)+2}, \ldots, P^\sigma_{j(i+1)}, E_{i+1})\ , \tag{8}$$

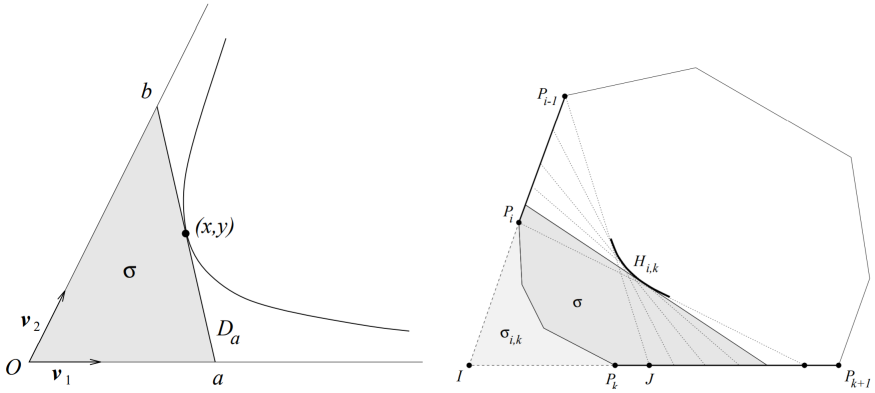
a set of middle points of σ-chords with unchanged inflection points [7]. This is important here because the convex and concave parts are glued together after each iteration. A σ-chord C_i^σ is defined as a segment connecting two points of the (discrete) curve that cuts off a (polygonal) area of size σ between itself and the curve (see Fig. 9).

The larger σ, the less the accuracy of the geometric scheme. Hence the affine shortening process is iterated with a small σ as often as needed to achieve the desired smoothness. The smoothing process can be described with Algorithm 1.

Algorithm 1 *Smooth Curves*

```
1: for all curves c ∈ LLTree do
2:     while (desired scale t not reached) do
3:         split c into convex and concave parts
4:         resample c
5:         affineErosion(c)
6:     resample c
```

For all curves ∈ LLTree the process of splitting, resampling and affine erosion is iterated until the desired scale t is reached (lines 2–5). As a last step the new curve (σ-chord) is resampled again.



(a) (b)

Fig. 9: Affine erosion of a corner, from [15]. (a) The affine erosion of a corner results in a hyperbola. σ displays the area cut after several iterations. (b) Evolution of a hyperbola ($H_{i,k}$) resulting from two edges. $\sigma_{i,k}$ is the area cut after a couple of iterations, whereas σ (note that σ includes $\sigma_{i,k}$) analogously to (a) displays the area trimmed after many iterations.

2.3 Reconstruction

For scalar-valued images the regions can be filled straightforward by iterating the LLTree from top to bottom and printing the inside of the regions with the level lines' pvalue. Herein the children's pregion overprint their parents'. As siblings do not affect each other, it is irrelevant which region is filled first on the same level of the LLTree.

3 Curvature-Based Denoising of Vector-Valued Images

This section is devoted to adapting the denoising method recalled in the previous section to vector-valued images.

3.1 Pseudo-Level Lines

Unlike grey-scale images, vector-valued images are not filled by curves of constant intensity, i.e. level lines. As a surrogate for these, it is proposed in [6] to consider the integral curves of the directions of minimal vectorial change, also denoted as level lines there. For a clear distinction, we will use the term *pseudo-level lines* in the following. The first step towards computing pseudo-level lines is the computation of gradients of vector-valued images; following [6] this can be done using standard tools from Riemannian geometry [14].

For a (space-continuous) vector-valued image $\boldsymbol{u}(\boldsymbol{x}) : \mathbb{R}^2 \to \mathbb{R}^n$, at each location $\boldsymbol{x} = (x, y)^\mathsf{T}$ the directional derivative of the vector-valued image $\boldsymbol{u}$ at $\boldsymbol{x}$

in the direction of $\boldsymbol{v}$ then is the vector

$$\partial_{\boldsymbol{v}} \boldsymbol{u}(\boldsymbol{x}) = D\boldsymbol{u}(\boldsymbol{x})\boldsymbol{v} \tag{9}$$

where

$$D\boldsymbol{u}(\boldsymbol{x}) = \begin{pmatrix} \partial_x u_1(\boldsymbol{x}) & \partial_y u_1(\boldsymbol{x}) \\ \vdots & \vdots \\ \partial_x u_n(\boldsymbol{x}) & \partial_y u_n(\boldsymbol{x}) \end{pmatrix} \tag{10}$$

is the Jacobian matrix of $\boldsymbol{u}$ at $\boldsymbol{x}$. The norm $\|\partial_{\boldsymbol{v}} \boldsymbol{u}(\boldsymbol{x})\|$ yields the rate of change of the values of $\boldsymbol{u}$ in the direction of $\boldsymbol{v}$ and can be written as a positive semidefinite quadratic form of $\boldsymbol{v}$ by

$$\|\partial_{\boldsymbol{v}} \boldsymbol{u}(\boldsymbol{x})\|^2 = (D\boldsymbol{u}(\boldsymbol{x})\boldsymbol{v})^{\mathsf{T}} D\boldsymbol{u}(\boldsymbol{x})\boldsymbol{v} = \boldsymbol{v}^{\mathsf{T}} \big(D\boldsymbol{u}(\boldsymbol{x})^{\mathsf{T}} D\boldsymbol{u}(\boldsymbol{x})\big)\boldsymbol{v} \; . \tag{11}$$

As a result, in each non-singular point $\boldsymbol{x}$ of the image domain there are two mutually orthogonal directions $\boldsymbol{v}_{1,2}(\boldsymbol{x})$ in which the greatest and least rates of change, respectively, are found; $\boldsymbol{v}_{1,2}$ are the eigenvectors of $\boldsymbol{J}(\boldsymbol{x}) := D\boldsymbol{u}(\boldsymbol{x})^{\mathsf{T}} D\boldsymbol{u}(\boldsymbol{x})$. (The matrix $\boldsymbol{J}(\boldsymbol{x})$ is also known as *structure tensor*.) The vector field $\boldsymbol{v}_1(\boldsymbol{x})$ denoting the directions of fastest change can be understood as surrogate of a gradient vector field of $\boldsymbol{u}$, and the vector field $\boldsymbol{v}_2(\boldsymbol{x})$ of directions of slowest change as surrogate of a level-line direction field; thus the pseudo-level lines are the integral curves of the vector field $\boldsymbol{v}_2(\boldsymbol{x})$. Note that in the continuous case, under regularity conditions, pseudo-level lines are closed curves, see [3], [10], [11].

The following subsections 3.2–3.4 correspond to the three steps of our overall algorithm for curvature-based smoothing of vector-valued images.

3.2 First Step: Construction of the Pseudo-Level Line Tree

From here on we assume that the vector-valued image is an RGB colour image with the colour channels R, G, B. Unlike for grey-scale images, a unique pvalue can not be defined for vector-valued images. Nevertheless, we define the mean value of all IIPs as pvalue in order to push a pixel $\hat{p}$, that fulfills the criteria

$$\big(R_{\hat{p}} - R_{\mathrm{nIIP}}\big)^2 + \big(G_{\hat{p}} - G_{\mathrm{nIIP}}\big)^2 + \big(B_{\hat{p}} - B_{\mathrm{nIIP}}\big)^2 \;\; < \;\; s^2 \; , \tag{12}$$

$$\big(R_{\hat{p}} - R_{\mathrm{nIEP}}\big)^2 + \big(G_{\hat{p}} - G_{\mathrm{nIEP}}\big)^2 + \big(B_{\hat{p}} - B_{\mathrm{nIEP}}\big)^2 \;\; < \;\; s^2 \; , \tag{13}$$

where nIIP, nIEP represent the nearest IIP, IEP of the actual level line, and $s \in \mathbb{R}$ is a tolerance limit, to a pseudo-level line's pregion. Additionally, $\hat{p}$ must have a pixel $\hat{q_n}$ in its immediate neighbourhood with radius $r = 5$, that is already part of a pseudo-level line.

Pseudo-level line extraction. In order to build a discrete pseudo-level line for vector-valued images, we need strict criteria for a pixel r to either be the next IIP or IEP. To this end, we start from the 2×2 structure tensor

$$\boldsymbol{J} = \nabla R \, \nabla R^{\mathsf{T}} + \nabla G \, \nabla G^{\mathsf{T}} + \nabla B \, \nabla B^{\mathsf{T}} \tag{14}$$

where $\nabla c = (\partial_x c, \partial_y c)^\mathsf{T}$ for $c = R, G, B$ can be computed using central differences involving the four immediate neighbours of the dummy pixel. (In each step, only one of the three pixels p, q, r represents a dummy pixel, the others are integer.)

The spectral decomposition of the structure tensor $\boldsymbol{J} = \lambda_1 \boldsymbol{v}_1 \boldsymbol{v}_1^\mathsf{T} + \lambda_2 \boldsymbol{v}_2 \boldsymbol{v}_2^\mathsf{T}$ with the eigenvalues $\lambda_1 \geq \lambda_2 \geq 0$ and eigenvectors $\boldsymbol{v}_1 \perp \boldsymbol{v}_2$ yields the (pseudo-) gradient direction $\boldsymbol{v}_1$ and pseudo-level line direction $\boldsymbol{v}_2$. The projection matrix

$$\boldsymbol{Z} = \begin{pmatrix} \langle \nabla R,\, \boldsymbol{v}_1 \rangle \\ \langle \nabla G,\, \boldsymbol{v}_1 \rangle \\ \langle \nabla B,\, \boldsymbol{v}_1 \rangle \end{pmatrix}, \tag{15}$$

is a 3×1 matrix that projects a 3×1 Red-Green-Blue (RGB)-vector (the intensity of a pixel) onto the gradient in the colour space. Thus,

$$p(p) = p_p = \left\langle \boldsymbol{Z},\, \begin{pmatrix} R_p & G_p & B_p \end{pmatrix}^\mathsf{T} \right\rangle \tag{16}$$

is the projection of the pixel p onto this gradient; p_q and p_r are computed analogously. To give a certain criterion for r to be the next IIP or IEP, let $\hat{\alpha} \in [0, 1]$ be the division ratio that splits p_p and p_q into two parts, then

$$p_{\hat{\alpha}} = (1 - \hat{\alpha}) \cdot p_p + \hat{\alpha} \cdot p_q \ . \tag{17}$$

If $p_p > p_q$, the eigenvector $\boldsymbol{v_1}$ is replaced with $-\boldsymbol{v}_1$ such that $p_p \leq p_{\hat{\alpha}} \leq p_q$ is applicable. The pixel r can now certainly be chosen as next IIP if

$$p_r \leq p_{\hat{\alpha}} \ , \tag{18}$$

and as IEP otherwise.

For the first edgel after the start edgel, $\hat{\alpha}$ is initialised with 0.5. Every time a new edgel is added to the level line, $\hat{\alpha}$ is updated for the next step via

$$\hat{\alpha} = \frac{p_{\hat{\alpha}} - p_r}{p_q - p_r} \quad \text{if} \ \ r = \text{IIP} \ , \qquad \text{or} \qquad \hat{\alpha} = \frac{p_{\hat{\alpha}} - p_p}{p_r - p_p} \quad \text{if} \ \ r = \text{IEP} \ . \tag{19}$$

Crash handling. Although in the continuous domain pseudo-level lines are closed curves, the discrete algorithm described so far can fail to yield a closed level line. This is essentially due to accumulated errors in the estimation of colour gradient projection matrices $\boldsymbol{Z}$ in the course of the computation. This means that the sequence of pseudo-level line edgels might not return to the start edgel exactly. If this is the case, the pseudo-level line LL crsahed into itself and must be modified. Crashes can both happen from the inside and the outside. An inside crash occurs if some IIP $\in LL$ is picked as a new IEP. Crashes from the outside are detected if some IEP $\in LL$ is selected as new IIP. Another possibility is, that a whole edgel $\hat{\beta}$ already $\in LL$ is found again. If the IIP($\hat{\beta}$) is found first, LL crashed into itself from the inside. If IEP($\hat{\beta}$) is reached first, the crash occured from the outside.

In both cases, one edgel $\in LL$ needs to be changed: $(p,\, r) \leftrightarrow (r,\, q)$. Starting from this edgel the pseudo-level line is recomputed. In the case of an inside crash,

candidate edgels $i \in \{1, \dots, l-1\}$ – with $l = \text{length}(LL)$ – for a change are of the type

$$(p_i, r_i) \to (r_i, q_{i-1}) \,, \tag{20}$$

forcing LL to make a turn to the outside. In the other case, LL is forced to make a turn to the inside, meaning admissible edgels are of the type

$$(r_i, q_i) \to (p_{i-1}, r_i) \,. \tag{21}$$

In most cases, more than one edgel lends itself as a candidate. To implement a reliable rule to determine the best candidate, we assign alternative edgels as described in (20) and (21) with costs ψ that measure how expensive it is for LL to take the alternative direction. The more clearly the decision is for r to be IEP or IIP (18), the higher the costs for the edgel to take r *wrongly* as IIP (20) or IEP (21), respectively. Depending on the decision for r, we compute

$$\psi = \frac{p_{\hat{\alpha}} - p_r}{p_q - p_p} \quad \text{if} \;\; r = \text{IIP} \,, \qquad \text{or} \qquad \psi = \frac{p_r - p_{\hat{\alpha}}}{p_q - p_p} \quad \text{if} \;\; r = \text{IEP} \,. \tag{22}$$

When LL is approximated, from all potential candidates the one with the lowest costs is modified. Each level line modified has a handicap Ψ, initialised with the costs ψ_i of the first changed edgel e_i. Let the second crash occur from the same side with e_j with $i \neq j$, then Ψ is either set to ψ_j, if $j < i$, or ψ_j is added to Ψ ($\Psi = \psi_i + \psi_j$), if $j > i$. Note that Ψ is added to all costs $\psi_{i+1}, \dots, \psi_{l-1}$ of the edgels $\in$ alternative path $(\mathrm{e}_{i+1}, \dots, \mathrm{e}_{l-1})$ within each crash.

Intensity handling. For vector-valued images the intensities must be carried along the pseudo-level lines for each node in order to reconstruct a clean image after having applied affine morphological scale space (AMSS) smoothing. Special attention is paid to IIPs shared by parents and children: In such a case $\text{RGB}(\text{IIP}_{\text{parent}})$ is replaced with $\text{RGB}(\text{IEP}_{\text{child}})$ in order to have a consistent treatment of which RGB values are associated to the inner and outer sides of both pseudo-level lines, respectively.

Similar modifications regarding coincidences between IIP/IEP pixels of siblings are under investigation but currently not part of our implementation.

3.3 Second Step: Smoothing

Smoothing the pseudo-level lines: AMSS. Affine morphological scale space works equally to scalar-valued images with respect to process of curve evolution itself. Additionally, for vector-valued images the curveIEP is evolved too.

Furthermore, the intensities of each subpixel carried along the pseudo-level line must be stored and, if necessary, modified correctly. This is only possible with huge expense, because the number of subpixels $\in \; c$ changes within the smoothing process. New points inserted to the curve get the arithmetic mean value, computed with the intensities of the immediate former and immediate next subpixel, associated.

Smoothing the intensities along curves. The second step of denoising is to smooth the vectorial intensities along the curves. As these intensities affect the quality of the denoised image, a smooth colour gradient is desired. Therefore linear explicit diffusion is applied to all curves.

Denoting by v_i the RGB value of the pixel $p_i \in$ curve c with $i \in \{0, \dots, l\}$, $l = \text{length}(c)$, we smooth the discrete 1-D signal $(v_0, \dots, v_l)$ by linear diffusion [21, Chap. 1]. In doing so, we approximate the diffusion PDE $v_t = v_{xx}$ by the standard explicit finite-difference scheme

$$v_i^{k+1} = v_i^k + \tau(v_{i+1}^k - 2v_i^k + v_{i-1}^k) \,. \tag{23}$$

for iteration numbers $k \geq 0$, starting with the given signal in step $k = 0$ and assuming a spatial step size of 1. This explicit scheme is stable for time step sizes $\tau \leq 1/2$. In the present paper, we run three iterations, amounting to a diffusion time $t = 1.5$.

3.4 Third Step: Reconstruction

Given the curvature image – shortened level lines printed equipped correct intensities – it is necessary to reconstruct, finally, a clean denoised image.

To this end, the intensities (RGB) of IIPs and IEPs from the curvature image are fixed. Using these as Dirichlet boundary conditions, intensities on the remaining pixels can be inpainted by linear diffusion [21, Chap. 1] which should in principle be computed until numerical convergence to a steady state. Using a standard explicit finite-difference scheme for 2D diffusion, one computes

$$u_{i,j}^k = \left(1 - 4\frac{\tau}{h^2}\right) u_{i,j}^{k-1} + \frac{\tau}{h^2}\left(u_{i-1,j}^{k-1} + u_{i,j-1}^{k-1} + u_{i,j+1}^{k-1} + u_{i+1,j}^{k-1}\right) \tag{24}$$

for all non-IIP/IEP pixels (i, j) and iterations $k = 1, 2, \dots$ until

$$|u_{i,j}^k - u_{i,j}^{k-1}| < \varepsilon \qquad \text{for all } i, j. \tag{25}$$

In (24), τ denotes the time step size and h the spatial grid step size of the image; assuming $h = 1$ the scheme is stable for $\tau \leq 1/4$.

The number of iterations until the stopping condition (25) is met can be reduced by a suitable initialisation; to this end, the pregion of each node nd_l with $l = 0, \dots, \text{number(nodes)} - 1$ can be prefilled line by line by colouring the pixels $u_{i,j}^0 \in \text{pregion}(nd_l)$ with the intensity of the last met $\text{IIP}(nd_l)$ in this line.

4 Experimental Demonstration

In this section we will illustrate curvature-based denoising of grey-scale and colour images with some example images. All methods were implemented entirely in C++ on the basis of the standard library, some components being adapted from the published implementation of [7].

Before we turn to show actual image smoothing examples, we discuss the visualisation of curvature maps.

Fig. 10: Viridis colour bar.

Fig. 11: Rescaled Viridis colour bar.

4.1 Curvature maps and visualisation

Curvature maps give a coloured information about the curvatures present in the denoised image. As not all pixels are part of a level line, we compute first by (4) the curvatures in those pixels that are IIPs of the shortened level lines. Fixing these as Dirichlet boundary conditions, we inpaint the curvature map to the remaining pixels by running linear diffusion until numerical convergence to a steady state is reached, analogous to Section 3.4.

For visualisation, the so obtained dense curvature field $(c_{i,j})$ with values in $[-1, 1]$ is coloured on a modification of the Viridis colour scale [20] (see also [8]) reaching from dark blue/purple, $(R, G, B) = (68.0, 1.0, 84.0)$, for $c_{i,j} = -1$, via blue/green, $(R, G, B) = (32.0, 146.0, 140.0)$, for $c_{i,j} = 0$, to yellow/orange, $(R, G, B) = (253.0, 231.0, 37.0)$, for $c_{i,j} = 1$.

Whereas the original Viridis colour scale, Fig. 10, ensures linear contrast between different values visualised, we use a modification in which instead of $c_{i,j}$ itself the quantity $\tanh(10\, c_{i,j})$ is coloured by the original Viridis scale, resulting in enhanced colour contrast for curvatures around zero as shown in Fig. 11.

4.2 Image Smoothing Experiments

First, we show an experiment on a grey-scale image, Fig. 12, to demonstrate the effect of the modified neighbourhood setting with dummy pixels as introduced in Section 2.1. Note that the denoising results in Fig. 12b and Fig. 12c are visibly biased to the respective diagonal directions of the chosen 6-neighbourhoods.

For vector-valued images, first the effect of AMSS is highlighted with Fig. 13. Further, in Fig. 14 we show a denoising result for a colour image with Gaussian noise. In Fig. 15 we demonstrate the effect for impulse noise.

With our non-optimised implementation, run times ranged from about 2 min (grey-scale experiment, Fig. 12) to about 10 min (noise-free colour experiment, Fig. 13); surprisingly, the noisy colour images in Fig. 14 and Fig. 15 were processed much faster than Fig. 13, probably due to the dominance of much shorter pseudo-level lines. At any rate, we expect that run times can be significantly reduced by future algorithmic optimisations.

5 Summary and Outlook

In this work, we have extended the curvature-based denoising algorithm for grey-scale images from [7] to vector-valued, such as RGB colour, images. In

(a) Noisy source (b) SC2, E_6^1

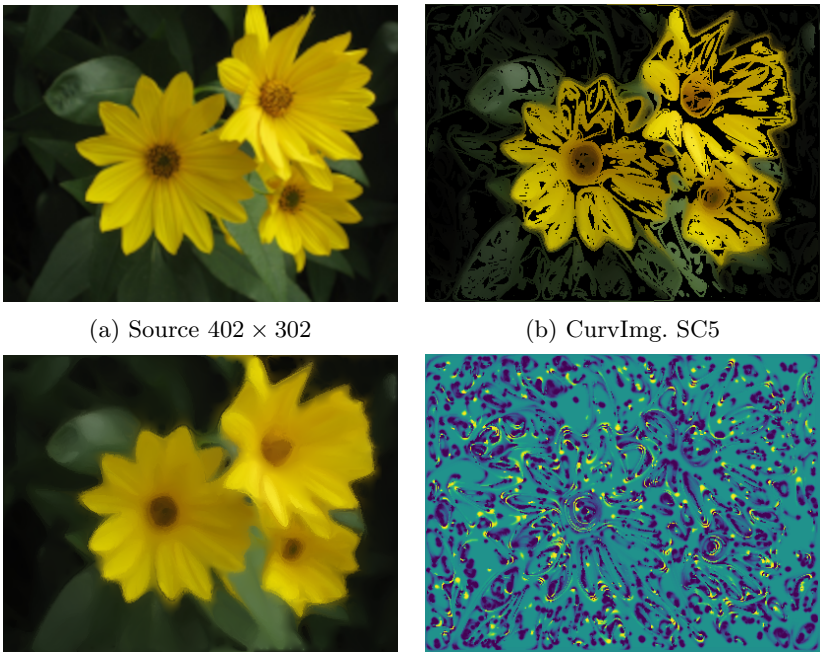
(c) SC2, E_6^2 (d) SC2, $E_4 + E_8$

Fig. 12: The level lines in the source image *camera40.pgm* [256 × 256] (a) noisy with Gaussian noise ($\sigma = 40$) are extracted using different edgel types (cf. Fig. 2 in Section 2.1). (b)–(d) Denoised images after having applied AMSS smoothing with SC2. (b) 6-connectedness of type 1 (E_6^1), (c) 6-connectedness of type 2 (E_6^2) used. (b) shows stripes 45° in lower right, (c) 45° in upper right direction. The 4- (for dummy pixels) and 8-connectedness edgels (for integer pixels) – $E_4 + E_8$ – used in (d) remove the artifacts.

the course of this extension, we have designed a robust extraction algorithm for pseudo-level lines. Due to the absence of a usable pvalue in vector-valued images, the intensities must be carried along each line in a properly manner. Special handling is required in certain configurations where nested discrete pseudo-level lines touch each other.

The smoothing process for level lines using AMSS could be transferred verbatim to pseudo-level lines. However, along with smoothing the pseudo-level lines their attached intensity information needs to be smoothed as well.

Finally, the reconstruction step required again an adaptation because for vector-valued images it is no longer sufficient to fill private regions with constant values. Diffusion inpainting was used to overcome this difficulty.



(a) Source 402 × 302 (b) CurvImg. SC5

(c) Smoothed SC5 (d) CurvMap(b)

Fig. 13: Smoothing a colour image. (a) Image *flowers*. (b)–(d): Result of AMSS smoothing with SC5 applied to the 96 744 level lines. (Image source: `https://cs.colby.edu/courses/S19/cs151-labs/labs/lab04/Flowers.png`, accessed 2022-02-02. Author: Colby)

(a) Noisy source 502 × 207 (b) Smoothed SC2

Fig. 14: Denoising of a colour image with Gaussian noise. (a) Image *HoheMunde40* degraded by Gaussian noise ($\sigma = 40$). – (b) Denoised by applying AMSS smoothing with SC2 to the 206 842 level lines. (Image source: `https://www.telfs.at/files/user_upload/915x375/wohnen-leben-hohe-munde-hausberg-telfs-02.jpg`, accessed: 2022-02-02.)

By experiments the viability of the approach was demonstrated. We presented processed colour images as well as exemplary curvature maps. Ongoing work is, on one hand, directed at algorithmic optimisations.

(a) Noisy source 400 × 300 (b) Smoothed SC1

Fig. 15: Denoising of a colour image with impulse noise. (a) Image *MTBIN20_cbc* degraded by synthetic impulse noise where 20% of all pixels are replaced with a random RGB value. – (b) Denoised by applying AMSS smoothing with SC1 to the 206 726 level lines.

On the other hand, as already pointed out in [7], denoising is just one application of the curvature-based image processing paradigm underlying this work. The sub-pixel localised curvature information extracted in the course of the method bears potential for a range of further applications like image registration, segmentation, sharpening, or feature extraction for computer vision applications; note that points with extremal curvature such as corners are, in different formulations, long-established features in computer vision. With our curvature-based analysis method for vector-valued images the new paradigm can also be made available for colour images.

References

1. Alvarez, L., Morales, F.: Affine morphological multiscale analysis of corners and multiple junctions. International Journal of Computer Vision **25**(2) (1997)
2. Alvarez, L., Morel, J.M.: Formalization and computational aspects of image analysis. Acta Numerica **3**, 1–59 (1994). https://doi.org/10.1017/S0962492900002415
3. Ambrosio, L., Caselles, V., Masnou, S., Morel, J.M.: Connected components of sets of finite perimeter and applications to image processing. Journal of the European Mathematical Society **3**, 39–92 (2 2001). https://doi.org/10.1007/PL00011302
4. Attneave, F.: Some informational aspects of visual perception. Psychological Review **61**(3), 183–193 (1954)
5. Bertalmío, M., Levine, S.: Denoising an image by denoising its curvature image. SIAM Journal on Imaging Sciences **7**(1), 187–211 (2014). https://doi.org/10.1137/120901246
6. Chung, D.H., Sapiro, G.: On the level lines and geometry of vector-valued images. IEEE Signal Processing Letters **7**(9), 241–243 (2000). https://doi.org/10.1109/97.863143
7. Ciomaga, A., Monasse, P., Morel, J.M.: The image curvature microscope: Accurate curvature computation at subpixel resolution. Image Processing On Line **7**, 197–217 (2017). https://doi.org/10.5201/ipol.2017.212

8. Crameri, F., Shephard, G.E., Heron, P.J.: The misuse of colour in science communication. Nature Communications **11** (2020). https://doi.org/10.1038/s41467-020-19160-7
9. Crandall, M.G., Lions, P.L.: Convergent difference schemes for nonlinear parabolic equations and mean curvature motion. Numerische Mathematik **75**(1), 17–41 (1996)
10. Cumani, A.: Edge detection in multispectral images. CVGIP: Graphical Models and Image Processing **53**(1), 40–51 (1991). https://doi.org/10.1016/1049-9652(91)90018-F
11. Di Zenzo, S.: A note on the gradient of a multi-image. Computer Vision, Graphics, and Image Processing **33**(1), 116–125 (1 1986). https://doi.org/10.1016/0734-189X(86)90223-9
12. Guggenheimer, H.W.: Differential Geometry. McGrawHill, New York (1963)
13. Guichard, F., Morel, J.M., Ryan, R.: Contrast invariant image analysis and PDE's. Tech. rep., Image Processing On Line (2004), `http://dev.ipol.im/~morel/LivreGMR/JMMBookOct04.ps`
14. Kreyszig, E.: Differential Geometry. University of Toronto Press, Toronto (2019). https://doi.org/10.3138/9781487589455
15. Moisan, L.: Affine plane curve evolution: a fully consistent scheme. IEEE Transactions on Image Processing **7**(3), 411–420 (1998). https://doi.org/10.1109/83.661191
16. Mondelli, M., Ciomaga, A.: Finite difference schemes for MCM and AMSS. Image Processing On Line **1**, 127–177 (2011). https://doi.org/10.5201/ipol.2011.cm_fds
17. Sapiro, G.: Geometric Partial Differential Equations and Image Analysis. Cambridge University Press, Cambridge (2001)
18. Sapiro, G., Tannenbaum, A.: On affine plane curve evolution. Journal of Functional Analysis **119**, 79–120 (1994)
19. Song, Y.: A topdown algorithm for computation of level line trees. IEEE Transactions on Image Processing **16**(8), 2107–2116 (2007). https://doi.org/10.1109/TIP.2007.899616
20. van der Walt, S., Smith, N.: MPL colour maps. Online Ressource (2020), `https://bids.github.io/colormap`, accessed 2022-03-12
21. Weickert, J.: Anisotropic Diffusion in Image Processing. Teubner, Stuttgart (1998)