Saarland University
Faculty of Natural Sciences and Technology I
Computer Science Department
Master's Program in Visual Computing

Master's Thesis

Moment Preserving Diffusion in Image Processing

submitted by

Martin Peter Grochulla

martin@mgrochulla.de

March 10, 2010

Supervisor	Prof. Dr. Joachim Weickert
Adviser	Prof. Dr. Joachim Weickert
Reviewers	Prof. Dr. Joachim Weickert
	Dr. Andrés Bruhn

Statement

Hereby I confirm that this thesis is my own work and that I have documented all sources used.

Saarbrücken, March 10, 2010

Martin Peter Grochulla

Declaration of Consent

Herewith I agree that my thesis will be made available through the library of the Computer Science Department.

Saarbrücken, March 10, 2010

Martin Peter Grochulla

Abstract

We present a new approach for denoising noisy images with a diffusion filter. Noise is present in many digital images. Based on the imaging system there are different types of noises, one of them is additive noise. We present an approach for denoising gray value images with additive noise. Our approach is based on nonlinear isotropic diffusion. One property of nonlinear isotropic diffusion is the preservation of the average gray level of the original image during the image evolution. In contrast to nonlinear isotropic diffusion we do not preserve the average gray level during the image evolution but a (shifted) moment of the gray values of the original image. Since the average gray level is equivalent to the first (central) moment our approach is a generalization of nonlinear isotropic diffusion in that sense.

We first introduce linear and nonlinear isotropic diffusion, explain how to discretize the partial differential equations arising from these diffusion processes, briefly show properties of those processes using discrete scale-space theory, and present efficient ways to implement them.

Based on the presented diffusion processes we show how to extend nonlinear isotropic diffusion in order to preserve higher moments. We then discuss which properties from the nonlinear isotropic diffusion carry over to our generalization. Finally we show in our experiments that our approach provides good denoising results in case of noisy, binary gray level images.

Acknowledgement

I owe my deepest gratitude to my adviser and mentor Prof. Joachim Weickert. He has been the person who has drawn my attention to the field of visual computing. With this thesis he has given me the opportunity to work with him and gain insight into this broad and interesting field of research. Without his valuable advice and help have this thesis would not have been possible.

Furthermore, I offer my regards and blessings to all of those who supported me in any respect during the completion of the thesis.

Contents

Chapter 1

Introduction

Human beings perceive their environment to a great amount by images. Many areas in our brain are involved in processing the visual information obtained by our eyes. Hence the visual perception is the most important kind of perception for us. Combining the images from both eyes to one three dimensional image, extracting relevant information in the image from irrelevant information, and adding missing visual information to a complete image are some examples of the visual processing in our brain.
The brain is not the only part of the human visual system that has evolved in the past: Our eyes are able to cope with different and difficult circumstances. We can see on a bright day and in a dark night. With sufficient ambient light we are also able to perceive color information.
The capabilities of the human visual system become clear when one compares the images we perceive with the images acquired by other imaging systems.

1.1 Image Acquisition

Such imaging systems are for example digital cameras or medical equipment (computer tomography, magnetic resonance imaging, ultrasonography). Due to physical limitations of the imaging equipment the acquired images are distorted. Basically there are two types of distortions: blur and noise.
Blurring occurs when for instance the digital camera does not focus on the object properly or when the object is moving relatively to the camera. The resulting images then look smeared.
Noise describes random deviations in brightness of the acquired image. Based on the sensor and the method used for image acquisition there are different types of noise: impulse noise, multiplicative noise, and additive noise. We will focus on additive noise.

1.2 Image Processing and Computer Vision

The goal of image processing and computer vision is to process images in such a way that they are easier to interpret by human beings or better to process by further algorithms.
Computer vision focuses on the processing of several related images in order to obtain information that cannot be extracted from a single image only. Computer vision algorithms are used for

- optic flow computation: The optic flow describes the direction and speed of movement of objects in two consecutive images from an image sequence,

- stereo reconstruction: Here one uses two images of the same object taken at the same time from different directions. From those two images one can then reconstruct the depth information of the object, and
- particle image velocimetry: Here one uses particles in a fluid to compute how the fluid behaves near obstacles by monitoring the movement of the particles.

Image processing focuses on enhancing the quality of single images. Image processing algorithms (filters) are used for

- extraction of important image structures such as edges and corners (since the human visual system is very sensitive to this kind of discontinuities),
- segmentation: dividing the image into regions of constant color where one has discontinuities at the region boundaries,
- deblurring: Reconstruction of a visually better image from a blurred image (since in a blurred image edges are smeared and dislocated), and
- denoising: Removing noise from a noisy image.

In this thesis we will focus on the task of denoising. There are different filters for image denoising. One popular class of filters are diffusion filters.

1.3 Diffusion Filters

Diffusion filters are inspired by the physical process of diffusion which is governed by Fick's law and the continuity equation. Fick's law describes the property of the process to equilibrate concentration differences. The continuity equation describes the preservation of mass in different processes. Combining both leads to partial differential equations expressing the behavior of diffusion filters: equilibrating differences while preserving the mass. Diffusion filters are able simplify the image while preserving important image structures. This simplification allows the filter to reduce the (random) noise in the image and hence to enhance it.
There are different types of diffusion (see [14]):

- linear diffusion, which is the simplest type of diffusion,
- nonlinear isotropic diffusion, where a diffusivity function is used to reduce diffusion at edges which leads to the ability to enhance edges, and
- nonlinear anisotropic diffusion, where a diffusion and structure tensor is used for edge-enhancing and coherence-enhancing diffusion, respectively. In contrast to isotropic diffusion in anisotropic diffusion the diffusion in different directions can be steered separately.

In this thesis we will deal with linear and nonlinear isotropic diffusion only. We will first introduce linear and nonlinear isotropic diffusion, explain how they can be implemented, and what properties they have. Then we explain how we can use those diffusion filters in order to preserve higher moments. Finally we demonstrate the effects of the different parameters and show some filtering results.

Chapter 2

Background

Before we present our approach to enhance images using diffusion filters, we first introduce the notation we will use throughout this thesis. Then we present linear and nonlinear isotropic diffusion. Our approach is based on nonlinear isotropic diffusion which is a generalization of linear diffusion.
Linear diffusion is the simplest possible diffusion filter. Although the results of this filter are inferior to other more sophisticated diffusion filters it has some preferable properties for an image filter and is often used inside more sophisticated diffusion filters.

2.1 Gray Scale Images

In our setting a continuous gray scale image is a mapping

$$f : \mathbb{R}^2 \supset \Omega \mapsto \mathbb{R}$$

from a rectangular subset of $\mathbb{R}^2$, $\Omega =]0, a_x[\times]0, a_y[$, to $\mathbb{R}$. Here the domain Ω is called the image domain. The co-domain specifies gray values, usually low gray values are dark and high gray values are bright.
Digital gray scale images on the other hand are sampled and quantized. Sampling is the discretization of the image domain. Here an image consists of gray values of a rectangular point grid $\{f_{i,j} | i = 1, \ldots, n_x, j = 1, \ldots, n_y\}$ within the image domain Ω instead of gray values of the entire image domain (see figure 2.1). A grid point (i, j) is called pixel. Here n_x and n_y is the width and height of the image in pixels, respectively. $f_{i,j}$ denotes the gray value of pixel (i, j). We have $f_{i,j} = f\big((i - \frac{1}{2}) \cdot h_x, (j - \frac{1}{2}) \cdot h_y\big)$, so in the sampled image the gray value of pixel (i, j) is the gray value of the continuous image f in the center of $](i - 1) \cdot h_x, i \cdot h_x[\times](j - 1) \cdot h_y, j \cdot h_y[$. Here h_x and h_y is the distance between adjacent pixels in x- and y-direction, respectively. Usually the distances are the same for the x- and y-dimension and both distances are normalized to 1. Quantization is the discretization of the co-domain. If the gray values are coded byte-wise then the discrete co-domain is given by $\{0, 1, \ldots, 255\}$.
We assume that we are given digital gray scale images. So, the input data for our filter is sampled and quantized: gray values are from $\{0, 1, \ldots, 255\}$ and they are sampled at a regular point grid with $h_x = h_y = 1$. Throughout this thesis we will use n_x to denote the width of the image in pixel and n_y to denote the height of the image in pixels.

2.2 Image Distortions

Digital images are distorted due to limitations of the imaging equipment or image transmission. There are basically two types of image distortions. On the one hand there is blur. Blurring

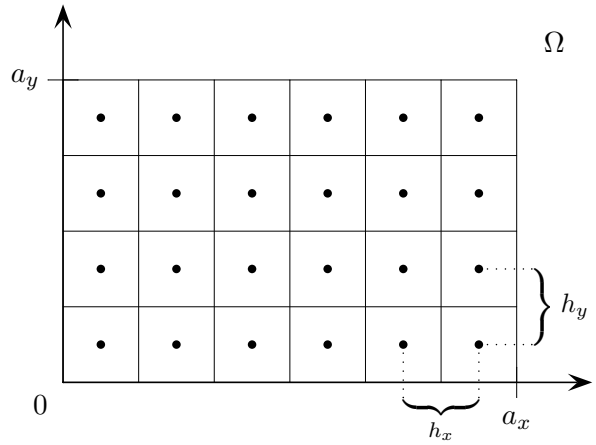


Figure 2.1: Point grid within the image domain $\Omega \supset]0, a_x[\times]0, a_y[$ consisting of $6 \cdot 4$ pixels. If $h_x = h_y = 1$ then we have $n_x = a_x$ and $n_y = a_y$ in this case.

occurs when the camera does not focus properly on the targeted object, when the camera is moved while the image is acquired, or the object moves while the image is acquired. The former two types of blur are also known as motion blur. In all cases the resulting images look smeared. On the other hand there is noise. In images noise is present due to the sensor quality of the CCD chip in digital cameras, specific acquisition methods which create characteristic noise, or other disturbances occurring during image acquisition. There are different types of noise. Additive noise is an important and common type of noise. Here the noise is independent of the gray values of the image and can be expressed as:

$$f_{i,j} = g_{i,j} + n_{i,j},$$

where f is the acquired image, g is the ideal image one wishes to acquire and n is the additive noise (see figure 4.1). The noise n may have different distributions. If the distribution is a Gaussian distribution one speaks of Gaussian noise, here many small and few large gray value perturbations occur. In contrast to additive noise multiplicative noise is signal-dependent:

$$f_{i,j} = g_{i,j} + n_{i,j} \cdot g_{i,j} = g_{i,j} \cdot \big(1 + n_{i,j}\big).$$

The third type of noise is the so-called impulse noise. Here not all pixels of the image are changed but only a fraction of pixels have erroneous gray values. If the erroneous gray values are either 0 (the lowest possible gray value) or 255 (the highest possible gray value) one speaks of salt and pepper noise. If the erroneous gray values are uniformly distributed over $\{0, 1, \ldots, 255\}$ one speaks of uniform noise.

2.3 Continuous Differential Equation for Diffusion

Diffusion is a physical process equilibrating concentration differences in a medium while preserving the total mass of that medium (see [2], [4], [5], or [6]). If u describes the concentration of the medium then the equilibration property is described by Fick's law

$$j = -g \cdot \nabla u.$$

Here a (spatial) concentration gradient ∇u creates a flux j in direction of the steepest descend. The flux is proportional to the concentration gradient. Bigger concentration differences lead to bigger flux equilibrating the given concentration differences. Furthermore, the flux depends on the medium: in different media concentration differences are balanced out in different times. That differing speed of the diffusion process is expressed by the (non-negative) diffusivity g. The preservation of the total mass of the medium is described by the continuity equation

$$\partial_t u + \operatorname{div} j = 0,$$

where t denotes the time and hence ∂_t is the partial derivative in time. The equation states that the sum of the change of u in time and the divergence of the flux j of u is zero. In other words: when the amount of the medium in one location is reduced (the concentration in that point falls) the same amount of the medium is increased in another locations (the concentration in those point rises). Hence the total amount or the total mass of the medium does not change. By rewriting the continuity equation

$$\partial_t u + \operatorname{div} j = 0 \quad \Leftrightarrow \quad \partial_t u = -\operatorname{div} j,$$

we can now combine the equation stating Fick's law and the rewritten continuity equation to

$$\partial_t u = -\operatorname{div} (-g \cdot \nabla u)$$

leading to

$$\partial_t u = \operatorname{div} (g \cdot \nabla u),$$

which is the continuous partial differential equation for diffusion.
We will use diffusion filters for denoising images: Gaussian noise creates many small gray value perturbations in images, but only few big ones. The diffusion process will equilibrate those differences and hence remove the noise from the image.

2.4 Linear Diffusion

Linear diffusion is the simplest diffusion process possible. For linear diffusion g is set to a constant. For simplicity one usually sets, without loss of generality as we will later explain, $g = 1$ and obtains

$$\partial_t u = \operatorname{div} (\nabla u).$$

Using the Laplace operator Δ this can be written to

$$\partial_t u = \Delta u,$$

which is also known as the heat equation (see [2]).

In image processing applying a diffusion filter to a digital image f is done in the following way: The pixel (i, j) of the image f represents a grid point in the rectangular point grid and the gray value $f_{i,j}$ of that pixel represents a concentration at that point in the image domain. The continuous linear diffusion filter now computes iteratively a filtered version $u(x, y, t)$ of $f(x, y)$ as a solution of the diffusion equation

$$\partial_t u = \Delta u$$

with the original image $f(x, y)$ as initial condition $u(x, y, 0) = f(x, y)$ and suitable boundary conditions. Here t is the diffusion time. So, beginning with $t = 0$ (the original image) one computes an evolution of the image according to the linear diffusion process.
Boundary conditions are necessary in order to control the behavior of the diffusion filter at the boundary of the image domain and the boundary pixels, respectively. A commonly used boundary conditions is the Neumann boundary condition. In the Neumann boundary condition or reflecting boundary condition one chooses the gray values at the image boundary in such a way that the gradient across the boundary vanishes:

$$\partial_n u = 0.$$

Here n is the (unit) normal vector at the image boundary: $n = \frac{1}{|\nabla u|}(u_y, -u_x)^\top$, where $\nabla u = (u_x, u_y)^\top$. ∂_n is the directional derivative in direction of vector n: $\partial_n u = n^\top \nabla u$. The easiest way to achieve that condition is to mirror the given image at its boundary hence the term reflection boundary condition. So, the Neumann boundary condition prohibits any flux across the boundary.
Furthermore, the Neumann boundary condition has another preferable property: It does not violate the mass preservation property of the diffusion process. Because of this property and the fact that we only have to reflect the gray values at the image boundary we will use the Neumann boundary condition throughout the thesis.

2.4.1 Discretization

In order to develop an algorithm for the linear diffusion filter from the continuous linear diffusion equation we have to discretize the continuous partial differential equation from section 2.3:

$$\partial_t u = \Delta u.$$

We first rewrite the equation to

$$\partial_t u = \partial_{xx} u + \partial_{yy} u,$$

where ∂_{xx} and ∂_{yy} are the second partial derivative in x- and y-direction, respectively. We abbreviate the notation from above further be writing u_t, u_{xx}, u_{yy} for the first partial derivative in time and the second partial derivatives in x- and y-direction, respectively:

$$u_t = u_{xx} + u_{yy}.$$

Derivative Approximation

The derivatives occurring in the continuous differential equation have to be approximated, since the digital image is sampled at the discrete positions of the point grid as described in section 2.1. For the approximation we use finite difference methods (see [11], [9]). Derivatives at a certain pixel position (i, j) are approximated by the gray values $u_{i,j}$ at this position (i, j) and the gray values of the neighbor pixels, for instance $u_{i-1,j}$, and $u_{i+1,j}$. The approximation is done by a Taylor series: if $f : \mathbb{R} \mapsto \mathbb{R}$ is infinitely many times differentiable around a, then

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x - a)^n,$$

where $f^{(n)}(a)$ denotes the nth derivative of f at a and $n!$ denotes the nth factorial.

Let us demonstrate the derivative approximation: Assume we are in a one-dimensional setting. We want to approximate u'', the second partial derivative of u, at pixel (i) using the gray values u_{i-1}, u_i, and u_{i+1}. We denote the pixel positions as subscripts. Taylor expansion around position (i) gives us:

$$\begin{aligned} u_{i-1} &= u_i - h \cdot u_i' + \frac{h^2}{2} \cdot u_i'' - \frac{h^3}{6} \cdot u_i''' + \frac{h^4}{24} \cdot u_i'''' + \mathcal{O}\left(h^5\right), \\ u_i &= u_i, \\ u_{i+1} &= u_i + h \cdot u_i' + \frac{h^2}{2} \cdot u_i'' + \frac{h^3}{6} \cdot u_i''' + \frac{h^4}{24} \cdot u_i'''' + \mathcal{O}\left(h^5\right), \end{aligned}$$

where $\mathcal{O}$ denotes the Landau symbol and h the grid size. Now comparison of the coefficients in

$$\begin{aligned} 0 \cdot u_i + 0 \cdot u_i' + 1 \cdot u_i'' &\stackrel{!}{=} \alpha_{-1} \cdot u_{i-1} + \alpha_0 \cdot u_i + \alpha_1 \cdot u_{i+1} \\ &= (\alpha_{-1} + \alpha_0 + \alpha_1) \cdot u_i \\ &+ h \cdot (-\alpha_{-1} + \alpha_1) \cdot u_i' \\ &+ h^2 \cdot (\alpha_{-1} + \alpha_1) \cdot u_i'' + \mathcal{O}\left(h^3\right) \end{aligned}$$

leads to the following linear system of equations

$$\begin{pmatrix} 1 & 1 & 1 \\ -1 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \alpha_{-1} \\ \alpha_0 \\ \alpha_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \frac{2}{h^2} \end{pmatrix}$$

which has the following solution:

$$\alpha_{-1} = \frac{1}{h^2}, \quad \alpha_0 = -\frac{2}{h^2}, \quad \text{and} \quad \alpha_1 = \frac{1}{h^2}.$$

Note that in order to derive a derivative approximation for the nth (partial) derivative one needs at least $n+1$ values, otherwise the linear system does not have a unique solution. Plugging the solution from above into the Taylor expansion leads to

$$\frac{1}{h^2} \cdot u_{i-1} - \frac{2}{h^2} \cdot u_i + \frac{1}{h^2} \cdot u_{i+1} = u_i'' + \frac{h^2}{12} \cdot u_i'''' + \mathcal{O}\left(h^4\right),$$

giving an approximation of consistency order 2 due to the quadratic error term $\frac{h^2}{12}$ in the grid size h. Basically higher consistency orders give better accuracies. In this case we have used pixels left and right of pixel (i) to approximate u''. One could also think of only using pixels either left of or right of pixel (i) for the approximation. But it turns out that one can achieve higher consistency orders by choosing the neighbor pixels in such a way that pixel (i) is the center of all considered pixels (a so-called central difference).

Explicit Scheme

The partial derivative of u in time at $k \cdot \tau$ is approximated by a so called forward difference

$$u_t \approx \frac{u_{i,j}^{(k+1)} - u_{i,j}^{(k)}}{\tau},$$

and the second partial derivatives in x- and y-directions at positions $(i-\frac{1}{2})\cdot h_x$ and $(j-\frac{1}{2})\cdot h_y$ are approximated by central differences:

$$u_{xx} \approx \frac{u_{i-1,j}^{(k)} - 2u_{i,j}^{(k)} + u_{i+1,j}^{(k)}}{h_x^2}, \quad \text{and}$$
$$u_{yy} \approx \frac{u_{i,j-1}^{(k)} - 2u_{i,j}^{(k)} + u_{i,j+1}^{(k)}}{h_y^2}.$$

Here $\tau > 0$ is the time step size. The diffusion time is now the product of the time step size τ and the number of iterations. Furthermore, h_x and h_y are the spatial grid sizes in x- and y-direction, respectively. The time step is denoted as superscript. The resulting numerical scheme is then given by

$$\frac{u_{i,j}^{(k+1)} - u_{i,j}^{(k)}}{\tau} = \frac{u_{i-1,j}^{(k)} - 2u_{i,j}^{(k)} + u_{i+1,j}^{(k)}}{h_x^2} + \frac{u_{i,j-1}^{(k)} - 2u_{i,j}^{(k)} + u_{i,j+1}^{(k)}}{h_y^2}.$$

Solving this equation for $u_{i,j}^{(k+1)}$ one can see that given the gray value from time step (k) one can compute the gray value at time step $(k+1)$ explicitly:

$$u_{i,j}^{(k+1)} = u_{i,j}^{(k)} + \tau \cdot \left(\frac{u_{i-1,j}^{(k)} - 2u_{i,j}^{(k)} + u_{i+1,j}^{(k)}}{h_x^2} + \frac{u_{i,j-1}^{(k)} - 2u_{i,j}^{(k)} + u_{i,j+1}^{(k)}}{h_y^2} \right).$$

Hence this scheme is called *explicit scheme*. Setting the spatial grid sizes h_x and h_y to 1 we get

$$u_{i,j}^{(k+1)} = u_{i,j}^{(k)} + \tau \cdot \left(u_{i-1,j}^{(k)} - 2u_{i,j}^{(k)} + u_{i+1,j}^{(k)} + u_{i,j-1}^{(k)} - 2u_{i,j}^{(k)} + u_{i,j+1}^{(k)} \right),$$

which we can rewrite to

$$u_{i,j}^{(k+1)} = \tau \cdot u_{i-1,j}^{(k)} + \tau \cdot u_{i+1,j}^{(k)} + \tau \cdot u_{i,j-1}^{(k)} + \tau \cdot u_{i,j+1}^{(k)} + (1 - 4 \cdot \tau) \cdot u_{i,j}^{(k)},$$

showing that the new gray value of pixel (i,j) at time step $k+1$ is a convex combination of the gray values of pixels $(i-1,j), (i+1,j), (i,j-1), (i,j+1)$, and (i,j) at time step (k) if

$$1 - 4 \cdot \tau > 0 \quad \Leftrightarrow \quad \tau < \frac{1}{4}.$$

A different notation can be given by the stencil which is an equivalent but graphical representation of the above equation:

0	τ	0
τ	$1 - 4 \cdot \tau$	τ
0	τ	0

In this notation the middle cell represents the pixel (i,j) and the other cells represent the adjacent cells (we use the same orientation as a two-dimensional Cartesian coordinate system):

$(i-1, j+1)$	$(i, j+1)$	$(i+1, j+1)$
$(i-1, j)$	(i, j)	$(i+1, j)$
$(i-1, j-1)$	$(i, j-1)$	$(i+1, j-1)$

Each cell contains the weight of the gray value of the represented pixel. For the computation of the gray value of pixel (i, j) at time step $(k + 1)$ one multiplies the given weights with the corresponding gray values and sums up.
In order to not violate the Neumann boundary conditions there are two possibilities. The first possibility is to reflect the boundary pixel before iterating the filter: extending the image at the image boundary by one pixel and taking the gray value of a former boundary pixel as gray value of the pixel that has extended the image. By doing so, one has always two pixels with the same gray value at the image boundary. Those two pixels with equal gray values will give a vanishing gradient across the boundary and hence the Neumann boundary condition will be respected. Before performing another iteration one then has to update the new boundary pixels. This is because they have not been changed by the iteration but they have to be equal to the boundary pixels of the original image for the next iteration. After the last iteration one only has to discard the additionally created boundary pixels to obtain the final filtering result.
The second possibility is to treat pixels at boundaries differently from inner pixels: for instance the equation for computing the new gray value of an edge pixel at the left boundary ($i = 1, j \neq 1 \vee j \neq n_y$) is given by

$$u_{1,j}^{(k+1)} = \tau \cdot u_{2,j}^{(k)} + \tau \cdot u_{1,j-1}^{(k)} + \tau \cdot u_{1,j+1}^{(k)} + (1 - 3 \cdot \tau) \cdot u_{1,j}^{(k)},$$

while the equation for the left bottom corner pixel ($i = j = 1$) is given by:

$$u_{1,1}^{(k+1)} = \tau \cdot u_{2,1}^{(k)} + \tau \cdot u_{1,2}^{(k)} + (1 - 2 \cdot \tau) \cdot u_{1,j}^{(k)}.$$

Here we can see that for an edge pixel only three neighboring pixels are considered and for a corner pixel only two neighboring pixels are considered. There is no $u_{0,j}^{(k)}, u_{0,1}^{(k)}$, or $u_{1,0}^{(k)}$ pixel as it would occur in the unmodified equation of the explicit scheme).

2.4.2 Properties

We will briefly investigate some useful properties of the diffusion filter presented in section 2.4. In order to do so, we will use the so-called Discrete Scale-Space Theory. We will not go into further detail about these theories, instead, we will present the results of applying Discrete Scale-Space Theory.
But first let us introduce another notation of the linear diffusion filter. The notation is required in order to analyze the diffusion filter. In the section above we have presented a formulation of the filter that describes how one has to compute the new value of one pixel from the old values of pixels. We will now formulate the filter as a matrix-vector multiplication. The image $u^{(k)}$ at time step (k) with its gray values $u_{i,j}^{(k)}, i = 1, \ldots, n_x,\ j = 1, \ldots, n_y$ will be written as a vector

$$u^{(k)} = \big(u_{1,1}^{(k)}, u_{2,1}^{(k)}, \ldots, u_{n_x,1}^{(k)}, u_{1,2}^{(k)}, u_{2,2}^{(k)}, \ldots, u_{n_x,2}^{(k)}, \ldots, u_{1,n_y}^{(k)}, \ldots, u_{n_x,n_y}^{(k)}\big)^\top,$$

(see figure 2.2). So, in $u^{(k)}$ we have a consecutive ordering of all pixels of the image. We first have the gray values of the bottom line (starting with the left pixel and ending with the right pixel) of the image, followed by the second line from the bottom, followed by the third line, and so forth. Then the filter can be written as

$$\frac{u^{(k+1)} - u^{(k)}}{\tau} = A\big(u^{(k)}\big) \cdot u^{(k)},$$

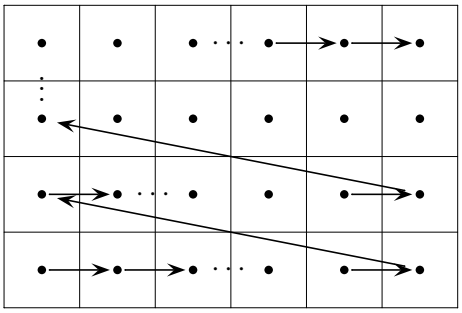


Figure 2.2: Consecutive ordering of all image pixels for matrix-vector notation.

where the system matrix $A\left(u^{(k)}\right)$ looks as follows:

$$A\big(u^{(k)}\big) = \Big(a_{i',j'}\big(u^{(k)}\big)\Big) = \begin{cases} -\left|\mathcal{N}(i')\right|, & \text{if } i' = j' \\ 1, & \text{if } j' \in \mathcal{N}(i') \\ 0, & \text{if } j' \notin \mathcal{N}(i'). \end{cases}$$

Here $\mathcal{N}(i')$ denotes the set of indices j' such that the pixel represented by the j'th component of $u^{(k)}$ is a neighbor of the pixel represented by the i'th component of $u^{(k)}$. $|\mathcal{N}(i')|$ then denotes the number of neighbor pixels. If the original image is n_x pixels wide and n_y pixels high then the following holds:

$$\begin{aligned} \mathcal{N}(i') = &\begin{cases} \{i'-1\}, & (i'-1) \nmid n_x \\ \emptyset, & \text{else} \end{cases} && \text{left neighbor} \\ \cup &\begin{cases} \{i'+1\}, & i' \nmid n_x \\ \emptyset, & \text{else} \end{cases} && \text{right neighbor} \\ \cup &\begin{cases} \{i'-n_x\}, & i' > n_x \\ \emptyset, & \text{else} \end{cases} && \text{bottom neighbor} \\ \cup &\begin{cases} \{i'+n_x\}, & i' \leqslant n_x \cdot (n_y - 1) \\ \emptyset, & \text{else} \end{cases} && \text{top neighbor} \end{aligned}$$

Depending what pixel we are considering that pixel has either two neighbor pixels (corner pixel), three neighbors (edge pixel) or four neighbors (inner pixel). We can now rewrite the equation in such a way that one can compute the next iteration by one matrix-vector-multiplication:

$$\begin{aligned} \frac{u^{(k+1)} - u^{(k)}}{\tau} &= A\big(u^{(k)}\big) \cdot u^{(k)}, \\ u^{(k+1)} &= \big(I + \tau \cdot A\big(u^{(k)}\big)\big) \cdot u^{(k)}. \end{aligned}$$

Here I is the $(n_x \cdot n_y) \times (n_x \cdot n_y)$ unit matrix. Setting $Q := I + \tau \cdot A$ we obtain

$$u^{(k+1)} = Q\big(u^{(k)}\big) \cdot u^{(k)}.$$

As we can see from the definition $Q\left(u^{(k)}\right)$ does not depend on $u^{(k)}$, so, in this case we can simply write Q. Q is a pentadiagonal matrix. The five diagonals arise from the fact that the new gray value of an (inner) pixel is computed using the weighted old gray value at that pixel and the weighted four old gray values of its four neighbor pixels (compare the stencil notation of the linear diffusion filter).

Discrete Scale-Space Theory

Given a discretized diffusion filter that fulfills certain conditions the Discrete Scale-Space Theory states properties the given filter has. Besides the Discrete Scale-Space Theory there is a Continuous and a Semi-Discrete Scale-Space Theory as well. Those theories can be used if one wants to analyze continuous or semi-discrete (continuous in time, but discrete in space) diffusion filters.
The Discrete Scale-Space Theory now states (see [14], [1]): Let $f \in \mathbb{R}^n$, $J = \{1, \ldots, n\}$ and compute $\left(u^{(k)}\right)_{k \geqslant 0}$ by

$$\begin{aligned} u^{(0)} &= f, \\ u^{(k+1)} &= Q\left(u^{(k)}\right) \cdot u^{(k)}, \end{aligned}$$

where $Q\left(u^{(k)}\right)$ is an $n \times n$-matrix continuously depending on $u^{(k)}$ and satisfying the following six conditions $\forall i, j \in J$:

1. Continuity of the Argument: $Q \in C\left(\mathbb{R}^n, \mathbb{R}^{n \times n}\right)$,
2. Symmetry: $q_{ij} = q_{ji}$,
3. Unit Row Sum: $\sum_{j \in J} q_{ij} = 1 \ \forall i$,
4. Non-negativity: $q_{ij} \geqslant 0$,
5. Positive Diagonal: $q_{ii} > 0$,
6. Irreducibility: $\forall i, j \, \exists k_0, \ldots, k_r : k_0 = i \wedge k_r = j \wedge \forall p = 0, \ldots, r-1 : q_{k_p k_{p+1}} \neq 0$.

Then the scheme provides the following properties:

1. Well-Posedness: a unique solution $u^{(k)}$ exists for every k, the solution depends continuously on f,
2. Average Gray Level Invariance: $\frac{1}{n} \sum_{j \in J} u_j^{(k)} = \mu, \ \forall k \geqslant 0$, with $\mu := \frac{1}{n} \sum_{j \in J} f_j$,
3. Extremum Principle: $\min_{j \in J} f_j \leqslant u_i^{(k)} \leqslant \max_{j \in J} f_j, \ \forall i \in J, \ \forall k \geqslant 0$,
4. Smoothing Lyapunov Sequences:
   - the p-norms $||u^{(k)}||_p := \left(\sum_{i=1}^N |u_i^{(k)}|^p \right)^{\frac{1}{p}}$ are decreasing in k for all $p \geqslant 1$,
   - all even central moments $M_{2m}[u^{(k)}] := \dfrac{1}{n} \sum_{j=1}^n (u_j^{(k)} - \mu)^{2 \cdot m}, m \in \mathbb{N}$ are decreasing in k,
   - the entropy $S[u^{(k)}] := -\sum_{j=1}^n u_j^{(k)} \cdot \ln u_j^{(k)}$ is increasing in k (if f_j is positive for all j),
5. Convergence to a Constant Steady-State: $\lim_{k \to \infty} u_i^{(k)} = \mu, \ \forall i \in J$.

Now let us verify that the linear diffusion filter satisfies the conditions mentioned above (for further information see [15], [17], [7], [12]):

1. Continuity of the argument is satisfied trivially, since Q does not depend on $u^{(k)}$.

2. Symmetry follows from symmetry of A and this on the other hand follows from the fact that i' is a neighbor of j' if and only if j' is a neighbor of i'.

3. By the definition of A one can see that A has vanishing row sum. Then by construction Q has unit row sum.

4. Non-negativity and positive diagonal is satisfied if and only if $1 - 4 \cdot \tau > 0$ which holds if $\tau < \frac{1}{4}$. Hence we are only allowed to use time step sizes smaller than $\frac{1}{4}$.

5. Irreducibility follows from the fact that for any two pixels i', j' and for any path $i' = k_0, \ldots, k_r = j'$ connecting them $q_{k_i, k_{i+1}} > 0$ for all $i = 0, \ldots, r-1$.

Hence the linear diffusion filter

$$u^{(k+1)} = Q \cdot u^{(k)}$$

with Q from above and $\tau < \frac{1}{4}$ has a unique solution for every time step. The average gray value or average gray level is preserved in every iteration step. The filter fulfills the extremum principle or minimum maximum principle. This means a gray value never becomes smaller than the minimal gray value of the original image and it never becomes greater than the maximal gray value of the original image. So, no over- or undershoots occur. Furthermore, the filter creates with increasing time simpler versions of the original image. And lastly the filter converges to a steady state: the average gray value.
Above we have chosen $g = 1$ as diffusivity for the sake of simplicity. Note that we could also choose any other constant as diffusivity: g represents the speed of the continuous diffusion process. Choosing g different to 1 would simply change the restriction on the time step size τ in the explicit scheme. Carrying g through the discretization one would obtain

$$1 - 4 \cdot \tau \cdot g > 0$$

as time step size restriction for τ. This is equivalent to

$$\tau < \frac{1}{4 \cdot g.}$$

We see that choosing, for instance, $g = 0.5$ a time step size smaller than 0.5 would be admissible for the explicit scheme then. Remember that the product of iterations and time step size represents the diffusion time of the continuous diffusion process. So, slowing down the diffusion process allows us to use greater a time step size. Hence it does not matter how g is chosen: the maximal diffusion time per iteration is always the same.

Gaussian Convolution and Linear Diffusion

One can show that the linear diffusion as described above is an approximation of a convolution of the image with a (two-dimensional) Gaussian kernel. The (continuous) convolution of two functions g and w is defined as

$$\big(g * w\big)(x) := \int g(x - x') \cdot w(x') \, \mathrm{d}x',$$

the discrete convolution of $g = (g_i)$ and $w = (w_i)$ is given by

$$\big(g * w\big)_i := \sum_k g_{i-k} \cdot w_k.$$

Convolution of g with w can be regarded as averaging of the components of g with weights. The weights are the mirrored components of w.

An m-dimensional Gaussian kernel with kernel width σ is defined by

$$K_\sigma(x) := \frac{1}{\left(2 \cdot \pi \cdot \sigma^2\right)^{\frac{m}{2}}} \cdot \exp\left(-\frac{|x|^2}{2 \cdot \sigma^2}\right),$$

σ is also called standard deviation and σ^2 is the variance. The convolution with a Gaussian kernel has two interesting properties: First consecutive convolution of an image with Gaussian kernels of variances σ_1^2 and σ_2^2 is equivalent to convolution of the image with one Gaussian kernel of variance $\sigma_1^2 + \sigma_2^2$. This means that iterating the explicit scheme of the linear diffusion simply approximates a (continuous) convolution of the image with a Gaussian kernel of increasing standard deviation and variance. In fact, the following relation between the diffusion time t of linear diffusion and the kernel width σ of a Gaussian convolution exists:

$$t = \frac{1}{2} \cdot \sigma^2,$$

and hence

$$\sigma = \sqrt{2 \cdot t}.$$

The second interesting property of Gaussian convolution is its separability: Instead of convolving an image with a two-dimensional Gaussian kernel of width σ one can first convolve the image with an one-dimensional Gaussian kernel K_σ in x-dimension and then convolve the result with a one-dimensional Gaussian kernel K_σ in y-dimension. Since linear diffusion is an approximation of continuous Gaussian convolution and Gaussian convolution is separable the linear diffusion process is also separable:

$$u^{(k+1)} = Q \cdot u^{(k)} \approx Q_y \cdot Q_x \cdot u^{(k)},$$

with

$$Q_x = \begin{cases} 1 - \sum_{\mathcal{N}_x(i')} \tau, & \text{if } i' = j' \\ \tau, & \text{if } j' \in \mathcal{N}_x(i') \\ 0, & \text{if } j' \notin \mathcal{N}_x(i'). \end{cases}$$

$$Q_y = \begin{cases} 1 - \sum_{\mathcal{N}_y(i')} \tau, & \text{if } i' = j' \\ \tau, & \text{if } j' \in \mathcal{N}_y(i') \\ 0, & \text{if } j' \notin \mathcal{N}_y(i'). \end{cases}$$

and

$$\begin{aligned} \mathcal{N}_x(i') = & \begin{cases} \{i' - 1\}, & (i' - 1) \nmid n_x \\ \emptyset, & \text{else} \end{cases} && \text{left neighbor} \\ \cup & \begin{cases} \{i' + 1\}, & i' \nmid n_x \\ \emptyset, & \text{else} \end{cases} && \text{right neighbor} \\ \mathcal{N}_y(i') = & \begin{cases} \{i' - n_x\}, & i' > n_x \\ \emptyset, & \text{else} \end{cases} && \text{bottom neighbor} \\ \cup & \begin{cases} \{i' + n_x\}, & i' \leqslant n_x \cdot (n_y - 1) \\ \emptyset, & \text{else} \end{cases} && \text{top neighbor} \end{aligned}$$

2.4.3 Semi-implicit Scheme

Let us have a look at the linear diffusion filter formulated as matrix-vector-multiplication:

$$\frac{u^{(k+1)} - u^{(k)}}{\tau} = A\big(u^{(k)}\big) \cdot u^{(k)}.$$

As we have seen this discretization allows us to compute the new gray values directly from the old ones. However, this is not the only possibility to formulate this filter. One can also consider the following formulations:

$$\frac{u^{(k+1)} - u^{(k)}}{\tau} = A\big(u^{(k)}\big) \cdot u^{(k+1)},$$

leading to the so-called *semi-implicit scheme*. Here we exchange the old values $u^{(k)}$ which are multiplied with $A\big(u^{(k)}\big)$ by the new ones $u^{(k+1)}$. By this exchange the new gray values are no longer explicitly computable. In order to compute them one has to solve the linear system of equations

$$\big(I - \tau \cdot A\big(u^{(k)}\big)\big) \cdot u^{(k+1)} = u^{(k)}.$$

This scheme is called semi-implicit, since one has to solve a system of equations in order to compute the gray values at the next iteration. So, the new values are given implicitly be the system of equations. For the matrix $A\big(u^{(k)}\big)$ still the old values are used. So, the system of equations is linear. Now one has to solve a system of equations which is more difficult than simply computing the new values explicitly. By setting

$$Q := \big(I - \tau \cdot A\big(u^{(k)}\big)\big)^{-1}$$

one can also analyze the properties of the semi-implicit scheme using the Discrete Scale-Space Theory. It turns out that the semi-implicit scheme has the same properties as the explicit scheme. However, it does not have a time step size restriction as it is the case for the explicit scheme. So arbitrarily large time steps can be used. In order to achieve a certain filtering result one can now either perform few iterations of the semi-implicit scheme (with a big time step size τ) or perform many iterations of the explicit scheme (with a time step size smaller than $\frac{1}{4}$). Using the new values for matrix $A\big(u^{(k+1)}\big)$ as well we obtain

$$\frac{u^{(k+1)} - u^{(k)}}{\tau} = A\big(u^{(k+1)}\big) \cdot u^{(k+1)},$$

the *fully-implicit scheme*. In order to compute the new gray values in this case one has to solve the following system of equations

$$\big(I - \tau \cdot A\big(u^{(k+1)}\big)\big) \cdot u^{(k+1)} = u^{(k)},$$

which is in general no longer linear. An iteration of the fully-implicit scheme is much more expensive, since one has to solve a nonlinear system of equations. We will not go into further detail about the fully-implicit scheme here. However, let us point out that since in linear diffusion the matrix A does not depend on $u^{(k)}$ (in the semi-implicit case) or $u^{(k+1)}$ (in the fully-implicit case) the semi-implicit scheme and fully-implicit scheme are in this case identical.

For solving the linear system of equations

$$Q \cdot u^{(k+1)} = u^{(k)},$$

of the semi-implicit scheme of linear diffusion one could use common algorithms, for instance the Jacobi method, the Gauß-Seidel method or Successive Over-relaxation (see [18]). Since the linear diffusion is separable we do not have to use those methods. Instead, we make usage of the separability:

$$Q \cdot u^{(k+1)} \approx Q_y \cdot Q_x \cdot u^{(k+1)} = u^{(k)}.$$

By splitting the linear diffusion into a diffusion along the x-axis first and then into a diffusion along the y-axis we now do not have to solve the linear system of equations

$$Q \cdot u^{(k+1)} = u^{(k)},$$

where Q is a pentadiagonal matrix. Instead, we first solve the tridiagonal system

$$Q_y \cdot u^{(k')} = u^{(k)}$$

for $u^{(k')}$ and then solve another tridiagonal system

$$Q_x \cdot u^{(k+1)} = u^{(k')}$$

for $u^{(k+1)}$. Tridiagonal systems of equations can be solved efficiently by the Thomas Algorithm.

Thomas Algorithm

The Thomas algorithm or tridiagonal matrix algorithm is a fast and efficient algorithm to solve tridiagonal system of equations (see [13]). It is a special case of solving linear systems of equations by Gaussian elimination. Given the tridiagonal system of equations

$$B \cdot x = d,$$

where the system matrix A is of the form

$$B = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ \gamma_1 & \alpha_2 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \gamma_{n-2} & \alpha_{n-1} & \beta_{n-1} \\ & & & \gamma_{n-1} & \alpha_n \end{pmatrix}$$

and $x \in \mathbb{R}^n$ and $d \in \mathbb{R}^n$. The Thomas algorithm solves this system in three steps:

- In the first step the matrix B is decomposed into two matrices: a lower bidiagonal matrix L and an upper bidiagonal matrix R with

$$L = \begin{pmatrix} 1 & & & & \\ l_1 & 1 & & & \\ & \ddots & \ddots & & \\ & & l_{n-2} & 1 & \\ & & & l_{n-1} & 1 \end{pmatrix} \quad \text{and} \quad R = \begin{pmatrix} m_1 & r_1 & & & \\ & m_2 & r_2 & & \\ & & \ddots & \ddots & \\ & & & m_{n-1} & r_{n-1} \\ & & & & m_n \end{pmatrix}.$$

Comparing the coefficients of

$$L \cdot R = \begin{pmatrix} m_1 & r_1 & & & \\ l_1 m_1 & l_1 r_1 + m_2 & r_2 & & \\ & \ddots & \ddots & \ddots & \\ & & l_{n-1} m_{n-1} & l_{n-2} r_{n-2} + m_{n-1} & r_{n-1} \\ & & & l_n m_n & l_{n-1} r_{n-1} + m_n \end{pmatrix}$$

with those of A show that $r_i = \beta_i$ for $1 \leqslant i \leqslant n-1$. The coefficients l_i and m_i are given by

$$\begin{aligned} & m_1 := \alpha_1 \\ & \texttt{for } i = 1, \ldots, n-1 : \\ & \qquad l_i := \frac{\gamma_i}{m_i} \\ & \qquad m_{i+1} := \alpha_{i+1} - l_i \cdot \beta_i \end{aligned}$$

- In the second step the system $L \cdot y = d$ is solved for y by

$$\begin{aligned} & y_1 := d_1 \\ & \texttt{for } i = 2, \ldots, n : \\ & \qquad y_i := d_i - l_{i-1} \cdot y_{i-1} \end{aligned}$$

- In the third step the system $R \cdot x = y$ is solved for x by

$$\begin{aligned} & x_n := \frac{y_n}{m_n} \\ & \texttt{for } i = n-1, \ldots, 1 : \\ & \qquad x_i := \frac{y_i - \beta_i \cdot x_{i+1}}{m_i} \end{aligned}$$

The Thomas algorithm is stable for strictly diagonally dominant tridiagonal matrices. This is the case for our semi-implicit discretization. Since the semi-implicit discretization does not have a restriction on the time step size τ we are able to perform a Gaussian convolution of the image by one single iteration in which we have to apply the Thomas algorithm twice.

2.5 Nonlinear Isotropic Diffusion

In the previous section we have presented the linear diffusion filter. The linear diffusion filter is inspired by the differential equation

$$\partial_t u = \operatorname{div}(g \cdot \nabla u),$$

where g is set to 1. Although that filter has some preferable properties the visual results are not very good: the image is smoothed in the same way in every point. In this section we will present a generalization of linear diffusion filter: the nonlinear isotropic diffusion (see [3], [14]). We will present discretizations of the nonlinear isotropic diffusion. Furthermore, we will present properties of this diffusion process.

Nonlinear isotropic diffusion is motivated by the same differential equation as linear diffusion:

$$\partial_t u = \operatorname{div}(g \cdot \nabla u).$$

However, here we do not set g to a constant. Instead, g will depend on the image u. More precisely g will depend on the length of the image gradient $|\nabla u|$:

$$\partial_t u = \operatorname{div}\big(g(\,|\nabla u|\,) \cdot \nabla u\big).$$

The idea behind it is simply: In order to preserve edges in the image for a longer time than in the linear diffusion filter the diffusivity depends on the image gradient ∇u. The length of the image gradient is high at edges, low in regions with small gray value perturbations, and zero in a flat constant region. The higher the gradient length the bigger the differences between neighboring pixels. g is now a function of $|\nabla u|^2$ which is equal to 1 if the gradient vanishes and which decreases for increasing gradient.
Noise is often present as small gray value perturbations in the images. These small gray value perturbations create high gradients in the image which can be misinterpreted as edges that one would like to preserve. To circumvent this misinterpretations one performs a presmoothing of the image before the computation of the gradients. The presmoothing is done by linear diffusion as described in section 2.4.3:

$$\partial_t u = \operatorname{div}\big(g(\,|\nabla u_\sigma|\,) \cdot \nabla u\big),$$

where σ is a parameter indicating the diffusion time of linear diffusion of the image for g.
One uses the term nonlinear isotropic diffusion to distinguish this type of diffusion from nonlinear anisotropic diffusion. In the isotropic case the diffusivity function g returns a real value $g : \mathbb{R} \mapsto \mathbb{R}$. Here the diffusion process equilibrates concentration differences in all directions in the same way. On the other hand in the anisotropic case g returns a 2×2-matrix, $g : \mathbb{R} \mapsto \mathbb{R}^{2\times 2}$. This allows to treat concentration differences in different directions separately. For instance nonlinear anisotropic diffusion allows to reduce diffusion across an edge while smoothing along the edge as it is the case in edge-enhancing diffusion. Nonlinear anisotropic diffusion is a generalization of nonlinear isotropic diffusion. We will not go into further detail about anisotropic diffusion here.

2.5.1 Explicit Scheme

In order to develop an explicit scheme for nonlinear diffusion we have to discretize the partial differential equation

$$\partial_t u = \operatorname{div}\big(g(\,|\nabla u_\sigma|\,) \cdot \nabla u\big)$$

(see section 2.4.1). This equation can be rewritten to

$$\partial_t u = \operatorname{div}\left(g(\,|\nabla u_\sigma|\,) \cdot \begin{pmatrix}\partial_x u \\ \partial_y u\end{pmatrix}\right),$$

which is equivalent to

$$\partial_t u = \partial_x\big(g(\,|\nabla u_\sigma|\,) \cdot \partial_x u\big) + \partial_y\big(g(\,|\nabla u_\sigma|\,) \cdot \partial_y u\big).$$

Discretization

Using our abbreviations for partial derivatives and writing g instead of $g_\sigma\big(|\nabla u_\sigma|\big)$, we obtain

$$u_t = (g \cdot u_x)_x + (g \cdot u_y)_y \,.$$

Now we will use the following approximations:

$$\begin{aligned}
u_t &\approx \frac{u_{i,j}^{(k+1)} - u_{i,j}^{(k)}}{\tau}, \\
(g \cdot u_x)_x &\approx \frac{1}{h_x} \cdot \Big(\big(g \cdot u_x\big)_{i+\frac{1}{2},j}^{(k)} - \big(g \cdot u_x\big)_{i-\frac{1}{2},j}^{(k)} \Big) \\
&\approx \frac{1}{h_x} \cdot \left(g_{i+\frac{1}{2},j}^{(k)} \cdot \frac{u_{i+1,j}^{(k)} - u_{i,j}^{(k)}}{h_x} - g_{i-\frac{1}{2},j}^{(k)} \cdot \frac{u_{i,j}^{(k)} - u_{i-1,j}^{(k)}}{h_x} \right) \\
&= \frac{1}{h_x^2} \cdot \left(g_{i+\frac{1}{2},j}^{(k)} \cdot \Big(u_{i+1,j}^{(k)} - u_{i,j}^{(k)}\Big) - g_{i-\frac{1}{2},j}^{(k)} \cdot \Big(u_{i,j}^{(k)} - u_{i-1,j}^{(k)}\Big) \right)
\end{aligned}$$

and accordingly

$$\begin{aligned}
(g \cdot u_y)_y &\approx \frac{1}{h_y} \cdot \Big(\big(g \cdot u_y\big)_{i,j+\frac{1}{2}}^{(k)} - \big(g \cdot u_y\big)_{i,j-\frac{1}{2}}^{(k)} \Big) \\
&\approx \frac{1}{h_y} \cdot \left(g_{i,j+\frac{1}{2}}^{(k)} \cdot \frac{u_{i,j+1}^{(k)} - u_{i,j}^{(k)}}{h_y} - g_{i,j-\frac{1}{2}}^{(k)} \cdot \frac{u_{i,j}^{(k)} - u_{i,j-1}^{(k)}}{h_y} \right) \\
&= \frac{1}{h_y^2} \cdot \left(g_{i,j+\frac{1}{2}}^{(k)} \cdot \Big(u_{i,j+1}^{(k)} - u_{i,j}^{(k)}\Big) - g_{i,j-\frac{1}{2}}^{(k)} \cdot \Big(u_{i,j}^{(k)} - u_{i,j-1}^{(k)}\Big) \right).
\end{aligned}$$

The resulting explicit scheme is then given by

$$\begin{aligned}
\frac{u_{i,j}^{(k+1)} - u_{i,j}^{(k)}}{\tau} &= \frac{1}{h_x^2} \cdot \left(g_{i+\frac{1}{2},j}^{(k)} \cdot \Big(u_{i+1,j}^{(k)} - u_{i,j}^{(k)}\Big) - g_{i-\frac{1}{2},j}^{(k)} \cdot \Big(u_{i,j}^{(k)} - u_{i-1,j}^{(k)}\Big) \right) \\
&\quad + \frac{1}{h_y^2} \cdot \left(g_{i,j+\frac{1}{2}}^{(k)} \cdot \Big(u_{i,j+1}^{(k)} - u_{i,j}^{(k)}\Big) - g_{i,j-\frac{1}{2}}^{(k)} \cdot \Big(u_{i,j}^{(k)} - u_{i,j-1}^{(k)}\Big) \right),
\end{aligned}$$

which can be rewritten to

$$\begin{aligned}
u_{i,j}^{(k+1)} = u_{i,j}^{(k)} &+ \frac{\tau}{h_x^2} \cdot \left(g_{i+\frac{1}{2},j}^{(k)} \cdot \Big(u_{i+1,j}^{(k)} - u_{i,j}^{(k)}\Big) - g_{i-\frac{1}{2},j}^{(k)} \cdot \Big(u_{i,j}^{(k)} - u_{i-1,j}^{(k)}\Big) \right) \\
&+ \frac{\tau}{h_y^2} \cdot \left(g_{i,j+\frac{1}{2}}^{(k)} \cdot \Big(u_{i,j+1}^{(k)} - u_{i,j}^{(k)}\Big) - g_{i,j-\frac{1}{2}}^{(k)} \cdot \Big(u_{i,j}^{(k)} - u_{i,j-1}^{(k)}\Big) \right).
\end{aligned}$$

The computation of the next iteration can be also expressed as weighted averaging of the central pixel and its neighbor pixels

$$\begin{aligned}
u_{i,j}^{(k+1)} &= \frac{\tau}{h_x^2} \cdot g_{i+\frac{1}{2},j}^{(k)} \cdot u_{i+1,j}^{(k)} + \frac{\tau}{h_x^2} \cdot g_{i-\frac{1}{2},j}^{(k)} \cdot u_{i-1,j}^{(k)} \\
&+ \frac{\tau}{h_y^2} \cdot g_{i,j+\frac{1}{2}}^{(k)} \cdot u_{i,j+1}^{(k)} + \frac{\tau}{h_y^2} \cdot g_{i,j-\frac{1}{2}}^{(k)} \cdot u_{i,j-1}^{(k)} \\
&+ \left(1 - \frac{\tau}{h_x^2} \cdot g_{i+\frac{1}{2},j}^{(k)} - \frac{\tau}{h_x^2} \cdot g_{i-\frac{1}{2},j}^{(k)} - \frac{\tau}{h_y^2} \cdot g_{i,j+\frac{1}{2}}^{(k)} - \frac{\tau}{h_y^2} \cdot g_{i,j-\frac{1}{2}}^{(k)} \right) \cdot u_{i,j}^{(k)}
\end{aligned}$$

or in stencil notation (for $h_x = h_y = 1$):

0	$\tau \cdot g^{(k)}_{i,j+\frac{1}{2}}$	0
$\tau \cdot g^{(k)}_{i-\frac{1}{2},j}$	$-\tau \cdot g^{(k)}_{i,j+\frac{1}{2}}$ $-\tau \cdot g^{(k)}_{i-\frac{1}{2},j} + 1 - \tau \cdot g^{(k)}_{i+\frac{1}{2},j}$ $-\tau \cdot g^{(k)}_{i,j-\frac{1}{2}}$	$\tau \cdot g^{(k)}_{i+\frac{1}{2},j}$
0	$\tau \cdot g^{(k)}_{i,j-\frac{1}{2}}$	0

Here $g^{(k)}_{i\pm\frac{1}{2},j}$ and $g^{(k)}_{i,j\pm\frac{1}{2}}$ are approximations of the image gradient at time step (k) at positions $(i \pm \frac{1}{2}, j)$ and $(i, j \pm \frac{1}{2})$, respectively:

$$g^{(k)}_{i\pm\frac{1}{2},j} \approx \frac{g^{(k)}_{i\pm1,j} + g^{(k)}_{i,j}}{2},$$

and

$$g^{(k)}_{i,j\pm\frac{1}{2}} \approx \frac{g^{(k)}_{i,j\pm1} + g^{(k)}_{i,j}}{2}.$$

The approximations of the image gradient between the pixel grid positions are computed as arithmetic mean of the corresponding approximations of the image gradient at pixel grid positions. Note that for computing the image gradients one performs a presmoothing $u^{(k)}_\sigma$ of the original image u in order to avoid misinterpretations of noise as edges. This presmoothing is performed before each iteration (k). $g^{(k)}_{i,j}$ is approximated as follows:

$$\begin{aligned} g^{(k)}_{i,j} &= g\left(\sqrt{(u_\sigma)^2_x + (u_\sigma)^2_y}\right)\Big|^{(k)}_{i,j} \\ &\approx g\left(\sqrt{\frac{\left(u^{(k)}_\sigma\right)_{i+1,j} - \left(u^{(k)}_\sigma\right)_{i-1,j}}{2 \cdot h_x} + \frac{\left(u^{(k)}_\sigma\right)_{i,j+1} - \left(u^{(k)}_\sigma\right)_{i,j-1}}{2 \cdot h_y}}\right), \end{aligned}$$

where g is a diffusivity function.

Diffusivity Functions

Let us now specify the diffusivity function g. As explained above the diffusivity function steers the behavior of the diffusion process. In order to preserve edges we want to reduce the diffusion at edges. This means the diffusivity function should decrease towards 0 for increasing lengths of image gradients. In constant regions were the image gradient vanishes we want to perform linear diffusion. This means the diffusivity function should increase towards 1 for decreasing lengths of image gradients and become 1 for vanishing image gradients. Common choices for g are the *Charbonnier diffusivity*

$$g\big(\,|\nabla u|\,\big) = \frac{1}{\sqrt{1 + \frac{|\nabla u|^2}{\lambda^2}}}$$

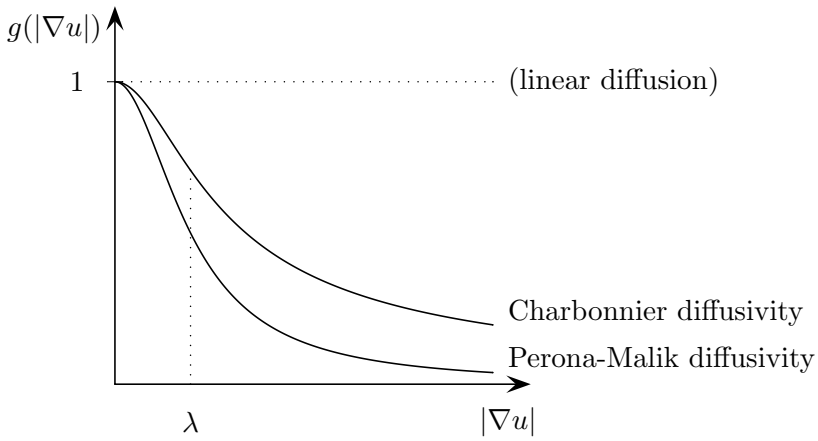


Figure 2.3: Behavior of different diffusivity functions for increasing image gradient $|\nabla u|$.

and the *Perona-Malik diffusivity* (see [10])

$$g\big(\,|\nabla u|\,\big) = \frac{1}{1 + \frac{|\nabla u|^2}{\lambda^2}}$$

(see figure 2.3).
Here λ is a parameter distinguishing between edges and constant regions, the so-called contrast parameter. For edges we have $|\nabla u| > \lambda$. Then the denominator of both diffusivity functions becomes large and the whole fraction becomes small. When the diffusivity becomes small the diffusion process is slowed down which preserves edges for a longer time. Whereas for constant regions and regions with small gray value perturbations, we have $|\nabla u| < \lambda$. Then the denominator in both diffusivity functions comes close to 1 and the value of the whole fraction comes close to 1 as well. The diffusivity then equilibrates small gray value differences faster. When choosing a smaller λ more image features will be treated as edges where the diffusion will be slowed down. Choosing a larger λ less images features will be treated as edges and bigger parts of the image will be diffused like in the linear diffusion process. In the limiting case where λ goes to infinity ($\lambda \to \infty$) all image features are treated as regions where normal linear diffusion is performed. Hence by choosing large values for λ the nonlinear isotropic diffusion filter will have the same behavior as the linear diffusion filter.
Given the same λ the Charbonnier diffusivity function treats less image features as edges and hence performs more similar to a linear diffusion than the Perona-Malik diffusivity function.

2.5.2 Semi-implicit Scheme

For deriving the semi-implicit scheme let us first rewrite the explicit scheme in matrix-vector notation. We use the same notation as in section 2.4.2. The gray values of the new time step for nonlinear isotropic diffusion can be computed as follows:

$$\frac{u^{(k+1)} - u^{(k)}}{\tau} = A\big(u^{(k)}\big) \cdot u^{(k)},$$

which can be rewritten as an iteration scheme

$$u^{(k+1)} = \Big(I + \tau \cdot A\big(u^{(k)}\big)\Big) \cdot u^{(k)},$$

with

$$A\big(u^{(k)}\big) = \Big(a_{i',j'}\big(u^{(k)}\big)\Big) = \begin{cases} -\frac{1}{2} \cdot \sum\limits_{n \in \mathcal{N}(i')} \big(g_{i'}^{(k)} + g_n^{(k)}\big), & \text{if } i' = j', \\ \frac{1}{2} \cdot \big(g_{i'}^{(k)} + g_{j'}^{(k)}\big), & \text{if } j' \in \mathcal{N}(i'), \\ 0, & \text{if } j' \notin \mathcal{N}(i'). \end{cases}$$

Here we have set the spatial grid size $h_x = h_y = 1$ for the sake of simplicity. $\mathcal{N}(i')$ denotes the set of neighbor pixels of pixel (i'), where all pixels are numbered consecutively (see figure 2.2). Note that in comparison to section 2.4.2 only the entries in matrix $A\big(u^{(k)}\big)$ have changed. Applying the same steps as in section 2.4.3 we get from the equation

$$\frac{u^{(k+1)} - u^{(k)}}{\tau} = A\big(u^{(k)}\big) \cdot u^{(k+1)},$$

the following semi-implicit scheme for nonlinear isotropic diffusion:

$$\Big(I - \tau \cdot A\big(u^{(k)}\big)\Big) \cdot u^{(k+1)} = u^{(k)},$$

with $A\big(u^{(k)}\big)$ from above. In contrast to the semi-implicit scheme for linear diffusion the matrix A now depends on $u^{(k)}$ since the $g_{i'}^{(k)}$ depend on $u^{(k)}$. Setting

$$Q\big(u^{(k)}\big) := I + \tau \cdot A\big(u^{(k)}\big)$$

in the explicit case and

$$Q\big(u^{(k)}\big) := I - \tau \cdot A\big(u^{(k)}\big)^{-1}$$

in the semi-implicit scheme we can now analyze the properties of the nonlinear isotropic diffusion filter.

2.5.3 Properties

Applying the Discrete Scale-Space Theory from section 2.4.2 we obtain the same properties as for linear diffusion:

1. Well-Posedness: a unique solution $u^{(k)}$ exists for every k, the solution depends continuously on f,
2. Average Gray Level Invariance,
3. Extremum Principle,
4. Smoothing Lyapunov Sequences:
   - the p-norms are decreasing in k for all $p \geqslant 1$,
   - all even central moments are decreasing in k,
   - the entropy is increasing in k (if f_j is positive for all j),
5. Convergence to a Constant Steady-State.

Let us come back to the semi-implicit scheme for nonlinear diffusion from section 2.5.2. Computing the gray values for the next time step now requires to solve linear system of equations. Common iterative algorithms for solving linear system of equations guarantee convergence and do not need additional storage. However, for increasing time step size τ those algorithms convergence slower towards the solution.
Although the nonlinear isotropic diffusion process is not separable it is efficiently solvable by applying a splitting-based technique: the Additive Operator Splitting.

2.5.4 AOS Scheme

The idea behind the additive operator splitting is to decompose a problem in higher dimensions into simpler one-dimensional problems (see [8], [16]). In our case, instead of computing the nonlinear diffusion of an image in two dimension at once, we compute a nonlinear diffusion of the image in x-dimension and a nonlinear diffusion of the image in y-dimension and average both results.
The semi-implicit discretization of nonlinear diffusion is given by

$$\Big(I - \tau \cdot A\big(u^{(k)}\big)\Big) \cdot u^{(k+1)} = u^{(k)}.$$

This can be rewritten to

$$\left(I - \tau \cdot \sum_{i=1}^{m} A_i\big(u^{(k)}\big)\right) \cdot u^{(k+1)} = u^{(k)},$$

where $A_i\big(u^{(k)}\big)$ is the matrix that arises if one considers the nonlinear diffusion process in the ith dimension only. Here m is the number of dimensions. The additive operator splitting scheme, *AOS scheme*, now considers instead of

$$u^{(k+1)} = \left(I - \tau \cdot \sum_{i=1}^{m} A_i\big(u^{(k)}\big)\right)^{-1} \cdot u^{(k)}$$

the split variant

$$u^{(k+1)} = \frac{1}{m} \cdot \sum_{i=1}^{m} \Big(I - m \cdot \tau \cdot A_i\big(u^{(k)}\big)\Big)^{-1} \cdot u^{(k)}.$$

In our case of two-dimensional gray scale images we have

$$u^{(k+1)} = \frac{1}{2} \cdot \sum_{i=1}^{2} \Big(I - 2 \cdot \tau \cdot A_i\big(u^{(k)}\big)\Big)^{-1} \cdot u^{(k)}.$$

Note that all $A_i\big(u^{(k)}\big)$ are now tridiagonal matrices:

$$A_1\big(u^{(k)}\big) = \Big(a_{1,i',j'}\big(u^{(k)}\big)\Big) = \begin{cases} -\frac{1}{2} \cdot \sum\limits_{n \in \mathcal{N}_x(i')} \big(g_{i'}^{(k)} + g_n^{(k)}\big), & \text{if } i' = j', \\ \frac{1}{2} \cdot \big(g_{i'}^{(k)} + g_{j'}^{(k)}\big), & \text{if } j' \in \mathcal{N}_x(i'), \\ 0, & \text{if } j' \notin \mathcal{N}_x(i') \end{cases}$$

and

$$A_2\big(u^{(k)}\big) = \Big(a_{2,i',j'}\big(u^{(k)}\big)\Big) = \begin{cases} -\frac{1}{2} \cdot \sum\limits_{n \in \mathcal{N}_y(i')} \big(g_{i'}^{(k)} + g_n^{(k)}\big), & \text{if } i' = j', \\ \frac{1}{2} \cdot \big(g_{i'}^{(k)} + g_{j'}^{(k)}\big), & \text{if } j' \in \mathcal{N}_y(i'), \\ 0, & \text{if } j' \notin \mathcal{N}_y(i'), \end{cases}$$

where $\mathcal{N}_x(i')$ and $\mathcal{N}_y(i')$ is the set representing the horizontal and vertical neighbor pixels of pixel (i'), respectively. One iteration of the AOS scheme with time step size τ is now performed by computing the nonlinear diffusion in x- and y-dimension, respectively with a time step size of $2 \cdot \tau$ and then averaging the both results. In order to perform the nonlinear diffusion in one dimension one has to solve the tridiagonal linear system of equations which can be done efficiently by the Thomas algorithm (see section 2.4.3). More precisely one also has to apply the Thomas algorithm twice to efficiently smooth the image $u^{(k)}$ before the approximation of the image gradients in the computation of the diffusivities. So, in one AOS iteration one performs the Thomas algorithm either twice (without presmoothing) or four times (with presmoothing). Note that the AOS scheme is not identical to the semi-implicit discretization, but it has the same consistency order as the semi-implicit discretization. Experiments show that one can achieve comparable result to the explicit scheme with a time step size of up to 5.0. Furthermore, the AOS scheme has the same properties as the semi-implicit scheme (see section 2.5.3).
We have introduced the notation, the linear diffusion process, the nonlinear isotropic diffusion process, and their properties. Let us now show our new approach for image denoising using diffusion filters.

Chapter 3

Moment Preserving Diffusion

We will now generalize the nonlinear isotropic diffusion in order to achieve visually better denoising results at minimal additional costs. One of the nice properties of linear and nonlinear isotropic diffusion is their average gray level or average gray value invariance (see sections 2.4.2, 2.5.3):

$$\frac{1}{n_x \cdot n_y} \cdot \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} u_{i,j}^{(k)} = \mu, \ \forall k \geqslant 0,$$

with

$$\mu := \frac{1}{n_x \cdot n_y} \cdot \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} f_{i,j}.$$

The average gray value invariance states that the average gray value of the image stays constant during its evolution. The average gray value can also be expressed as a moment. A moment of order p is defined as:

$$m_p := \frac{1}{n_x \cdot n_y} \cdot \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} f_{i,j}^p.$$

So, the average gray value μ represents the first moment m_1. Furthermore, the average gray value can also be expressed as a central moment. The central moment of order p is defined as:

$$M_p := \frac{1}{n_x \cdot n_y} \cdot \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} \left(f_{i,j} - \mu\right)^p,$$

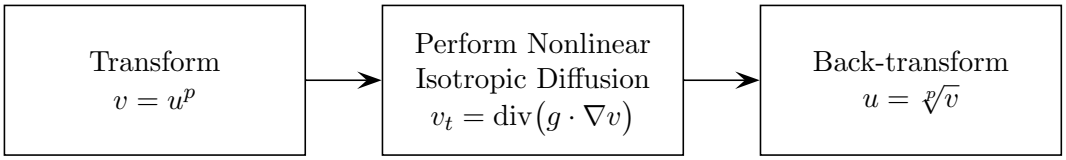
where μ is defined as above. So, in this case we have

$$\begin{aligned} M_1 &= \frac{1}{n_x \cdot n_y} \cdot \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} \left(f_{i,j} - \mu\right)^1 \\ &= \frac{1}{n_x \cdot n_y} \cdot \left(\left(\sum_{i=1}^{n_x} \sum_{j=1}^{n_y} f_{i,j} \right) - \mu \cdot n_x \cdot n_y \right) \\ &= \frac{1}{n_x \cdot n_y} \cdot \left(\sum_{i=1}^{n_x} \sum_{j=1}^{n_y} f_{i,j} \right) - \mu \\ &= m_1 - \mu \qquad = 0, \end{aligned}$$

since $m_1 = \mu$. Hence linear and nonlinear isotropic diffusion preserve the first moment m_1 as well as the first central moment M_1. In our approach we will now generalize nonlinear isotropic diffusion to a nonlinear isotropic diffusion preserving higher moments. Let us explain the idea behind preserving higher moments. Nonlinear isotropic diffusion preserves the first (central) moment $m_1(M_1)$. As already said above it redistributes the gray values of the pixels. This means that all pixels are treated in the same way: independent from their gray values. Given a (noisy) image with dominating bright and dark areas preserving a higher moment promises to achieve better denoising results, since the preservation of a higher moment will lead to different treatment of different gray values of the image.
First we present our approach for a moment preserving diffusion. Then we will investigate which properties of the nonlinear isotropic diffusion carries over the our new approach.

3.1 Preservation of the pth Moment

The idea behind our approach for a moment preserving diffusion process is the average gray level invariance of the linear and nonlinear isotropic diffusion: Given u those processes do not change the average gray value, hence they do not change the total mass. Instead, they only redistribute the given mass. So, in order to preserve the moment of order p we do not diffuse u but u^p instead. The gray level invariance of u^p now translate to a preservation of the pth moment.
Given an initial image u, the moment to preserve p, and the contrast parameter λ for the diffusivity function g our approach looks basically as follows:



For one iteration we first compute $v = u^p$, then we perform nonlinear isotropic diffusion as described in section 2.5 (see [14]) and finally we compute $u = \sqrt[p]{v}$.

3.1.1 Details

Let us explain the details of our approach. One single iteration of the moment preserving diffusion consists of three steps:

1. We compute
$$v = u^p.$$
Here we first perform a linear transformation. All gray values $u^{(k)}$ are scaled in such a way that the gray values of the initial image f fit into the range $[0.0, 1.0]$. Then we raise the scaled gray values to the pth power:
$$\begin{aligned} &\texttt{for } i = 1, \ldots, n_x : \\ &\quad \texttt{for } j = 1, \ldots, n_y : \\ &\quad\quad v_{i,j}^{(k)} := \left(u_{i,j}^{(k)} / 255.0\right)^p \end{aligned}$$
Since we assume that the gray values of the image are in the range $[0.0, 255.0]$ a division of the gray values by 255.0 maps them to the range $[0.0, 1.0]$. The scaled gray values then remain in the interval of $[0.0, 1.0]$ after raising them to their pth power.

2. We perform nonlinear isotropic diffusion of the transformed gray values $v^{(k)}$:

$$v_t = \operatorname{div}\Big(g\big(\nabla v_\sigma\big) \cdot \nabla v\Big),$$

with smoothing parameter σ, diffusivity function g, and contrast parameter λ^p. If we use the Charbonnier diffusivity as diffusivity function we would have:

$$g\big(\,|\nabla v_\sigma|\,\big) = \frac{1}{\sqrt{1 + \dfrac{|\nabla v_\sigma|^2}{\lambda^{2\cdot p}}}}.$$

We use the AOS scheme as described in section 2.5.4 for the nonlinear isotropic diffusion. We perform one iteration on $v^{(k)}$ in order to obtain $v^{(k+1)}$. This can be done efficiently by applying the Thomas algorithm twice (without presmoothing, if $\sigma = 0$) or four times (with presmoothing). For the diffusivity function we use λ^p as contrast parameter in order to treat the contrast parameter λ in a similar way as the gray values $u^{(k)}$: the gray values are raised to their pth power, so, we do the same with λ. Note that this does not imply that for the same value of λ images with different parameter p will be comparable.

3. We compute

$$u = \sqrt[p]{v}.$$

Here we perform the back-transformation corresponding to the transformation in the first step. By computing the pth root from $v^{(k+1)}$ and scaling we obtain $u^{(k+1)}$:

$$\begin{aligned}
&\texttt{for } i = 1, \ldots, n_x : \\
&\quad \texttt{for } j = 1, \ldots, n_y : \\
&\qquad u_{i,j}^{(k+1)} := \sqrt[p]{v_{i,j}^{(k+1)}} \cdot 255.0
\end{aligned}$$

One can see that by performing two consecutive iterations the back-transform of the first iteration and the transform of the second iteration cancel out. So one can perform all necessary iterations consecutively before the back-transform, if one is not interested in the evolution of the image but only in the end result.
The computational overhead for transform and back-transform in comparison to the nonlinear isotropic diffusion is fairly low: one has one additional transformation and one additional back-transformation per iteration. Taking advantage of the cancellation effect of a back-transform followed by a transform one even needs only one transformation and back-transformation regardless of the number of iterations.

3.1.2 Properties

Let us now check which properties from the nonlinear isotropic diffusion carry over to our moment preserving nonlinear isotropic diffusion and which properties do not. Note that since we already know the properties of nonlinear isotropic diffusion from section 2.5.3 we only have to check whether those properties are preserved by the transformation and back-transformation. We have to investigate the following properties:

1. Well-Posedness: a unique solution $u^{(k)}$ exists for every k, the solution depends continuously on f.

   Our approach fulfills the well-posedness property. First the nonlinear isotropic diffusion is well-posed. Furthermore, the transformation and back-transformation are well-posed:

the scaling and the computation of the pth power are invertible since the gray values are always non-negative. Hence computing the pth root from $v^{(k+1)}$ followed by a scaling has always a unique result $u^{(k+1)}$. The result depends continuously on f.

2. Average Gray Level Invariance:

$$\frac{1}{n_x \cdot n_y} \cdot \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} u_{i,j}^{(k)} = \frac{1}{n_x \cdot n_y} \cdot \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} f_{i,j}, \quad \forall k \geqslant 0.$$

The average gray level invariance is not fulfilled. Our motivation is to generalize nonlinear isotropic diffusion to preserve the moment of order p. Hence we have in our case a moment invariance:

$$\frac{1}{n_x \cdot n_y} \cdot \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} \left(u_{i,j}^{(k)}\right)^p = \frac{1}{n_x \cdot n_y} \cdot \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} f_{i,j}^p, \quad \forall k \geqslant 0.$$

As one can see the average gray level invariance is the special case of the moment invariance for $p = 1$. Note that for all other cases $p > 1$ the pth moment is preserved but due to the non-linearity of the transformation and back-transformation the average gray level of the image is not preserved any more.

3. Extremum Principle:

$$\min_{\substack{i=1,\ldots,n_x \\ j=1,\ldots,n_y}} f_{i,j} \leqslant u_{i,j}^{(k)} \leqslant \max_{\substack{i=1,\ldots,n_x \\ j=1,\ldots,n_y}} f_{i,j}, \quad \forall k \geqslant 0.$$

The extremum principle (or minimum maximum principle) of the nonlinear diffusion carries over to our approach.

4. Smoothing Lyapunov Sequences:

   - The p-norms
   $$||u^{(k)}||_p := \left(\sum_{i=1}^{n_x} \sum_{j=1}^{n_y} |u_{i,j}^{(k)}|^p \right)^{\frac{1}{p}}$$
   are decreasing in k for all $p \geqslant 1$.
   - All even central moments
   $$M_{2m}[u^{(k)}] := \frac{1}{n} \cdot \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} \left(u_{i,j}^{(k)} - \mu\right)^{2 \cdot m}, m \in \mathbb{N}$$
   are decreasing in k, where $\mu = \frac{1}{n_x \cdot n_y} \cdot \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} f_{i,j}$.
   - The entropy
   $$S[u^{(k)}] := -\sum_{i=1}^{n_x} \sum_{j=1}^{n_y} u_{i,j}^{(k)} \cdot \ln u_{i,j}^{(k)}$$
   is increasing in k (if $f_{i,j}$ is positive for all i, j).

Due to the non-linearity of the transformation none of these properties are fulfilled by our approach.

5. Convergence to a Constant Steady-State:

$$\lim_{k\to\infty} u_{i,j}^{(k)} = m_p, \quad \forall i,j,$$

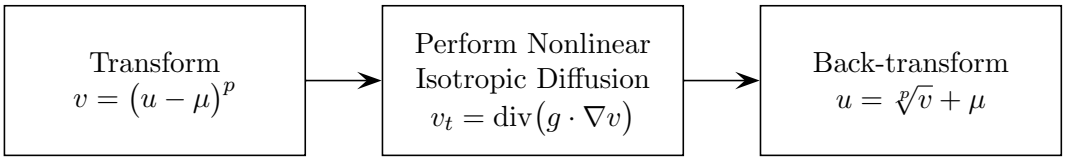
with

$$m_p := \frac{1}{n_x \cdot n_y} \cdot \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} f_{i,j}^p.$$

The convergence to a constant steady-state of our approach follows from the convergence to a steady-state of the nonlinear isotropic diffusion. The steady-state is in our case m_p, the moment of order p that has been preserved during the evolution of the image.

3.2 Preservation of the pth Central Moment

In the previous section we have presented a nonlinear isotropic diffusion that preserves the pth moment. By preserving a higher moment ($p > 1$) pixels with different gray values are treated differently: brighter pixels which have high gray values have more impact on the evolution than darker pixels which have low gray values. However, it would be preferable to treat dark and bright pixels in the same way, since for instance Gaussian noise increases and decreases gray values in the same way.
Handling dark and bright pixels in the same way can be accomplished by extending the filter presented in section 3.1 to preserve the pth central moment M_p. Given an initial image u, the central moment to preserve p, and the contrast parameter λ for the diffusivity function g our approach looks basically as follows:



Here μ is the average gray value of the original image f. For one iteration we first compute $v = (u-\mu)^p$, then we perform nonlinear isotropic diffusion as described in section 2.5 and finally we compute $u = \sqrt[p]{v} + \mu$.

3.2.1 Details

The only difference of the central moment preserving diffusion to the moment preserving diffusion is the subtraction of μ in the transformation step and the addition of μ in the back-transformation step. The subtraction of μ is motivated by the definition of the central moment of order p,

$$M_p := \frac{1}{n_x \cdot n_y} \cdot \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} (f_{i,j} - \mu)^p,$$

where

$$\mu := \frac{1}{n_x \cdot n_y} \cdot \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} f_{i,j}.$$

The addition of μ at the end of the third step is performed in order to undo the subtraction of μ in the transformation step.
One single iteration of the central moment preserving diffusion looks as follows:

1. We compute
$$v = \big(u - \mu\big)^p.$$
Precisely we do the following:
$$\begin{aligned}
&\mu := 0\\
&\texttt{for } i = 1, \ldots, n_x:\\
&\quad \texttt{for } j = 1, \ldots, n_y:\\
&\quad\quad \mu := \mu + f_{i,j}\\
&\mu := \mu / \big(n_x \cdot n_y\big)\\
&\texttt{for } i = 1, \ldots, n_x:\\
&\quad \texttt{for } j = 1, \ldots, n_y:\\
&\quad\quad v_{i,j}^{(k)} := \Big(\big(u_{i,j}^{(k)} - \mu\big)/255.0\Big)^p
\end{aligned}$$
First we perform a linear transformation. All gray values are shifted by the average gray value μ. Since $\mu \in [0.0, 255.0]$ the shifted gray values are in the interval $[-255.0, 255.0]$. Dividing the shifted gray values by 255.0 results in gray values which lie in the range $[-1.0, 1.0]$. By raising the shifted and scaled gray values to their pth power they will stay in the interval $[-1.0, 1.0]$.

2. We perform nonlinear isotropic diffusion of the transformed gray values $v^{(k)}$:
$$v_t = \operatorname{div}\Big(g\big(\nabla v_\sigma\big) \cdot \nabla v\Big),$$
with diffusivity function g and contrast parameter λ^p. This step is identical to the diffusion step of the moment preserving diffusion.

3. We compute
$$u = \sqrt[p]{v} + \mu.$$
Here we perform the back-transformation corresponding to the transformation in the first step. By computing the pth root from $v^{(k+1)}$, scaling, and shifting we obtain $u^{(k+1)}$:
$$\begin{aligned}
&\mu := 0\\
&\texttt{for } i = 1, \ldots, n_x:\\
&\quad \texttt{for } j = 1, \ldots, n_y:\\
&\quad\quad \mu := \mu + f_{i,j}\\
&\mu := \mu / \big(n_x \cdot n_y\big)\\
&\texttt{for } i = 1, \ldots, n_x:\\
&\quad \texttt{for } j = 1, \ldots, n_y:\\
&\quad\quad u_{i,j}^{(k+1)} := \sqrt[p]{v_{i,j}^{(k+1)}} \cdot 255.0 + \mu
\end{aligned}$$

Again a back-transformation followed by a transformation cancel out. Transformation and back-transformation are computationally cheap in comparison to the nonlinear isotropic diffusion.

3.2.2 Properties

Let us now investigate the properties of the central moment preserving diffusion as we have done it in section 3.1.2:

1. Well-Posedness: a unique solution $u^{(k)}$ exists for every k, the solution depends continuously on f.

   Our approach fulfills the well-posedness property for odd p. Nonlinear isotropic diffusion is well-posed. In the transformation step shifting, scaling, and computing the pth power of the gray values result in gray values which lie in the interval $[-1.0, 1.0]$. In order to be able to back-transform the resulting gray values p has the be odd. Furthermore, the result depends continuously on f.

   Raising the gray values to the pth power where p is even results in non-negative values. Taking the pth root has no unique result any more. Hence we can only use an odd p for exponentiation.

2. Average Gray Level Invariance:

$$\frac{1}{n_x \cdot n_y} \cdot \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} u_{i,j}^{(k)} = \frac{1}{n_x \cdot n_y} \cdot \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} f_{i,j}, \quad \forall k \geqslant 0.$$

   The average gray level invariance is not fulfilled. Again our motivation is to generalize nonlinear isotropic diffusion to preserve the central moment of order p. Hence we have in our case a central moment invariance:

$$\frac{1}{n_x \cdot n_y} \cdot \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} \left(u_{i,j}^{(k)} - \mu \right)^p = \frac{1}{n_x \cdot n_y} \cdot \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} \left(f_{i,j} - \mu \right)^p, \quad \forall k \geqslant 0.$$

   As one can see the average gray level invariance is the special case of the central moment invariance for $p = 1$. Note that for all other cases $p > 1$, p odd the pth central moment is preserved but due to the non-linearity of the transformation and back-transformation the average gray level of the image is not preserved.

3. Extremum Principle:

$$\min_{\substack{i=1,\ldots,n_x \\ j=1,\ldots,n_y}} f_{i,j} \leqslant u_{i,j}^{(k)} \leqslant \max_{\substack{i=1,\ldots,n_x \\ j=1,\ldots,n_y}} f_{i,j}, \quad \forall k \geqslant 0.$$

   The extremum principle (or minimum maximum principle) of the nonlinear diffusion carries over.

4. Smoothing Lyapunov Sequences:

   - The p-norms are decreasing in k for all $p \geqslant 1$.
   - All even central moments are decreasing in k,
   - The entropy is increasing in k.

   As for the case of preserving moments of order p due to the non-linearity of the transformation none of these properties are fulfilled.

5. Convergence to a Constant Steady-State:

$$\lim_{k\to\infty} u_{i,j}^{(k)} = m_p, \quad \forall i,j,$$

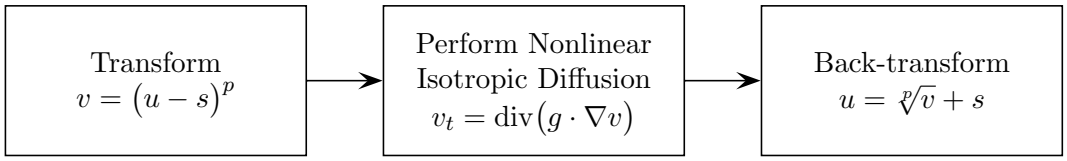
with

$$m_p := \frac{1}{n_x \cdot n_y} \cdot \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} f_{i,j}^p.$$

The convergence to a constant steady-state follows from the convergence to a steady-state of the nonlinear isotropic diffusion. The steady-state is in our case M_p, the central moment of order p that has been preserved during the evolution of the image.

3.3 Combining both Approaches

In the previous sections we have presented an approach to preserve a moment of order p during nonlinear isotropic diffusion and an approach to preserve a central moment of order p. For the latter we simply shifted all gray values by the average gray value μ before scaling, raising to the pth power, and performing nonlinear diffusion. Let us now combine both approaches by introducing a new parameter s describing the shift of the gray values:



We have essentially the same approach as in the previous section expect we shift all gray values by s instead of μ. One can easily see that by setting

$$s := 0$$

we obtain the nonlinear isotropic diffusion preserving the m_p, the moment of order p as described in section 3.1. By setting

$$s := \mu = \frac{1}{n_x \cdot n_y} \cdot \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} f_{i,j}$$

we obtain the nonlinear isotropic diffusion preserving M_p, the central moment of order p as described in section 3.2. Furthermore, we can also use other values as shift parameter, as for example the mid-range of f:

$$s := \frac{\min_{\substack{i=1,\dots,n_x \\ j=1,\dots n_y}} f_{i,j} + \max_{\substack{i=1,\dots,n_x \\ j=1,\dots n_y}} f_{i,j}}{2}.$$

Now we end up with a nonlinear diffusion preserving the moment of order p shifted by s:

$$\frac{1}{n_x \cdot n_y} \cdot \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} \left(u_{i,j}^{(k)} - s\right)^p = \frac{1}{n_x \cdot n_y} \cdot \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} (f_{i,j} - s)^p, \quad \forall k \geqslant 0.$$

For $p = 1$ we obtain the average gray level invariance from the original nonlinear isotropic diffusion (see 2.5) — independent from the choice of s. For $s = 0$ we obtain the invariance of the moment of order p, and finally for an odd p and $s = \mu$ we obtain the invariance of the central moment of order p. One can check that the properties of well-posedness, the minimum maximum principle, and the property of convergence to a constant steady-state are fulfilled.
By having an additional shift parameter s we are not only able to express both approaches from above but by choosing appropriate values for s we are able to steer the behavior of the filter. Choosing a small s in comparison to the maximal gray value will result in a preferred diffusion of dark image areas while the bright image areas will be significantly less diffused. Choosing a high s will result in a preferred diffusion of bright image areas while dark image ares will be less diffused. Because of the exponentiation the differences between gray values around the value of s become smaller than gray values differences far apart from s. Hence the following nonlinear diffusion step will diffuse areas with a gray values close to s much more than areas with gray values far apart from s.

3.4 Limitations

Before we present results from our experiments let us mention some limitations of our approach:

- One can see that from the construction of our approach even central moments cannot be preserved. The shifted gray values are positive and negative. Raising them to a power p that is even we would lose the sign of the gray value.

- Raising the shifted and scaled gray values to higher powers leads to an amplification of minimal and maximal gray values. If minimal and maximal gray values are created by a specific type of noise — as it is the case with uniform noise and especially salt and pepper noise — our approach amplifies the noise. Hence our filter is only suitable for noise types that do not introduce many extreme gray values. This is, for instance, the case with Gaussian noise where the majority of the gray values are often changed slightly and only few gray values are changed severely.

- The amplification of the gray values by exponentiation in combination with the shifting by s of the gray values leads to a symmetric approach: gray values close to the value s are smoothed more by the diffusion process than gray values significantly smaller or greater than s. This approach is especially useful if the original image consists mainly of similar areas of two different gray values. Then by choosing the average gray level μ as shift parameter s values below μ will by mostly diffused to a darker area and values above μ will by mostly diffused to a brighter area. Unfortunately this behavior is not so useful for images consisting of areas of more than two different gray values as it is the case in typical gray value images.

Chapter 4

Experiments

In this chapter we will show results of our moment preserving diffusion filter as presented in section 3.3. First we will compare results of our approach to results obtained from already existing approaches. Then we will investigate the influence of different parameters on the behavior of our filter.
Let us first show our test image. As already said we expect visually good results for images consisting mainly of areas of two different gray levels. Hence we have taken the images depicted in figure 4.1. For our experiments we use the (right) noisy image as input and we would like to achieve an image similar to the original (left) image.
In sections 2.4 and 2.5 we have presented linear and nonlinear isotropic diffusion, respectively. Applying these diffusion filters leads to results as shown in figure 4.2. On the one hand linear diffusion reduces the noise in the image significantly, but on the other hand the edges of the triangle and the rectangle are blurred strongly. The more noise is reduced the more edges are blurred and dislocated. This is the reason why linear diffusion is not used for the task of denoising images.
Applying the nonlinear isotropic diffusion filter shows the advantages of nonlinear filters to linear filters: the areas inside the triangle and the rectangle are smoothed while the diffusion at the edges is reduced in order to preserve them for a longer time. Note the differences between the Charbonnier diffusivity (middle image) and the Perona-Malik diffusivity (right image, see [10]). The Charbonnier diffusivity reduces the diffusion at edges less then the Perona-Malik diffusivity. Here one can see the edge enhancing property of the Perona-Malik diffusivity. Note that the diffusion times for the results of linear diffusion and nonlinear diffusion are different. Basically the results of equal diffusion times of linear and nonlinear diffusion are not comparable,

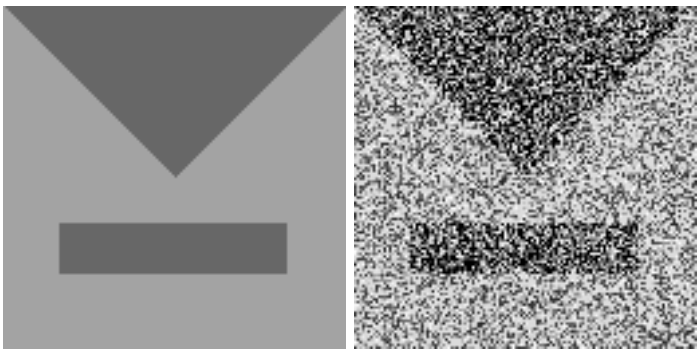


Figure 4.1: **Left:** Original gray scale image, 128×128 pixel; **Right:** Original image with added noise.

Figure 4.2: Noisy image filtered with different diffusivity functions; **Left:** Linear diffusion, implicit scheme, $\tau = 0.1$, 10 iterations; **Middle:** Nonlinear isotropic diffusion, AOS scheme with Charbonnier diffusivity, $\tau = 5.0, \sigma = 0.0, \lambda = 0.25$, 100 iterations; **Right:** Nonlinear isotropic diffusion, AOS scheme with Perona-Malik diffusivity, $\tau = 5.0, \sigma = 0.0, \lambda = 2.0$, 100 iterations.

since nonlinear diffusion reduces diffusion at locations where the lengths of gradients are big.

Image evolution

In figure 4.3 we can see an image evolution of our approach. In this particular case we have chosen to preserve the third moment $p = 3$. As shift parameter s we have chosen the mid-range 127.5. One can see that in the first iterations the noise is removed while edges are well preserved. Later the edges are smoothed and finally image structures begin to disappear as well.
We will now investigate the influence of the different parameters.

Influence of parameter p

First we have a look at the influence of the parameter p. Choosing $p = 1$ we get the nonlinear isotropic diffusion filter which is a special case of our approach (see section 2.5). Note that in the case of $p = 1$ the choice of the shift parameter does not play a role. For the remainder of this chapter we focus on the general case of our approach. Hence we assume that $p \geqslant 3$ and p odd.
In figure 4.4 we have fixed all parameters expect for p. For the parameter p we use the values $3, 5$, and 7, respectively. We can see that for $p = 3$ the diffusion process has advanced too much, since the rectangle is already vanished to some extent and the corners of the triangle is rounded. For $p = 5$ the diffusion process has to advance further since the (dark) noise in the bright background has disappeared but there is still (bright) noise in the triangle and the rectangle. For $p = 7$ the diffusion is advanced even slower than for $p = 5$. There is both, noise in the background as well as noise in the triangle and the rectangle. In this case we have used the mid-range 127.5 as shift parameter s.
So, increasing p results in a slow down of the diffusion process. This is because we use λ^p as contrast parameter in the nonlinear diffusion step. λ has to be chosen from $[0.0, 1.0]$ since we shift and scale the gray values of $u^{(k)}$ to an interval of length 1 that lies in $[-1.0, 1.0]$. Hence the bigger p the more the diffusion process is slowed down. Note that choosing λ instead of λ^p as contrast parameter would have the opposite effect. The differences between the gray values of $u^{(k)}$ would be smaller for increasing p. Hence the diffusion process would be sped up since less image features would be recognized as edges.

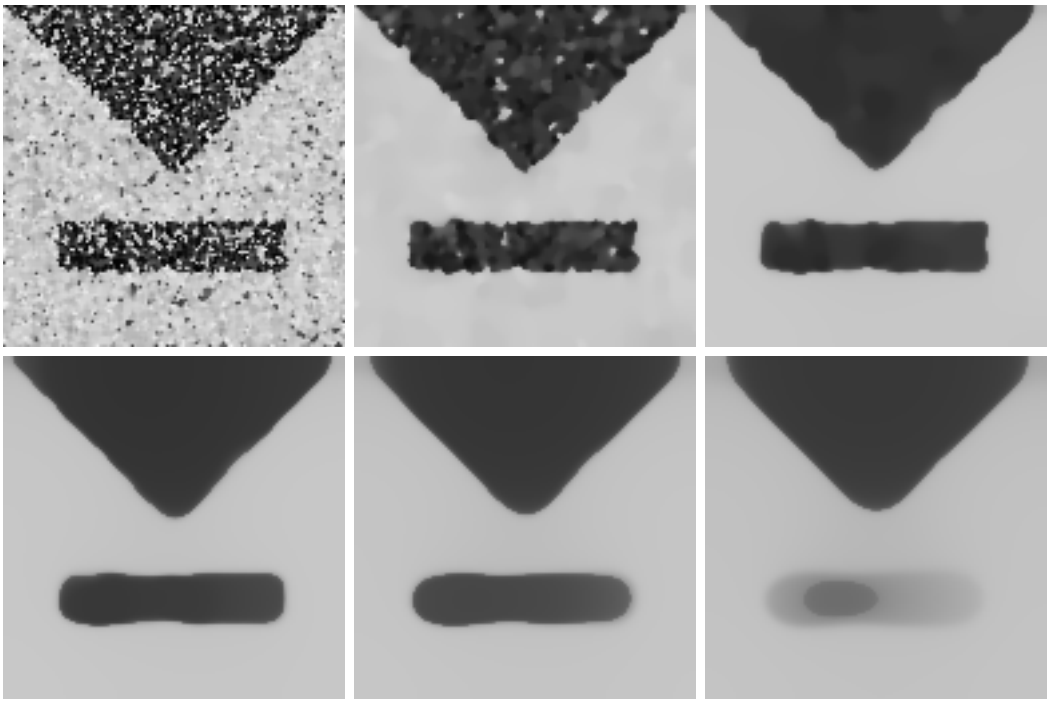


Figure 4.3: Noisy image filtered with moment preserving diffusion, AOS scheme with Charbonnier diffusivity; $p = 3, s = 127.5, \tau = 5.0, \sigma = 0.0, \lambda = 0.05$; **Top**, **Left:** 10 iterations; **Middle:** 40 iteration; **Right:** 90 iterations; **Bottom**, **Left:** 160 iterations; **Middle:** 250 iteration; **Right:** 360 iterations;

Figure 4.4: Influence of parameter p on the moment preserving diffusion, AOS scheme with Charbonnier diffusivity, $s = 127.5, \tau = 5.0, \sigma = 0.0, \lambda = 0.1$, 50 iterations; **Left:** $p = 3$; **Middle:** $p = 5$; **Right:** $p = 7$.

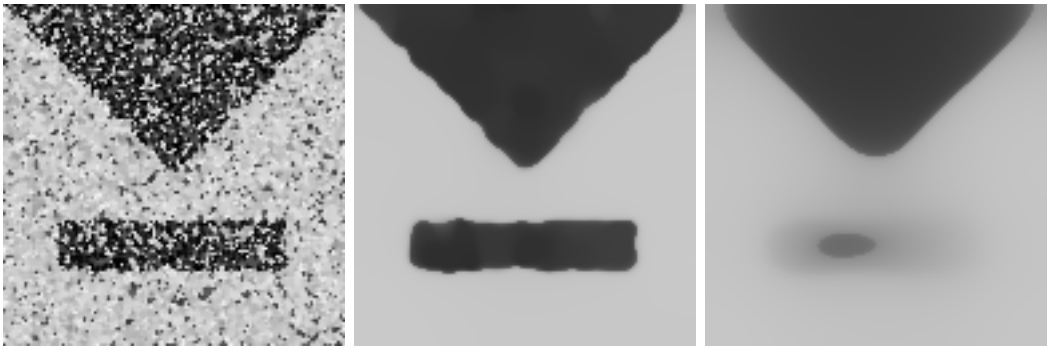


Figure 4.5: Influence of parameter λ on the moment preserving diffusion, AOS scheme with Charbonnier diffusivity, $p = 3, s = 127.5, \tau = 5.0, \sigma = 0.0$, 100 iterations; **Left:** $\lambda = 0.02$; **Middle:** $\lambda = 0.05$; **Right:** $\lambda = 0.08$.

Influence of contrast parameter λ

We fix all parameters except for λ. λ^p is used as the contrast parameter for the diffusion process. The bigger λ the less image features are recognized as edges. Less recognized edges mean the diffusion process will be slowed down less often. Hence a bigger λ speeds up the diffusion process as we can see in figure 4.5. While for $\lambda = 0.02$ the image has to be denoised further, for $\lambda = 0.05$ and $\lambda = 0.08$ the images are denoised well. But in the case of $\lambda = 0.08$ the diffusion has already started to diffuse the image structures. The rectangle has nearly completely vanished.

Choice of parameters p and λ

As we have described above the diffusion is sped up for increased p and fixed λ as well as for increased λ and fixed p. So, a parameter λ giving a visually good filtering result for a given p will probably not give a nice filtering result for a different p. Hence when changing p one also has to adjust λ appropriately. Our diffusion process is very sensitive to the choice of the contrast parameter λ. For our noisy test image we can achieve the filtering results as depicted in figure 4.6. With increasing parameter p and an appropriately adjusted parameter λ one can achieve good denoising results with increasing contrast enhancement. Furthermore, the edges are less blurred.

Influence of the time step size τ

In figure 4.7 we demonstrate the behavior of our filter for increasing time step size τ. As diffusion time we have chosen $t = 500.0$. We have chosen increasing time step sizes of $5.0, 20.0$, and 50.0. This means that we have to perform $100, 25$, and 10 iterations, respectively. One can see that for increasing time step size τ the filtering results look less smooth and the edges are more ragged. Remember that in this example we have used a time step size that is ten times as big as in the other examples. This means that we only have to perform one tenth of the iterations. So, if one is interested in a fast computation one can increase the time step size τ and reduce the number of iterations. But if one is interested in an accurate computation one can reduce the time step size τ but then one has to perform more iterations.

Influence of shift parameter s

In figure 4.8 the influence of the shift parameter s on the filtering result is depicted. For the shift parameter we chosen values of 110 to 160. First of all gray values close to the value s will be

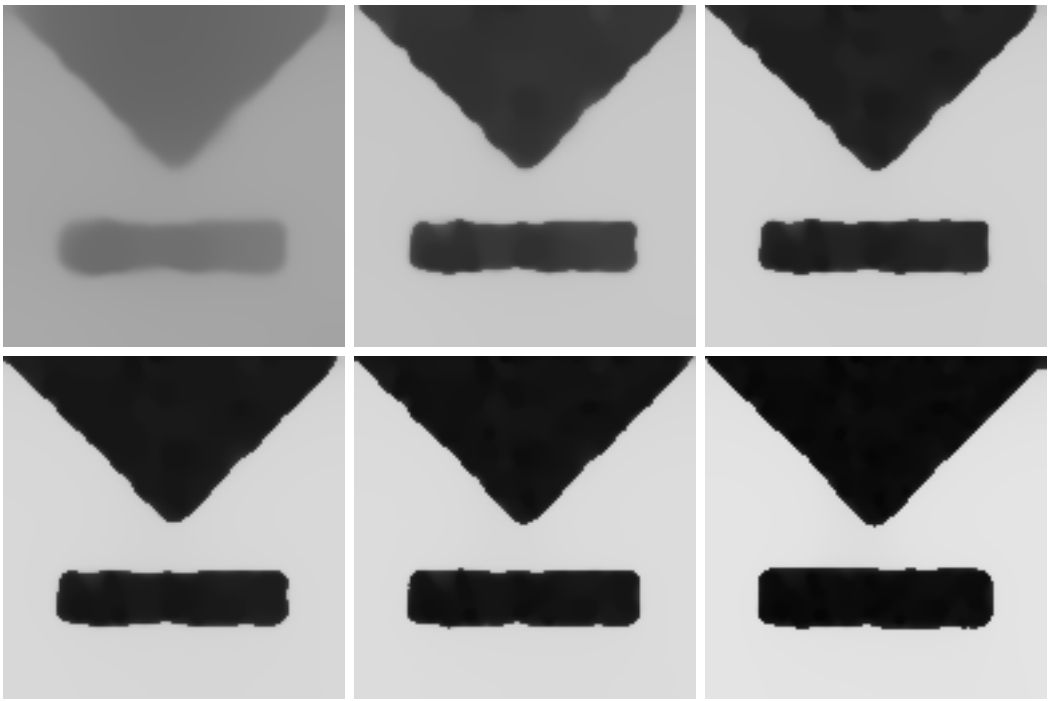


Figure 4.6: Noisy image filtered with moment preserving diffusion and different parameters p and λ, AOS scheme with Charbonnier diffusivity, $s = 127.5, \tau = 5.0, \sigma = 0.0$, 100 iterations; **Top**, **Left:** $\lambda = 0.001, p = 1$; **Middle:** $\lambda = 0.05, p = 3$; **Right:** $\lambda = 0.11, p = 5$; **Bottom**, **Left:** $\lambda = 0.18, p = 7$; **Middle:** $\lambda = 0.22, p = 9$; **Right:** $\lambda = 0.3, p = 15$.

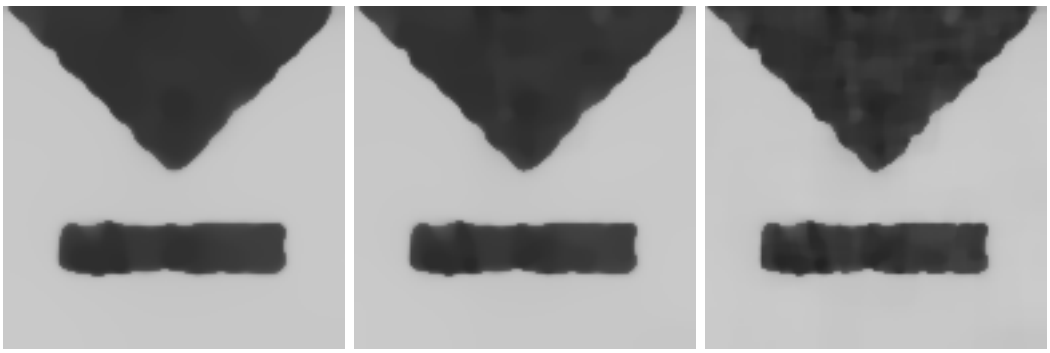


Figure 4.7: Influence of parameter τ on the moment preserving diffusion, AOS scheme with Charbonnier diffusivity, $p = 3, s = 127.5, \sigma = 0.0, \lambda = 0.05$; **Left:** $\tau = 5.0$, 100 iterations; **Middle:** $\tau = 20.0$, 25 iterations; **Right:** $\tau = 50.0$, 10 iterations.

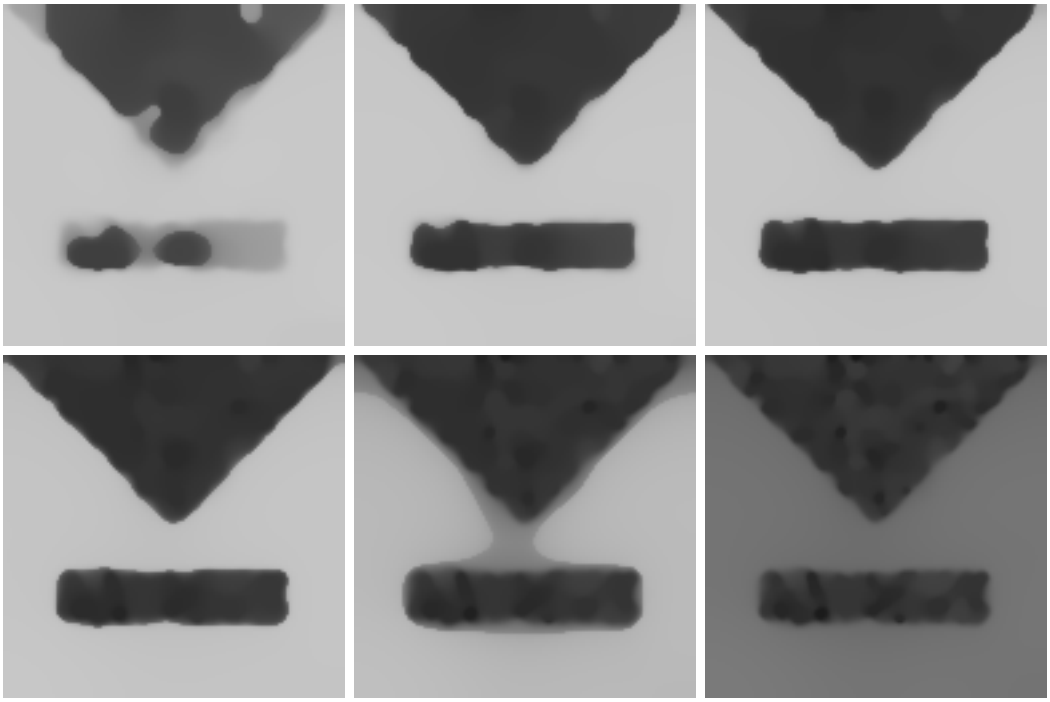


Figure 4.8: Influence of parameter s on the moment preserving diffusion, AOS scheme with Charbonnier diffusivity, $p = 3, \tau = 5.0, \sigma = 0.0, \lambda = 0.05$, 100 iterations; **Top**, **Left:** $s = 110.0$; **Middle:** $s = 120.0$; **Right:** $s = 130.0$; **Bottom**, **Left:** $s = 140.0$; **Middle:** $s = 150.0$; **Right:** $s = 160.0$.

diffused more than gray values far apart from s. Due to the exponentiation of the shifted gray values with p a difference between gray values close to s is smaller than a difference between gray values far apart from s. Hence differences of gray values close to the gray value s are equilibrated faster than differences of gray values which are far apart.
One can think of the exponentiation of the gray values as a weighting of the different pixels: pixels with gray values far apart from the value s have a bigger weight than those with gray values close to s. If we choose s to be small there are many bright pixels far apart from s that dominate the evolution of the image. This results in an increasing average gray level of the image. Similarly a high value of s results in many dark pixels dominating the diffusion process leading to a decreasing average gray value. So, the average gray value of the evolving image can be steered by s.

Influence of diffusivity function g

In contrast to the Charbonnier diffusivity in nonlinear isotropic diffusion the Perona-Malik diffusivity allows an enhancement of edges. In figure 4.9 we have a comparison of the Charbonnier diffusivity in nonlinear diffusion, the Perona-Malik diffusivity in nonlinear diffusion, and linear diffusion. Linear diffusion corresponds to choosing the diffusivity function $g(|\nabla u|) = 1$ which is independent from the image gradient. In figures 4.10 and 4.11 we can see the results of applying linear diffusion, nonlinear diffusion with Charbonnier diffusivity, and nonlinear diffusion with Perona-Malik diffusivity as diffusion step in our approach. When it comes to edge preservation and edge enhancement nonlinear diffusion with the Perona-Malik diffusivity provides better results than nonlinear diffusion with the Charbonnier diffusivity which in turn provides better results than linear diffusion.

Figure 4.9: Noisy image filtered with different diffusivity functions; **Left:** Linear diffusion, implicit scheme, $\tau = 0.1$, 10 iterations; **Middle:** Nonlinear isotropic diffusion, AOS scheme with Charbonnier diffusivity, $\tau = 5.0, \sigma = 0.0, \lambda = 0.25$, 100 iterations; **Right:** Nonlinear isotropic diffusion, AOS scheme with Perona-Malik diffusivity, $\tau = 5.0, \sigma = 0.0, \lambda = 2.0$, 100 iterations.

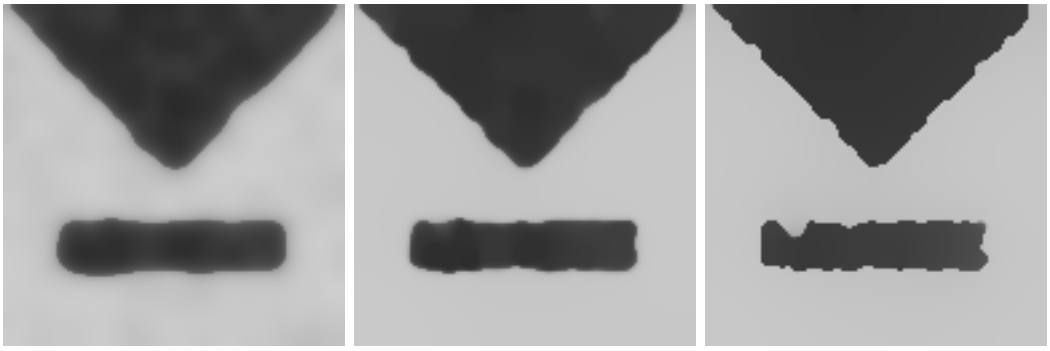


Figure 4.10: Noisy image filtered with moment preserving diffusion and different diffusivity functions g, $p = 3, s = 127.5$; **Left:** Linear diffusion, implicit scheme, $\sigma = 0.1$, 100 iterations; **Middle:** Charbonnier diffusivity, AOS scheme, $\tau = 5.0, \sigma = 0.0, \lambda = 0.05$, 100 iterations; **Right:** Perona-Malik diffusivity, AOS scheme, $\tau = 5.0, \sigma = 0.0, \lambda = 0.12$, 100 iterations.

Figure 4.11: Noisy image filtered with moment preserving diffusion and different diffusivity functions g, $p = 5, s = 127.5$; **Left:** Linear diffusion, implicit scheme, $\sigma = 0.1$, 100 iterations; **Middle:** Charbonnier diffusivity, AOS scheme, $\tau = 5.0, \sigma = 0.0, \lambda = 0.12$, 100 iterations; **Right:** Perona-Malik diffusivity, AOS scheme, $\tau = 5.0, \sigma = 0.0, \lambda = 0.21$, 100 iterations.

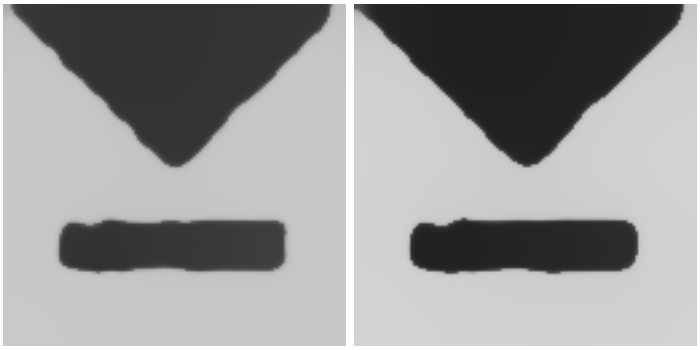


Figure 4.12: Noisy image filtered with moment preserving diffusion, AOS scheme with Perona-Malik diffusivity and presmoothing; **Left:** $p = 3, s = 127.5, \tau = 5.0, \sigma = 1.0, \lambda = 0.10$, 100 iterations; **Right:** $p = 5, s = 127.5, \tau = 5.0, \sigma = 1.0, \lambda = 0.19$, 100 iterations.

Perona-Malik diffusivity with presmoothing

Although the Perona-Malik diffusivity gives the best denoising results with edge enhancement the edges are ragged in comparison to the original image. One possibility to overcome this effect is to use presmoothing of the image u for the gradient estimation $|\nabla u|$ used in the diffusivity function g. By presmoothing the image for the gradient estimate in combination with a lower λ parameter we are able to obtain results as depicted in figure 4.12.

Chapter 5

Conclusion

5.1 Summary

In this thesis we have presented a new diffusion filter based on nonlinear isotropic diffusion. One property of the nonlinear isotropic diffusion is the preservation of the average gray level of the original image during the image evolution. With our approach we are also able preserve higher (central and shifted) moments by generalizing nonlinear isotropic diffusion. We achieve this property by

- raising the (shifted) gray values of the image to the odd power p,
- performing a nonlinear isotropic diffusion step, and
- taking the pth root from the gray values (and shifting them back if necessary).

We have seen that our approach can achieve better denoising results than nonlinear isotropic diffusion in case of noisy binary images. However, the filtering results for binary images with impulse noise (salt and pepper noise or uniform noise for example) are inferior to existing filtering approaches like median filtering. Furthermore, our approach does not lead to better denoising results when it comes to typical gray value images. Here we can only obtain the same results as nonlinear isotropic diffusion, since this is a special case of our approach.
Our approach is efficiently computable for two reasons: first we use an AOS scheme for nonlinear isotropic diffusion and second for the final filtering result it is sufficient to perform the transformation step and the back-transformation step only once.

5.2 Future Work

The design of our approach allows to preserve a higher moment. The choice to preserve a higher moment p instead of the first moment leads to a change of the average gray level during the image evolution. The preservation of the average gray level is a preferable property of a diffusion filter. It turns out that the change of the average gray level can be steered by the shift parameter s. By a suitable choice of the shift parameter s one can preserve the average gray level. However, this parameter s is in general different for each iteration and its computation hard. By preserving a higher moment of order p and the average gray value the presented diffusion filter can be improved further. To be able to do this an efficient way to compute the appropriate value for s in each iteration has to be found.
Due to the exponentiation extreme gray values are more important for the evolution of the image than gray values close to s. This amplification, however, is a disadvantage when it comes

to impulse noise that creates many extreme gray values such as salt and pepper noise. For that type of noise our approach is inferior to other filters. But there can be other, more suitable, transformations for handling impulse noise with our approach.
We have used nonlinear isotropic diffusion in our approach. Using a nonlinear anisotropic diffusion process instead of a nonlinear isotropic one can lead to better denoising results in other applications where anisotropic diffusion has to be used.

Bibliography

[1] L. Alvarez, F. Guichard, P.-L. Lions, and J.-M. Morel. Axioms and fundamental equations in image processing. In *Archive for Rational Mechanics and Analysis*, volume 123, pages 199–257, 1993. 2.4.2

[2] H. S. Carslaw and J. C. Jaeger. *Conduction of Heat in Solids.* Oxford University Press, second edition, 1975. 2.3, 2.4

[3] F. Catté, P.-L. Lions, J.-M. Morel, and T. Coll. Image selective smoothing and edge detection by nonlinear diffusion. In *SIAM Journal on Numerical Analysis*, volume 29, pages 182–193, 1992. 2.5

[4] J. Crank. *The Mathematics of Diffusion.* Oxford University Press, second edition, 1959. 2.3

[5] L. C. Evans. *Partial Differential Equations.* American Mathematical Society, Providence, 1998. 2.3

[6] S. J. Farlow. *Partial Differential Equations for Scientists and Engineers.* Dover, New York, 1993. 2.3

[7] T. Lindeberg. *Scale-Space Theory in Computer Vision.* Kluwer, boston edition, 1994. 2.4.2

[8] T. Lu, P. Neittaanmäki, and X.-C. Tai. A parallel splitting up method and its application to navier-strokes equations. In *Applied Mathematics Letters*, volume 4, pages 25–29, 1991. 2.5.4

[9] K. W. Morton and D. F. Mayers. *Numerical Solution of Partial Differential Equations.* Cambridge University Press, Cambridge, 1994. 2.4.1

[10] P. Perona and J. Malik. Scale space and edge detection using anisotropic diffusion. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 12, pages 629–639, 1990. 2.5.1, 4

[11] G. D. Smith. *Numerical Solution of Partial Differential Equations: Finite Difference Methods.* Clarendon Press, Oxford, third edition, 1985. 2.4.1

[12] J. Sporring, M. Nielsen, L. Florack, and P. J. (Eds.). *Gaussian Scale-Space Theory.* Kluwer, dordrecht edition, 1997. 2.4.2

[13] L. H. Thomas. Elliptic problems in linear difference equations over a network. Technical report, Watson Scientific Computing Laboratory, Columbia University, New York, NJ, 1949. 2.4.3

[14] J. Weickert. *Anisotropic Diffusion in Image Processing.* Teubner, Stuttgart, 1998. See also `www.mia.uni-saarland.de/weickert/Papers/diss.ps.gz`. 1.3, 2.4.2, 2.5, 3.1

[15] J. Weickert, S. Ishikawa, and A. Imiya. Linear scale space has first been proposed in japan. In *Journal of Mathematical Imaging and Vision*, volume 10, pages 237–252, 1999. (`www.mia.uni-saarland.de/weickert/publications.shtml`). 2.4.2

[16] J. Weickert, B. M. ter Haar Romeny, and M. A. Viergever. Efficient and reliable schemes for nonlinear isotropic diffusion filtering. In *IEEE Transactions on Image Processing*, volume 7, pages 398–410, 1998. 2.5.4

[17] A. P. Witkin. Scale-space filtering. In *Proc. Eighth International Joint Conference on Artifical Intelligence (Karlsruhe, Germany)*, volume 2, pages 945–951, 1983. 2.4.2

[18] D. M. Young. *Iterative Solution of Large Linear Systems.* Academic Press, New York, 1971. 2.4.3