A Discrete Theory and Efficient Algorithms for Forward-and-Backward Diffusion Filtering

Martin Welk
Private University for Health Sciences, Medical Informatics and Technology (UMIT)
Eduard-Wallnöfer-Zentrum 1
6060 Hall/Tyrol, Austria
Tel.: +43-50-86483974
martin.welk@umit.at

Joachim Weickert
Mathematical Image Analysis Group
Faculty of Mathematics and Computer Science, Campus E1.7
Saarland University
66041 Saarbrücken, Germany
Tel.: +49-681-30257340
weickert@mia.uni-saarland.de

Guy Gilboa
Electrical Engineering Department
Technion – IIT
Technion City, Haifa 32000, Israel
Tel.: +972-4-8294653
guy.gilboa@ee.technion.ac.il

September 3, 2018 (revised version)

Abstract

Image enhancement with forward-and-backward (FAB) diffusion lacks a sound theory and is numerically very challenging due to its diffusivities that are negative within a certain gradient range. In our paper we address both problems. First we establish a comprehensive theory for space-discrete and time-continuous FAB diffusion processes. It requires approximating the gradient magnitude with a nonstandard discretisation. Then we show that this theory carries over to the fully discrete case, when an explicit time discretisation with a fairly restrictive step size limit is applied. To come up with more efficient algorithms we propose three accelerated schemes: (i) an explicit scheme with global time step size adaptation that is also well-suited for parallel implementations on GPUs, (ii) a randomised two-pixel scheme that offers optimal adaptivity of the time step size, (iii) a deterministic two-pixel scheme which benefits from less restrictive consistency bounds. Our experiments demonstrate that these algorithms allow speed-ups by up to three orders of magnitude without compromising stability or introducing visual artifacts.

Keywords: image enhancement • diffusion filtering • backward parabolic PDEs • dynamical systems • nonstandard finite differences • ill-posed problems

1 Introduction

Partial differential equations (PDEs) and variational approaches for enhancing digital images have been investigated intensively in the last thirty years. An overview can be found e.g. in [1, 33]. As continuous frameworks, these approaches excel by their concise and transparent formulation and their natural representation of rotational invariance.

However, some highly interesting models are affected by well-posedness problems, making their analysis in a continuous setting difficult. Well-posedness properties of space-discrete and fully-discrete formulations therefore receive increasing attention.

Regarding the Perona–Malik filter, a space-discrete and fully discrete theory for smooth nonnegative diffusivities was established by Weickert [33]. The corresponding explicit scheme was proven in [34] to preserve monotonicity in 1D. This explains that staircasing is the worst phenomenon that can happen. An extension of this analysis to singular nonnegative diffusivities was accomplished by Pollak et al. [25] who verified the well-posedness of dynamical systems with discontinuous right hand sides arising from a space-discrete Perona-Malik model.

Using backward diffusion to enhance image features has a long tradition, going back already to a 1955 publication by Kovasznay and Joseph [12] and a 1965 paper by Gabor [8, 15]. Since backward diffusion is a classical example of an ill-posed problem [29], designing appropriate numerical schemes for these processes continues to be a difficult research topic; see e.g. the recent papers [5, 6, 40] and the references therein. For the stabilised inverse linear diffusion process introduced by Osher and Rudin, a continuous well-posedness theory is lacking, but a stable minmod discretisation could be devised [21]. Breuß and Welk [3] showed that staircasing cannot be avoided by suitable space discretisations.

For shock filtering [13, 22] which, too, is difficult to analyse in the continuous setting, discrete well-posedness results are found in [38], including an analytic solution of the corresponding dynamical system.

On the variational side, Nikolova has published a number of impressive papers that provide deep insights into the behaviour of minimisers of space-discrete energies, even if they are highly nonconvex or nondifferentiable; see e.g. [19, 20]. It would have been extremely difficult if not impossible to obtain similar results in the continuous setting.

The *forward-and-backward (FAB) diffusion* model of Gilboa et al. [10] is another example for such difficulties. Designed for the sharpening of images, it is basically a Perona–Malik type PDE filter. However, its diffusivities take positive values in some regions and negative values in others. Thus, it is not surprising that no well-posedness results are available in the continuous setting and experiments with standard explicit discretisations show violations of a maximum–minimum principle. On the other hand, FAB diffusion has been modified and generalised in a number of ways [9, 26, 27, 30, 31, 32]. Thus, it would clearly be desirable to have some theoretical underpinnings and reliable and efficient algorithms for this class of methods. However, in view of the difficulties described above, it is most promising to establish a sound theory if one focuses on discrete FAB models.

Our Contribution. The goal of our paper is to address these two problems. In a first step, we establish a space-discrete diffusion theory that generalises the results from Weickert [33] which are only applicable for positive diffusivities. By relaxing some of its requirements we end up with a framework that can also be applied to space-discrete FAB diffusion: In particular, we can establish well-posedness, a preservation of the average grey value, a maximum–

minimum principle, an interesting Lyapunov function and convergence to a discrete steady state. However, these results hold only if we replace the standard discretisation of the gradient magnitude by a nonstandard one that vanishes at discrete extrema. We show that this theory also carries over to the fully discrete setting with an explicit time discretisation, if the time step size stays below a very severe bound that results from worst case a priori estimates. By replacing them with more realistic a posteriori estimates we end up with much more efficient schemes. They may either adapt the time step size for all locations simultaneously, or they can be based on estimates with maximal locality / adaptivity by splitting the diffusion into pairs of neighbouring pixels. Our experiments show that these acceleration strategies can lead to speed-ups of up to three orders of magnitude.

The present paper is based on two conference publications [35, 37]: The first one has presented early results on a space-discrete theory and one-dimensional experiments, whereas the second one focuses on a fully discrete theory and an efficient randomised two-pixel scheme for 2D images. Our present manuscript extends these preliminary findings in several ways:

- In the space-discrete and time-continuous setting, we establish Lyapunov functions and come up with convergence results.
- In the fully discrete framework, we propose two novel schemes:
  (i) An explicit scheme with global time step size adaptation and its parallelisation on a GPU.
  (ii) A deterministic two-pixel scheme that outperforms its randomised predecessor from [37] w.r.t. its consistency properties and its efficiency.
- Our experiments are more comprehensive and include e.g. also comparisons between different types of FAB diffusivities.

Structure of the Paper. Sections 2 and 3 provide concepts that are essential for understanding the remainder of our paper by reviewing FAB diffusion [10] and the classical space-discrete diffusion framework from [33], respectively. In Section 4 we introduce our novel space-discrete theory that is also applicable to FAB diffusion. A corresponding fully discrete theory for explicit schemes is established in Section 5. Section 6 discusses several algorithmic variants which are substantially more efficient by exploiting local adaptivity or parallelism, and Section 7 confirms this by experiments. The paper is concluded with a summary and an outlook in Section 8.

2 Forward-and-Backward Diffusion Filtering

Forward-and-backward (FAB) diffusion filtering has been proposed by Gilboa, Sochen and Zeevi in 2002 [10]. We recall the basic definitions for the 2D case (generalisation to arbitrary dimensions is straightforward).

The starting point is the well-known nonlinear diffusion model of Perona and Malik [24]. Let a greyscale image $f : \Omega \to \mathbb{R}$ on a rectangular image domain $\Omega \subset \mathbb{R}^2$ be given. To enhance this image, filtered versions $u(\boldsymbol{x}, t)$ of $f(\boldsymbol{x})$ are created by solving an initial–boundary value problem for the PDE

$$\partial_t u \;=\; \operatorname{div}\big(g(|\boldsymbol{\nabla} u|^2)\,\boldsymbol{\nabla} u\big) \tag{1}$$

with the input image f as initial condition,

$$u(\boldsymbol{x}, 0) = f(\boldsymbol{x}) \ , \tag{2}$$

and homogeneous Neumann boundary conditions,

$$\partial_{\boldsymbol{n}} u = 0 \ , \tag{3}$$

where $\boldsymbol{n}$ denotes a vector normal to the image boundary $\partial\Omega$. Here $\boldsymbol{x}$ stands for $(x, y)^\top$. Writing partial derivatives by subscripts, we denote by $\boldsymbol{\nabla} := (\partial_x, \partial_y)^\top$ the spatial gradient and by div its corresponding divergence operator.

The specificity of FAB diffusion is the choice of the diffusivity function. The Perona–Malik framework [24] requires g to take positive values. In contrast, a FAB diffusivity is positive for small image gradients, while it becomes negative for larger ones. Different models for such diffusivities g have been proposed, see for example [9, 26]. In [9] the diffusivity

$$g(s^2) = \frac{1}{\sqrt{1 + (s/k_f)^2}} - \frac{\alpha}{1 + (s/k_b)^2}, \tag{4}$$

is proposed, where k_f and k_b control the gradient magnitudes for forward and backward diffusion, respectively, and α is the weight between these terms. Note that for suitable values of the parameters, this diffusivity is positive for small image gradients, while it becomes negative for larger ones, and finally becomes positive again, see [9]. Another example, adapted from [26], is

$$g(s^2) \ = \ 2 \exp\left(-\frac{\kappa^2 \ln 2}{\kappa^2 - 1} \cdot \frac{s^2}{\lambda^2}\right) - \exp\left(-\frac{\ln 2}{\kappa^2 - 1} \cdot \frac{s^2}{\lambda^2}\right) \tag{5}$$

with admissible parameters $\lambda > 0$ and $\kappa > 1$. In contrast to (4), this diffusivity does not become positive again for large gradient magnitudes: Asymptotically it approaches 0 from below when the gradient magnitude grows to infinity. We will discuss different types of FAB diffusivities in more detail in Subsection 7.5.

The main motivation for FAB was to introduce a general-purpose stable image sharpening process for which dominant gradients, above the noise level, are strongly enhanced.

To understand the difference between a Perona–Malik filter and FAB diffusion, let us rewrite the evolution (1) in terms of the flowline direction $\eta \parallel \boldsymbol{\nabla} u$ and the isophote direction $\xi \perp \boldsymbol{\nabla} u$. With the flux function $\phi(s) := g(s^2)s$ this yields

$$\partial_t u \ = \ \phi'(|\boldsymbol{\nabla} u|)\, u_{\eta\eta} \ + \ g(|\boldsymbol{\nabla} u|^2)\, u_{\xi\xi} \, . \tag{6}$$

Both the Perona–Malik filter and FAB diffusion allow flux functions which may be decreasing in an interval: Consider e.g. the positive diffusivity

$$g(s^2) \ = \ \frac{1}{1 + s^2/\lambda^2} \qquad (\lambda > 0) \tag{7}$$

in the Perona–Malik case. Thus, we may encounter backward diffusion in the flowline direction η for both filters, producing a desirable sharpening across edges which enhances contrast. Along the isophote direction ξ, however, both processes differ: Since the Perona-Malik filter

permits only positive diffusivities, it always shows forward diffusion behaviour. On the contrary, the FAB diffusion may also exhibit backward behaviour. Thus, it is a more consequent realisation of a sharpening process.

Like some other types of nonlinear diffusion processes, FAB diffusion can be related to variational approaches. In [9] FAB diffusion with the diffusivity (4) has been interpreted as an energy minimisation process of a nonmonotone potential in the shape of a triple-well. FAB diffusion has also been put into relation with wavelet methods for image enhancement [18]. A schematic form of the FAB diffusivity is shown in Figure 1. Several examples of diffusivities and their corresponding potentials (penalisers) are plotted within the experimental section in Figure 6.

Beyond these works, there is not much theoretical analysis of the fully continuous FAB process documented in the literature. In particular, no existence, uniqueness and stability results have been proven. In [10] it was conjectured that FAB diffusion violates a maximum–minimum principle due to the effect of negative diffusivities.

Indeed, such violations can be observed in numerical experiments, where FAB diffusion is discretised using standard numerical methods. However, this is no longer true if more sophisticated space discretisations are used: As [35] brought out, one can discretise FAB diffusion in a way such that the space-discrete process obeys the maximum–minimum principle, and further useful theoretical results on the space-discrete process could be established. Stability properties of fully discrete FAB diffusion were considered in [35], too, but limited to the 1D case. This analysis was extended to the 2D case in [37]. In Sections 4 and 5, we will also detail the analytical results from [35] and [37], focusing on the 2D case.

3 A Space-Discrete Diffusion Framework

Let us now review the space-discrete diffusion framework of Weickert [33], since parts of it can be extended to the FAB setting.

To study diffusion in the space-discrete 2D case, we consider the discrete image domain

$$\Gamma := \{1, 2, \ldots, m\} \times \{1, 2, \ldots, n\} \; . \tag{8}$$

Each pixel $(i, j) \in \Gamma$ is assumed to be centred in the location $\big((i - \frac{1}{2})h_1, (j - \frac{1}{2})h_2\big)$, where h_1 and h_2 denote the grid size (pixel width) in x- and y-direction, respectively.

Denoting by $u_{i,j}$ an approximation to u in pixel $(i, j) \in \Gamma$, a standard discretisation of a Perona-Malik type diffusion equation

$$\partial_t u \;=\; \partial_x \Big(g(|\boldsymbol{\nabla} u|^2)\, \partial_x u \Big) \;+\; \partial_y \Big(g(|\boldsymbol{\nabla} u|^2)\, \partial_y u \Big) \tag{9}$$

in some inner pixel (i, j) yields the ordinary differential equation

$$\begin{aligned} \frac{\mathrm{d}u_{i,j}}{\mathrm{d}t} = \; & \frac{1}{h_1} \left(\frac{g_{i+1,j} + g_{i,j}}{2} \, \frac{u_{i+1,j} - u_{i,j}}{h_1} - \frac{g_{i,j} + g_{i-1,j}}{2} \, \frac{u_{i,j} - u_{i-1,j}}{h_1} \right) \\ & + \frac{1}{h_2} \left(\frac{g_{i,j+1} + g_{i,j}}{2} \, \frac{u_{i,j+1} - u_{i,j}}{h_2} - \frac{g_{i,j} + g_{i,j-1}}{2} \, \frac{u_{i,j} - u_{i,j-1}}{h_2} \right) . \end{aligned} \tag{10}$$

This formula even holds for boundary pixels, provided that the homogeneous Neumann boundary conditions (3) are implemented by mirroring boundary pixels into dummy pixels:

$$u_{0,j}^k := u_{1,j}^k \ , \qquad u_{m+1,j}^k := u_{m,j}^k \ , \qquad u_{i,0}^k := u_{i,1}^k \ , \qquad u_{i,n+1}^k := u_{i,n}^k \tag{11}$$

for all indices i and j. A suitable discretisation for the diffusivity g will be discussed later.

In a more compact notation, one can represent a pixel $(i,j) \in \Gamma$ by a single index $k(i,j) \in J$ with the new index set $J = \{1,\ldots,N\}$, where $N = n \cdot m$. This leads to

$$\frac{\mathrm{d}u_k}{\mathrm{d}t} = \sum_{\nu=1}^{2} \sum_{l \in \mathcal{N}_\nu(k)} \frac{g_l + g_k}{2h_\nu^2} \, (u_l - u_k) \ , \tag{12}$$

where $\mathcal{N}_\nu(k)$ are the neighbours of pixel k in the ν-th coordinate direction (boundary pixels may have less neighbours). This can be written as a system of ordinary differential equations (ODEs):

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} = \boldsymbol{A}(\boldsymbol{u}) \, \boldsymbol{u} \ , \tag{13}$$

where $\boldsymbol{u} = (u_1,\ldots,u_N)^\top$, and the $N \times N$ matrix $\boldsymbol{A}(\boldsymbol{u}) = \big(a_{k,l}(\boldsymbol{u})\big)$ satisfies

$$a_{k,l} = \begin{cases} \frac{g_k+g_l}{2h_\nu^2} & \text{if } l \in \mathcal{N}_\nu(k) \ , \\ -\sum\limits_{\nu=1}^{2} \sum\limits_{l \in \mathcal{N}_\nu(k)} \frac{g_k+g_l}{2h_\nu^2} & \text{if } l = k \ , \\ 0 & \text{else} \ . \end{cases} \tag{14}$$

A space-discrete problem class (P_s) is defined in the following way.

Let $\boldsymbol{f} \in \mathbb{R}^N$. Find a function $\boldsymbol{u} \in \mathrm{C}^1([0,\infty),\mathbb{R}^N)$ that satisfies the initial value problem

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} = \boldsymbol{A}(\boldsymbol{u}) \, \boldsymbol{u} \ , \qquad \boldsymbol{u}(0) = \boldsymbol{f} \ ,$$

where $\boldsymbol{A} = (a_{i,j})$ has the following properties:

(S1) Lipschitz-continuity of $\boldsymbol{A} \in \mathrm{C}(\mathbb{R}^N, \mathbb{R}^{N\times N})$ for every bounded subset of $\mathbb{R}^N$,

(S2) symmetry: $a_{i,j}(\boldsymbol{u}) = a_{j,i}(\boldsymbol{u}) \quad \forall\, i,j \in J, \ \forall\, \boldsymbol{u} \in \mathbb{R}^N$,

(S3) vanishing row sums: $\sum_{j\in J} a_{i,j}(\boldsymbol{u}) = 0 \quad \forall\, i \in J, \ \forall\, \boldsymbol{u} \in \mathbb{R}^N$,

(S4) nonnegative off-diagonals: $a_{i,j}(\boldsymbol{u}) \geq 0 \quad \forall\, i \neq j, \ \forall\, \boldsymbol{u} \in \mathbb{R}^N$,

(S5) irreducibility for all $\boldsymbol{u} \in \mathbb{R}^N$.

(P_s)

One should remember that a matrix $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ is called irreducible if for any $i,j \in J$ there exist $k_0,\ldots,k_r \in J$ with $k_0 = i$ and $k_r = j$ such that $a_{k_p k_{p+1}} \neq 0$ for $p = 0,\ldots,r-1$. In other words: There is a path from pixel i to pixel j along which the diffusivities do not vanish.

Under these requirements the subsequent result is proven in [33]:

Proposition 1 (Properties of Space-Discrete Diffusion Filtering)**.** *For the space-discrete filter class (P_s) the following statements are valid:*

(a) (Well-Posedness)
For every $T > 0$ the problem (P_s) has a unique solution $\boldsymbol{u}(t) \in \mathrm{C}^1([0,T],\mathbb{R}^N)$. This solution depends continuously on the initial value and the right-hand side of the ODE system.

(b) (Maximum–Minimum Principle)
Let $a := \min_{j\in J} f_j$ and $b := \max_{j\in J} f_j$. Then, $a \leq u_i(t) \leq b$ for all $i \in J$ and $t \in [0,T]$.

(c) (Average Grey Level Invariance)
The average grey level $\mu := \frac{1}{N}\sum_{j\in J} f_j$ is not affected by the space-discrete diffusion filter: $\frac{1}{N}\sum_{j\in J} u_j(t) = \mu$ for all $t > 0$.

(d) (Lyapunov Functions)
For all $r \in \mathrm{C}^1[a,b]$ with increasing derivative r' on $[a,b]$, the function $V(t) := \Phi(\boldsymbol{u}(t)) := \sum_{i\in J} r(u_i(t))$ is a Lyapunov function, i.e. $V(t)$ is decreasing and bounded from below by $\Phi(\boldsymbol{c})$, where $\boldsymbol{c} := (\mu,\ldots,\mu)^\top \in \mathbb{R}^N$.

(e) (Convergence to a Constant Steady State)
$\lim\limits_{t\to\infty} \boldsymbol{u}(t) = \boldsymbol{c}$.

The proof shows that not all of the requirements (S1)–(S5) are necessary for each of the theoretical results above: Requirement (S1) is needed for local well-posedness, while proving a maximum–minimum principle requires (S3) and (S4). Local well-posedness together with the maximum–minimum principle implies global well-posedness. The average grey value invariance is based on (S2) and (S3). The existence of Lyapunov functionals can be established by means of (S2)–(S4), and convergence to a constant steady state requires (S5) in addition to (S2)–(S4).

4 Analysis of Space-Discrete FAB Diffusion

We turn now to apply the results from the previous section to space-discrete FAB diffusion. Herein we follow mostly [35], adding in Subsection 4.2 new findings on Lyapunov functions and convergence to a constant steady state.

Whereas we write down our theory focused on the 2D regular grid Γ as introduced in (8), the analysis in the previous as well as in most of this section can be generalised straightforwardly to diffusion in arbitrary dimensions, and even to diffusion on graphs, where the image domain J is the node set of a graph, edges connecting neighbouring nodes are weighted by distances, and intensities propagate in the diffusion process via these edges, see [7]. For example, by rewriting (12) as

$$\frac{\mathrm{d}u_k}{\mathrm{d}t} = \sum_{l\in\mathcal{N}(k)} \frac{g_l + g_k}{2h_{k,l}^2}\,(u_l - u_k)\,, \tag{15}$$

(and (14) accordingly) where $\mathcal{N}(k)$ are all neighbours of pixel k, and $h_{k,l} := h_1$ for horizontal and $h_{k,l} := h_2$ for vertical neighbours, it becomes obvious that (15) can be used verbatim also for diffusion on graphs, by just redefining $h_{k,l}$ based on the edge weights (distances).

Specific to regular grids is Subsection 4.3 where discretisations of the diffusivity are discussed.

4.1 Direct Consequences from the Space-Discrete Framework

It is straightforward to verify the prerequisites (S1)–(S5) for the popular *positive* diffusivity functions, such that Proposition 1 is applicable. However, for FAB diffusion negative diffusivities are possible and the situation becomes much more complicated. One immediately sees that space-discrete FAB diffusion satisfies (S1: smoothness), (S2: symmetry), and (S3: vanishing row sums). However, this just implies local well-posedness and average grey level invariance.

By inspecting (14) it becomes clear that (S4: nonnegative off-diagonals) and (S5: irreducibility) cannot be satisfied for typical FAB diffusivities: These diffusivities may vanish (which violates (S5)) and they may even become negative (violating (S4)). As a consequence, global well-posedness, a maximum–minimum principle, Lyapunov functions and convergence to a constant steady state cannot be proven in this way.

4.2 Scale-Space Theory under Weaker Prerequisites

For the practical applicability of FAB diffusion it would be highly desirable to have at least global well-posedness and a maximum–minimum principle. Is there a remedy for these properties? Fortunately the answer is affirmative, since (S4: nonnegative off-diagonals) can be replaced by a less restrictive condition that only holds at extrema:

Proposition 2 (Maximum–Minimum Principle for Space-Discrete Diffusion Filtering under Weaker Conditions)**.** *Assume that a space-discrete filter satisfies only the properties* (S1)–(S3) *of the framework* (P_s)*, and*

(S4a) *nonnegative off-diagonals at extrema:*
$a_{i,j}(\boldsymbol{u}) \geq 0$ *for all* $j \in J$ *with* $j \neq i$ *if* $\boldsymbol{u}$ *has an extremum in* i*.*

Then the well-posedness result (a), the maximum–minimum principle (b), and the average grey level invariance (c) of Proposition 1 are still satisfied.

Proof. Following [33], one observes that in some pixel k that is a discrete global maximum (i.e. $u_k \geq u_j$ for all $j \in J$), condition (S4a) implies that

$$\begin{aligned} \frac{\mathrm{d}u_k}{\mathrm{d}t} &= \sum_{j \in J} a_{kj}(\boldsymbol{u})\, u_j = a_{kk}(\boldsymbol{u})\, u_k + \sum_{j \in J \setminus \{k\}} \underbrace{a_{kj}(\boldsymbol{u})}_{\geq 0} \underbrace{u_j}_{\leq u_k} \\ &\leq u_k \cdot \sum_{j \in J} a_{kj}(\boldsymbol{u}) \overset{\text{(S3)}}{=} 0 \,. \end{aligned} \tag{16}$$

In the same way one can prove that if k is a minimum, one has $\mathrm{d}u_k/\mathrm{d}t \geq 0$.

This nonenhancement behaviour in extrema is the only place where nonnegativity is required in the entire proof of the maximum–minimum principle in [33]. As a consequence, the maximum–minimum principle still holds if (S4) is replaced by the weaker condition (S4a). Moreover, together with local well-posedness, global well-posedness is obtained. This completes the proof. □

Further adapting the conditions of the general theory in Section 3 to FAB diffusivities, we can also devise Lyapunov functions.

Proposition 3 (Lyapunov Condition for Space-Discrete Diffusion Filtering under Weaker Conditions)**.** *Assume that a space-discrete filter satisfies only the properties* (S1)–(S3) *of the framework* (P_s) *and* (S4a) *from Proposition 2. Assume further that there is a symmetric neighbourhood relation* $\sim$ *on* J *such that the following two conditions are satisfied:*

(S4b) *positive off-diagonals in neighbourhood of extrema:*
If $\boldsymbol{u}$ *has an extremum in* i, $a_{i,j}(\boldsymbol{u}) > 0$ *holds for all* $j \in J \setminus \{i\}$ *with* $i \sim j$.

(S5a) *irreducibility of neighbourhood relation:*
The neighbourhood graph in J *based on the neighbourhood relation* $\sim$ *is connected.*

Using $u_{\max} := \max_j u_j$, $u_{\min} := \min_j u_j$ *and*

$$\Phi(\boldsymbol{u}) := u_{\max} - u_{\min} \ , \tag{17}$$

define then the function $V : [0, \infty) \to \mathbb{R}_0^+$ *by* $V := \Phi \circ \boldsymbol{u}$, *i.e.*

$$V(t) := u_{\max}(t) - u_{\min}(t) \ . \tag{18}$$

Then the Lyapunov property (d) and steady state property (e) of Proposition 1 are satisfied with the Lyapunov function $V(t)$.

Remark 1. For images on regular grids which are in the focus of the present paper, the symmetric neighbourhood relation $\sim$ will relate a pixel to its immediate neighbours in the coordinate directions (resulting in a 4-neighbourhood for interior pixels of 2D images). However, the proposition is equally applicable to arbitrary neighbourhood graphs.

The proof of this proposition relies on the following statement.

Lemma 1. *Under the assumptions of Proposition 3, the functions* Φ *and* V *satisfy the following three conditions:*

(i) Φ *is continuous on* $[a, b]^N$, *where* a *and* b *are the minimum and maximum of* $\boldsymbol{u}^0$, *respectively.*

(ii) V *is continuous, bounded and piecewise differentiable.*

(iii) Let $V'_+(t)$ *denote the right-sided derivative of* V. *Then* $V(t) > 0$ *and* $V'_+(t) \leq 0$ *hold for all* $t \geq 0$, *with* $V'_+(t) = 0$ *only for discrete times* t, *unless the image* $\boldsymbol{f}$ *is flat.*

Proof of the Lemma. Continuity of Φ w.r.t. $\boldsymbol{u}$, and of $\boldsymbol{u}$ w.r.t. t, is straightforward, as (P_s) with the modified condition set (S1)–(S3), (S4a), (S4b), (S5a) is a dynamical system with bounded right-hand side. By composition, $V(t)$ is continuous, too. By the Lipschitz-continuity of $\boldsymbol{A}$ the right-hand side of the dynamical system is even continuous, ensuring continuous differentiability of $\boldsymbol{u}(t)$. Therefore V is also differentiable, except at those times t when the global maximum, or minimum, of $\boldsymbol{u}(t)$ is attained by two equal-valued pixels $u_i(t)$ and $u_j(t)$ with $\dot{u}_i(t) \neq \dot{u}_j(t)$, such that the global maximum or minimum property is

transferred from one to the other pixel at time t. As the number N of pixels is finite, this can happen only at discrete times t, and at each of these times the pixels representing the global maximum and minimum in the subsequent time interval establish a well-defined right-sided derivative of V. This completes the proof of (i).

Next we prove that for a non-trivial image $V'_+(t) = 0$ cannot stay true throughout a time interval $[t_1, t_2]$, $t_1 < t_2$.

For any pixel $u_i(t)$ representing the global maximum or minimum at time t, the weights $a_{i,j}(t)$ for all neighbours j of i are positive by hypothesis (S4b), while $a_{i,i}(t)$ is positive by (S4a) and (S3). Thus, $\dot{u}_{i,j} = 0$ can hold only if all neighbours of $u_{i,j}$ have the same grey value as $u_{i,j}$ at time t.

Assume $u_i(t)$ is a global maximum with $\dot{u}_i(t) = 0$ throughout $[t_1, t_2]$. Then all neighbour pixels u_j with $j \sim i$ satisfy $u_j = u_i$ throughout $[t_1, t_2]$, thus also $\dot{u}_j = 0$. As the image domain J is irreducible (connected by the neighbourhood relation) by (S5), this implies by recursive application that the image is a steady state. By reversibility of the dynamical system under consideration, this is possible only if $\boldsymbol{f}$ is trivial.

Thus, $V'_+(t)$ can vanish only at discrete time points. For all other times, $V'_+(t)$ is negative because of the negative matrix entries $a_{i,i}$ and nonnegative $a_{i,j}$ for $j \neq i$ at extrema $i \in J$.

Finally, the reversibility of the aforementioned dynamical system also ensures $V(t) > 0$ for all times if $\boldsymbol{f}$ is nontrivial. □

Proof of Proposition 3. Since the quantity $\Phi(\boldsymbol{u})$ is nonnegative, and vanishes exactly for the steady states $\boldsymbol{u}(t) = \boldsymbol{c}$, Lemma 1 characterises V as a Lyapunov function for semidiscrete FAB diffusion on each interval $[t_0, \infty)$ with $t_0 > 0$.

The convergence result (e) will follow now by slight adaptation of a standard proof from the theory of dynamical systems. Note that properties (ii) and (iii) from Lemma 1 but with the stronger inequality $V'(t) < 0$ for all t, are a standard argument about global asymptotic stability of dynamical systems; see e.g. [23, p. 127, Theorem 3]. However, the sole purpose of $V' < 0$ in the proof of the result is to ensure that $V(t)$ is strictly decreasing. The latter can equally be inferred from the slightly weaker property in Lemma 1(iii). With this modification, the proof from [23] carries over. □

4.3 A Nonstandard Space Discretisation for FAB Diffusion

While the preceding results are encouraging, and the standard pixel grid obviously induces a neighbourhood relation $\sim$ that satisfies condition (S5a), we have not yet shown that a suitable space-discretisation satisfies the modified requirements (S4a) and (S4b) at extrema. Unfortunately, this issue is a bit more delicate than one might assume: A standard discretisation of the diffusivity $g\big(|\boldsymbol{\nabla} u|^2\big)$ in some pixel (i, j) of a regular grid Γ is given by the central difference approximation

$$g_{i,j} := g\left(\left(\frac{u_{i+1,j} - u_{i-1,j}}{2h_1}\right)^2 + \left(\frac{u_{i,j+1} - u_{i,j-1}}{2h_2}\right)^2\right). \tag{19}$$

However, even if $\boldsymbol{u}$ has an extremum in (i, j), the approximation (19) of $|\boldsymbol{\nabla} u|^2$ may become positive – and not 0 as one would expect from the continuous theory. Since the FAB diffusivities only guarantee that $g(0) > 0$, it can happen that this finite difference approximation creates negative diffusivities in extrema and (S4a) is violated.

Fortunately there is an interesting alternative to the standard discretisation of the diffusivity on a regular grid that solves these problems immediately. As a prerequisite, let us first specify our requirements for FAB diffusivities more precisely.

Definition 1 (Admissible FAB diffusivity)**.** Let $g : \mathbb{R}_0^+ \to \mathbb{R}$ be a Lipschitz-continuous function. Assume that there are two constants $c_1 > c_2 > 0$ such that $g(0) = c_1$, and $g(z) > -c_2$ for all $z > 0$. Then g is called *admissible FAB diffusivity.*

Figure 1 illustrates the requirements of this definition.

Proposition 4 (Properties of Space-Discrete FAB Diffusion)**.** *Let an admissible FAB diffusivity g according to Definition 1 be given. Then the space discretisation* (10) *of FAB diffusion is well-posed, satisfies a maximum–minimum principle and average grey level invariance, if the diffusivity is evaluated by the nonstandard finite difference approximation*

$$g_{i,j} := g\left(\max\left(\frac{u_{i+1,j} - u_{i,j}}{h_1} \cdot \frac{u_{i,j} - u_{i-1,j}}{h_1}, 0\right) + \max\left(\frac{u_{i,j+1} - u_{i,j}}{h_2} \cdot \frac{u_{i,j} - u_{i,j-1}}{h_2}, 0\right)\right). \tag{20}$$

Under these conditions, the FAB evolution also satisfies the Lyapunov property with the Lyapunov function $V(t) = u_{\max}(t) - u_{\min}(t)$, and converges to the trivial steady state $\boldsymbol{u} = \boldsymbol{c}$ for $t \to \infty$.

It should be noted that this approximation has the same quadratic order of consistency as the previous one. However, it guarantees a vanishing discrete gradient approximation in extrema. Since according to the conditions of Definition 1 the diffusivity $g(0) = c_1$ at an extremum yields a positive average with any other value of the diffusivity g, (S4a) and (S4b) are guaranteed. Interestingly, the Lipschitz continuity of g and the property $g(0) > -\inf_{z>0} g(z)$ are the only requirements that are necessary to establish well-posedness, maximum–minimum principle, Lyapunov property and convergence to the trivial steady state for space-discrete FAB diffusion.

Nonstandard finite difference approximations that approximate nonlinear expressions by a combination of forward and backward differences have been advocated by Mickens [17] as an appropriate tool to design algorithms which capture essential physical properties of their underlying differential equations. Our paper confirms this.

Last but not least, our results are not restricted to the two-dimensional case: With a similar nonstandard approximation, it is straightforward to verify that space-discrete FAB diffusion is well-posed and satisfies an extremum principle on regular grids in any dimension.

5 Analysis of Fully Discrete FAB Diffusion in 2D

Combining the spatial discretisation from Section 4 with a forward difference discretisation in time, we obtain a simple explicit scheme for (1) with time step size τ and grid sizes h_1 and

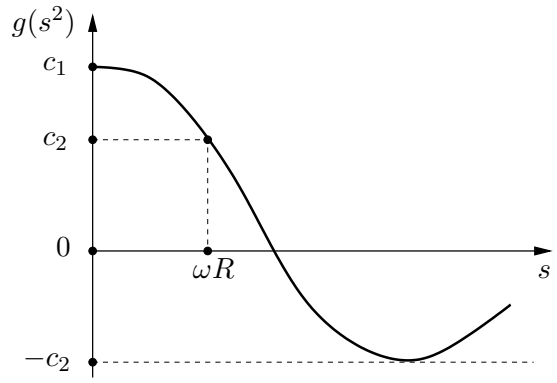


Figure 1: Schematic view of an admissible FAB diffusivity satisfying the conditions of Proposition 5.

h_2 in x- and y-direction:

$$u_{i,j}^{k+1} = u_{i,j}^k + \tau \cdot \left(\frac{g_{i+1,j}^k + g_{i,j}^k}{2} \cdot \frac{u_{i+1,j}^k - u_{i,j}^k}{h_1^2} - \frac{g_{i,j}^k + g_{i-1,j}^k}{2} \cdot \frac{u_{i,j}^k - u_{i-1,j}^k}{h_1^2} \right.$$
$$\left. + \frac{g_{i,j+1}^k + g_{i,j}^k}{2} \cdot \frac{u_{i,j+1}^k - u_{i,j}^k}{h_2^2} - \frac{g_{i,j}^k + g_{i,j-1}^k}{2} \cdot \frac{u_{i,j}^k - u_{i,j-1}^k}{h_2^2} \right) . \tag{21}$$

Here, $u_{i,j}^k$ approximates u at the grid location of $(i,j) \in \Gamma$ and time $k\tau$. Furthermore, we use the nonstandard approximation (20) for the admissible FAB diffusivity, computing diffusivities $g_{i,j}^k$ in all pixels in time step k from the values $u_{i,j}^k$ in the same time step.

Our requirements to the discrete FAB diffusion problem and the diffusivity function are summarised in the following definition.

Definition 2 (Discrete FAB evolution)**.** Let an initial 2D image $\boldsymbol{f} = (f_{i,j})$ on $\Gamma = \{1, 2, \ldots, m\} \times \{1, 2, \ldots, n\}$ be given. Let the grey values $f_{i,j}$ be restricted to a finite interval $[a, b]$ of length $R := b - a$.

The *discrete FAB evolution* is the sequence of images $\boldsymbol{u}^k = (u_{i,j}^k)$ that evolves according to (21), (20) with the initial condition $\boldsymbol{u}^0 = \boldsymbol{f}$, mirroring boundary conditions, and an admissible FAB diffusivity function g as specified in Definition 1.

The *stabilisation range constant* of g is the largest constant $\omega = \omega(g) > 0$ such that $g(s^2) > c_2$ holds for all s with $0 < s < \omega R$ (such an ω exists due to the continuity of g; compare Figure 1).

5.1 Maximum–Minimum Principle

Our first result is a 2D analogue for the first statement of [35, Prop. 4], i.e. the maximum–minimum principle. Our bound for τ is adapted to the 2D grid geometry.

Proposition 5 (Maximum–Minimum Principle for Discrete FAB Diffusion)**.** *Consider the discrete FAB evolution as specified in Definition 2. Let the time step τ satisfy the inequality*

$$\tau \leq \vartheta := \frac{\omega^2 h_1^4 h_2^4}{2c_1(h_1^2 + h_2^2)(\omega^2 h_1^2 h_2^2 + h_1^2 + h_2^2)} . \tag{22}$$

Then $\boldsymbol{u}$ obeys the following maximum–minimum principle: If the initial signal is bounded by $f_{i,j} \in [a,b]$ for all $(i,j) \in \Gamma$, then $u_{i,j}^k \in [a,b]$ for all $(i,j) \in \Gamma$, $k \geq 0$.

The proof of this statement relies on two local properties.

Lemma 2. *Under the assumptions of Proposition 5, local maxima do not increase.*

Proof. Let $u_{i,j}^k$ be a local maximum of $\boldsymbol{u}^k$. Then $g_{i,j}^k = c_1$, and we have that $g_{i+1,j}^k + g_{i,j}^k$ etc. are positive while $u_{i+1,j}^k - u_{i,j}^k$ etc. are negative such that all summands in the bracket on the r.h.s. of (21) are negative. This holds independent of τ.

The r.h.s. of (21) is a convex combination of $u_{i,j}^k$, $u_{i\pm1,j}^k$, $u_{i,j\pm1}^k$ if

$$1 - \frac{\tau}{2h_1^2}\left(g_{i+1,j}^k + 2g_{i,j}^k + g_{i-1,j}^k\right) - \frac{\tau}{1h_2^2}\left(g_{i,j+1}^k + 2g_{i,j}^k + g_{i,j-1}^k\right) \geq 0 \;, \tag{23}$$

which is certainly the case if

$$\tau \leq \frac{1}{2\,c_1\left(\frac{1}{h_1^2} + \frac{1}{h_2^2}\right)} \;. \tag{24}$$

Note that this is the well-known time step limit for the standard explicit scheme in the case of nonnegative diffusivity. The bound in (24) is greater than ϑ in Proposition 5 since

$$\begin{aligned} \frac{1}{2\,c_1\left(\frac{1}{h_1^2} + \frac{1}{h_2^2}\right)} &= \frac{\omega^2 h_1^4 h_2^4}{2\,c_1(h_1^2+h_2^2)\omega^2 h_1^2 h_2^2} \\ &\geq \frac{\omega^2 h_1^4 h_2^4}{2\,c_1(h_1^2+h_2^2)\omega^2 h_1^2 h_2^2 + 2\,c_1(h_1^2+h_2^2)^2} = \vartheta \;. \end{aligned} \tag{25}$$

This completes the proof. □

Nonenhancement of local extrema is a frequently used scale-space property for linear and nonlinear diffusion processes; see e.g. [2, 14, 33]. However, in the case of FAB diffusion, we need one more property to prove a maximum–minimum principle:

Lemma 3. *Within one time step of the discrete FAB evolution from Definition 2 with the time step size τ bounded by (22), the value of a pixel which is not a local maximum cannot grow larger than the greatest value among its neighbouring pixels before that time step.*

Proof. Let $u_{i,j}^k$ be the non-maximal pixel under consideration. Assume first that the largest grey value among its neighbours is attained by a horizontal neighbour, say $u_{i-1,j}^k$. So $u_{i-1,j}^k$ is greater than $u_{i,j}^k$ and not less than each other neighbour of $u_{i,j}^k$.

To outline the proof first, notice that basically two situations can happen: If $u_{i,j}^k$ is only slightly smaller than $u_{i-1,j}^k$, it turns out that $u_{i,j}^k$ is "not far from maximality". In this case, its diffusivity $g_{i,j}^k$ will be positive and large enough to ensure forward diffusion (Case 1 in the following), such that again the time step limit of explicit forward diffusion applies. Otherwise, negative diffusivity may occur, but at the same time there is quite some way to go before $u_{i,j}^{k+1}$ could exceed $u_{i-1,j}^k$. The overall bounds on the image contrast limit the diffusion flow and thereby the "speed" of the pixel. So one can state also in this case a time step size limit

that ensures the desired inequality (Case 2). In the following, these two cases are treated exactly.

Case 1: Assume that one has

$$\frac{1}{h_1^2}(u_{i,j}^k - u_{i-1,j}^k)(u_{i+1,j}^k - u_{i,j}^k) + \frac{1}{h_2^2}(u_{i,j}^k - u_{i,j-1}^k)(u_{i,j+1}^k - u_{i,j}^k) \leq \omega^2 R^2 \ . \tag{26}$$

Then $g_{i,j}^k \geq c_2$, and thus $g_{i,j}^k + g_{i+1,j}^k$, $g_{i,j}^k + g_{i,j\pm1}^k \geq 0$. As a consequence, $u_{i,j}^{k+1}$ is a convex combination of $u_{i,j}^k$, $u_{i\pm1,j}^k$, $u_{i,j\pm1}^k$ (and thereby certainly not greater than the maximum of these) if

$$1 - \frac{\tau}{2h_1^2}(g_{i+1,j}^k + 2g_{i,j}^k + g_{i-1,j}^k) - \frac{\tau}{2h_2^2}(g_{i,j+1}^k + 2g_{i,j}^k + g_{i,j-1}^k) \geq 0 \ , \tag{27}$$

which is certainly fulfilled if

$$1 - \frac{4\tau c_1}{2h_1^2} - \frac{4\tau c_1}{2h_2^2} \geq 0 \ , \tag{28}$$

i.e. (24). As pointed out in the proof of Lemma 2, the time step bound ϑ from (22) satisfies this condition which completes the proof for the present case.

Case 2: Let

$$\frac{1}{h_1^2}(u_{i,j}^k - u_{i-1,j}^k)(u_{i+1,j}^k - u_{i,j}^k) + \frac{1}{h_2^2}(u_{i,j}^k - u_{i,j-1}^k)(u_{i,j+1}^k - u_{i,j}^k) > \omega^2 R^2 \ . \tag{29}$$

We prove first that the difference between pixel $u_{i,j}^k$ and its greatest neighbour $u_{i-1,j}^k$ fulfils the inequality

$$u_{i-1,j}^k - u_{i,j}^k > \frac{\omega^2 R}{\frac{1}{h_1^2} + \frac{1}{h_2^2}} \ . \tag{30}$$

To this end, assume the contrary

$$u_{i-1,j}^k - u_{i,j}^k \leq \frac{\omega^2 R}{\frac{1}{h_1^2} + \frac{1}{h_2^2}} \ . \tag{31}$$

Since $|u_{i+1,j}^k - u_{i,j}^k| \leq R$ by Definition 2, we have then

$$\frac{1}{h_1^2}(u_{i,j}^k - u_{i-1,j}^k)(u_{i+1,j}^k - u_{i,j}^k) \leq \frac{\omega^2 R^2 h_2^2}{h_1^2 + h_2^2} \ . \tag{32}$$

Assume first that $u_{i,j+1}^k \geq u_{i,j}^k \geq u_{i,j-1}^k$. By (31), the maximality of $u_{i-1,j}^k$ within the neighbourhood of $u_{i,j}^k$ implies

$$u_{i,j+1}^k - u_{i,j}^k \leq \frac{\omega^2 R}{\frac{1}{h_1^2} + \frac{1}{h_2^2}} \ , \tag{33}$$

which together with $u^k_{i,j} - u^k_{i,j-1} \le R$ yields

$$\frac{1}{h_1^2}(u^k_{i,j} - u^k_{i,j-1})(u^k_{i,j+1} - u^k_{i,j}) \le \frac{\omega^2 R^2 h_1^2}{h_1^2 + h_2^2} \,, \tag{34}$$

The same estimate holds also if $u^k_{i,j+1} \le u^k_{i,j} \le u^k_{i,j-1}$. In the remaining cases, $u^k_{i,j+1} < u^k_{i,j}$ or $u^k_{i,j-1} > u^k_{i,j}$, the product $(u^k_{i,j} - u^k_{i,j-1})(u^k_{i,j+1} - u^k_{i,j})$ is negative such that (34) holds in all cases.

As adding (32) and (34) yields the hypothesis of Case 1, we see that (31) contradicts the condition of Case 2, which completes the proof of (30).

Further, we have by the hypothesis of Case 2 that $-c_2 \le g^k_{i,j} \le c_2$. Together with the hypotheses from Proposition 5 on the range of g and the image range, one has

$$\begin{aligned}
-c_2 &\le \frac{g^k_{i+1,j} + g^k_{i,j}}{2} \le \frac{c_1 + c_2}{2} \,, & -\frac{R}{h_1^2} &\le \frac{u^k_{i+1,j} - u^k_{i,j}}{h_1^2} \le \frac{R}{h_1^2} \,, \\
-c_2 &\le \frac{g^k_{i-1,j} + g^k_{i,j}}{2} \le \frac{c_1 + c_2}{2} \,, & 0 &\le \frac{u^k_{i-1,j} - u^k_{i,j}}{h_1^2} \le \frac{R}{h_1^2} \,, \\
-c_2 &< \frac{g^k_{i,j\pm1} + g^k_{i,j}}{2} \le \frac{c_1 + c_2}{2} \,, & -\frac{R}{h_2^2} &\le \frac{u^k_{i,j\pm1} - u^k_{i,j}}{h_2^2} \le \frac{u^k_{i-1,j} - u^k_{i,j}}{h_2^2} \,.
\end{aligned} \tag{35}$$

Inserting these into (21) gives

$$u^{k+1}_{i,j} \le u^k_{i,j} + \tau\left(\frac{c_1 + c_2}{2h_1^2}R + \frac{c_1 + c_2}{2h_1^2}(u^k_{i-1,j} - u^k_{i,j}) + 2\,\frac{c_1 + c_2}{2h_2^2}R\right) , \tag{36}$$

$$u^k_{i-1,j} - u^{k+1}_{i,j} \ge (u^k_{i-1,j} - u^k_{i,j})\left(1 - \frac{\tau(c_1 + c_2)}{2h_1^2}\right) - \frac{\tau(c_1 + c_2)R}{2}\left(\frac{1}{h_1^2} + \frac{2}{h_2^2}\right) . \tag{37}$$

The r.h.s. of (37) is certainly nonnegative if

$$\tau \le \frac{2h_1^2}{c_1 + c_2} \cdot \frac{u^k_{i-1,j} - u^k_{i,j}}{u^k_{i-1,j} - u^k_{i,j} + R\left(1 + 2\,h_1^2/h_2^2\right)} \,, \tag{38}$$

for which by our initial estimate for $u^k_{i-1,j} - u^k_{i,j}$ and monotonicity it suffices that

$$\tau \le \frac{2\omega^2 h_1^4 h_2^4}{(c_1 + c_2)\big(\omega^2 h_1^2 h_2^4 + (h_2^2 + 2h_1^2)(h_1^2 + h_2^2)\big)} \,. \tag{39}$$

The bound from (39) is greater than ϑ from (22). To see this, notice that

$$c_1 + c_2 \le 2c_1 \,, \tag{40}$$

$$\omega^2 h_1^2 h_2^4 \le 2\,\omega^2 h_1^2 h_2^2 (h_1^2 + h_2^2) \,, \tag{41}$$

$$(h_2^2 + 2h_1^2)(h_1^2 + h_2^2) \le 2(h_1^2 + h_2^2)^2 \,. \tag{42}$$

Inserting these into the r. h. s. of (39) yields

$$\begin{aligned}
&\frac{2\omega^2 h_1^4 h_2^4}{(c_1 + c_2)\big(\omega^2 h_1^2 h_2^4 + (h_2^2 + 2h_1^2)(h_1^2 + h_2^2)\big)} \\
&\quad \le \frac{2\omega^2 h_1^4 h_2^4}{2c_1(h_1^2 + h_2^2)\big(2\omega^2 h_1^2 h_2^2 + 2(h_1^2 + h_2^2)\big)} = \vartheta \,.
\end{aligned} \tag{43}$$

As a result, for $\tau \leq \vartheta$ as required by (22) one has $u_{i,j}^{k+1} \leq u_{i-1,j}^{k}$. This concludes the proof in Case 2.

So far, the results of both Case 1 and Case 2 were based on the assumption that the greatest neighbour of the pixel under consideration is a horizontal neighbour. If $u_{i,j}^{k}$ has its greatest neighbour in vertical direction, analogous considerations lead to versions of the estimates (28) and (39) with h_1 and h_2 exchanged, from which the sufficiency of the bound (22) can be established in the same way as before. This concludes the proof. □

Proof of Proposition 5. The preservation of the global maximum of $\boldsymbol{u}$ follows immediately from Lemmas 2 and 3: Let U be the global maximum of $\boldsymbol{u}^k$, i.e. $u_{i,j}^{k} \leq U$ for all $(i,j) \in \Gamma$. We prove that $u_{i,j}^{k+1} \leq U$, too, holds for all $(i,j) \in \Gamma$.

If $u_{i,j}^{k}$ is a local maximum, Lemma 2 implies $u_{i,j}^{k+1} \leq u_{i,j}^{k} \leq U$.

If $u_{i,j}^{k}$ is not a local maximum, Lemma 3 ensures that $u_{i,j}^{k+1} \leq \max\{u_{i\pm1,j}^{k}, u_{i,j\pm1}^{k}\} \leq U$.

Analogous statements for minima arise by considering the evolution of the inverted image $-\boldsymbol{f}$. Thus, the maximum–minimum principle of the Proposition has been proven. □

Remark 2. Unlike in the 1D case [35, Prop. 4], the statement that an extremum may not split into two does not hold. Similar to ordinary homogeneous diffusion, a "dumbbell" configuration with a narrow ridge between two more extended plateaus can serve as a counterexample; see e.g. [14]. Note that for sufficiently small grey value differences between the ridge and the adjacent plateaus all diffusivities in this region are positive.

5.2 Strict Lyapunov Condition

The maximum–minimum principle from Proposition 5 suggests the use of the difference between global maximum and global minimum of the image as a Lyapunov function to investigate the possible convergence of discrete FAB diffusion. Our proofs from the previous subsection, however, still leave the possibility that the global maximum and minimum of the image stay constant, and different from each other, forever.

To rule out this possibility, we will refine our analysis and construct a strictly decreasing Lyapunov function by incorporating multiplicities of maxima and minima as additional information. From now on, we require that τ is strictly smaller than the bound (22) from Proposition 5. We introduce notations for the global extremal grey values of images with their multiplicities, and an ordering for pairs of values with multiplicities.

Definition 3. For any image $\boldsymbol{u} = (u_{i,j})_{(i,j)\in\Gamma}$, let $u_{\max} := \max_{i,j} u_{i,j}$, $u_{\min} := \min_{i,j} u_{i,j}$ denote its maximal and minimal grey value, respectively, and $n_{\max} := \#\{(i,j) \mid u_{i,j} = u_{\max}\}$, $n_{\min} := \#\{(i,j) \mid u_{i,j} = u_{\min}\}$ their multiplicities.

Definition 4. Let the relation $\prec$ on $\mathbb{R} \times \mathbb{N}$ be given by

$$(u_1, n_1) \prec (u_2, n_2) \quad :\Longleftrightarrow \quad (u_1 < u_2) \;\text{ or }\; (u_1 = u_2 \;\text{ and }\; n_1 < n_2)\,. \tag{44}$$

Clearly, $\prec$ is a strict total order. We can now establish the maximum–minimum difference with multiplicities as Lyapunov function for discrete FAB diffusion.

Proposition 6 (Lyapunov Function for Discrete FAB Diffusion)**.** *Consider the fully discrete FAB evolution from Definition 2. If the time step size is chosen as $\tau < \vartheta$ with ϑ as in Proposition 5, then*

$$(u_{\max}^{k+1} - u_{\min}^{k+1},\ n_{\max}^{k+1} + n_{\min}^{k+1}) \prec (u_{\max}^{k} - u_{\min}^{k},\ n_{\max}^{k} + n_{\min}^{k}) \tag{45}$$

holds, unless $u_{\max}^k = u_{\min}^k$.

Proof. Let $u_{i,j}^k$ be a local maximum of $\boldsymbol{u}^k$. As in the proof of Lemma 2, we have $g_{i,j}^k = c_1$. Thus, the new value $u_{i,j}^{k+1}$ of that pixel will be a convex combination of the old grey values of pixel (i,j) and its neighbours, with all neighbours having positive weights. Therefore, $u_{i,j}^{k+1} = u_{i,j}^k$ can happen only if all neighbours have the same value as $u_{i,j}$ in time step k. As long as not all pixels of the image have the same value, there will be at least one pixel $u_{i,j}^k = u_{\max}^k$ with a neighbour of smaller grey value.

Following the proof of Lemma 3 we see that for $\tau < \vartheta$ the new pixel value $u_{i,j}^{k+1}$ remains strictly below the old value of its largest neighbour $u_{i-1,j}^k$. Thus, $u_{i,j}^{k+1} = u_{\max}^k$ cannot hold for a pixel with $u_{i,j}^k < u_{\max}^k$.

Combining both arguments, we see that the number of pixels attaining the value $u_{\max}^k$ decreases in time step $k+1$. As a consequence, one has $u_{\max}^{k+1} < u_{\max}^k$ (if no pixel with that value remains), or $n_{\max}^{k+1} < n_{\max}^k$ (if the maximal value remains equal).

Analogous reasoning for minima completes the proof. □

5.3 Convergence to a Flat Steady State

An immediate consequence of Proposition 6 is the following statement.

Corollary 1 (Steady States of Discrete FAB Diffusion)**.** *The only fixed points of the discrete FAB diffusion process* (21), (20) *are the flat images given for each $\hat{u} \in \mathbb{R}$ by*

$$u_{i,j} = \hat{u} \quad \text{for all } (i,j) \in \Gamma\ . \tag{46}$$

By average grey value invariance, the only steady state that could be reached from a given initial image $\boldsymbol{f}$ is that for which $\hat{u}$ equals the average grey value of $\boldsymbol{f}$, i.e. $\hat{u} = \mu = \frac{1}{N} \sum_{(i,j)\in\Gamma} f_{i,j}$. We will now prove convergence to this steady state.

Proposition 7 (Convergence of Discrete FAB Diffusion)**.** *Fully discrete FAB diffusion* (21), (20) *with $\boldsymbol{u}^0 = \boldsymbol{f}$ and time step size $\tau < \vartheta$ converges to the fixed point* (46) *where $\hat{u} = \mu$ is the average grey value of $\boldsymbol{f}$.*

Proof. Consider the strictly decreasing (w.r.t. $\prec$) sequences $\big((u_{\max}^k, n_{\max}^k)\big)_{k\in\mathbb{N}}$ and $\big((-u_{\min}^k, n_{\min}^k)\big)_{k\in\mathbb{N}}$ from Proposition 6. These sequences are bounded from below by $(f_{\min}, N)$ and $(-f_{\max}, N)$, respectively. By an easy adaptation of the standard argument for sequences in $\mathbb{R}$ to sequences in $\mathbb{R}\times\mathbb{N}$ it follows that the sequences $(u_{\max}^k)$, $(u_{\min}^k)$ converge. Denote by $\bar{u}$, $\underline{u}$ their respective limits.

Assume $\underline{u} < \bar{u}$. By the maximum–minimum principle, $\boldsymbol{u}^k \in [a,b]^N$ holds for all k. Since $[a,b]^N$ is compact, the sequence $(\boldsymbol{u}^k)$ has an accumulation point. Because of the monotonicity of $(u_{\max}^k)$, $(u_{\min}^k)$ each accumulation point satisfies $u_{\max}^* = \bar{u}$, $u_{\min}^* = \underline{u}$.

We choose one accumulation point $\boldsymbol{u}^*$ and consider the FAB evolution $(\tilde{\boldsymbol{u}}^k)_{k\in\mathbb{N}_0}$ with initial condition $\tilde{\boldsymbol{u}}^0 = \boldsymbol{u}^*$. By Proposition 6, there exists a natural number K such that $\tilde{u}^K_{\max} = \max \tilde{\boldsymbol{u}} < \bar{u}$. Let therefore $\delta := \bar{u} - \tilde{u}^K_{\max} > 0$.

Moreover, the evolution (21), (20) satisfies a Lipschitz condition on $[a,b]^N$ with respect to the maximum norm $\|\cdot\|$, i.e.

$$\|u^k - \hat{u}^k\| < B \quad \Rightarrow \quad \|u^{k+1} - \hat{u}^{k+1}\| < LB \tag{47}$$

with some Lipschitz constant $L > 0$. Since $\boldsymbol{u}^*$ is an accumulation point of $(\boldsymbol{u}^k)$, we can choose k such that $\|\boldsymbol{u}^k - \boldsymbol{u}^*\| < \delta/L^K$. Consequently, $\|\boldsymbol{u}^{k+K} - \tilde{\boldsymbol{u}}^K\| < \delta$, and by the triangle inequality it follows that

$$u^{k+K}_{\max} \;\le\; \tilde{u}^K_{\max} + \|\boldsymbol{u}^{k+K} - \tilde{\boldsymbol{u}}^K\| \;<\; \bar{u} - \delta + \delta = \bar{u}\ , \tag{48}$$

contradicting the convergence of $(u^k_{\max})$ to $\bar{u}$. Thus, our assumption $\underline{u} < \bar{u}$ must be wrong, and we have $\underline{u} = \bar{u}$, i.e. convergence to a flat steady state. □

6 Efficient Numerics for FAB Diffusion

Based on the stability result from Proposition 5 it is possible to compute FAB diffusion by a stable explicit scheme. The limit (22) on the time step size, however, leads to time step sizes that are much too small for most practical application.

In fact, (22) is an *a priori* estimate for the time step size which is obtained by the combination of several worst-case estimates in the proofs in Subsection 5.1. It is very common for this kind of estimates that the worst-case estimates involved will in general be reached only at a few pixel locations within a larger image, and also rarely apply simultaneously. Indeed, in practical computation one notices that even with drastically larger time step sizes, violations of the maximum–minimum principle occur only in some time steps, affect only a few pixels, and the bounds are exceeded only by small amounts.

This motivates us to devise explicit schemes for FAB diffusion with adaptive step-size control based on *a posteriori* estimates of the time step size.

6.1 A Stable Explicit Scheme with Global Step Size Adaptation

We start by equipping the explicit scheme (21), (20) with a step-size control that provides in each iteration for a global time step τ which prevents violations of stability. Our theory from Section 5 already contains the blueprint for such a step-size control, as the criteria formulated in Lemma 2 and Lemma 3 can immediately be used to obtain the desired a posteriori estimate.

To this end, notice that our explicit scheme (21) can be written as $u^{k+1}_{i,j} = u^k_{i,j} + \tau\, \dot{u}^k_{i,j}$ where the finite-difference approximation $\dot{u}^k_{i,j}$ of $\operatorname{div}(g\,\boldsymbol{\nabla} u)$ at pixel (i,j) in time step k can be precomputed via the diffusivities (20) independent of the time step size τ.

Our first criterion, non-enhancement of extrema, is stated in Lemma 2. To ensure this criterion, the time step size bound from Lemma 2 can be applied directly. This time step size limit is the same as for standard forward diffusion processes and does not constitute a limiting factor here.

The more severe limitation comes from the second criterion, local monotonicity preservation, see Lemma 3. It requires that a non-extremal pixel must not become larger than its

largest neighbour, or smaller than its smallest neighbour, within one time step. Instead of using the restrictive a priori time step limit from Lemma 3, local monotonicity preservation can be enforced directly in each time step of an explicit scheme. First one computes the diffusivities (20) and the values $\dot{u}_{i,j}^k$ for all pixels. Then one checks for each non-maximal pixel $u_{i,j}^k$ and its maximal neighbour $u_{i',j'}^k$ whether the condition $u_{i,j}^{k+1} \leq u_{i',j'}^{k+1}$ could be violated for large time step sizes τ, and limits the time step size accordingly. An analogous check is performed for non-minimal pixels with regard to their minimal neighbours. The global time step size is then set to the most restrictive bound among all pixels.

Unfortunately, the classification of pixels as non-maximal or non-minimal is sensitive to numerical roundoff errors. As a result, it can happen that for neighbouring pixels which are almost equal and actually represent the same discrete local extremum, a local monotonicity violation is anticipated, and a very small spurious time step limit is computed. To overcome this problem, the a priori estimate from Proposition 5 itself is used: Whenever a time step bound less than that of (22) is computed, it is discarded because we know that no local monotonicity violation can occur up to (22).

In Table 1 we formulate the complete algorithm for one explicit time step $\boldsymbol{u}^k \to \boldsymbol{u}^{k+1}$ with global step-size adaptation. Its inputs are the image $\boldsymbol{u}^k$, the upper time step limit $\tau_{\max}$ computed from (24), and the lower time step limit $\tau_{\min} := \vartheta$ from (22).

6.2 Parallel GPU Implementation

Explicit schemes for PDEs of image processing are often excellent candidates for highly parallel computation, due to the fact that computations within each iteration are structured in few steps, each of which can be performed independently for each pixel. This is also the case for our explicit scheme for FAB diffusion.

We describe here a parallel implementation suitable for implementation on a programmable GPU.

Steps 1., 2., and 4. of the algorithm presented in the previous subsection can immediately be parallelised by assigning pixels to different threads. For example, an implementation in the CUDA framework can perform each of these steps by a kernel that is launched in a grid of parallel threads.

The step that requires closer consideration is Step 3, i.e. the adaptive step size control. The sequential formulation of the previous subsection cannot be implemented efficiently in a parallel framework. Instead, the step-size limits for all pixels are computed independently. The minimum of all these step sizes is then determined by a dyadic cascade of aggregation steps, a standard strategy for computing commutative and associative aggregation operations across a grid in parallel computing with logarithmic time complexity. The cascade proceeds by a sequence of steps. In each step of the cascade, the number of threads is halved. Those threads which continue aggregate two values each by a minimum operation, then all threads are synchronised.

As a result, Step 3 of the sequential algorithm is replaced with the Steps 3a and 3b given in Table 2.

The so modified Step 3 is suitable for parallelisation. In a practical CUDA implementation, the dyadic cascade in Step 3b will actually be carried out within each block of synchronous

Table 1: Time step of explicit scheme for FAB diffusion with global adaptive step size control

Input: $\boldsymbol{u}^k$, $\tau_{\max}$, $\tau_{\min}$

1. **Diffusivity computation:** For each pixel (i,j), compute the diffusivity $g_{i,j}$ according to (20):

$$g_{i,j}^k = g\left(\max\left(\frac{u_{i+1,j}^k - u_{i,j}^k}{h_1} \cdot \frac{u_{i,j}^k - u_{i-1,j}^k}{h_1}, 0\right) + \max\left(\frac{u_{i,j+1}^k - u_{i,j}^k}{h_2} \cdot \frac{u_{i,j}^k - u_{i,j-1}^k}{h_2}, 0\right)\right).$$

2. **Flow computation:** For each pixel (i,j), compute the right-hand side $\dot{u}_{i,j}^k$ via (21):

$$\dot{u}_{i,j}^k = \frac{g_{i+1,j}^k + g_{i,j}^k}{2} \cdot \frac{u_{i+1,j}^k - u_{i,j}^k}{h_1^2} - \frac{g_{i,j}^k + g_{i-1,j}^k}{2} \cdot \frac{u_{i,j}^k - u_{i-1,j}^k}{h_1^2} + \frac{g_{i,j+1}^k + g_{i,j}^k}{2} \cdot \frac{u_{i,j+1}^k - u_{i,j}^k}{h_2^2} - \frac{g_{i,j}^k + g_{i,j-1}^k}{2} \cdot \frac{u_{i,j}^k - u_{i,j-1}^k}{h_2^2}.$$

3. **Step-size determination:** Let $\tau := \tau_{\max}$.

   For each pixel (i,j),

   - If (i,j) is not a discrete local maximum, let (i',j') be its maximal neighbour.

     If $u_{i,j}^k + \tau\, \dot{u}_{i,j}^k > u_{i',j'}^k + \tau\, \dot{u}_{i',j'}^k$, set $\tau^* := -\dfrac{u_{i',j'}^k - u_{i,j}^k}{\dot{u}_{i',j'}^k - \dot{u}_{i,j}^k}$.

     If $\tau^* \geq \tau_{\min}$, let $\tau = \tau^*$.
   - If (i,j) is not a discrete local minimum, let (i',j') be its minimal neighbour.

     If $u_{i,j}^k + \tau\, \dot{u}_{i,j}^k < u_{i',j'}^k + \tau\, \dot{u}_{i',j'}^k$, set $\tau^* := -\dfrac{u_{i',j'}^k - u_{i,j}^k}{\dot{u}_{i',j'}^k - \dot{u}_{i,j}^k}$.

     If $\tau^* \geq \tau_{\min}$, let $\tau = \tau^*$.

4. **Global update:** For each pixel (i,j), compute $u_{i,j}^{k+1} := u_{i,j}^k + \tau\, \dot{u}_{i,j}^k$.

Table 2: Modification for GPU implementation of the time step from Table 1

3a. **Pixel-wise step-size determination:** For each pixel (i,j),

- Let $\tau_{i,j} := \tau_{\max}$.
- If (i,j) is not a discrete local maximum, let (i',j') be its maximal neighbour.
  If $u^k_{i,j} + \tau_{i,j}\,\dot{u}^k_{i,j} > u^k_{i',j'} + \tau_{i,j}\,\dot{u}^k_{i',j'}$, set $\tau^* := -\dfrac{u^k_{i',j'} - u^k_{i,j}}{\dot{u}^k_{i',j'} - \dot{u}^k_{i,j}}$.
  If $\tau^* \geq \tau_{\min}$, let $\tau_{i,j} = \tau^*$.
- If (i,j) is not a discrete local minimum, let (i',j') be its minimal neighbour.
  If $u^k_{i,j} + \tau_{i,j}\,\dot{u}^k_{i,j} < u^k_{i',j'} + \tau_{i,j}\,\dot{u}^k_{i',j'}$, set $\tau^* := -\dfrac{u^k_{i',j'} - u^k_{i,j}}{\dot{u}^k_{i',j'} - \dot{u}^k_{i,j}}$.
  If $\tau^* \geq \tau_{\min}$, let $\tau_{i,j} = \tau^*$.

3b. **Global step-size determination:** Compute $\tau = \min\limits_{i,j} \tau_{i,j}$ by a dyadic cascade of minimum operations.

threads. The final aggregation of step sizes across all blocks involves a comparatively small number of values and is usually implemented sequentially.

6.3 A Randomised Two-Pixel Scheme with Local Step-Size Adaptation

In Subsection 6.1 we have used our theoretical findings to obtain a *global* time step τ in each iteration of the otherwise unchanged explicit scheme (21), (20). We obtain a higher efficiency by following a radically *localised* approach: The diffusion flow $\dot{u}^k_{i,j}$ in (21) is composed of four two-pixel flows, each between the location (i,j) and one of its neighbours. Each of these flows appears with opposite signs in the flows of its two participating pixels, which ensures that the average grey value conservation property of the continuous diffusion process is exactly fulfilled also in its discretisation (21). In order to maintain this conservation property, we choose the two-pixel flows as the elementary units for our scheme; compare also the proceeding in [4, 28] and related ideas based on four-pixel interactions [36].

Starting from an approximation $\boldsymbol{u}^k$ pertaining to evolution time $k\tau_{\max}$, a global time step of size $\tau_{\max}$ is carried out by updating two-pixel flows in random order, each one with an appropriate time step. In the case of forward diffusion the time step size is chosen so that the two interacting pixels preserve their monotonicity order. In the case of backward diffusion it is chosen such that they are prevented from growing above the maximum, or decreasing below the minimum of their respective neighbourhoods. Updated values $u_{i,j}$ enter immediately the computation of other pixels. This is repeated until all flows have reached the new time level, yielding the new approximation $\boldsymbol{u}^{k+1}$.

The time step $k \mapsto k+1$ is summarised in Table 3. It starts by initialising an evolution time account for each pair of neighbouring pixels with $\tau_{\max}$. Then pixel pairs are randomly selected and updated until all time accounts reach zero, indicating that we have progressed from sync time $k\tau_{\max}$ to $(k+1)\tau_{\max}$.

Table 3: Time step of randomised two-pixel scheme for FAB diffusion with locally adaptive step size control

Input: $\boldsymbol{u}^k$, $\tau_{\max}$

Initialisation: For each pair of neighbouring pixels, $\{(i,j),(i',j')\}$ with $(i',j') = (i+1,j)$ or $(i',j') = (i,j+1)$, let $T_{i,j;i',j'} := \tau_{\max}$.

Asynchronous update: Repeat the following steps 1–5 until all $T_{i,j;i',j'}$ are zero.

1. **Random selection:** Select a pixel pair $\{(i,j),(i',j')\}$ randomly, with the probability of each pair to be selected being proportional to $T_{i,j;i',j'}$.

2. **Diffusivity computation:** Compute $g_{i,j}$ and $g_{i',j'}$ as in (20); let $g := \frac{1}{2}(g_{i,j} + g_{i',j'})$.

3. **Flow computation:** Compute the flow $\dot{u} := g \cdot (u_{i',j'} - u_{i,j})/h^2$, using $h = h_1$ for horizontal or $h = h_2$ for vertical neighbours.

4. **Step-size determination:** Let $\tau^* := T_{i,j;i',j'}$.

   - If $g > 0$ and $\tau^* > \dfrac{h^2}{2g}$, reduce τ^* to $\dfrac{h^2}{2g}$.
   - If $g < 0$ and (i,j) is not a discrete local maximum, let (i^*,j^*) be its maximal neighbour.

     If $u_{i,j} + \tau^*\dot{u} > u_{i^*,j^*}$, reduce τ^* to $\tau^* := \dfrac{u_{i^*,j^*} - u_{i,j}}{\dot{u}}$.
   - If $g < 0$ and (i,j) is not a discrete local minimum, let (i^*,j^*) be its minimal neighbour.

     If $u_{i,j} + \tau^*\dot{u} < u_{i^*,j^*}$, reduce τ^* to $\tau^* := \dfrac{u_{i^*,j^*} - u_{i,j}}{\dot{u}}$.
   - If $g < 0$ and (i',j') is not a discrete local maximum, let (i^*,j^*) be its maximal neighbour.

     If $u_{i',j'} - \tau^*\dot{u} > u_{i^*,j^*}$, reduce τ^* to $\tau^* := \dfrac{u_{i^*,j^*} - u_{i',j'}}{-\dot{u}}$.
   - If $g < 0$ and (i',j') is not a discrete local minimum, let (i^*,j^*) be its minimal neighbour.

     If $u_{i',j'} - \tau^*\dot{u} < u_{i^*,j^*}$, reduce τ^* to $\tau^* := \dfrac{u_{i^*,j^*} - u_{i',j'}}{-\dot{u}}$.

5. **Two-pixel flow update:** Update $u_{i,j}$ and $u_{i',j'}$ by replacing them with the new values $\tilde{u}_{i,j} := u_{i,j} + \tau^*\dot{u}$ and $\tilde{u}_{i',j'} := u_{i',j'} - \tau^*\dot{u}$. Decrease $T_{i,j;i',j'}$ by τ^*.

Our theoretical results from Section 5 imply that this process terminates: The calculation of the time step size τ^* in each step is based on local evaluations of the same inequality conditions that were used in the proof of Proposition 5 via Lemmas 2 and 3. Worst-case estimates led to the conclusion that for $\tau = \vartheta$ as given in (22), these conditions are globally satisfied. Therefore, the local evaluations of the same conditions will always result in $\tau^* \geq \vartheta$. As a result, each individual flow needs not more than $\lceil \tau_{\max}/\vartheta \rceil$ update steps to reduce its $T_{i,j;i',j'}$ from $\tau_{\max}$ to 0.

6.3.1 Consistency of the Scheme

Our scheme is conditionally consistent. To understand this, let us assume for simplicity $h_1 = h_2 = h$. The forward difference approximation in t and central difference approximations in the spatial domain generate approximation errors of $\mathcal{O}(\tau + h^2)$ as in conventional explicit schemes. Additional approximation errors result from the asynchronous update process. When computing a two-pixel flow $\dot{u} = g \cdot (u_{i',j'} - u_{i,j})/h^2$, the value $u_{i,j}$ is the sum of $u^k_{i,j}$ from the previous synchronisation time and several independent updates of the flows between (i, j) and its neighbours. Thus, $u_{i,j}$ carries an approximation error of $\mathcal{O}(u_x \cdot \tau/h + u_y \cdot \tau/h)$; similarly for $u_{i',j'}$. Each two-pixel flow therefore incurs an approximation error due to asynchronicity of $\mathcal{O}(\tau/h^3)$, making the overall approximation error of our scheme $\mathcal{O}(\tau + h^2 + \tau/h^3)$. This yields a conditionally consistent approximation to the FAB diffusion PDE if τ/h^4 is bounded by a constant when $\tau, h \to 0$.

On the other hand, sending the grid size h to 0 is not necessarily what one does in typical image processing applications. If one keeps the grid size constant and is only interested in the limit $\tau \to 0$, our randomised two-pixel scheme gives an $\mathcal{O}(\tau)$ approximation to the space-discrete FAB process. In that respect, it has the same consistency quality as the explicit scheme.

6.3.2 Efficient Random Selection and Time Accounting

For an efficient implementation of the algorithm, the performance of the selection in Step 1 is crucial. To this end, the bookkeeping of time step accounts $T_{i,j;i',j'}$ is done within a binary tree structure as follows.

Data Structure. Let $M = (m-1)n + m(n-1)$ be the number of pairs $\{(i, j), (i', j')\}$ of neighbouring pixels. Let $P = 2^p$ be the smallest power of two greater than M. We index pixel pairs by indices $\ell \in \{0, \ldots, M-1\}$ from which the corresponding i, j, i', j' can be computed in constant computation time as follows:

- If $\ell < (m-1)n$, create a horizontal neighbour pair by

$$\begin{aligned} i &:= \left\lfloor \frac{\ell}{n} \right\rfloor + 1 && \in \{1, \ldots, m-1\} \,, \\ j &:= \ell - n(i-1) + 1 && \in \{1, \ldots, n\} \,, \\ i' &:= i + 1 \,, \quad j' := j \,. \end{aligned} \tag{49}$$

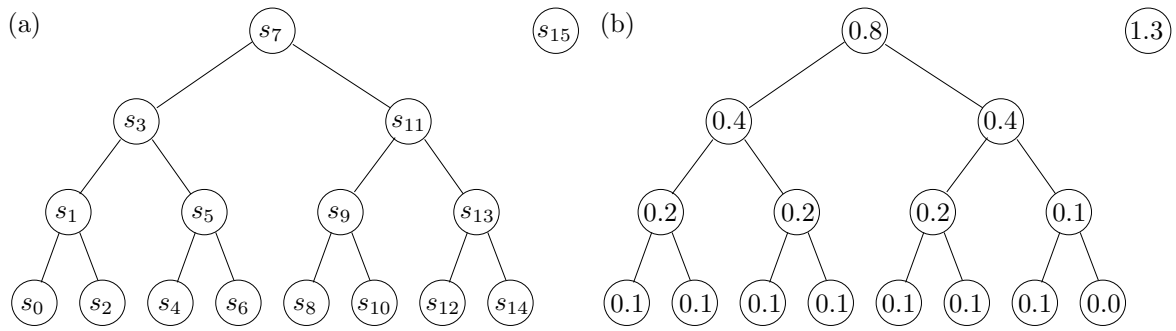


Figure 2: (a) Binary tree used for the time accounting of two-pixel flows in the randomised scheme, schematic for $P = 16$, and its representation in an array $\boldsymbol{s}$ with additional entry s_{15}. (b) Same binary tree with entries corresponding to $M = 13$ (e.g. for a 2×5-image) and $T_\ell = 0.1$ for $\ell = 0, \ldots, 12$.

- If $\ell \geq (m-1)n$, create a vertical neighbour pair by

$$\begin{aligned} \ell' &:= \ell - (m-1)n && \in \{0, \ldots, m(n-1)-1\} \,, \\ i &:= \left\lfloor \frac{\ell'}{n-1} \right\rfloor + 1 && \in \{1, \ldots, m\} \,, \\ j &:= \ell' - (n-1)(i-1) + 1 && \in \{1, \ldots, n-1\} \,, \\ i' &:= i \,, \quad j' := j + 1 \,. \end{aligned} \tag{50}$$

Denote by T_ℓ the time account value for the flow indexed by ℓ, for $\ell = 0, \ldots, M-1$, and $T_\ell = 0$ for $\ell = M, \ldots, P-1$. To store the extended sequence $(T_0, \ldots, T_{P-1})$, we use a complete binary tree with $P-1$ nodes (thus, with p levels). The nodes can be stored by in-order traversal in the first $P-1$ entries of a plain array $\boldsymbol{s} = (s_0, \ldots, s_{P-1})$, leaving s_{P-1} unused so far. In this representation, the tree structure is simply encoded in the binary representation of the indices: Let $\ell \in \{0, \ldots, P-2\}$ be an index in $\boldsymbol{s}$, and let its binary representation be

$$\ell = \sum_{q=0}^{p-1} b_q \cdot 2^q \tag{51}$$

with all $b_q \in \{0, 1\}$. Let q^* be the smallest index with $b_{q^*} = 0$ (i.e. the position of the least significant 0-bit). Then $p - 1 - q^*$ gives the level of the node corresponding to ℓ: If ℓ is even, it represents a leaf, with level number $p-1$. The index $\ell = P/2 - 1$, with $p - 1 - q^* = 0$, represents the root of the tree. For any non-root node ℓ (with $q^* < p-1$) its parent has the index $\ell^\wedge := \ell + 2^{q^*} - b_{q^*+1} 2^{q^*+1}$, whereas for any inner node ℓ (with $q^* > 0$) its children have the indices $\ell_\pm := \ell \pm 2^{q^*-1}$. Formally we can also compute $\ell^\wedge = P - 1$ for the root node $\ell = P/2 - 1$, and $\ell_- = P/2 - 1$ for $\ell = P - 1$. All these index computations can be done very efficiently by logical bit operations.

An exemplary tree for $P = 16$ ($p = 4$) is shown in Fig. 2 (a). As an example, for node $\ell = 5 = (0101)_2$ we have $q^* = 1$. Indeed, the node is on level $p - 1 - q^* = 2$ counted from the root. Further, $\ell^\wedge = 5 + 2^1 - 1 \cdot 2^2 = 3$ is the parent node, whereas $\ell_\pm = 5 \pm 2^0$ yields the two children $\ell_- = 4$ and $\ell_+ = 6$.

Now we store in each s_ℓ, $0 \leq \ell \leq P-1$, the sum of T_ℓ itself and all $T_{\ell'}$ where ℓ' belongs to the left sub-tree below the node of ℓ. Additionally, s_{P-1} is used to hold the sum of all T_ℓ. A simple example is shown in Fig. 2 (b).

This data structure allows to perform all operations required for the time accounting in the algorithm of Table 3 in not more than logarithmic time w.r.t. M per two-pixel flow update. As at least one update is needed per two-pixel flow in the global time step $k \mapsto k+1$, this means that the overall complexity of the time accounting is $\mathcal{O}(M \log M)$, thus also $\mathcal{O}(N \log N)$, per global time step. Let us detail in the following the exact operations needed in our algorithm; these occur in the initialisation (once per global time step) and in Steps 1 (random selection) and 5 (two-pixel flow update).

Random Selection. To select a two-pixel flow, with the remaining diffusion times as probabilities (Step 1 of the algorithm in Table 3, a single uniformly distributed random number $z \in [0, s_{P-1}]$ is needed, with s_{P-1} being the current total of the time accounts for all flows. By binary subdivision of the interval using the threshold values stored in the search tree, the flow index ℓ with

$$\sum_{\ell'=0}^{\ell-1} T_{\ell'} < z \leq \sum_{\ell'=0}^{\ell} T_{\ell'} \tag{52}$$

is determined. In detail, this is done by the following algorithm:

1. Compute a (pseudo-) random number r with uniform distribution in $[0, 1]$.
   Let $z = r \cdot s_{P-1}$, and $\ell = P/2 - 1$.
2. If $z \leq s_\ell$, let $\beta := 0$, otherwise $\beta := 1$.
3. If ℓ is even, select the flow with index $\ell + \beta$, stop.
4. If $\beta = 0$, replace ℓ with its left child index ℓ_-.
   if $\beta = 1$, decrease z by s_ℓ, then replace ℓ with its right child index ℓ_+.
5. Repeat from Step 2.

As an example, let the tree from Fig. 2 be given. With the pseudo-random number $r = 0.45$ we get $z = r \cdot s_{15} = 0.585$. Starting from $\ell = 7$ we compare $z = 0.585 \leq s_7 = 0.8$, so descend to the left sub-tree, setting $\ell_- = 3$ as the new ℓ. The next comparison $z = 0.585 > s_3 = 0.4$ sends us down the right sub-tree, setting $\ell_+ = 5$ as new ℓ, and $z - s_3 = 0.185$ as new z. Due to $z = 0.185 \leq s_5 = 0.2$ we replace ℓ with $\ell_- = 4$ to descend to the left again. Now, $z = 0.185 > s_4 = 0.1$ implies that $\beta = 1$, and $\ell + \beta = 5$ is returned as index of the selected pixel pair. Indeed, in the example $\sum_{\ell'=0}^{4} T_{\ell'} = 0.5 < 0.585 \leq \sum_{\ell'=0}^{5} T_{\ell'} = 0.6$ holds. Assuming $m = 2$, $n = 5$, the index $\ell = 5$ would be translated via (50) into the pixel pair $\{(1,1),(1,2)\}$.

Updating of the Time Accounts. After updating the flow with index ℓ by the diffusion time chunk τ^*, its time account must be updated (in Step 5 of the algorithm in Table 3). This means that all partial sums of time accounts stored in $\boldsymbol{s}$ which include T_ℓ as summand must be decreased by τ^*. This is done as follows.

1. Decrease s_ℓ by τ^*.
2. If $\ell = P - 1$, Stop.
3. Determine $\ell^\wedge$.
   If ℓ is the left child of $\ell^\wedge$, i.e., $(\ell^\wedge)_- = \ell$, replace ℓ with $\ell^\wedge$ and go to Step 1.
   Otherwise replace ℓ with $\ell^\wedge$ and go to Step 2.

Alternatively, one can generate a set of indices $\{\ell_0, \ldots, \ell_{p-1}\}$ where $\ell_0 := \ell$, and each ℓ_{q+1} is obtained by flipping the b_q bit of ℓ_q to 1. For each index ℓ' in the set, the corresponding $s_{\ell'}$ is decreased by τ^*. Note that whenever $\ell_{q+1} = \ell_q$, these count as a single set element, and the decrement is done only once.

Continuing the previous example, an update of the pixel pair belonging to $\ell = 5$ would trigger updates of the entries s_5, s_7 and s_{15}.

Initialisation of the Data Structure. In the initialisation step of Table 3 that is performed once per global time step, the initial entries s_ℓ must be computed. In the simplest case this is done by filling $\boldsymbol{s}$ with zeros, and performing one update with $-\tau_{\max}$ per flow. The cost of these operations is $\mathcal{O}(M \log M)$. Further algorithmic optimisations allow to reduce the effort to $\mathcal{O}(M)$. We do not pursue this optimisation here because of the marginal contribution of the initialisation to the total computational expense, and the fact that the random selection and two-pixel updates do not allow to reduce the overall complexity of time accounting below $\mathcal{O}(M \log M)$ anyway.

6.4 Deterministic Two-Pixel Scheme with Local Step-Size Adaptation

The $\mathcal{O}(\tau/h^3)$ contribution to the approximation error that limits the conditional consistency of the scheme from Table 3 results from the asynchronicity in the flow updates entering the values $u_{i,j}$, $u_{i',j'}$ in the computation of each next upcoming two-pixel flow $\dot{u}$. The order of this approximation error can be improved by one factor h if it is ensured that all flows contributing to each of the pixels $u_{i,j}$, $u_{i',j'}$ are updated to a synchronous time level, and by another factor h if the time levels of both pixels are synchronous when the new two-pixel flow is computed. To this end, we devise another variant of an explicit two-pixel scheme with local time step adaptation.

Overview of the New Scheme. One synchronisation step of our new scheme is summarised in Tables 4–5. Here, Table 4 describes the initialisations, and Table 5 contains the(loop for asynchronous updates.

In this scheme, time accounting is done for individual pixels: each pixel (i, j) is assigned a time account $t_{i,j}$ that starts from zero, and must reach $\tau_{\max}$ at the end of the asynchronous update loop. Pixel intensities are always updated with the total velocities incorporating all two-pixel flows that affect these pixels.

As a rule, updates are still carried out for pairs of neighbouring pixels, and by time steps that are governed by expiry times that guarantee that no violations of local stability conditions occur within a time step. These expiry times are precomputed and determine the order of updates to be performed. There are two kinds of stability conditions that are naturally associated with pairs of neighbouring pixels and single pixels, respectively. Therefore we will

Table 4: Time step of deterministic two-pixel scheme for FAB diffusion with locally adaptive step size control, first part (continued in Table 5)

Input: $\boldsymbol{u}^k$, $\tau_{\max}$

Initialisation:

1. **Time levels for pixels:** For each pixel (i,j), let $t_{i,j} := 0$.
2. **Diffusivity computation:** For each pixel (i,j), compute $g_{i,j}$ as in (20).
3. **Two-pixel flow computation:** For each pair of neighbouring pixels, $\{(i,j),(i',j')\}$ with $(i',j') = (i+1,j)$ or $(i',j') = (i,j+1)$, compute

$$\varphi_{i,j;i',j'} := \frac{g_{i,j} + g_{i',j'}}{2} \cdot \frac{u_{i',j'} - u_{i,j}}{h^2} \,, \tag{53}$$

   using $h = h_1$ for horizontal or $h = h_2$ for vertical neighbours.
4. **Pixelwise total velocity computation:** For each pixel (i,j), compute the right-hand side $\dot{u}_{i,j}$ as in (21):

$$\dot{u}_{i,j} = \varphi_{i+1,j;i,j} - \varphi_{i,j;i-1,j} + \varphi_{i,j+1;i,j} - \varphi_{i,j;i,j-1} \,. \tag{54}$$

5. **Flow expiry times:** For each pair of neighbouring pixels, $\{(i,j),(i',j')\}$ with $(i',j') = (i+1,j)$ or $(i',j') = (i,j+1)$,
   - Let $T_{i,j;i',j'} := \tau_{\max}$.
   - If $u_{i,j} \neq u_{i',j'}$ and $\operatorname{sgn}(\dot{u}_{i,j} - \dot{u}_{i',j'}) = -\operatorname{sgn}(u_{i,j} - u_{i',j'})$ and $|u_{i,j} - u_{i',j'}| < T_{i,j;i',j'}\,|\dot{u}_{i,j} - \dot{u}_{i',j'}|$, let

$$T_{i,j;i',j'} := -\frac{u_{i,j} - u_{i',j'}}{\dot{u}_{i,j} - \dot{u}_{i',j'}} \,.$$

6. **Pixel expiry times:** For each pixel (i,j),
   - Let $T_{i,j} := \tau_{\max}$.
   - If $\dot{u}_{i,j} > 0$, let $u_{\max} := \max\{u_{i,j}, u_{i',j'} \mid (i',j') \in \mathcal{N}(i,j)\}$. If $u_{i,j} + T_{i,j} \cdot \dot{u}_{i,j} > u_{\max}$, let

$$T_{i,j} := \frac{u_{\max} - u_{i,j}}{\dot{u}_{i,j}} \,.$$

   - If $\dot{u}_{i,j} < 0$, let $u_{\min} := \min\{u_{i,j}, u_{i',j'} \mid (i',j') \in \mathcal{N}(i,j)\}$. If $u_{i,j} + T_{i,j} \cdot \dot{u}_{i,j} < u_{\min}$, let

$$T_{i,j} := \frac{u_{\min} - u_{i,j}}{\dot{u}_{i,j}} \,.$$

7. **Priority queue:** Establish a priority queue Q of all neighbour pairs $\{(i,j),(i',j')\}$ and pixels (i,j) with priorities $-T_{i,j;i',j'}$ and $-T_{i,j}$, respectively, thus associating highest priority with lowest expiry time.

Table 5: Time step of deterministic two-pixel scheme for FAB diffusion with locally adaptive step size control, second part (continued from Table 4)

Asynchronous update: Repeat the following steps 1–7 until all $t_{i,j}$ are equal to $\tau_{\max}$.

1. **Priority-based selection:** Select from Q the pixel pair $\{(i,j),(i',j')\}$ or pixel (i,j) with highest priority (lowest flow expiry time $T_{i,j;i',j'}$ or pixel expiry time $T_{i,j}$). Let T^* be the expiry time of the selected pixel pair or pixel.

   If a pixel pair $\{(i,j),(i',j')\}$ has been selected, let $\mathcal{I}_1 := \{(i,j),(i',j')\}$ be the two-pixel set. If a pixel (i,j) has been selected, let $\mathcal{I}_1 := \{(i,j)\} \cup \mathcal{N}(i,j)$ be the set containing the selected pixel and its neighbours.

   In both cases, let $\mathcal{I}_2 := \mathcal{I}_1 \cup \bigcup_{(i^*,j^*)\in\mathcal{I}_1} \mathcal{N}(i^*,j^*)$ be the set containing all pixels from $\mathcal{I}_1$ and their neighbours. (Except at the image boundaries, $\mathcal{I}_2$ will consist of eight pixels if a pixel pair has been selected, or of thirteen pixels if a pixel has been selected.)

2. **Pixel updates:** For each $(i^*,j^*) \in \mathcal{I}_2$,

   - Update u_{i^*,j^*} by replacing it with the new value $\tilde{u}_{i^*,j^*} := u_{i^*,j^*} + (T^* - t_{i^*,j^*})\dot{u}_{i^*,j^*}$.
   - Update t_{i^*,j^*} by replacing it with T^*.

3. **Diffusivity computation:** For each $(i^*,j^*) \in \mathcal{I}_1$, recompute g_{i^*,j^*} as in (20).

4. **Two-pixel flow computation:** For each pair of neighbouring pixels $\{(i^*,j^*),(i^{**},j^{**})\}$ with $(i^*,j^*) \in \mathcal{I}_1$ and $(i^{**},j^{**}) \in \mathcal{I}_1$, recompute $\varphi_{i^*,j^*;i^{**},j^{**}}$ as in (53).

5. **Pixel velocity computation:** For each pixel $(i^*,j^*) \in \mathcal{I}_1$, recompute $\dot{u}_{i^*,j^*}$ according to (54).

6. **Flow expiry time updates:** For each pair $\{(i^*,j^*),(i^{**},j^{**})\}$ of neighbouring pixels with $(i^*,j^*) \in \mathcal{I}_2$ and $(i^{**},j^{**}) \in \mathcal{I}_2$,

   - Let $\tau^* := \tau_{\max} - t_{i^*,j^*}$
   - If $u_{i^*,j^*} \neq u_{i^{**},j^{**}}$ and $\operatorname{sgn}(\dot{u}_{i^*,j^*} - \dot{u}_{i^{**},j^{**}}) = -\operatorname{sgn}(u_{i^*,j^*} - u_{i^{**},j^{**}})$ and $|u_{i^*,j^*} - u_{i^{**},j^{**}}| < \tau^* |\dot{u}_{i^*,j^*} - \dot{u}_{i^{**},j^{**}}|$, let
   $$\tau^* := -\frac{u_{i^*,j^*} - u_{i^{**},j^{**}}}{\dot{u}_{i^*,j^*} - \dot{u}_{i^{**},j^{**}}} \,.$$
   - Let $T_{i^*,j^*;i^{**},j^{**}} := t_{i^*,j^*} + \tau^*$.
   - Correct the priority of $\{(i^*,j^*),(i^{**},j^{**})\}$ in Q according to the new value $-T_{i^*,j^*;i^{**},j^{**}}$.

7. **Pixel expiry time updates:** For each pixel $(i^*,j^*) \in \mathcal{I}_1$,

   - Let $\tau^* := \tau_{\max} - t_{i^*,j^*}$
   - If $\dot{u}_{i^*,j^*} > 0$, let $u_{\max} := \max\{u_{i^*,j^*}, u_{i^{**},j^{**}} \mid (i^{**},j^{**}) \in \mathcal{N}(i^*,j^*)\}$. If $u_{i^*,j^*} + \tau^* \cdot \dot{u}_{i^*,j^*} > u_{\max}$, let
   $$\tau^* := \frac{u_{\max} - u_{i^*,j^*}}{\dot{u}_{i^*,j^*}} \,.$$
   - If $\dot{u}_{i^*,j^*} < 0$, let $u_{\min} := \min\{u_{i^*,j^*}, u_{i^{**},j^{**}} \mid (u^{**},j^{**}) \in \mathcal{N}(i^*,j^*)\}$. If $u_{i^*,j^*} + \tau^* \cdot \dot{u}_{i^*,j^*} < u_{\min}$, let
   $$\tau^* := \frac{u_{\min} - u_{i^*,j^*}}{\dot{u}_{i^*,j^*}} \,.$$
   - Let $T_{i^*,j^*} := t_{i^*,j^*} + \tau^*$.
   - Correct the priority of (i^*,j^*) in Q according to the new value $-T_{i^*,j^*}$.

assign expiry times to pairs of neighbouring pixels as well as to single pixels. Both kinds of expiry times will be collected in a joint priority queue from which we select always the next item for update based on earliest expiry time.

Whenever the next item for update is a two-pixel flow $\{(i,j),(i',j')\}$,

- both pixels and all their neighbours are updated to a common target time with their respective total velocities; the common target time is determined by the expiry time of the two-pixel flow;
- the diffusivities $g_{i,j}$ and $g_{i',j'}$ are recomputed;
- with these new diffusivities, the flow $\varphi_{i,j;i',j'} := g \cdot (u_{i,j} - u_{i',j'})/h^2$ between the two pixels is recomputed, and the total velocities $\dot{u}_{i,j}$ and $\dot{u}_{i',j'}$ are updated;
- the flow expiry times for the two-pixel flow between (i,j) and (i',j') and all two-pixel flows between (i,j), (i',j') and their neighbours are updated as well, and so are the pixel expiry times of (i,j) and (i',j').

Whenever the next item for update is a single pixel (i,j), the update step is essentially the combination of updates of all flows adjacent to this pixel with the same target time. This means that

- the pixel (i,j) and its neighbours up to second order are updated to the common target time determined by the expiry time of the pixel;
- the diffusivities of pixel (i,j) and its immediate neighbours are recomputed;
- the flows $\varphi_{i,j;i',j'}$ between (i,j) and its neighbours are recomputed, and the velocities of (i,j) and its neighbours are updated;
- the expiry times of pixel (i,j) and its neighbour pixels and of all flows adjacent to these are updated.

As is evident from this outline, the scheme differs from the previous one in several important details which we will now discuss one by one. We discuss in detail the update of a pixel pair $\{(i,j),(i',j')\}$.

Pixel Updating by Total Velocities. From the randomised two-pixel scheme of Subsection 6.3, we retain the principle of asynchronous updates for two-pixel flows between neighbouring pixels, where the maximal evolution time updates are determined such that stability conditions are warranted. However, whereas in Section 6.3 the isolated flows $\varphi_{i,j;i',j'}$ were used for updating pixels (i,j), (i',j'), we use now total velocities: Each update to a pixel $u_{i,j}$ is made using the total velocity $\dot{u}_{i,j}$ that incorporates the contributions $\varphi_{i,j;i'',j''}$ from all neighbours (i'',j'') of (i,j). The same goes for pixel (i',j').

The use of total velocities for pixel updates ensures that at each intermediate step, each $u_{i,j}$ is an approximation of u at location (i,j) and time $k\tau_{\max} + t_{i,j}$, where $k\tau_{\max} \leq k\tau_{\max} + t_{i,j} \leq (k+1)\tau_{\max}$, and improves the approximation error by a factor h from $\mathcal{O}(\tau/h^3)$ to $\mathcal{O}(\tau/h^2)$. Time levels $t_{i,j}$ for individual pixels are tracked in the algorithm.

Flow Expiry Times. The first restriction that must be obeyed so that an update step of pixels does not violate the stability condition of Lemma 3 is that neighbouring pixels must not change their order within one update step. This condition is evaluated for each pair of neighbouring pixels and gives rise to a time limit that is naturally attributed to the pixel pair. We will denote it by $T_{i,j;i',j'}$ and call it the flow expiry time. Depending on the current values $u_{i,j}$, $u_{i',j'}$ and velocities $\dot{u}_{i,j}$, $\dot{u}_{i',j'}$, we compute $T_{i,j;i',j'}$ as the time at which the intensities of pixels (i,j) and (i',j') evolving with their current velocities would become equal, or the time $\tau_{\max}$ of the next global synchronisation step, whatever is earlier.

Pixel Expiry Times. The second restriction for the time step size of local updates relates to both Lemma 2 and Lemma 3. Within one update step, a pixel $u_{i,j}$ must not increase in excess of the maximum of intensities of the pixel and its immediate neighbours. (If the maximum intensity is that of pixel (i,j) itself, it must not grow at all, thus ensuring the condition of Lemma 2; otherwise this restriction implements the condition of Lemma 3.) Analogously, it must not decrease below the minimum of intensities of the pixel and its neighbours. This condition needs to be evaluated for each pixel and implies a time limit that is naturally associated with the pixel itself. We will denote it by $T_{i,j}$ and call it the pixel expiry time. Based on the current intensities of pixel (i,j) and its neighbours as well as the current velocity $\dot{u}_{i,j}$, we compute $T_{i,j}$ as the time at which the intensity of pixel (i,j) with its current velocity would leave the range bounded by the current minimal and maximal intensities within its neighbourhood, or $\tau_{\max}$, whatever is earlier.

In each update step, we select the pixel pair or single pixel with earliest expiry time, whatever comes first.

Once a flow $\varphi_{i,j;i',j'}$ is computed, it is used until either its underlying pixel pair or one of the two pixels expires. Thus, it enters all possible recomputations of $\dot{u}_{i,j}$ and $\dot{u}_{i',j'}$ as long as $t_{i,j}, t_{i',j'} < \min\{T_{i,j;i',j'}, T_{i,j}, T_{i',j'}\}$. Recomputation of a neighbouring flow $\varphi_{i,j;i'',j''}$ or $\varphi_{i',j';i'',j''}$ with $(i'',j'') \neq (i,j),(i',j')$ does not trigger a recomputation of $\varphi_{i,j;i',j'}$. However, the flow expiry time $T_{i,j;i',j'}$ as well as one of the pixel expiry times $T_{i,j}$ and $T_{i',j'}$ are adjusted in this case.

Updating Two-Pixel Flows and Total Velocities. After updating the flow between two neighbouring pixels (i,j), (i',j') to its expiry time, it is necessary to recompute the isolated flow $\varphi_{i,j;i',j'}$ between these two pixels, and the total velocities $\dot{u}_{i,j}$, $\dot{u}_{i',j'}$. With regard to the above discussion of the approximation error, we should ensure that for the computation of $\varphi_{i,j;i',j'}$ all participating pixels share the same time level. This means that we cannot restrict the pixel updates for the flow $\{(i,j);(i',j')\}$ to the two pixels (i,j), (i',j') but must even update all their neighbours (i'',j'') to the same time level, $t_{i'',j''} = t_{i,j} = t_{i',j'}$. Note that we have selected $\{(i,j),(i',j')\}$ as the pixel pair with earliest expiry time. Therefore, the updates to both pixels and their neighbours can safely be done within the expiry times of all these pixels and their adjacent flows.

After synchronising in this way all participating pixels, we compute new values $g_{i,j}$, $g_{i',j'}$ and a new $\varphi_{i,j;i',j'}$. The synchronisation of all pixels entering the computation of $\varphi_{i,j;i',j'}$ warrants a further improvement of the approximation error by a factor h from $\mathcal{O}(\tau/h^2)$ to $\mathcal{O}(\tau/h)$.

Adjusting Expiry Times. It is worth noticing that despite the update of all neighbouring pixels (i'', j'') of (i, j) and (i', j'), the flow $\varphi_{i,j;i',j'}$ is the only one that is updated. The flows $\varphi_{i,j;i'',j''}$ and $\varphi_{i',j';i'',j''}$ to neighbours $(i'', j'') \neq (i, j), (i', j')$ are left unchanged, despite the fact that they depend on $g_{i,j}$, $g_{i',j'}$, too. However, their recomputation would require updates to further neighbour pixels, and entail total velocity updates for further pixels, and by recursion finally lead to a synchronous time step for the entire image. By updating only $\varphi_{i,j;i',j'}$, total velocity updates are limited to the two pixels (i, j), (i', j').

We must, however, adjust the flow expiry times $T_{i,j;i'',j''}$, $T_{i',j';i'',j''}$ for all neighbour pairs $\{(i, j), (i'', j'')\}$ and $\{(i', j'), (i'', j'')\}$, because these depend on the total velocities $\dot{u}_{i,j}$, $\dot{u}_{i',j'}$ that have changed.

Moreover, we must update the pixel expiry times $T_{i,j}$, $T_{i',j'}$. In some cases, the new expiry time of one of these pixels will be equal to the time level of the current update step. Then, an update step for this pixel with recomputation of all adjacent flows will follow immediately. By this mechanism, further neighbour pixels are updated to the extent necessary to guarantee stability but avoiding unnecessary recursion across the entire image.

Comparison with Randomised Two-Pixel Scheme. As mentioned above, we have to select two-pixel flows to update in a priority order given by their expiry times. Thereby, the selected flow always has an expiry time less than or equal the expiry times of all neighbouring pairs.

The necessity to update flows in priority order implies that our new scheme is deterministic. The randomisation of the update order that served to avoid directional bias in Subsection 6.3 is not feasible here. However, there is also no need for such a measure here: Since each two-pixel diffusivity is recomputed if and only if it has expired, the order of updates among flows with equal expiry times does not change the intensities computed.

A final point to consider is that the conservation property of discretised diffusion was ensured in the randomised two-pixel scheme by always applying a two-pixel flow to both participating pixels at the same time. In our new scheme, these updates may be distributed to different iterations of the asynchronous update. However, each particular value of the two-pixel flow $\varphi_{i,j;i',j'}$ lives for a well-defined time interval $[t_1, t_2]$. It is computed when both pixels (i, j), (i', j') are synchronised at time level $t_{i,j} = t_{i',j'} = t_1$, and expires finally at a time level t_2. All updates of $u_{i,j}$ and $u_{i',j'}$ until they are both synchronised at $t_{i,j} = t_{i',j'} = t_2$ use the same value of $\varphi_{i,j;i',j'}$, which again guarantees the conservation property.

6.4.1 Consistency of the Scheme

Adapting our argument from Section 6.3.1, we see that in the computation of $u_{i,j}$, asynchronicity takes place only between the time levels of the approximations of flows $\varphi_{i,j;i',j'}$ between (i, j) and its neighbours (i', j'). As these flows approximate $g \cdot u_x/h$ for horizontal or $g \cdot u_y/h$ for vertical neighbours, the resulting approximation error of $\dot{u}_{i,j}$ due to the asynchronicity is $\mathcal{O}((u_x + u_y)\tau/h) = \mathcal{O}(\tau/h)$. As a result, the scheme from Tables 4–5 has an approximation error of $\mathcal{O}(\tau + h^2 + \tau/h)$. This ensures a conditionally consistent approximation to the FAB diffusion PDE if τ/h^2 is bounded by a constant when $\tau, h \to 0$. However, this is always satisfied by the stability conditions of our automatic time step size adaptation.

6.4.2 Efficient Time Accounting and Flow Selection

Time accounting occurs in this algorithm in two forms: on one hand the evolution times reached by the individual pixels have to be tracked; on the other hand two-pixel flows are equipped with expiry times. Each of the two sets of time variables can be stored in an array.

However, it is crucial for the algorithm that the selection of the next two-pixel flow to be updated, namely the one with earliest expiry time, is performed efficiently. This requires an efficient implementation of the priority queue Q that stores pixel pairs with their expiry times. For this purpose, a binary heap can be used, see e.g. [16, Ch. 6]. Each heap element stores the index of the flow pair by which its expiry time can be accessed by $\mathcal{O}(1)$ array access. The initialisation of Q can be performed in $\mathcal{O}(M)$ where M is the number of pixel pairs as before (note that the number of pixels N is also within $\mathcal{O}(M)$). In the asynchronous update cycle, the highest-priority entry can be determined in $\mathcal{O}(1)$.

As it is necessary to adjust priorities during the operation of the algorithm, the implementation must be extended into an addressable priority queue. As heap elements already store the indices of their flow pairs, this is easily achieved by establishing an array addressed by the flow pairs, which stores for each pair a pointer to the corresponding heap element. During heap operations, these pointers must be kept current. This does not change the asymptotic complexities stated before. The heap order corrections needed to adjust the priority of an element require not more than $\mathcal{O}(\log M)$ operations.

Data Structure. For convenience, we shortly describe how the priority queue Q is realised in our algorithm, adapting the binary heap construction from [16, Ch. 6]. First, the expiry times $T_{i,j;i',j'}$, $T_{i,j}$ are stored in an array T of size $M+N$, where each flow is identified by a unique *flow index* $l = l_{i,j;i',j'} \in \{1,\dots,M\}$, and each pixel by a unique *pixel index* $l = l_{i,j} \in \{M+1,\dots,M+N\}$, and $T[l]$ for $l \in \{1,\dots,M+N\}$ stores the respective expiry time. The heap array H of size $M+N$, addressed by *heap indices* k running from 1 to $M+N$, stores a permutation of the indices $1,\dots,M+N$. A third array P, addressed by flow/pixel-indices, stores for each l the corresponding heap index $P[l] = k$ that addresses l in the heap array, such that $H[P[l]] = l$ and $P[H[k]] = k$ for all k, l. Whenever entries of H are permuted, P must be updated accordingly.

For two flow/pixel indices l, l' we define $l \preceq l'$ if and only if $T[l] \leq T[l']$, i.e. flow/pixel l expires not later than flow/pixel l'. The array H is in heap order (w.r.t. the expiry times T) if and only if $H[\lfloor k/2 \rfloor] \preceq H[k]$ for all $k \in \{2,\dots,M+N\}$. By requiring the $\preceq$ relation between entries $\lfloor k/2 \rfloor$ and k, the array H is implicitly equipped with a binary tree structure where $H[1]$ is the root, and each $H[k]$ has the children $H[2k]$ and $H[2k+1]$ as long as these are inside H. Each node is required to be less or equal each of its children w.r.t. $\preceq$. As a consequence, the root $H[1]$ is the minimal element w.r.t. $\preceq$. The binary tree has the minimal possible depth, with $\lceil \log_2(M+N) \rceil$ as the maximal distance of a node from the root.

Priority Queue Operations. As mentioned before, the flow l with minimal expiry time is given by $H[1]$, and can therefore be selected in $\mathcal{O}(1)$ time. This is exactly what is needed in the priority-based selection step of our algorithm (asynchronous update, Step 1).

In the course of the algorithm, we need furthermore to establish the heap order (in the initialisation Step 6), and to correct the priority of an individual flow l after a change of $T[l]$ (end of asynchronous update, Step 5).

The latter task is accomplished by two recursive procedures called *sift-up* and *sift-down*. The procedure *sift-up* takes a heap index k as argument. Its pre-condition is that the heap order in H holds with the possible exception that element $H[k]$ may violate this order by being placed "too low" in the tree. Then, *sift-up* acts by swapping $H[k]$ up the tree until complete heap order of H is restored:

sift-up(k) :

1. If $k = 1$, stop.
2. Let $k' = \lfloor k/2 \rfloor$.
   Check if $H[k'] \preceq H[k]$. If this is the case, stop.
3. Swap $H[k]$ and $H[k']$.
   Update $P[H[k]] = k$ and $P[H[k']] = k'$.
   Call *sift-up*(k').

The other procedure, *sift-down*, too, takes a heap index k as argument. Contrary to *sift-up*, it has the pre-condition that the heap order in H holds in the sub-trees below k (i.e. the roots of which are the children of k) but $H[k]$ may violate the heap order by being placed "too high" in the tree. Then, *sift-down* extends the heap order to the complete tree rooted at k by swapping $H[k]$ down the tree as far as necessary:

sift-down(k) :

1. If $2k > M + N$, stop.
2. If $2k + 1 \leq M + N$ and $H[2k + 1] \preceq H[2k]$, let $k' = 2k + 1$; otherwise, let $k' = 2k$.
3. Check if $H[k] \preceq H[k']$. If this is the case, stop.
4. Swap $H[k]$ and $H[k']$.
   Update $P[H[k]] = k$ and $P[H[k']] = k'$.
   Call *sift-down*(k').

As the recursion in both *sift-up* and *sift-down* visits each depth-plane of the binary tree at most once, the complexity of each is $\mathcal{O}(\log(M + N)) = \mathcal{O}(\log M)$. The correction of the priority of a two-pixel flow (Step 5 of Table 5) is done by calling either *sift-down* or *sift-up*, depending on whether the expiry time has been increased or decreased.

The remaining task is to establish the heap order to initialise the priority queue. To this end, H is filled by an arbitrary permutation of $\{1, \ldots, M + N\}$ (initially just $(1, \ldots, M + N)$, in later time steps just the left-over from the previous iteration). Then one calls *sift-down*(k) for $k = \lfloor (M + N)/2 \rfloor, \ldots, 1$ in descending order. As the heap order condition is trivially satisfied in each single-node subtree, the heap order in H holds right of k before the call to *sift-down*(k), and holds from k on to the right after this call. Analysis of the run time, see [16, Ch. 6], shows that the complexity of establishing the heap order on H is $\mathcal{O}(M + N) = \mathcal{O}(M)$.

Final Remark. For our algorithm, the binary heap structure as described offers a fair balance of simplicity and efficiency. Further (slight) performance gains can possibly be achieved by using more sophisticated implementations of addressable priority queues; see [16, Ch. 6].

7 Experiments

For our experiments below, we use test images with quadratic pixels, where we assume unit grid size $h_1 = h_2 = 1$. The greyscale range lies in the interval $[0, 255]$. We have implemented our methods in ANSI C and compiled the code with a GNU gcc compiler (version 5.4.0). The parallel algorithm from Section 6.2 was implemented in CUDA C, and compiled using the Nvidia® CUDA compilation tools (version 8.0.44). Compiler optimisation was set to `-O2` in all cases, CUDA architecture was set to 6.1. No advanced code optimisations took place.

For CPU algorithms, we report run times on a single core of a workstation with an Intel® Core™ i7-6800K CPU running at 3.40 GHz. CUDA computations were performed on the same machine, using a Zotac GeForce GTX 1080 graphics card with 8 GB RAM and 2560 compute units.

7.1 Standard vs. Nonstandard Discretisation

In our first experiment we compare two implementation of our explicit scheme (21): one with the standard discretisation (19), the other one with the nonstandard discretisation (20). We use the diffusivity function (5). Fig. 3 shows the results for a standard test image that is processed with FAB diffusion with $\lambda = 4$, $\kappa = 2.5$, and stopping time $t = 10$. We observe that an explicit scheme with standard discretisation is unstable, even for very small time step sizes (Fig. 3(b)). Thus, it is not considered any further.

Equipping the explicit scheme with the nonstandard discretisation enables a stable result, provided that the time step size limit (22) is obeyed (Fig. 3(c)). For the diffusivity (5) with $\lambda = 4$ and $\kappa = 2.5$, the constant ω in (22) is given by $\omega = 0.009568$. Together with the grid sizes $h_1 = h_2 = 1$ this yields a time step size restriction of $\tau \leq 1.14 \cdot 10^{-5}$. Choosing $\tau := 1 \cdot 10^{-5}$ requires as many as 1 million iterations to reach a stopping time of $t = 10$. The corresponding CPU time was approx. 50 minutes. Since this is too slow for most applications, let us have a look at our various acceleration strategies.

Table 6: Run times (in seconds) for the computation of the results in Fig. 3 by CPU and CUDA implementations. CUDA-d refers to a version with double precision, whereas in CUDA-df the diffusivities g are computed in single precision. For CUDA, upright numbers show the gross run time of the program, whereas numbers in italics show the net computation time without system overheads.

Algorithm	CPU	CUDA-d	CUDA-df
explicit/standard discr.	2784		
explicit/nonstandard discr.	3004	42.5	26.3
		36.3	*19.9*
global time step adaptation	24.1	0.84	0.79
		0.30	*0.23*
randomised two-pixel	8.6	—	—
deterministic two-pixel	6.0	—	—

7.2 Faster Algorithms

The remaining images (d)–(f) in Fig. 3 show the results for three accelerations of the explicit scheme with nonstandard discretisation: global step size adaptation (Subsection 6.1), the randomised two-pixel scheme (Subsection 6.3), and its deterministic variant (Subsection 6.4). We observe that all approaches are visually of the same quality than its nonaccelerated counterpart. However, as evident from Table 6, they offer substantial speed-ups:

- The scheme with global time step adaptation and a maximal time step size of 0.24 requires only 4710 iterations to reach the stopping time of $t = 10$. Thus, its average time step size is approximately 0.0021, i.e. about 210 times larger than the baseline scheme. This shows that our time step size restriction (22) based on a priori estimates is indeed very pessimistic. On the other hand, in the course of the iterative computation a few time steps with sizes close to the theoretically estimated value 1.14×10^{-5} do indeed occur which confirms that no substantially larger time step size could be used in a non-adaptive regime.

  With 24 seconds, the computation time has reduced by a factor of about 120. Comparing this to the factor 210 for the time step size, the additional expense of the time step adaptation is visible.

- For the randomised two-pixel variant of the explicit scheme with nonstandard discretisation we use a sync step size of $\tau_{\max} = 0.1$. Thus, only 100 sync steps are necessary to reach a stopping time of $t = 10$. This requires a CPU time of 8.6 seconds, i.e. a speed-up of 2.8 compared to the scheme with global time step adaptation, or about 350 compared to the explicit scheme with constant time step size.

  The fact that the average time step size of 0.0991 is very close to the sync step size of $= 0.1$ demonstrates that time step size reductions for stability reasons are very rare events.

- The deterministic two-pixel scheme with a sync step size of $\tau_{\max} = 0.1$ is even faster than its probabilistic counterpart: Its run time comes down to 6.0 seconds, amounting to a speed-up of 1.4 over the scheme from Section 6.3, of 4 over the scheme with global time step adaptation, and of 500 over the scheme with fixed time steps.

  The average time step size for the pixel updates amounts to 0.0551 which indicates that time steps are split finer than in the randomised scheme but not dramatically so.

CUDA implementations were done for the explicit scheme with nonstandard discretisation, and for the scheme with global time step adaptation. Asynchronous schemes with local time step adaptation do not lend themselves to a straightforward parallelisation. The explicit scheme with standard discretisation could easily be parallelised but is not worth the effort due to its instability. A word of care should be said about numerical precision. Unlike in many diffusion-based algorithms in image processing, where single-precision float numbers are accurate enough, double precision is necessary in all of our algorithms due to the extremely small time steps that occur. Whereas on modern CPUs there is no noticeable difference in run times between single and double precision, precision still matters for performance on graphics cards. Our comparison therefore includes two variants of the CUDA implementation: one in which all numerical computations are performed in double precision (CUDA-d in Table 6),

and one in which double precision is used for u, $\dot{u}$ and time steps, whereas the diffusivities g are computed only in single precision, as a closer analysis reveals that the higher precision is not necessary for g (CUDA-df in Table 6). Numerical deviations between the CPU and both CUDA implementations for each algorithms were negligible (note that differences in the order of machine precision occur regularly between CPU and CUDA).

For the CUDA implementations, we report in Table 6 the gross run times of the program run and (in italics) the net computation times. The latter exclude some system overheads that occur due to the management of the graphics card, and contribute considerably to the overall program run time. For CPU computations, such system overheads are not observed.

As expected, the CUDA implementations allow further accelerations of the two first algorithms. With 42 (*36*) seconds for the explicit scheme with fixed time steps, and 0.84 *(0.30)* seconds for the scheme with global time step adaptation, the double-precision implementation CUDA-d provides for an 80-fold speed-up over CPU computation, from which some is chipped away by the system overheads. With the mixed-precision implementation CUDA-df the speed-up of the actual computation raises to about 150 for the scheme with fixed time steps, or 100 for the scheme with global time step adaptation, which is again diminished by system overheads. The reason why the adaptive scheme does not profit as much from the CUDA-df implementation as the non-adaptive one is that the computations for the very time step adaptation fully add to the double-precision part of the code (as high precision is needed for accurate time step accounting).

These experiments demonstrate that exploiting adaptivity, locality and parallelism are essential mechanisms to turn a prohibitively slow scheme into efficient algorithms that offer speed-ups by three orders of magnitude.

7.3 Scale-Space Behaviour

Since our accelerated algorithms are highly efficient, they can also be used for long term computations, arising e.g. in scale-space analysis [11, 39]. Fig. 4 depicts the temporal evolution of a noisy test image when the scale-space is governed by FAB diffusion with diffusivity (5), and the deterministic two-pixel scheme is employed. We observe the high robustness of the FAB scale-space in spite of the fact that its diffusities may become negative. Moreover, the experiment confirms convergence to a flat steady state for $t \to \infty$. This is in full accordance with our theoretical results.

7.4 Denoising Qualities

Since scale-spaces are simplifying evolutions with smoothing properties, it appears plausible that they also offer certain denoising qualities. While Fig. 4 has already confirmed this for a synthetic image, Fig. 5 studies the denoising capabilities of FAB diffusion for a highly textured real-world image: We have degraded the classical *barbara* test image with additive Gaussian noise. Although FAB diffusion has not been designed to be a noise removal method, we observe that it has clear denoising qualities which resembles those of a classical Perona–Malik filter. While this may appear suprising at first glance for an evolution which is of backward diffusion type in a certain gradient range, there is a simple explanation: In extrema, FAB diffusion always shows a forward diffusion behaviour, and our nonstandard discretisation takes

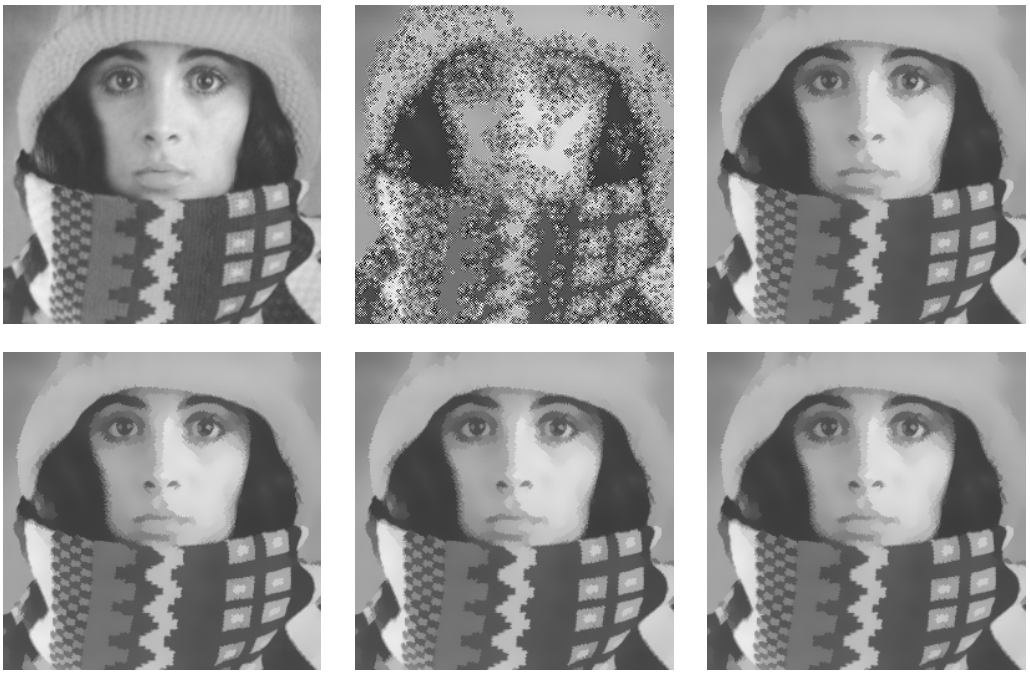


Figure 3: Influence of the numerical scheme on the result of FAB diffusion ($\lambda = 4$, $\kappa = 2.5$, $t = 10$). **(a) Top left:** Test image, 256×256 pixels. **(b) Top middle:** Explicit scheme (21) with standard discretisation (19). Computing 1 million iterations with time step size $\tau = 10^{-5}$ requires 2784 seconds, while the result is unstable. **(c) Top right:** Explicit scheme (21) with nonstandard discretisation (19). Performing 1 million iterations with $\tau = 10^{-5}$ takes 3004 seconds. **(d) Bottom left:** Explicit scheme with nonstandard discretisation and globally adaptive time step size control. It requires 4717 iterations, which are performed in 24.1 seconds. **(e) Bottom middle:** Randomised two pixel scheme with 100 iterations with sync time step size $\tau_{\max} = 0.1$, leading to a run time of 8.6 seconds. The average time step size is 0.0991. **(f) Bottom right:** Deterministic two-pixel scheme, using 100 iterations with synchronisation step size $\tau_{\max} = 0.1$. This requires 6.0 seconds. The average time step size is 0.0551.

care of its adequate algorithmic realisation. Thus, noise pixels which create local outliers are identified as extrema and attenuated accordingly.

7.5 Different Types of FAB Diffusivities

The crucial assumption for our theory is that $g(0) > 0$, which ensures that extrema undergo forward diffusion.

Thus, we do not necessarily require the diffusivity to take positive values for large gradients. Our framework can handle even diffusivities tending to negative limits for infinite gradients. An example is shown in Figure 6(g). For gradients going to infinity, the corresponding penaliser functions always decrease unboundedly; see the example in Figure 6(h).

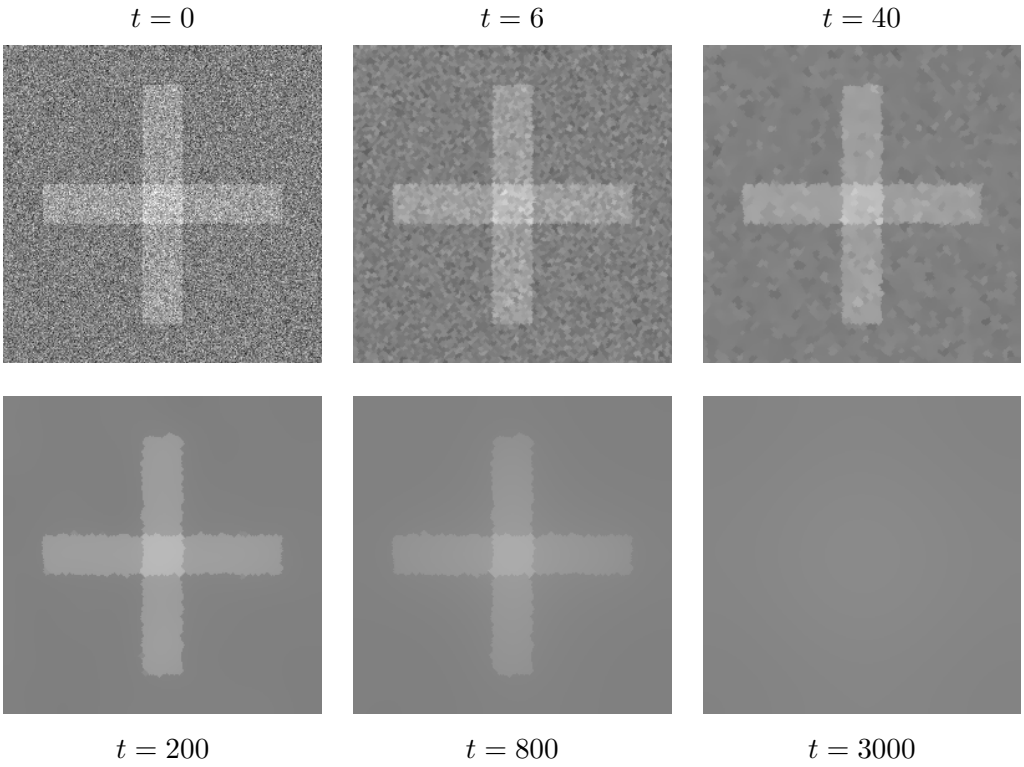
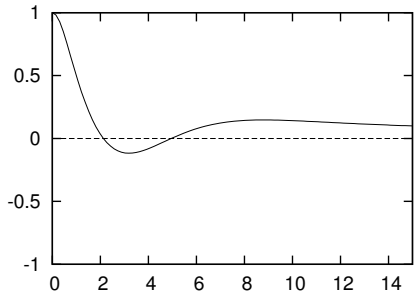
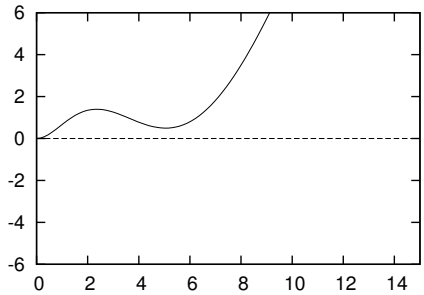


Figure 4: Temporal evolution of FAB diffusion ($\lambda = 2$, $\kappa = 2.5$), applied to a noisy synthetic test image with 256×256 pixels. The computations have been done with the deterministic two-pixel scheme with sync step size $\tau_{\max} = 0.1$.

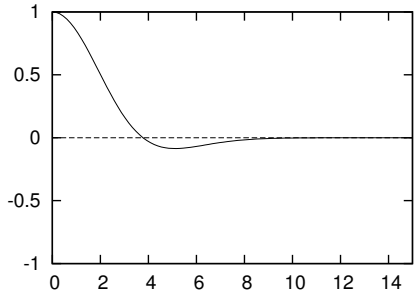
Figure 5: Denoising qualities of FAB diffusion for a real-world image. **(a) Left:** Test image *barbara* with 512×512 pixels. **(b) Middle:** With additive Gaussian noise, where the result is trucated outside $[0, 255]$. **(c) Right:** Restoration with FAB diffusion ($\lambda = 2$, $\kappa = 2.5$, $t = 120$). We have used the deterministic two-pixel scheme with sync step size $\tau_{\max} = 0.1$.



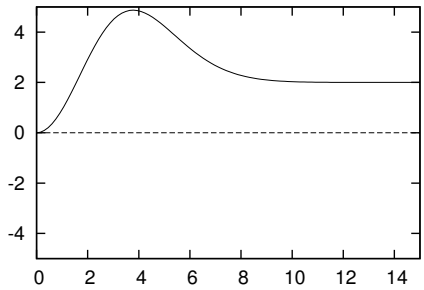
(a) Diffusivity, Type I



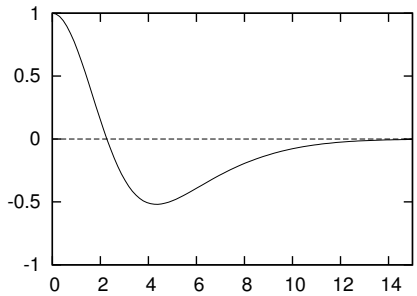
(b) Penaliser corresponding to (a)



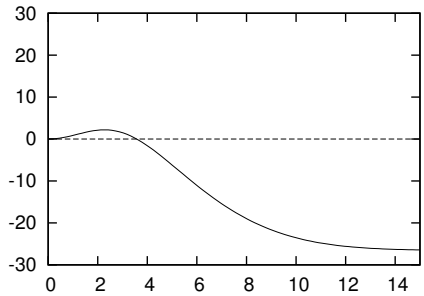
(c) Diffusivity, Type II



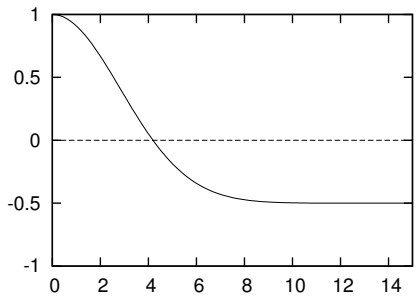
(d) Penaliser corresponding to (c)



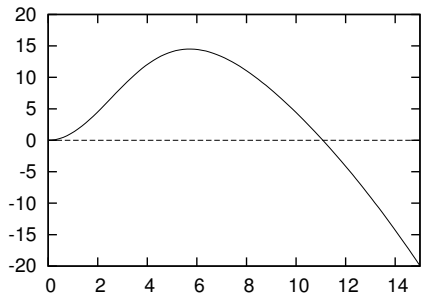
(e) Diffusivity, Type II



(f) Penaliser corresponding to (e)



(g) Diffusivity, Type III



(h) Penaliser corresponding to (g)

Figure 6: Forward-and-backward diffusivity functions $g(s^2)$ of Type I, Type II (two cases) and Type III, and corresponding penaliser functions $\Psi(s^2)$. The horizontal axis shows s, the vertical axis function values.

Let us now focus on three main classes of diffusivities:

Type I. Diffusivities similar to that from [9] that take positive values for large gradient magnitudes and tend to zero in the limit of infinite gradient magnitudes; see Figure 6(a).

Diffusivities of this kind can be associated to triple-well potentials, as shown in Figure 6(b). Some of these potentials are bounded from below by convex functions, which admits the application of some theoretical results in the space-continuous case.

Type II. Diffusivities similar to those from [26] that take negative values for large gradient magnitudes and tend to zero in the limit of infinite gradient magnitudes; see Figure 6(c) and (e).

The penaliser functions

$$\Psi(s^2) := \int_0^{s^2} g(t)\,\mathrm{d}t \tag{55}$$

corresponding to these diffusivities may have positive or negative horizontal asymptotes, as shown in Figure 6(d) and (f), or even decrease unboundedly in other cases.

Type III. Diffusivities that tend to a negative limit for gradient magnitudes going to infinity; see Figure 6(g).

For gradient magnitudes going to infinity, the corresponding penaliser functions always decrease unboundedly, see the example in Figure 6(h).

Specific representatives for these three types of diffusivities are presented in Table 7, where we see that our FAB diffusivity (5) is of Type II. We compare the impact of the three diffusivity types experimentally in Fig. 7. We observe that our deterministic two-pixel scheme can indeed handle all three scenarios. As expected, the Type I diffusivity reveals the strongest smoothing behaviour. The Type III diffusivity leads to the most extreme enhancement: Its pronounced backward diffusion along the edge contour produces fairly jagged edges. Nevertheless, since all three diffusivity types are compatible with our theory, their corresponding FAB evolutions will finally lead to a flat steady state.

8 Summary and Conclusions

Backward diffusion suffers from an extremely bad reputation of being a highly ill-posed process. We have seen in our paper that it can be turned into a highly stable evolution, provided that some essential requirements are met:

First of all, it must be stabilised at extrema in order to avoid under- and overshoots. The FAB diffusion paradigm does take care of this. Our discrete analysis is based on a smooth FAB diffusivity that attains a positive diffusivity in zero which is larger than the moduli of all negative diffusivities. Under these mild model assumptions we were able to establish well-posedness, a maximum–minimum principle, and convergence to a flat steady state for space-discrete FAB diffusions with a nonstandard discretisation of the gradient magnitude. These properties carry over to the fully discrete case with an explicit scheme, if one adheres to a very restrictive time step size limit.

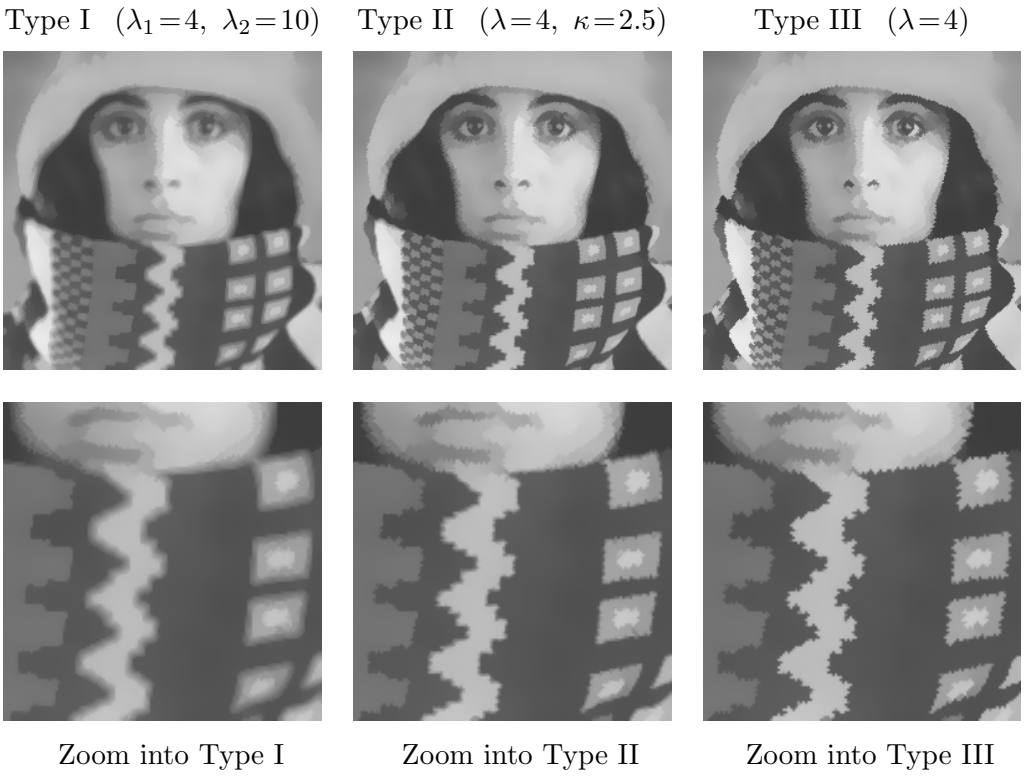


Figure 7: Comparison of the influence of the diffusivity functions. We use the diffusivities from Table 7. All computations are performed with the deterministic two-pixel scheme applied to the test image from Fig. 3(a) with sync step size $\tau_{\max} = 0.1$ and stopping time $t = 10$.

Table 7: Specific representatives for our three types of FAB diffusivities.

Diffusivity Type	Explicit Formula	Parameters
Type I	$g(s^2) = \dfrac{\cos(\pi\, y(s))}{1 + (y(s))^2}\,,$ $y(s) = \dfrac{5}{2}\left(\dfrac{(\lambda_2^2 - 6\lambda_1^2)\, s - (\lambda_2 - 6\lambda_1)\, s^2}{4\lambda_1\lambda_2(\lambda_2 - \lambda_1) + (\lambda_2^2 - 6\lambda_1^2)\, s - (\lambda_2 - 6\lambda_1)\, s^2}\right)$	$\lambda_1 > 0,$ $\lambda_2 > \sqrt{6}\,\lambda_1$
Type II	$g(s^2) = 2\exp\left(-\dfrac{\kappa^2 \ln 2}{\kappa^2 - 1} \cdot \dfrac{s^2}{\lambda^2}\right) - \exp\left(-\dfrac{\ln 2}{\kappa^2 - 1} \cdot \dfrac{s^2}{\lambda^2}\right)$	$\lambda > 0,\ \ \kappa > 1$
Type III	$g(s^2) = \dfrac{3}{2}\exp\left(-\dfrac{\ln 3}{\lambda^2}\, s^2\right) - \dfrac{1}{2}$	$\lambda > 0$

In order to make this concept practically viable, we came up with a number of novel schemes that exploit adaptivity or parallelism. In the simplest case, this adaptivity was applied globally by varying the time step size during the evolution. Allowing even space-variant time step sizes by splitting the diffusion process into two-pixel interactions is the most consequent and the most powerful implementation of the concept of adaptivity. To our knowledge this is the first time in PDE-based image analysis where the idea of space-variant time step sizes is used. We showed that this allows to accelerate FAB diffusion by up to three orders of magnitude.

We hope that our results may help to improve the reputation of backward parabolic processes, since they can offer some very attractive image enhancement properties that have hardly been explored so far, mainly because of the lack of stable numerical schemes. In our ongoing work we are looking into other PDE-based image enhancement methods that suffer from ill-posed continuous formulations.

Acknowledgements

J.W. and G.G. would like to thank the Isaac Newton Institute for Mathematical Sciences for support and hospitality during the programme *Variational Methods and Effective Algorithms for Imaging and Vision*, when final work on this paper was undertaken. This work was supported by EPSRC Grant Number EP/K032208/1 and by a Rothschild Distinguished Visiting Fellowship for J.W.

References

[1] Aubert, G., Kornprobst, P.: Mathematical Problems in Image Processing: Partial Differential Equations and the Calculus of Variations, *Applied Mathematical Sciences*, vol. 147, second edn. Springer, New York (2006)

[2] Babaud, J., Witkin, A.P., Baudin, M., Duda, R.O.: Uniqueness of the Gaussian kernel for scale space filtering. IEEE Transactions on Pattern Analysis and Machine Intelligence **8**, 26–33 (1986)

[3] Breuß, M., Welk, M.: Staircasing in semidiscrete stabilised inverse diffusion algorithms. Journal of Computational and Applied Mathematics **206**(1), 520–533 (2007)

[4] Burgeth, B., Weickert, J., Tari, S.: Minimally stochastic schemes for singular diffusion equations. In: X.C. Tai, K.A. Lie, T.F. Chan, S. Osher (eds.) Image Processing Based on Partial Differential Equations, pp. 325–339. Springer, Berlin (2007)

[5] Carasso, A.S.: Stable explicit time marching in well-posed or ill-posed nonlinear parabolic equations. Inverse Problems in Science and Engineering **24**(8), 1364–1384 (2016)

[6] Carasso, A.S.: Stabilized Richardson leapfrog scheme in explicit stepwise computation of forward or backward nonlinear parabolic equations. Inverse Problems in Science and Engineering **25**(12), 1719–1742 (2017)

[7] Elmoataz, A., Lezoray, O., Ta, V.T., Bougleux, S.: Partial difference equations on graphs for local and nonlocal image processing. In: O. Lezoray, L. Grady (eds.) Image Processing

and Analysis with Graphs: Theory and Practice, chap. 7, pp. 174–206. CRC Press, Boca Raton (2012)

[8] Gabor, D.: Information theory in electron microscopy. Laboratory Investigation **14**, 801–807 (1965)

[9] Gilboa, G., Sochen, N., Zeevi, Y.Y.: Image sharpening by flows based on triple well potentials. Journal of Mathematical Imaging and Vision **20**, 121–131 (2004)

[10] Gilboa, G., Sochen, N.A., Zeevi, Y.Y.: Forward-and-backward diffusion processes for adaptive image enhancement and denoising. IEEE Transactions on Image Processing **11**(7), 689–703 (2002)

[11] Iijima, T.: Basic theory on normalization of pattern (in case of typical one-dimensional pattern). Bulletin of the Electrotechnical Laboratory **26**, 368–388 (1962). In Japanese

[12] Kovasznay, L.S.G., Joseph, H.M.: Image processing. Proceedings of the IRE **43**(5), 560–570 (1955)

[13] Kramer, H.P., Bruckner, J.B.: Iterations of a non-linear transformation for enhancement of digital images. Pattern Recognition **7**, 53–58 (1975)

[14] Lindeberg, T.: Scale-Space Theory in Computer Vision. Kluwer, Boston (1994)

[15] Lindenbaum, M., Fischer, M., Bruckstein, A.: On Gabor's contribution to image enhancement. Pattern Recognition **27**, 1–8 (1994)

[16] Mehlhorn, K., Sanders, P.: Algorithms and Data Structures – The Basic Toolbox. Springer, Berlin (2008)

[17] Mickens, R.E.: Nonstandard Finite Difference Models of Differential Equations. World Scientific, Singapore (1994)

[18] Mrázek, P., Weickert, J., Steidl, G.: Diffusion-inspired shrinkage functions and stability results for wavelet denoising. International Journal of Computer Vision **64**(2/3), 171–186 (2005)

[19] Nikolova, M.: Local strong homogeneity of a regularized estimator. SIAM Journal on Applied Mathematics **61**(2), 633–658 (2000)

[20] Nikolova, M.: Minimizers of cost-functions involving nonsmooth data fidelity terms. application to the processing of outliers. SIAM Journal on Numerical Analysis **40**(3), 965–994 (2002)

[21] Osher, S., Rudin, L.: Shocks and other nonlinear filtering applied to image processing. In: A.G. Tescher (ed.) Applications of Digital Image Processing XIV, *Proceedings of SPIE*, vol. 1567, pp. 414–431. SPIE Press, Bellingham (1991)

[22] Osher, S., Rudin, L.I.: Feature-oriented image enhancement using shock filters. SIAM Journal on Numerical Analysis **27**, 919–940 (1990)

[23] Perko, L.: Differential Equations and Dynamical Systems, third edition edn. Springer, New York (2001)

[24] Perona, P., Malik, J.: Scale space and edge detection using anisotropic diffusion. IEEE Transactions on Pattern Analysis and Machine Intelligence **12**, 629–639 (1990)

[25] Pollak, I., Willsky, A.S., Krim, H.: Image segmentation and edge enhancement with stabilized inverse diffusion equations. IEEE Transactions on Image Processing **9**(2), 256–266 (2000)

[26] Smolka, B.: Combined forward and backward anisotropic diffusion filtering of color images. In: L. Van Gool (ed.) Pattern Recognition, *Lecture Notes in Computer Science*, vol. 2449, pp. 314–320. Springer, Berlin (2002)

[27] Smolka, B., Plataniotis, K.N.: On the coupled forward and backward anistropic diffusion scheme for color image enhancement. In: M.S. Lew, N. Sebe, J.P. Eakins (eds.) Image and Video Retrieval, *Lecture Notes in Computer Science*, vol. 2383, pp. 70–80. Springer, Berlin (2002)

[28] Steidl, G., Weickert, J., Brox, T., Mrázek, P., Welk, M.: On the equivalence of soft wavelet shrinkage, total variation diffusion, total variation regularization, and SIDEs. SIAM Journal on Numerical Analysis **42**(2), 686–713 (2004)

[29] Tikhonov, A.N., Arsenin, V.Y.: Solutions of Ill–Posed Problems. Wiley, Washington, DC (1977)

[30] Wang, Y., Niu, R., Shen, H., Yu, X.: Forward-and-backward diffusion for hyperspectral remote sensing image smoothing and enhancement. In: D. Li, J. Gong, H. Wu (eds.) International Conference on Earth Observation Data Processing and Analysis (ICEODPA), *Proceedings of SPIE*, vol. 7285. SPIE Press, Bellingham (2008)

[31] Wang, Y., Niu, R., Zhang, L., Wu, K., Sahli, H.: A scale-based forward-and-backward diffusion process for adaptive image enhancement and denoising. EURASIP Journal on Advances in Signal Processing **2011**(22) (2011)

[32] Wang, Y., Zhang, L., Li, P.: Local variance-controlled forward-and-backward diffusion for image enhancement and noise reduction. IEEE Transactions on Image Processing **16**(7), 1854–1864 (2007)

[33] Weickert, J.: Anisotropic Diffusion in Image Processing. Teubner, Stuttgart (1998)

[34] Weickert, J., Benhamouda, B.: A semidiscrete nonlinear scale-space theory and its relation to the Perona–Malik paradox. In: F. Solina, W.G. Kropatsch, R. Klette, R. Bajcsy (eds.) Advances in Computer Vision, pp. 1–10. Springer, Wien (1997)

[35] Welk, M., Gilboa, G., Weickert, J.: Theoretical foundations for discrete forward-and-backward diffusion filtering. In: X.C. Tai, K. Mørken, M. Lysaker, K.A. Lie (eds.) Scale Space and Variational Methods in Computer Vision, *Lecture Notes in Computer Science*, vol. 5567, pp. 527–538. Springer, Berlin (2009)

[36] Welk, M., Steidl, G., Weickert, J.: Locally analytic schemes: A link between diffusion filtering and wavelet shrinkage. Applied and Computational Harmonic Analysis **24**, 195–224 (2008)

[37] Welk, M., Weickert, J.: An efficient and stable two-pixel scheme for 2D forward-and-backward diffusion. In: F. Lauze, Y. Dong, A.B. Dahl (eds.) Scale Space and Variational Methods in Computer Vision, *Lecture Notes in Computer Science*, vol. 10302, pp. 94–106. Springer, Cham (2017)

[38] Welk, M., Weickert, J., Galić, I.: Theoretical foundations for spatially discrete 1-D shock filtering. Image and Vision Computing **25**(4), 455–463 (2007)

[39] Witkin, A.P.: Scale-space filtering. In: Proc. Eighth International Joint Conference on Artificial Intelligence, vol. 2, pp. 945–951. Karlsruhe, West Germany (1983)

[40] Zakeri, A., Jannati, Q., Amiri, A.: A numerical scheme for solving nonlinear backward parabolic problems. Bulletin of the Iranian Mathematical Society **41**(6), 1453–1464 (2015)