Delft University of Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft Institute of Applied Mathematics

Linear and anisotropic diffusion in image processing: A study on implementation, parameters and segmentation

A thesis submitted to the
Delft Institute of Applied Mathematics
in partial fulfillment of the requirements

for the degree

MASTER OF SCIENCE
in
APPLIED MATHEMATICS
by

P. van Marlen

Delft, the Netherlands
April 2018

MSc THESIS APPLIED MATHEMATICS

"Linear and anisotropic diffusion in image processing: A study on implemetation, parameters and segmentation"

P. van Marlen

Delft University of Technology

Daily supervisor

Dr. ir. M.B. van Gijzen

Other thesis committee members

Dr.ir R.F. Remis

Prof. dr. ir. C. Vuik

April, 2018

Delft

ABSTRACT

The TU Delft and the LUMC are creating a low-field MRI scanner to use in third world countries. This type of MRI scanner has many advantages, but a downside is the amount of noise in the obtained images. A way to reduce noise is diffusion filtering. This thesis discusses the theory of some linear and nonlinear diffusion filtering methods and tests them on several test problems. The methods range from the basic linear method, the heat equation, to the more advanced Perona-Malik method. The results indeed show that the Perona-Malik method generates better results, sometimes when combined with a Gaussian kernel to decrease its ill-posedness.
Two numerical methods have been compared for these anisotropic diffusion filtering methods: the Forward Time, Central Space method and the Additive Operator Splitting method. The advantage of the AOS method is the unconditional stability for al positive time step sizes Δt, while FTCS implementation is only stable for $0 < \Delta t \leq 0.25$. The results for the AOS method were also slightly better than for the FTCS method and almost never lead to instability. The FTCS method showed instability more often.
The parameter choice is of significant importance for the outcome. Several options for determining the gradient threshold parameter K, time step size Δt and stopping time S have been investigated. A new estimation for Δt and stopping time S have been proposed and compared. The results are images with good visual quality for both introduced methods. Also, an adaptive time step has been tested to overcome the instability for unstable methods and this seems to be succesful.
Lastly, region growing segmentation has been applied to images obtained with the proposed stopping time S. Segmentation is an additional way to investigate the quality of these outcomes. For most test problems the partitioning resembles the partitioning of the original image.
However, choosing certain parameters stays a challenge and can be of interest for further examination. It also remains a goal to investigate the discussed methods on data obtained with the actual TUD/LUMC MRI scanner.

Contents

1

INTRODUCTION

This thesis is related to a project of the TU Delft and LUMC. The goal of this project is to develop a low-field MRI scanner for developing countries. Penn State University is working on a similar project and collaborates with the TU Delft to share information. Both projects are trying to build a low-field MRI scanner, but of different types. A low-field MRI scanner has a lot of advantages to make it suitable for developing countries: it is cheaper and easier to use than a conventional high-field MRI scanner. However, a problem of a low-field MRI scanner is that it produces more noise. Luckily a lot of this noise can be removed using mathematics. This thesis focuses on diffusion filtering methods for noise reduction in the obtained images ([1]) and is part of the final stage of image processing.

The goal of this thesis consists of three subgoals:
1. to give an overview of diffusion filtering methods and to evaluate them for two different numerical methods: the Forward Time, Central Space method and the Additive Operator Splitting method ([10]).
2. to describe several options for optimal parameter choice for the gradient threshold parameter K, time step size Δt and stopping time S.
3. to use region growing segmentation on some results to compare their visual quality.
These subgoals will lead to a better understanding of image filtering and results of higher quality.

The report is organized in the following way. Since the images are obtained using an MRI scanner, it is important to understand the basics of MRI theory. Therefore chapter 2 focuses on the theory of both high-field and low-field MRI scanners. The general physics is discussed, as well as the differences between high- and low-field scanners.
The subsequent chapters focus on the first subgoal of the thesis: diffusion filtering methods and their implentation. First the mathematics of diffusion filtering methods are described. Chapter 3 starts with linear diffusion filtering and the physical background of this method. This method is the most basic form of diffusion filtering and while it is easy to implement, it has a few disadvantages.
Chapter 4 considers nonlinear or anisotropic diffusion filtering methods. These methods are more advanced, but overcome the disadvantages of linear filtering methods. In this chapter the well-posedness of nonlinear filtering is discussed, which leads to different approaches. We distinguish three cases: a well-posed method, an ill-posed, but regularly used method and the well-known Perona-Malik methods ([2]). These Perona-Malik methods are considered ill-posed, but have many advantages. To enhance the stability of these methods a Gaussian kernel can be added to them, which is also mentioned in chapter 4. All discussed methods are analyzed in the following chapters.

Chapter 5 describes the numerical scheme named Forward Time, Central Space. Chapter 6 focuses on the other numerical method, the Additive Operator Splitting method.
In chapter 7 all theory discussed so far is tested. All methods and both numerical schemes are compared.
Chapter 8 and 9 are part of the second subgoal. Chapter 8 focuses on theory found in literature on optimal parameter choice. In chapter 9 this theory is tested and when necessary modified for better results.
Chapter 10 and 11 are about segmentation. First the method is described in chapter 10 and used to partition several test images in chapter 11.
We conclude with an overall conclusion and ideas for improvement in chapter 12.

2

MRI BACKGROUND

Since this thesis is about diffusion filtering methods of MRI images, it is important to know the theory of MRI scanners in general and in particular of the low-field MRI. This chapter uses the theory mentioned in [3] to give a summary of the MRI physics.

2.1. MRI PHYSICS

An MRI scanner uses a magnetic field to obtain images. Usually a uniform background magnetic field is applied in the longitudinal direction, the z-direction:

$$\vec{B}_0 = B_0\vec{k} \tag{2.1}$$

with B_0 the field strength. This magnetic field leads to an equilibrium magnetization given by:

$$\vec{M}_0 = M_x\vec{i} + M_y\vec{j} + M_z\vec{k} \tag{2.2}$$

The vectors $\vec{i}, \vec{j}, \vec{k}$ denote the unit vectors along the x, y, z axes respectively and $\vec{r} = (x, y, z)$ is the 3D spatial coordinate.
This equilibrium magnetization M_0 is aligned with the magnetic field B_0. In order to change the direction of the magnetization a time-dependent magnetic field is added to the background field leading to a total magnetic field $\vec{B}(\vec{r}, t)$. This field gives a magnetization depending on time and space:

$$\vec{M}(\vec{r}, t) = M_x(\vec{r}, t)\vec{i} + M_y(\vec{r}, t)\vec{j} + M_z(\vec{r}, t)\vec{k} \tag{2.3}$$

An MRI scan consists of alternations between two stages: the excitation stage and the readout stage. During the excitation stage the magnetic field $\vec{B}(\vec{r}, t)$ is chosen in such a way that the magnetization has a component in the transverse plane, the (x, y) plane. The magnetic field is oscillating at a certain frequency called the Larmor frequency. This frequency is given by the Larmor relation:

$$\omega = \gamma|\vec{B}| \tag{2.4}$$

with ω the Larmor frequency and γ the gyromagnetic ratio. Due to this oscillating field the magnetization will precess in the transverse plane around the z-axis with the same Larmor frequency (figure 2.1).
During the readout stage the magnetic field only has a longitudinal component and is given by:

$$\vec{B}(\vec{r}, t) = B_z(\vec{r}, t)\vec{k} \tag{2.5}$$

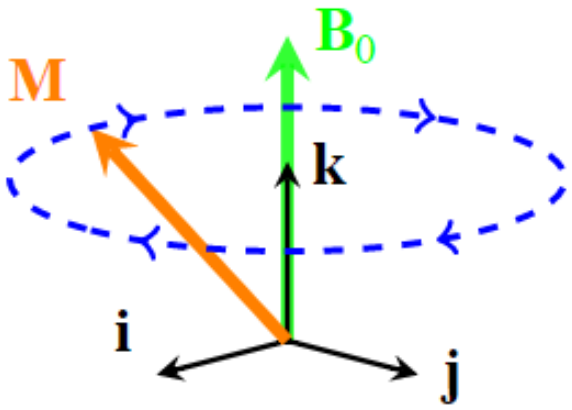

Figure 2.1: Precession of magnetization M due to magnetic field B [4]

The instantaneous Larmor frequency is then given by:

$$\omega(\vec{r},t) = \gamma B_z(\vec{r},t) \tag{2.6}$$

which leads to a magnetization

$$\begin{aligned} M(\vec{r},t) &= M(\vec{r},0)e^{-t/T_2(\vec{r})}e^{-i\int_0^t \omega(\vec{r},t')dt'} \\ &= f(\vec{r})e^{-t/T_2(\vec{r})}e^{-i\gamma\int_0^t B_z(\vec{r},t')dt'} \end{aligned} \tag{2.7}$$

The term $e^{-t/T_2(\vec{r})}$ gives the decay with time constant $T_2(\vec{r})$. The integral runs from $t = 0$, which is the time when the excitation stage ends, to a certain time $t > 0$ during the readout stage. We define $f(\vec{r}) := M(\vec{r},0)$, which is the magnetization immediately after the excitation stage. The goal is to obtain an image of this $f(\vec{r})$.

The magnetization $M(\vec{r},t)$ will induce an electromotive force in a coil creating an electric potential. After demodulation this electric potential can be written as the following signal:

$$s(t) = \int c(\vec{r})f(\vec{r})e^{-t/T_2(\vec{r})}e^{-i\varphi(\vec{r},t)}d\vec{r} \tag{2.8}$$

Here $c(\vec{r})$ is the sensitivity of the coil. We also define the phase:

$$\varphi(\vec{r},t) := \int_0^t \left(\gamma B_z(\vec{r},t') - \omega_0\right)dt' \tag{2.9}$$

where $\omega_0 = \gamma B_0$ is the demodulation frequency.

The obtained measurements are not exactly equal to this signal due to errors. The ith measurement of the coil at time t_i is given by:

$$y_i = s(t_i) + \varepsilon_i \tag{2.10}$$

with $i = 1,...,n_d$, n_d the number of samples and ε_i the measurement error of the ith sample.

As mentioned before the goal is to obtain an image of $f(\vec{r})$. In order to do this we approximate $f(\vec{r})$ using a finite series expansion, which means we write $f(\vec{r})$ as the sum of a finite number of pixel values:

$$f(\vec{r}) = \sum_{j=1}^{N} f_j b(\vec{r} - \vec{r}_j) \tag{2.11}$$

and rewrite equation 2.8 as:

$$s(t_i) = \sum_{j=1}^{n} a_{ij} f_j \tag{2.12}$$

with

$$\begin{aligned} a_{ij} &= \int c(\vec{r}) b(\vec{r} - \vec{r}_j) e^{-t_i / T_2(\vec{r})} e^{-i\varphi(\vec{r}, t_i)} d\vec{r} \\ &\approx c(\vec{r}_j) e^{-t_i / T_2(\vec{r}_j)} e^{-i\varphi(\vec{r}_j, t_i)} \Delta r \end{aligned} \tag{2.13}$$

Here Δr is the pixel area.
After substitution of 2.12 into 2.10 we get the following linear model, which can be solved for **f**:

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{f} + \boldsymbol{\varepsilon} \tag{2.14}$$

2.2. HIGH-FIELD AND LOW-FIELD MRI

The previous section discussed the physics of all MRI scanners. This section will discuss the differences between the most common high-field MRI scanner and the low-field MRI scanner used in this project and what the consequences of these differences are for matrix $\boldsymbol{A}$.

In general the longitudinal component of the magnetic field, $B_z(\vec{r}, t)$, mentioned in 2.5 can be written as:

$$B_z(\vec{r}, t) = B_0 + \Delta B_0(\vec{r}) + \vec{G}(t) \cdot \vec{r} \tag{2.15}$$

Here $\Delta B_0(\vec{r})$ is the spatial deviation of background field strength B_0 and $\vec{G}(t) = G_x\vec{i} + G_y\vec{j} + G_z\vec{k}$ is the field gradient. These variations $\vec{G}(t)$ are generated by gradient coils and are known.
Substituting 2.15 into 2.9 we get:

$$\varphi(\vec{r}, t) = \int_0^t \gamma \Delta B_0(\vec{r}) + \gamma \vec{G}(t) \cdot \vec{r}\, dt' \tag{2.16}$$

and

$$e^{i\varphi(\vec{r}, t)} = e^{-i\Delta\omega_0(\vec{r})t} e^{-i2\pi\vec{k}(t)\cdot\vec{r}} \tag{2.17}$$

with $\Delta\omega_0(\vec{r}) = \gamma\Delta B_0(\vec{r})$ and $\vec{k}(t) = \frac{1}{2\pi}\int_0^t \gamma\vec{G}(t')dt'$. $\vec{k}(t)$ is called the k-space trajectory.

We can now write the elements a_{ij} (equation 2.13) of matrix $\boldsymbol{A}$ as:

$$a_{ij} \approx c(\vec{r}_j) e^{-t_i / T_2(\vec{r}_j)} e^{-i\Delta\omega_0(\vec{r}_j)t_i} e^{-i2\pi\vec{k}(t_i)\cdot\vec{r}} \Delta r \tag{2.18}$$

High-field and low-field MRI scanners have differences in the magnetic field $B_z(\vec{r}, t)$ and in the time constant of the decay $T_2(\vec{r})$. These differences have an effect on a_{ij}.
High-field MRI scanners have a background magnetic field with a high strength B_0 and have a negligible deviating field: $\Delta B_0 = 0$. The time constant T_2 is quite high, usually in the range of seconds, which means the decay term can be neglected. The expression for a_{ij} becomes:

$$a_{ij} \approx c(\vec{r}_j) e^{-i2\pi\vec{k}(t_i)\cdot\vec{r}_j} \Delta r \tag{2.19}$$

The elements a_{ij} correspond to a standard Fourier transform, which means that operations with $\boldsymbol{A}$ and its inverse can be efficiently performed using a Fast Fourier Transform and Inverse Fast Fourier Transform, respectively.

All low-field MRI scanners have a very small background field $\vec{B}_0$, but can differ in the ΔB_0 and $\vec{G}(t)$ terms. The PSU scanner has an unknown ΔB_0 and a known gradient field $\vec{G}(t)$. Since ΔB_0 is assumed to be small, it is set to zero. In that case we obtain the same expression for a_{ij} as for a high-field scanner and the Inverse Fast Fourier transform can be applied to obtain the image. However, since ΔB_0 need not be close to zero, the images can be expected to be very noisy.
The scanner of the TUD/LUMC combines the low background field $\vec{B}_0$ with natural variations in the magnetic field: $\Delta B_0(\vec{r}) \neq 0$. This ΔB_0 is considered constant in time. In contrast to high-field scanners and the PSU scanner there is no gradient field: $\vec{G}(t) = 0$. The time constant $T_2(\vec{r})$ is in the range of milliseconds and therefore the decay term cannot be neglected. The elements of matrix $\boldsymbol{A}$ become:

$$a_{ij} \approx c(\vec{r}_j) e^{-t_i / T_2(\vec{r}_j)} e^{-i \Delta\omega_0(\vec{r}_j) t_i} \Delta r \tag{2.20}$$

This expression does not correspond to the Fourier transform, which means it is more difficult to obtain $\boldsymbol{f}$ from 2.14. This is a disadvantage of the TUD/LUMC scanner.
A disadvantage of all low-field MRI scanners is the lower signal-to-noise ratio. The SNR is linearly dependent on B_0, therefore the low background field causes a low SNR.

The PSU and TUD/LUMC use different magnets for their scanners. The TUD/LUMC scanner uses a permanent magnet, which means it is always ready for use and no heat is generated. The PSU scanner uses a resistive electromagnet. This type of magnet needs time to startup and does generate heat in the coils. However, due to the low background field not a lot of heat is generated and less cooling is necessary than for a high-field MRI scanner. High-field MRI scanners do not generate heat, but they need cooling to a temperature close to $0K$ to reach superconductivity. Needing less or no cooling makes both low-field MRI scanners a lot cheaper than a general high-field MRI scanner. This is one of the main reasons why a low-field MRI scanner is developed for these projects. Also, since less cooling is needed, the magnets can be used at room temperature, which makes the scanner more user-friendly ([5]).

3

LINEAR DIFFUSION FILTERING

Images obtained with the MRI scanner will contain noise. A way to reduce noise in images is diffusion filtering, which is the main topic of this thesis. The most basic form of diffusion filtering is linear diffusion filtering, which is the focus of this chapter.

3.1. PHYSICAL MOTIVATION OF DIFFUSION

The physical background of diffusion is explained in [1]. Diffusion is the process of reaching an equilibrium in concentration without losing or creating mass. Due to random movements particles move from a higher concentrated area to a lower concentrated area to reach this equilibrium state. The equilibrium property is given by Fick's first law:

$$J = -D \cdot \nabla u \tag{3.1}$$

This equation gives the relation between the diffusion flux J and the concentration gradient ∇u. D is the diffusion coefficient.
Since no mass is destroyed or created the continuity equation is also of importance:

$$\partial_t u = -\mathrm{div} J \tag{3.2}$$

Combining Fick's law with the continuity equation gives the diffusion equation:

$$\partial_t u = \mathrm{div}(D \cdot \nabla u) \tag{3.3}$$

The diffusion equation is applicable to many transport processes. By setting u as the gray value at a certain location instead of the concentration, we can apply this equation to image processing. In this section we assume the diffusion does not depend on the changing image and we set $D = 1$. This leads to linear diffusion filtering. In later chapters we will assume D is a function dependent on $|\nabla u|$, leading to nonlinear or anisotropic diffusion filters.

3.2. HEAT EQUATION

By setting $D = 1$ the diffusion equation in 2D becomes:

$$\begin{aligned} \partial_t u &= \mathrm{div}(\nabla u) \\ &= \Delta u \\ &= \partial_{xx} u + \partial_{yy} u \end{aligned} \tag{3.4}$$

which is also known as the heat equation. When using the heat equation as a filtering method for images it satisfies the following two criteria according to [2]:
- causality: any features at a coarse resolution need to be caused by features at a finer level. This means that no new features can be introduced at a coarse resolution.
- homogeneity and isotropy: the blurring is space invariant. Blurring is fading of the image due to diffusion. This criterium states that every part of the image fades similarly.

The causality criterion says that no new features can be introduced at a coarser level. In other words, no new minimum or maximum can be created in images other than the initial image. The criterium can be validated by using the Maximum Principle. This principle states that all minima and maxima of the solution exist in the original image and on the boundaries. When using Neumann boundary conditions, the Maximum Principle is stronger and says that the minima an maxima can only occur in the original image. Since the heat equation satisfies this principle, we can conclude that no new minima and maxima can occur on coarser levels, thereby meeting the causality criterium.

3.3. IMPLEMENTATION

To implement the heat equation we use the FTCS (forward time, central space) method, which is second order in space and first order in time. This method uses forward difference to estimate the time derivative and central difference for the space derivatives:

$$\begin{aligned}
\partial_t u(x_i, y_j, t_k) &\approx \frac{1}{\Delta t}(u_{i,j}^{k+1} - u_{i,j}^k) \\
\partial_{xx} u(x_i, y_j, t_k) &\approx \frac{1}{\Delta x^2}(u_{i+1,j}^k - 2u_{i,j}^k + u_{i-1,j}^k) \\
\partial_{yy} u(x_i, y_j, t_k) &\approx \frac{1}{\Delta y^2}(u_{i,j+1}^k - 2u_{i,j}^k + u_{i,j-1}^k)
\end{aligned} \tag{3.5}$$

Plugging this in equation 3.4 and setting $\Delta y = \Delta x$ and $\mu = \frac{\Delta t}{\Delta x^2}$ we obtain the following numerical scheme:

$$u_{i,j}^{k+1} = u_{i,j}^k + \mu(u_{i+1,j}^k + u_{i,j+1}^k - 4u_{i,j}^k + u_{i-1,j}^k + u_{i,j-1}^k) \tag{3.6}$$

The number of pixels in the x-direction is equal to the number of pixels in the y-direction and is denoted by n. On the boundaries we assume homogeneous Neumann boundary conditions: $\partial u / \partial \vec{n} = 0$, $\vec{n}$ the normal to the boundary. We have obtained the following system of equations:

$$\vec{U}^{k+1} = (I - \mu A)\vec{U}^k \tag{3.7}$$

with

$$A_{n^2 \times n^2} = \begin{bmatrix} A_1 & A_2 & 0 & 0 & \dots & 0 \\ A_3 & A_1 & A_3 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & A_3 & A_1 & A_3 \\ 0 & \dots & 0 & 0 & A_2 & A_1 \end{bmatrix}$$

$$A_{1,n \times n} = \begin{bmatrix} 4 & -2 & 0 & 0 & \dots & 0 \\ -1 & 4 & -1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & -1 & 4 & -1 \\ 0 & \dots & 0 & 0 & -2 & 4 \end{bmatrix}$$

$$A_{2,n\times n} = 2I$$
$$A_{3,n\times n} = I$$
$$\vec{U} = \begin{bmatrix} U_{1,1} & \dots & U_{n,1} & U_{1,2} & \dots & U_{n,2} & \dots & U_{1,n} & \dots & U_{n,n} \end{bmatrix}^T$$

and I the $n \times n$ identity matrix.

3.4. PROPERTIES OF THE METHOD

First we discuss the stability. To determine the largest possible $\mu = \Delta t/\Delta x^2$ which ensures stability, the Gershgorin circle theorem is used. In [6] it is written as follows:

GERSHGORIN CIRCLE THEOREM
The eigenvalues of a complex square matrix A with elements $a_{i,j}$ are in the union of the circles

$$|z - a_{i,i}| \leq \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{i,j}| \qquad with\ z \in \mathbb{C}$$

When applying this theorem to matrix A we see that $\lambda \leq 8$. We assume $\lambda_{max} = 8$ and set $\Delta x = 1$, which means that $\mu = \Delta t$. We see from 3.7 that we need for stability:

$$\begin{aligned} |1 - \Delta t\lambda| &\leq 1 \\ -1 \leq 1 - \Delta t\lambda &\leq 1 \\ -2 \leq -\Delta t\lambda \qquad &and \qquad -\Delta t\lambda \leq 0 \\ \Delta t \leq \frac{2}{\lambda} \qquad &and \qquad \Delta t\lambda \geq 0 \end{aligned} \tag{3.8}$$

Since $\lambda \geq 0$ and $\lambda_{max} = 8$ we conclude that the method is stable for $0 \leq \Delta t \leq \frac{2}{8} = \frac{1}{4}$. So when implementing this method we need to consider time steps smaller than or equal to 0.25. We shall verify this in chapter 7.

Next we want to prove that the total pixel value stays constant in time. In order to do this we use the Gauss divergence theorem on the difffusion equation given by 3.4:

$$\begin{aligned} \int_\Omega \partial_t u \, d\Omega &= \int_\Omega \nabla \cdot \nabla u \, d\Omega \\ &= \int_{\partial\Omega} \nabla u \cdot \vec{n} \, dS \\ &= \int_{\partial\Omega} \underbrace{\frac{\partial u}{\partial n}}_{=0} \, dS \\ &= 0 \\ \Rightarrow u_{tot}(t) &= \int_\Omega u \, d\Omega = constant \end{aligned} \tag{3.9}$$

For the numerical implementation of the method this means that the total pixel value should be constant over time. In the numerical experiments chapter, chapter 7, we will check this. Of course, the total numerical pixel value TPV is then given by the following sum:

$$TPV = \sum_{i=1}^{n} \sum_{j=1}^{n} u_{i,j} \tag{3.10}$$

Also, we know from [1] that the solution u converges to a constant value. This fact in combination with the constant total pixel value leads to the conclusion that it converges to APV, the average pixel value:

$$APV = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} u_{i,j} \tag{3.11}$$

4

ANISOTROPIC DIFFUSION FILTERING

In the previous chapter we used the heat equation as an image filtering method. As known from the literature, a big disadvantage of this method is the space-invariant blurring. This means that boundaries of regions are not preserved and edges lose their sharpness.
To overcome this problem the 'homogeneity and isotropy' criterium mentioned in the previous section is replaced by two other criteria. The new method needs to satisfy the following three criteria ([2]):
- causality: no new features can be introduced at a coarser grid. The 'cause' of features needs to be in a finer grid.
- immediate localization: at each resolution the region boundaries should have a sharpness suited for that resolution.
- piecewise smoothing: this means that at each resolution intraregion smoothing occurs over interregion smoothing.

To achieve these criteria we can adjust the diffusion coefficient D of equation 3.3. The heat equation was obtained by setting $D = 1$, but it is not necessary for D to be constant. By letting it be a suitable function of $|\nabla u|$ the three criteria mentioned above can be satisfied ([2]):
Let $D = c(|\nabla u|)$, then:

$$\partial_t u = \text{div}(c(|\nabla u|)\nabla u) \tag{4.1}$$

with $|\nabla u| = \sqrt{u_x^2 + u_y^2}$.

Since this equation belongs to a certain class of elliptic equations that satify the Maximum Principle ([2]), we can conclude that the causality criterium is achieved independent of $c(|\nabla u|)$. The other two criteria are depending on $c(|\nabla u|)$, proved by the fact that for $c = 1$ these criteria are not met. To determine which functions are suitable we need to look into edge enhancement.

4.1. EDGE ENHANCEMENT

The goal is to determine which functions $c(|\nabla u|)$ are suitable for image filtering with conservation of sharp boundaries. To do this we follow the steps taken in [2] and consider the 1D case only. First we model an edge as a step function convolved with a Gaussian (figure 4.1a).
In that case equation 4.1 simplifies to:

$$\partial_t u = \partial_x(c(\partial_x u)\partial_x u) \tag{4.2}$$

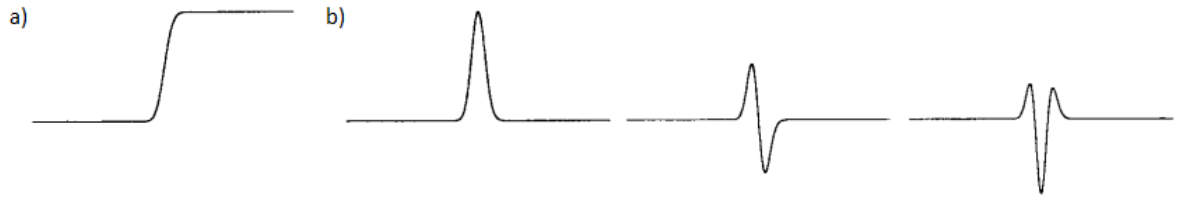

Figure 4.1: a) edge model b) 1st, 2nd, 3rd derivative of u

Let $\phi(\partial_x u) = c(\partial_x u) \cdot \partial_x u$ denote the flux, then we get:

$$\partial_t u = \partial_x \phi(\partial_x u) = \phi'(\partial_x u) \cdot \partial_{xx} u \tag{4.3}$$

Now we look at the variation in time of the slope of the edge, $\partial_t(\partial_x u)$. We assume $c(\cdot) > 0$, then $u(\cdot)$ is smooth and the order of differentiation can be inverted. We get:

$$\begin{aligned} \partial_t(\partial_x u) &= \partial_x(\partial_t u) \\ &= \partial_x(\partial_x \phi(\partial_x u)) \\ &= \partial_x(\phi'(\partial_x u) \cdot \partial_{xx} u) \\ &= \phi''(\partial_x u) \cdot \partial_{xx}^2 u + \phi'(\partial_x u) \cdot \partial_{xxx} u \end{aligned} \tag{4.4}$$

Without loss of generality we can assume that the edge is oriented as in figure 4.1a with $\partial_x u > 0$. We can see that in that case $\partial_{xx} u = 0$ and $\partial_{xxx} u << 0$ at the point of inflection. From equation 4.4 it follows that $\partial_t(\partial_x u)$ has a sign opposite to the sign of $\phi'(\partial_x u)$. Therefore we conclude that the slope of the edge will increase for $\phi'(\partial_x u) < 0$ and decrease for $\phi'(\partial_x u) > 0$.

This conclusion leads to the following threshold property for some threshold value τ:

$$\phi'(\partial_x u) = \begin{cases} > 0, & \partial_x u < \tau \\ < 0, & \partial_x u \geq \tau \end{cases} \tag{4.5}$$

This property states that for edge slopes $\partial_x u$ lower than τ edge blurring will occur and for slopes higher than τ edge sharpening will occur. So by choosing τ correctly, we achieve intraregion smoothing while maintaining sharp edges.

4.2. CHOICE OF DIFFUSION COEFFICIENT $c(\cdot)$

In choosing a correct $c(\cdot)$ it is also important to consider the well-posedness. [7] investigates well-posedness by minimizing the following energy function:

$$E(u) = \int_\Omega F(|\nabla u|) d\Omega = \int_\Omega F(\sqrt{u_x^2 + u_y^2}) d\Omega \tag{4.6}$$

with $F(|\nabla u|)$ a strictly increasing function: $F'(|\nabla u|) > 0$.
The energy function is a measure for the smoothness. Therefore minimizing the energy function is similar to smoothing. The minima of 4.6 are given by the solutions of the following parabolic equation:

$$\partial_t u = \operatorname{div}\left(\frac{F'(|\nabla u|)}{|\nabla u|} \nabla u\right) \tag{4.7}$$

which is similar to 4.1 if the diffusion coefficient is set to:

$$c(|\nabla u|) = \frac{F'(|\nabla u|)}{|\nabla u|} \tag{4.8}$$

As mentioned before, the flux is denoted by $\phi(s) = c(s)s = F'(s)$.

Now we look at the Hessian matrix of $F(|\nabla u|)$ and its eigenvalues. The Hessian is given by:

$$\begin{bmatrix} \partial_{xx}F(|\nabla u|) & \partial_{yx}F(|\nabla u|) \\ \partial_{xy}F(|\nabla u|) & \partial_{yy}F(|\nabla u|) \end{bmatrix} \tag{4.9}$$

and has eigenvalues $\lambda_1 = F'(|\nabla u|)/|\nabla u| = c(|\nabla u|)$ and $\lambda_2 = F''(|\nabla u|)$, as mentioned in [8].

To determine when the problem is well-posed, the surface of the energy function 4.6 and the initial conditions are of interest. When the energy surface is convex, it has a global minimum, which means that any starting point converges to this minimum. The problem is then well-posed. The energy surface is convex for:

$$\lambda_1 \geq 0 \qquad \textit{and} \qquad \lambda_2 \geq 0 \qquad \textit{for all } |\Delta u| \tag{4.10}$$

Since $F'(|\nabla u|) > 0$ we see that $\lambda_1 = F'(|\nabla u|)/|\nabla u| \geq 0$ is always true. So we need $\lambda_2 = F''(|\nabla u|) \geq 0$ for all $|\nabla u|$. However, $\lambda_2 > 0$ corresponds to smoothing in the gradient direction, which means that the edges will blur. Therefore we need to set $\lambda_2 = 0$ and it follows that $F(|\nabla u|) = |\nabla u|$. We conclude that for a well-posed problem a suitable diffusion coefficient is:

$$c(|\nabla u|) = \frac{F'(|\nabla u|)}{|\nabla u|} = \frac{1}{|\nabla u|} \tag{4.11}$$

Choosing λ_2 differently leads to inconvenient energy surfaces, thereby jeopardizing the well-posedness of the problem. For example, the energy surface has a unique saddle-point for:

$$\lambda_2 = F''(|\nabla u|) < 0 \tag{4.12}$$

In this case there are no local minima and the solution converges to a boundary point. Images with initial conditions close to each other will converge to the same solution as long as they are not across the saddle ridge. We get a problem of this type when setting $F(|\nabla u|) = |\nabla u|^{1/N}$, $N > 1$. The diffusion coefficient becomes:

$$c(|\nabla u|) = \frac{F'(|\nabla u|)}{|\nabla u|} = \frac{1}{N}|\nabla u|^{\frac{1}{N}-2}, \qquad N > 1 \tag{4.13}$$

In the previous section we found a threshold property for $\phi'(|\nabla u|)$. This property corresponds to a method which smooths the solution inside regions and sharpens the edges. Since $\phi'(|\nabla u|) = F''(|\nabla u|) = \lambda_2$ we can rewrite the threshold property given by 4.5 to:

$$\lambda_2 = \begin{cases} > 0, & |\nabla u| < \tau \\ < 0, & |\nabla u| \geq \tau \end{cases} \tag{4.14}$$

For this type of λ_2 the problem is ill-posed. The surface is likely to have many local minima, which means that images starting close to each other may converge to a different minimum.

However, the most common anisotropic filtering methods are of this type, due to the importance of the threshold propery. Perona and Malik ([2]) gave two possible functions satisfying the threshold property:

$$c(|\nabla u|) = e^{-(|\nabla u|/K)^2}, \qquad (K > 0) \tag{4.15}$$

and

$$c(|\nabla u|) = \frac{1}{1 + (|\nabla u|/K)^2}, \qquad (K > 0) \tag{4.16}$$

The constant K is chosen by hand or can be estimated in several ways, which are described in chapter 8.

Since the Perona-Malik methods satisfy the threshold property, they lead to ill-posed problems. The ill-posedness can be decreased by adding a smoothing operator to the diffusion coefficient $c(\cdot)$ ([9]). An example of such a smoothing operator is the Gaussian kernel, leading to:

$$\partial_t u = \text{div}(c(|\nabla (G(s) * u)|^2)\nabla u) \tag{4.17}$$

where $G(s) * u$ denotes a convolution of the image at time t with a Gaussian kernel of scale s. $G(s)$ can be seen as the solution at time s of the heat equation with $u(x, y, t)$ as initial data. A large s suppresses more noise, but leads to a blurrier image, while a small s may cause instability.

4.3. PROPERTIES OF THE METHOD

Again we will look into some properties of these methods. First we will discuss the stability. In the last chapter we concluded that the linear diffusion method is stable for all $0 < \Delta t \leq 0.25$ when using the FTCS scheme. This was proved with the Gershgorin circle theorem, also given in chapter 3. The theorem uses matrix A, given in the following system:

$$\vec{U}^{k+1} = \left(I - \mu A\right)\vec{U}^k \tag{4.18}$$

For the anisotropic case the matrix A is different than for the linear case. The matrix is now depending on values of the diffusion coefficient $c(\cdot)$. However, we can still apply the Gershgorin circle theorem to draw a few conclusions.
Consider the two Perona-Malik methods, given by 4.15 and 4.16. We know that both functions can take on values between 0 and 1. Therefore $c_{max} = 1$ in both cases. So the values in the matrix for the anisotropic case can maximally take on the values of the matrix A given in chapter 3. Therefore we can make the same conclusion after using the Gershgorin circle theorem, namely $\lambda \leq 8$ leading to a similar stability region of $0 < \Delta t \leq 0.25$.

Unfortunately the ill-posed and well-posed methods can be much larger than 1. Therefore the values in the matrix can be much higher as well. Using the Gershgorin circle theorem we conclude that λ can take on a much higher value, leading to a much smaller stability region. The exact region is unknown, but it is smaller than $0 < \Delta t \leq 0.25$. In the first section of chapter 7 on numerical experiments, the stability regions will be investigated.

Next we want to prove that also for anisotropic diffusion methods the total pixel value stays constant in time. We can do this in the same manner as for the linear method:

$$\begin{aligned}
\int_\Omega \partial_t u \, d\Omega &= \int_\Omega \nabla \cdot (c(|\nabla u|)\nabla u) \, d\Omega \\
&= \int_{\partial\Omega} (c(|\nabla u|)\nabla u) \cdot \vec{n} \, dS \\
&= \int_{\partial\Omega} \underbrace{\frac{c(|\nabla u|)\partial u}{\partial n}}_{=0} \, dS \\
&= 0 \\
\Rightarrow u_{tot}(t) &= \int_\Omega u \, d\Omega = constant
\end{aligned} \tag{4.19}$$

We see that for the anisotropic methods the total pixel value also stays constant. Numerically, this value is given by TPV, given by equation 3.10.
Similar to the linear case, the solution now also converges to a constant value. Since the total pixel value stays constant in time, we can conclude that u converges to APV, with APV the average pixel value (equation 3.11).

5

FTCS IMPLEMENTATION

All anisotropic methods have been implemented in two ways. This chapter will describe the first: implementation with the Forward Time Central Space (FTCS) method. First we will look into implementation of the Perona-Malik method, both with and without Gaussian kernel. The implementation of the other anisotropic methods are quite similar and are mentioned shortly in the final section of this chapter.

5.1. PERONA-MALIK

Recall the threshold property mentioned in the previous chapter:

$$\phi'(|\nabla u|) = \begin{cases} > 0, & |\nabla u| < \tau \\ < 0, & |\nabla u| \geq \tau \end{cases} \tag{5.1}$$

with $\phi(s) = c(s)s$ and $c(\cdot)$ the diffusion function of equation 4.1:

$$\partial_t u = \operatorname{div}(c(|\nabla u|)\nabla u) \tag{5.2}$$

A method satisfying this property is developed by Perona and Malik ([2]). Their article mentions two possible options for $c(\cdot)$:

$$c(|\nabla u|) = e^{-(|\nabla u|/K)^2}, \qquad (K > 0) \tag{5.3}$$

and

$$c(|\nabla u|) = \frac{1}{1 + (|\nabla u|/K)^2}, \qquad (K > 0) \tag{5.4}$$

To implement any filtering method we need to approximate equation 5.2. We can write $c(|\nabla u|)$ as $c(|\nabla u|^2)$, since both functions are also a function of $|\nabla u|^2$. We start by looking at $c(|\nabla u|^2)\nabla u$ and approximating it using second order central difference:

$$\begin{aligned} c(|\nabla u_{i,j}|^2)\nabla u_{i,j} &= c(|\nabla u_{i,j}|^2)\begin{bmatrix} \frac{\partial u_{i,j}}{\partial x} & \frac{\partial u_{i,j}}{\partial y} \end{bmatrix}^T \\ &= \begin{bmatrix} c(|\nabla u_{i,j}|^2)\frac{\partial u_{i,j}}{\partial x} & c(|\nabla u_{i,j}|^2)\frac{\partial u_{i,j}}{\partial y} \end{bmatrix}^T \\ &\approx \begin{bmatrix} c(|\nabla u_{i,j}|^2)\frac{1}{\Delta x}(u_{i+1/2,j} - u_{i-1/2,j}) & c(|\nabla u_{i,j}|^2)\frac{1}{\Delta y}(u_{i,j+1/2} - u_{1,j-1/2}) \end{bmatrix}^T \end{aligned} \tag{5.5}$$

Now we look at the whole equation 5.2:

$$\begin{aligned}
\partial_t u =& \operatorname{div}(c(|\nabla u|^2)\cdot\nabla u)\\
\approx& \operatorname{div}\left(\left[c(|\nabla u_{i,j}|^2)\tfrac{1}{\Delta x}(u_{i+1/2,j}-u_{i-1/2,j}) \quad c(|\nabla u_{i,j}|^2)\tfrac{1}{\Delta y}(u_{i,j+1/2}-u_{1,j-1/2})\right]^T\right)\\
=& \frac{\partial}{\partial x}\left(c(|\nabla u_{i,j}|^2)\frac{1}{\Delta x}(u_{i+1/2,j}-u_{i-1/2,j})\right)+\frac{\partial}{\partial y}\left(c(|\nabla u_{i,j}|^2)\frac{1}{\Delta y}(u_{i,j+1/2}-u_{1,j-1/2})\right)\\
\approx& \frac{1}{\Delta x}\left(c(|\nabla u_{i+1/2,j}|^2)\frac{1}{\Delta x}(u_{i+1,j}-u_{i,j})-c(|\nabla u_{i-1/2,j}|^2)\frac{1}{\Delta x}(u_{i,j}-u_{i-1,j})\right)\\
&+\frac{1}{\Delta y}\left(c(|\nabla u_{i,j+1/2}|^2)\frac{1}{\Delta y}(u_{i,j+1}-u_{1,j})-c(|\nabla u_{i,j-1/2}|^2)\frac{1}{\Delta y}(u_{i,j}-u_{1,j-1})\right)
\end{aligned} \tag{5.6}$$

We need to estimate the values for $c(|\nabla u_{i\pm1/2,j}|^2)$ and $c(|\nabla u_{i,j\pm1/2}|^2)$. We use the standard second order central difference formula to estimate $\nabla u_{i,j}$, which leads to:

$$\begin{aligned}
&\nabla u_{i,j} \approx \left[\tfrac{1}{2\Delta x}(u_{i+1,j}-u_{i-1,j} \quad \tfrac{1}{2\Delta y}(u_{i,j+1}-u_{i,j-1})\right]^T\\
\Rightarrow \qquad &|\nabla u_{i,j}|^2 \approx \frac{1}{(2\Delta x)^2}(u_{i+1,j}-u_{i-1,j})^2+\frac{1}{(2\Delta y)^2}(u_{i,j+1}-u_{i,j-1})^2
\end{aligned} \tag{5.7}$$

All these values are known and we can calculate $c(|u_{i,j}|^2)$ for all i,j.
To find $c(|\nabla u_{i\pm1/2,j}|^2)$ and $c(|\nabla u_{i,j\pm1/2}|^2)$ we take the average:

$$\begin{aligned}
c(|\nabla u_{i+1/2,j}|^2) &\approx \frac{1}{2}(c(|\nabla u_{i+1,j}|^2)+c(|\nabla u_{i,j}|^2))\\
c(|\nabla u_{i-1/2,j}|^2) &\approx \frac{1}{2}(c(|\nabla u_{i-1,j}|^2)+c(|\nabla u_{i,j}|^2))\\
c(|\nabla u_{i,j+1/2}|^2) &\approx \frac{1}{2}(c(|\nabla u_{i,j+1}|^2)+c(|\nabla u_{i,j}|^2))\\
c(|\nabla u_{i,j-1/2}|^2) &\approx \frac{1}{2}(c(|\nabla u_{i,j-1}|^2)+c(|\nabla u_{i,j}|^2))
\end{aligned} \tag{5.8}$$

We use forward difference (first order Euler forward) to estimate the time derivative. Setting $\Delta x = \Delta y = 1$, giving $\mu = \Delta t$, we obtain the following numerical scheme:

$$\begin{aligned}
u_{i,j}^{k+1} =& u_{i,j}^k+\Delta t\left(c(|\nabla u_{i+1/2,j}^k|^2)(u_{i+1,j}^k-u_{i,j}^k)-c(|\nabla u_{i-1/2,j}^k|^2)(u_{i,j}^k-u_{i-1,j}^k)\right)\\
&+\Delta t\left(c(|\nabla u_{i,j+1/2}^k|^2)(u_{i,j+1}^k-u_{1,j}^k)-c(|\nabla u_{i,j-1/2}^k|^2)(u_{i,j}^k-u_{1,j-1}^k)\right)
\end{aligned} \tag{5.9}$$

with the values for the diffusion function obtained by using 5.7 and 5.8.

5.2. PERONA-MALIK WITH GAUSSIAN KERNEL

As mentioned many times before, the Perona-Malik method leads to an ill-posed problem. We can add a smoothing operator to the method, such as a Gaussian kernel, to decrease this ill-posedness ([9]):

$$\partial_t u = \operatorname{div}(c(|\nabla (G(s)*u)|^2)\nabla u) \tag{5.10}$$

where $G(s)*u$ denotes a convolution of the image at time t with a Gaussian kernel of scale s.

The implementation is quite similar to the implementation in the previous chapter. There is only one difference: instead of $c(|\nabla u|^2)$ we have $c(|\nabla G(s) * u|^2$. This means that instead of calculating the norm of the gradient of u, we need the norm of the gradient of the convolution of u.

Let $\tilde{u} = G(s) * u$. The numerical scheme is then similar to 5.9, but with $\tilde{u}$ instead of u:

$$\begin{aligned} u_{i,j}^{k+1} =& u_{i,j}^k + \Delta t \left(c(|\nabla \tilde{u}_{i+1/2,j}^k|^2)(\tilde{u}_{i+1,j}^k - \tilde{u}_{i,j}^k) - c(|\nabla \tilde{u}_{i-1/2,j}^k|^2)(\tilde{u}_{i,j}^k - \tilde{u}_{i-1,j}^k) \right) \\ &+ \Delta t \left(c(|\nabla \tilde{u}_{i,j+1/2}^k|^2)(\tilde{u}_{i,j+1}^k - \tilde{u}_{1,j}^k) - c(|\nabla \tilde{u}_{i,j-1/2}^k|^2)(\tilde{u}_{i,j}^k - \tilde{u}_{1,j-1}^k) \right) \end{aligned} \tag{5.11}$$

with the values for the diffusion function obtained by using 5.7 and 5.8 with $\tilde{u}$ instead of u.

5.3. OTHER DIFFUSION FILTERING METHODS

In [7] two other diffusion coefficients $c(\cdot)$ were mentioned. One of these methods leads to a well-posed problem and was given by 4.11:

$$c(|\nabla u|) = \frac{1}{|\nabla u|} \tag{5.12}$$

The other diffusion coefficient, given by 4.13, leads to an ill-posed problem:

$$c(|\nabla u|) = \frac{1}{N} |\nabla u|^{\frac{1}{N}-2}, \qquad N > 1 \tag{5.13}$$

The implementation is again quite similar to the implementation of the Perona-Malik model (chapter 4). However, in this case it is not convenient to write $c(|\nabla u|^2)$ instead of $c(|\nabla u|)$, since the functions are only a function of $|\nabla u|$. We calculate $|\nabla u|$ by taking the square root of equation 5.7:

$$\begin{aligned} \nabla u_{i,j} &\approx \left[\tfrac{1}{2\Delta x}(u_{i+1,j} - u_{i-1,j} \quad \tfrac{1}{2\Delta y}(u_{i,j+1} - u_{i,j-1}) \right]^T \\ \Rightarrow \qquad |\nabla u_{i,j}| &\approx \sqrt{\frac{1}{(2\Delta x)^2}(u_{i+1,j} - u_{i-1,j})^2 + \frac{1}{(2\Delta y)^2}(u_{i,j+1} - u_{i,j-1})} \end{aligned} \tag{5.14}$$

After setting $\Delta x = \Delta y = 1$, the numerical scheme is similar to 5.9:

$$\begin{aligned} u_{i,j}^{k+1} =& u_{i,j}^k + \Delta t \left(c(|\nabla u_{i+1/2,j}^k|)(u_{i+1,j}^k - u_{i,j}^k) - c(|\nabla u_{i-1/2,j}^k|)(u_{i,j}^k - u_{i-1,j}^k) \right) \\ &+ \Delta t \left(c(|\nabla u_{i,j+1/2}^k|)(u_{i,j+1}^k - u_{1,j}^k) - c(|\nabla u_{i,j-1/2}^k|)(u_{i,j}^k - u_{1,j-1}^k) \right) \end{aligned} \tag{5.15}$$

with the values for the diffusion function again obtained by using 5.14 and taking the average:

$$\begin{aligned} c(|\nabla u_{i+1/2,j}|) &\approx \frac{1}{2}(c(|\nabla u_{i+1,j}|) + c(|\nabla u_{i,j}|)) \\ c(|\nabla u_{i-1/2,j}|) &\approx \frac{1}{2}(c(|\nabla u_{i-1,j}|) + c(|\nabla u_{i,j}|)) \\ c(|\nabla u_{i,j+1/2}|) &\approx \frac{1}{2}(c(|\nabla u_{i,j+1}|) + c(|\nabla u_{i,j}|)) \\ c(|\nabla u_{i,j-1/2}|) &\approx \frac{1}{2}(c(|\nabla u_{i,j-1}|) + c(|\nabla u_{i,j}|)) \end{aligned} \tag{5.16}$$

Due to the division by $|\nabla u|$, the methods diverge when $|\nabla u|$ approaches zero. This is the case when neighboring pixels have an almost similar pixel value. To overcome this problem we set $c(|\nabla u|)$ as follows for a certain constant $K > 0$:

$$c(|\nabla u|) = \begin{cases} \frac{1}{|\nabla u|}, & |\nabla u| \geq K \\ \frac{1}{K}, & |\nabla u| < K \end{cases} \tag{5.17}$$

and

$$c(|\nabla u|) = \begin{cases} \frac{1}{N}|\nabla u|^{\frac{1}{N}-2}, & |\nabla u| \geq K, N > 1 \\ \frac{1}{N}K^{\frac{1}{N}-2}, & |\nabla u| < K, N > 1 \end{cases} \tag{5.18}$$

By doing this we maintain the continuity of $c(|\nabla u|)$ and when choosing K correctly we keep the outcome of the original methods, without diverging to an unstable solution.

6

AOS IMPLEMENTATION

Another way of implementing nonlinear diffusion methods is by using the additive operator splitting (AOS) scheme. This scheme uses additive splitting, which ensures that all coordinate axes are treated in the same manner. The scheme preserves the average gray level, satisfies the Maximum Principle and converges to a constant steady state [10]. The scheme can be used for all methods, the only difference is the diffusion coefficient $c(\cdot)$.

6.1. 1-DIMENSIONAL CASE

Recall the nonlinear diffusion equation, given by equation 4.1 in chapter 3:

$$\partial_t u = \operatorname{div}(c(|\nabla u|)\nabla u) \tag{6.1}$$

First we consider the 1D case [11]. Equation 6.1 becomes:

$$\partial_t u = \partial_x(c(|\partial_x u|)\partial_x u) \tag{6.2}$$

An approximation of this 1D equation can be obtained as follows, by using first order central difference:

$$\begin{aligned}
\partial_t u_i =& \partial_x (c(u_i)\partial_x u_i) \\
\approx& \frac{1}{h}(c(u_{i+1/2})\partial_x u_{i+1/2} - c(u_{i-1/2})\partial_x u_{i-1/2}) \\
\approx& \frac{1}{h}\left(c(u_{i+1/2})\frac{u_{i+1}-u_i}{h} - c(u_{i-1/2})\frac{u_i - u_{i-1}}{h}\right) \\
\approx& \frac{1}{h^2}\left(\frac{c(u_{i+1})+c(u_i)}{2}(u_{i+1}-u_i) - \frac{c(u_{i-1})+c(u_i)}{2}(u_i - u_{i-1})\right) \\
=& \frac{c(u_{i+1})+c(u_i)}{2h^2}(u_{i+1}-u_i) + \frac{c(u_{i-1})+c(u_i)}{2h^2}(u_{i-1}-u_i)
\end{aligned} \tag{6.3}$$

Using the Euler forward method for the time derivative, we obtain the following discretization, known as the Perona-Malik scheme:

$$\frac{u_i^{k+1}-u_i^k}{\Delta t} = \sum_{j\in\mathcal{N}(i)} \frac{c_j^k + c_i^k}{2h^2}(u_j^k - u_i^k) \tag{6.4}$$

with
$u_i^k \approx u(x_i, t_k)$,
$\mathcal{N}(i)$ the set of the two neighbouring pixels of pixel i in the x-direction and
$c_i^k \approx c(|\partial_x u(x_i, t_k)|^2)$, which is defined as $c_i^k := c\left(\left(\frac{u_{i+1}-u_{i-1}}{2h}\right)^2\right)$.

By substituting $u_{i,j}^k$ for $u_{i,j}^{k+1}$ we can modify the explicit scheme to the following semi-implicit scheme:

$$\frac{u_i^{k+1} - u_i^k}{\Delta t} = \sum_{j \in \mathcal{N}(i)} \frac{c_j^k + c_i^k}{2h^2}(u_j^{k+1} - u_i^{k+1}) \tag{6.5}$$

Using matrix-vector notation we can rewrite this to:

$$\frac{\vec{U}^{k+1} - \vec{U}^k}{\Delta t} = A(\vec{U}^k) \cdot \vec{U}^{k+1} \tag{6.6}$$

which has the following solution:

$$\vec{U}^{k+1} = \left(I - \Delta t A(\vec{U}^k)\right)^{-1} \vec{U}^k \tag{6.7}$$

with

$$A(\vec{U}^k) = \frac{1}{2h^2}\begin{bmatrix} -2(c_1+c_2) & 2(c_1+c_2) & 0 & \dots & 0 \\ c_1+c_2 & -c_1-2c_2-c_3 & c_2+c_3 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & c_{n-2}+c_{n-1} & -c_{n-2}-2c_{n-1}-c_n & c_{n-1}+c_n \\ 0 & \dots & 0 & 2(c_{n-1}+c_n) & -2(c_{n-1}+c_n) \end{bmatrix}$$

$$\vec{U} = \begin{bmatrix} u_1 & u_2 & \dots & u_n \end{bmatrix}^T$$

.

6.2. HIGHER DIMENSIONAL CASE

In higher dimensions we can obtain a similar solution [11]. The nonlinear diffusion equation in m dimensions is given by:

$$\partial_t u = \sum_{\ell=1}^{m} \partial_{x_\ell}\left(c(|\nabla u|^2)\partial_{x_\ell} u\right) \tag{6.8}$$

Using the same approximation as for the 1D case we obtain the following explicit scheme:

$$\frac{u_i^{k+1} - u_i^k}{\Delta t} = \sum_{\ell=1}^{m} \sum_{j \in \mathcal{N}_\ell(i)} \frac{c_j^k + c_i^k}{2h^2}\left(u_j^k - u_i^k\right) \tag{6.9}$$

Again with the small modification we obtain the semi-implicit scheme:

$$\frac{u_i^{k+1} - u_i^k}{\Delta t} = \sum_{\ell=1}^{m} \sum_{j \in \mathcal{N}_\ell(i)} \frac{c_j^k + c_i^k}{2h^2}\left(u_j^{k+1} - u_i^{k+1}\right) \tag{6.10}$$

Using matrix-vector notation we can rewrite this to:

$$\frac{\vec{U}^{k+1} - \vec{U}^k}{\Delta t} = \sum_{\ell=1}^{m} A_\ell(\vec{U}^k)\vec{U}^{k+1} \tag{6.11}$$

with its solution given by:

$$\begin{aligned}\vec{U}^{k+1} &= \left(I - \Delta t \sum_{\ell=1}^{m} A_\ell(\vec{U}^k)\right)^{-1} \vec{U}^k \\ &= \left(\sum_{\ell=1}^{m} \frac{1}{m}\left(I - \Delta t m A_\ell(\vec{U}^k)\right)\right)^{-1} \vec{U}^k\end{aligned} \tag{6.12}$$

To obtain the additive operator splitting (AOS) scheme we need to modify this solution slightly. Instead of taking the inverse of the entire summation, we sum up the inverses of the operators $B_\ell(\vec{U}^k) = I - \Delta t m A_\ell(\vec{U}^k)$. This means the AOS scheme is given by:

$$\vec{U}^{k+1} = \frac{1}{m} \sum_{\ell=1}^{m} \left(I - \Delta t m A_\ell(\vec{U}^k)\right)^{-1} \vec{U}^k \tag{6.13}$$

The solution obtained with the AOS method is slightly different than the outcome of 6.12, but does have the same order of approximation. The operators $B_\ell(\vec{U}^k) = I - \Delta t m A_\ell(\vec{U}^k)$ describe the one-dimensional diffusion processes along the x_ℓ axes. When choosing the pixel numbering correctly, they are strictly diagonally dominant tridiagonal matrices, which makes solving the problem much simpler.
Another advantage of the AOS method is the fact that this scheme is stable for all step sizes $\Delta t > 0$, which is shown in [1] . This means that it may be possible to find adequate results in only one time step.

6.3. 2-DIMENSIONAL CASE

In MRI images we are dealing with two dimensions: the x-axis and the y-axis. Therefore we will look at the 2D case of the AOS method in detail using some of the notation and theory of [12].

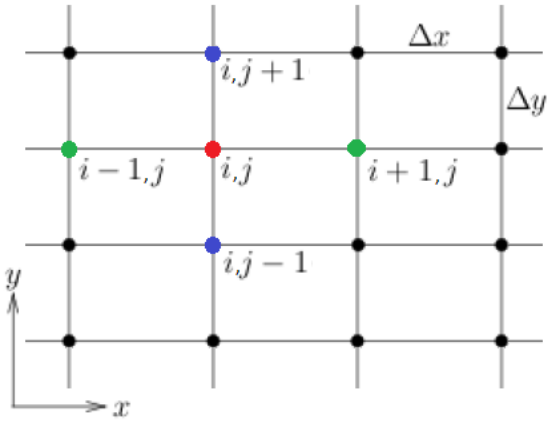

Figure 6.1: Discretization of the image: green dots are neighbours on the x-axis, blue dots are neighbours on the y-axis

As mentioned in the previous section we have the following semi-implicit approximation of the diffusion equation.

$$\frac{u_i^{k+1} - u_i^k}{\Delta t} = \sum_{\ell=1}^{m} \sum_{j \in \mathcal{N}_\ell(i)} \frac{c_j^k + c_i^k}{2h^2}\left(u_j^{k+1} - u_i^{k+1}\right) \tag{6.14}$$

When discretizing the image as shown in figure 6.1, we can rewrite this to:

$$\begin{aligned}\frac{u_{i,j}^{k+1}-u_{i,j}^{k}}{\Delta t} &= \underbrace{\frac{c_{i-1,j}^{k}+c_{i,j}^{k}}{2h^2}}_{c_{W\,i,j}}\left(u_{i-1,j}^{k+1}-u_{i,j}^{k+1}\right)+\underbrace{\frac{c_{i+1,j}^{k}+c_{i,j}^{k}}{2h^2}}_{c_{E\,i,j}}\left(u_{i+1,j}^{k+1}-u_{i,j}^{k+1}\right)\\ &\quad+\underbrace{\frac{c_{i,j-1}^{k}+c_{i,j}^{k}}{2h^2}}_{c_{S\,i,j}}\left(u_{i,j-1}^{k+1}-u_{i,j}^{k+1}\right)+\underbrace{\frac{c_{i,j+1}^{k}+c_{i,j}^{k}}{2h^2}}_{c_{N\,i,j}}\left(u_{i,j+1}^{k+1}-u_{i,j}^{k+1}\right)\\ \Rightarrow u_{i,j}^{k} &= \left(1+\Delta t(c_W+c_E+c_S+c_N)\right)u_{i,j}^{k+1}\\ &\quad-\Delta t c_W u_{i-1,j}^{k+1}-\Delta t c_E u_{i+1,j}^{k+1}-\Delta t c_S u_{i,j-1}^{k+1}-\Delta t c_N u_{i,j+1}^{k+1}\end{aligned} \tag{6.15}$$

The notation $c_{\{W,E,S,N\}i,j}$ come from the directions west, east, south and north.
We can write the equation in matrix-vector form, which leads to the following:

$$\begin{aligned}\vec{U}^{k} &= (I-\Delta t A)\vec{U}^{k+1}\\ \Rightarrow \vec{U}^{k+1} &= (I-\Delta t A)^{-1}\vec{U}^{k}\end{aligned} \tag{6.16}$$

with

$$A_{n^2\times n^2}=\begin{bmatrix} A_{C1,p=1} & 2A_{N,p=1} & 0 & 0 & \dots & 0\\ A_{S,p=2} & A_{C2,p=2} & A_{N,p=2} & 0 & \dots & 0\\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots\\ 0 & \dots & 0 & A_{S,p=n-1} & A_{C2,p=n-1} & A_{N,p=n-1}\\ 0 & \dots & 0 & 0 & 2A_{S,p=n} & A_{C3,p=n}\end{bmatrix}$$

$$A_{C1,p,n\times n}=\begin{bmatrix} -2c_{E,1,p}-2c_{N,1,p} & 2c_{E,1,p} & 0 & \dots & 0\\ \vdots & \ddots & \ddots & \ddots & \vdots\\ 0 & \dots & c_{W,n-1,p} & -c_{W,n-1,p}-c_{E,n-1,p}-2c_{N,n-1,p} & c_{E,n-1,p}\\ 0 & \dots & 0 & 2c_{W,n,p} & -2c_{W,n,p}-2c_{N,n,p}\end{bmatrix}$$

$$A_{C2,p,n\times n}=\begin{bmatrix} -2c_{E,1,p}-c_{S,1,p}-c_{N,1,p} & 2c_{E,1,p} & 0 & \dots & 0\\ \vdots & \ddots & \ddots & \ddots & \vdots\\ 0 & \dots & c_{W,n-1,p} & -(c_W+c_E+c_S+c_N)_{n-1,p} & c_{E,n-1,p}\\ 0 & \dots & 0 & 2c_{W,n,p} & -2c_{W,n,p}-c_{S,n,p}-c_{N,n,p}\end{bmatrix}$$

$$A_{C3,p,n\times n}=\begin{bmatrix} -2c_{E,1,p}-2c_{S,1,p} & 2c_{E,1,p} & 0 & \dots & 0\\ \vdots & \ddots & \ddots & \ddots & \vdots\\ 0 & \dots & c_{W,n-1,p} & -c_{W,n-1,p}-c_{E,n-1,p}-2c_{S,n-1,p} & c_{E,n-1,p}\\ 0 & \dots & 0 & 2c_{W,n,p} & -2c_{W,n,p}-2c_{S,n,p}\end{bmatrix}$$

$$A_{N/S,p,n\times n}=\begin{bmatrix} c_{N/S,1,p} & 0 & \dots & 0\\ \vdots & \ddots & \ddots & \vdots\\ 0 & \dots & c_{N/S,n-1,p} & 0\\ 0 & \dots & 0 & c_{N/S,n,p}\end{bmatrix}$$

$$\vec{U}=\begin{bmatrix}u_{1,1} & \dots & u_{n,1} & u_{1,2} & \dots & u_{n,2} & \dots & u_{1,n} & \dots u_{n,n}\end{bmatrix}^T$$

Note that $c_{N,i,j}=c_{S,i,j+1}$ and therefore $A_{N,p=1}=A_{S,p=2}$. Also, $c_{E,i,j}=c_{W,i+1,j}$.

The matrix A is the matrix for both axes and is not a tridiagonal matrix, which can lead to a long calculation time when solving the system. As mentioned in the previous section, we can obtain a tridiagonal matrix A_ℓ for each dimension when choosing the pixel numbering correctly.
Consider the pixel numbering given by figure 6.2. The left image is a standard numbering of the pixels and the right image gives its transverse. We see that the neighbours in the x-direction become neighbours in the y-direction, and the other way around. We can use this to obtain the two tridiagonal matrices $A_{1,2}$.

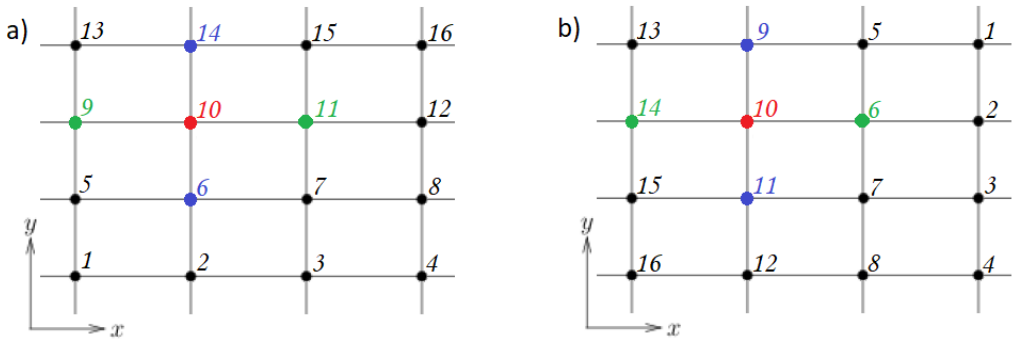

Figure 6.2: a) original numbering, b) transverse of the original numbering

First consider the original numbering and look at each row separately. By only taking the neighbours in the x-direction into account, we can simplify the problem to n 1D problems. The matrix A_1 corresponding to the x-direction becomes:

$$A_1, n^2 \times n^2 = \begin{bmatrix} A_{1D,p=1} & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & A_{1D,p=n-1} & 0 \\ 0 & \dots & 0 & A_{1D,p=n} \end{bmatrix} \tag{6.17}$$

with $A_{1D,p}$ the matrix A of the 1D case, given in the first section of this chapter:

$$A_{1D,p,n\times n} = \frac{1}{2h^2} \begin{bmatrix} -2c_{E,1,p} & 2c_{E,1,p} & 0 & \dots & 0 \\ c_{W,2,p} & -c_{W,2,p} - c_{E,2,p} & c_{E,2,p} & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & c_{W,n-1,p} & -c_{W,n-1,p} - c_{E,n-1,p} & c_{E,n-1,p} \\ 0 & \dots & 0 & 2c_{W,n,p} & -2c_{W,n,p} \end{bmatrix}$$

Now consider the right image with the transverse numbering. The neighbours in the y-direction have become the neighbours in the x-direction. So by using the same tridiagonal matrix A_1 on the transverse numbering, we can obtain the solution for the y-direction.

Recall the AOS scheme, given by 6.13:

$$\vec{U}^{k+1} = \frac{1}{m} \sum_{\ell=1}^{m} \left(I - \Delta t m A_\ell(\vec{U}^k) \right)^{-1} \vec{U}^k \tag{6.18}$$

By using both the original and the transpose numbering this becomes:

$$\begin{aligned}\vec{U}^{k+1} =& \frac{1}{2}\sum_{\ell=1}^{2}\left(I-2\Delta t A_\ell(\vec{U}^k)\right)^{-1}\vec{U}^k \\ =& \frac{1}{2}\left(I-2\Delta t A_1(\vec{U}^k)\right)^{-1}\vec{U}^k + \frac{1}{2}\left(I-2\Delta t A_1(\vec{U}^k_{tr})\right)^{-1}\vec{U}^k_{tr}\end{aligned} \tag{6.19}$$

with $\vec{U}_{tr}$ the pixelvalues in vector notation corresponding to the transverse numbering.

6.4. TRIDIAGONAL MATRIX ALGORITHM

Since $B_\ell(\vec{U}^k) = I-\Delta t m A_\ell(\vec{U}^k)$ are strictly diagonally dominant tridiagonal matrices, the tridiagonal matrix algorithm (TDMA) or Thomas algorithm ([13]) can be used to find the solution in linear time. The TDMA consists of two parts: a forward elimination phase and a backward substitution phase.

Consider the following tridiagonal system

$$\begin{bmatrix} b_1 & c_1 & 0 & \dots & 0 \\ a_2 & b_2 & c_2 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & \dots & 0 & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ . \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ . \\ y_{n-1} \\ y_n \end{bmatrix} \tag{6.20}$$

Now look at the following modification of the first two equations (forward substitution phase):

$$\begin{aligned}&\text{Eq}_{i=2}b_1 - \text{Eq}_{i=1}a_2 \\ \Rightarrow &(b_1b_2 - c_1a_2)x_2 + b_1c_2x_3 = b_1y_2 - a_2y_1\end{aligned} \tag{6.21}$$

By doing this modification x_1 has been eliminated from the equation. In the same way we can eliminate x_2, now using the modified equation for $i=2$ (given by 6.21) and the equation for $i=3$:

$$\begin{aligned}&\text{Eq}_{i=3}(b_1b_2 - c_1a_3) - \text{mod. Eq}_{i=2}a_3 \\ \Rightarrow &(b_3(b_1b_2 - c_1a_2) - c_2b_1a_3)x_3 + c_3(b_1b_2 - c_1a_2)x_4 = y_3(b_1b_2 - c_1a_2) - (y_2b_1 - y_1a_2)a_3\end{aligned} \tag{6.22}$$

We can repeat this process until the equation for $i=n$. The final equation will only involve the unknown x_n and can be used to solve the modified equation for $i=n-1$ and so on. In this backward substitution phase the total solution x is obtained.

Overall the algorithm is given by:

$$\begin{aligned}&\beta_1 = b_1 \\ &\gamma_1 = y_1/\beta_1 \\ &\texttt{For}\ \ i = 2,\dots,n \\ &\qquad \beta_i = b_i - (a_ic_{i-1}/\beta_{i-1}) \\ &\qquad \gamma_i = (y_i - a_i\gamma_{i-1})/\beta_i \\ &\texttt{End} \\ &x_n = \gamma_n \\ &\texttt{For}\ \ j = 1,\dots,n-1 \\ &\qquad x_{n-j} = \gamma_{n-j} - c_{n-j}x_{n-j+1}/\beta_{n-j} \\ &\texttt{End}\end{aligned}$$

The TDMA does have one limitation: it can only be used when

$$|b_i| > |a_i| + |c_i|, \qquad i = 1, \ldots, n \tag{6.23}$$

Since our matrix B_ℓ is strictly diagonally dominant it fulfills this requirement.

7

NUMERICAL EXPERIMENTS: IMPLEMENTATION

All diffusion filtering methods and implementations described so far are tested on three test problems. The first two test problems use the Shepp-Logan phantom, which is a model of the human head. The third problem uses a different phantom, the Ella-phantom.
The first problem is created by adding Gaussian white noise with zero mean and standard deviation 0.06 to the phantom. The second and third problem are the most realistic, since they represent the expected noise of the TUD/LUMC MRI scanner, which is thermal or Johnson noise. The second test problem is an image calculated by the MRI simulator described in [5]. The simulator models the ε given in 2.14 as thermal or Johnson noise. This type of noise is caused by small random voltage fluctuations. The Johnson noise is approximated by Gaussian noise with a zero mean and a standard deviation of the root mean square value given by the following equation:

$$N_{rms} = \sqrt{4k_B T_t R \Delta f} \tag{7.1}$$

Here $k_B = 1.38 \cdot 10^{-23}$ m^2 kg s^{-2} K^{-1} is the Boltzmann constant, T_t is the temperature, R is the resistance of the receiver and Δf the bandwidth. The maximum bandwidth of the TUD/LUMC prototype scanner is 20 kHz and assuming a coil resistance of 1Ω and room temperature of 293 K leads to a standard deviation of 20 nV. More details on the simulation can be found in [5].
The third problem also uses Johnson noise, but on the Ella phantom. Also, the standard deviation is different, it is smaller than 20nV.
All methods were tested for several values of time step size Δt, number of time steps T and parameter K. To determine which parameter combination has the best results for each method and to compare the methods, a difference between the solution and the original Shepp-Logan phantom needs to be calculated. This difference δ is chosen as the standard deviation of the noise in the image. This is given by the 2-norm of the difference between $u_{original}$ and u_{final} divided by the number of pixels in one direction, n:

$$\delta = \frac{1}{n} ||u_{original} - u_{final}||_2 \tag{7.2}$$

Before testing all diffusion filtering methods and their implementation, a section is dedicated to the properties of the methods. These properties are the stability, the total pixel value and the convergence to the average pixel value. All these qualities have also been tested on some of the test problems.

7.1. PROPERTIES OF THE METHODS

All experiments in this section have been implemented with the FTCS method.

7.1.1. LINEAR DIFFUSION FILTERING

This subsection discusses the properties of the linear diffusion filtering method, the heat equation. First the stability is tested. In chapter 3 it was shown the stability region was given by $0 < \Delta t \leq 0.25$. We test this region for the first test problem, Shepp-Logan with Gaussian noise ($n = 200$), and the third test problem, Ella with Johnson noise. The experiment is executed for $T = 50$ and $\Delta t = 0.25$ and 0.26. Figure 7.1 and 7.2 show the results.
The stability region seems to be correct. For $\Delta t = 0.25$ both test problems lead to a good result, while the outcome for $\Delta t = 0.26$ is clearly unstable.

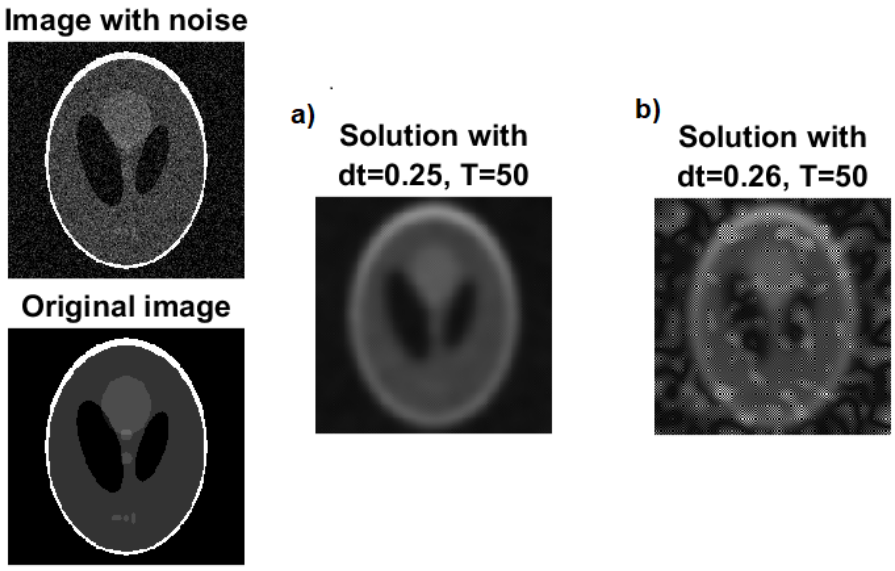

Figure 7.1: Stability for a) $\Delta t = 0.25$ and instability for b) $\Delta t = 0.26$ for the Shepp-Logan phantom

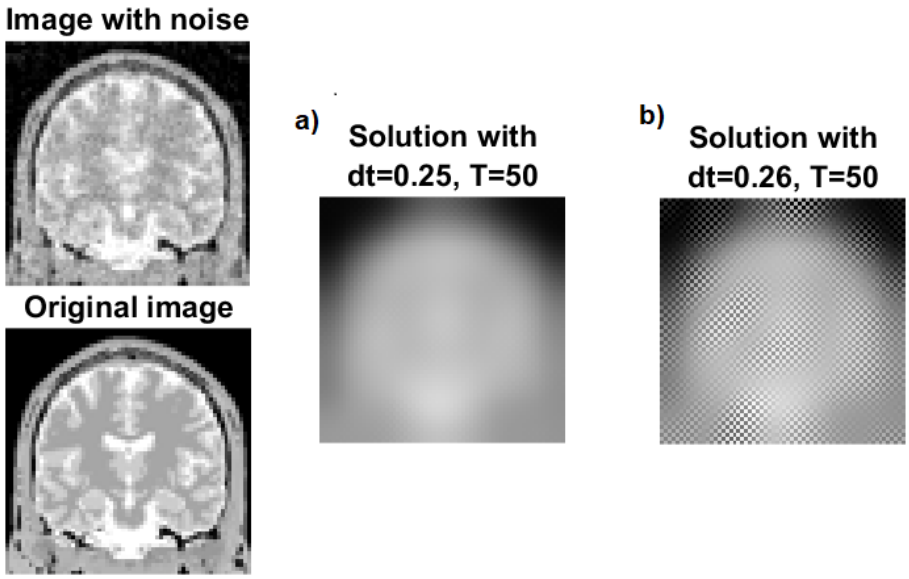

Figure 7.2: Stability for a) $\Delta t = 0.25$ and instability for b) $\Delta t = 0.26$ for the Ella phantom

Next the total pixel value is taken into consideration. We stated in chapter 3 that this TPV stays constant over time. Also, we stated that the solution u converges to the average pixel value (APV). Both statements will be checked for the Shepp-Logan and Ella test problems. We calculated the TPV for $T = 0$ (image with noise), 10, 20, 50, 100 and 100000, with a time step op $\Delta t = 0.1$. For $T = 100000$ we calculated the average pixel value and checked if this was the value of all pixels in the image. The values of TPV can be found in table 7.1 and 7.2. Figures 7.3 and 7.4 show the images corresponding to the tables.

From the tables we can see that the total pixel value is not exactly constant, but changes only slightly. For the Shepp-Logan phantom the TPV decreases at first, but comes back up to the original value. For the Ella phantom it increases from $2.3085 \cdot 10^3$ to $2.3575 \cdot 10^3$. An explanation for these variations in TPV can be found in the fact that the matrix for this problem has eigenvalues 0. These zero-eigenvalues come from the Neumann boundary conditions, which cause the problem to have multiple solutions. We know the amplification factor is 1 for the zero-eigenvalues, which means that the numerical errors are not damped. Each error in the direction of the corresponding eigenvectors are added to the exisiting error, causing the total pixel value to either increase or decrease.
For both problems the solution does converge to the average pixel value ($TPV_{T=100000}/n^2$).

T	0	10	20	50	100	100000
TPV ($\cdot 10^3$)	7.2484	7.2475	7.2473	7.2473	7.2484	7.2484
$APV(T = 100000)$	0.1823					

Table 7.1: Total pixel value for Shepp-Logan ($n = 200$)

T	0	10	20	50	100	100000
TPV ($\cdot 10^3$)	2.3085	2.3177	2.3216	2.3273	2.3317	2.3575
$APV(T = 100000)$	0.5756					

Table 7.2: Total pixel value for Ella

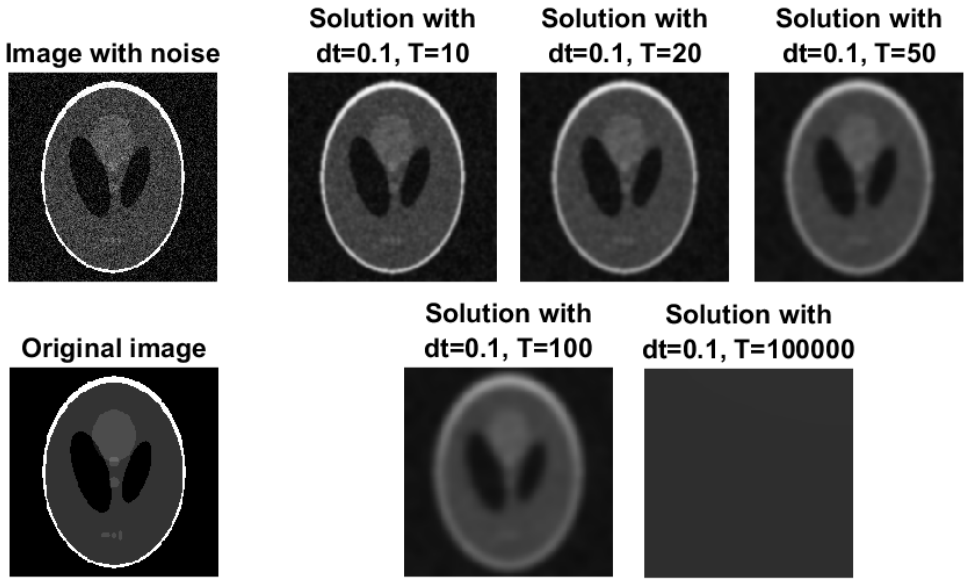

Figure 7.3: Results for $\Delta t = 0.1$ and various number of time steps T for the Shepp-Logan phantom

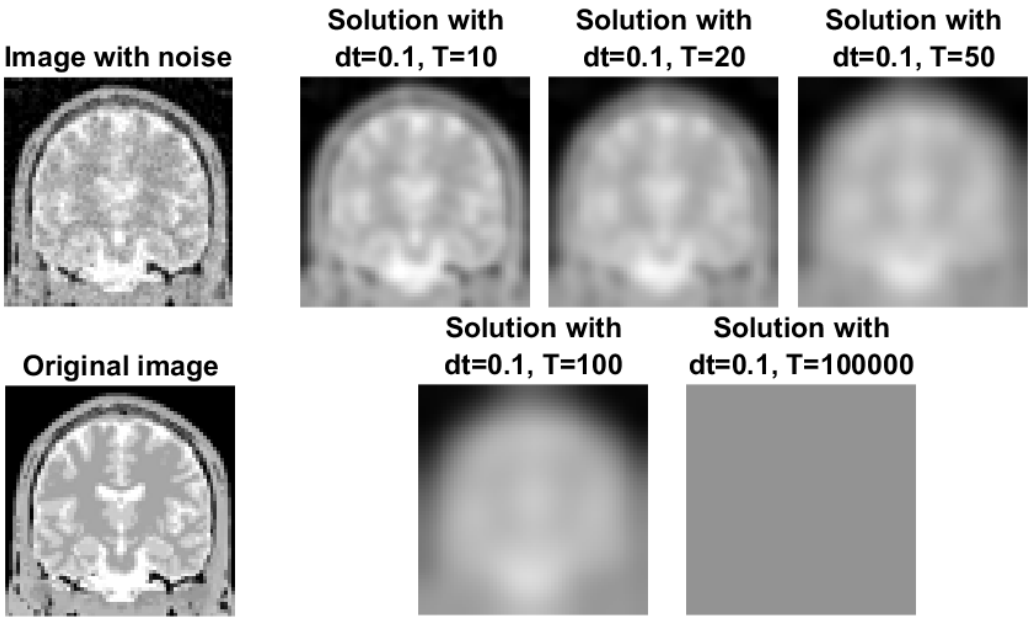

Figure 7.4: Results for $\Delta t = 0.1$ and various number of time steps T for the Ella phantom

7.1.2. ANISOTROPIC DIFFUSION FILTERING

Also for the anisotropic diffusion methods we looked into stability. For both Perona-Malik methods (5.3 and 5.4) the stability region should be similar to the region for linear diffusion filtering, so $0 < \Delta t \leq 0.25$. For the ill-posed and well-posed method we expect the region to be smaller. We will test these statements in this subsection.

First we will consider the Perona-Malik methods. We only discuss one of these methods here, since the other one has similar results. We will look at the results for Perona-Malik 5.4 with $T = 400$, $K = 0.5$ and $\Delta t = 0.25$ and 0.26. The images for the Shepp-Logan test problem ($n = 200$) and the Ella test problem can be found in figure 7.5 and 7.6, respectively.

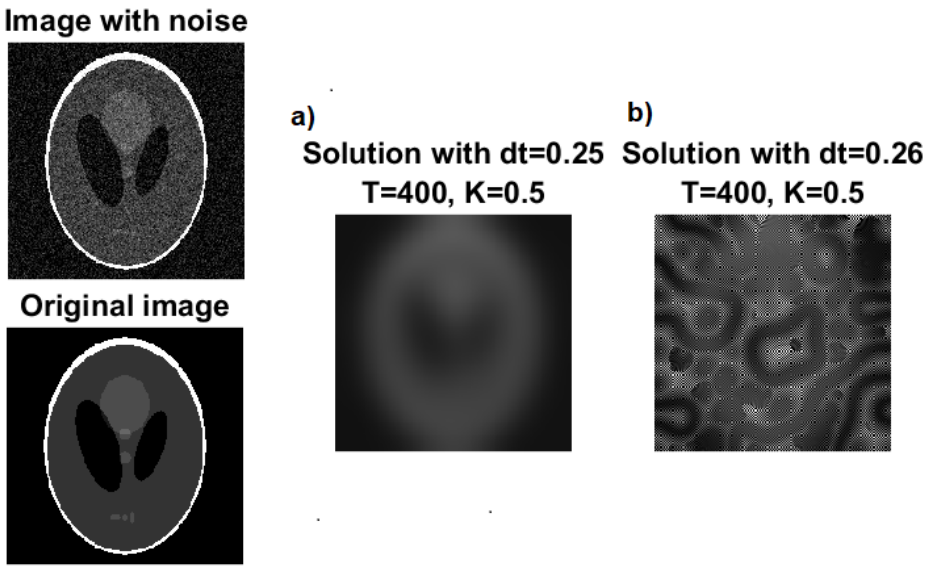

Figure 7.5: Stability for a) $\Delta t = 0.25$ and instability for b) $\Delta t = 0.26$ for the Shepp-Logan phantom

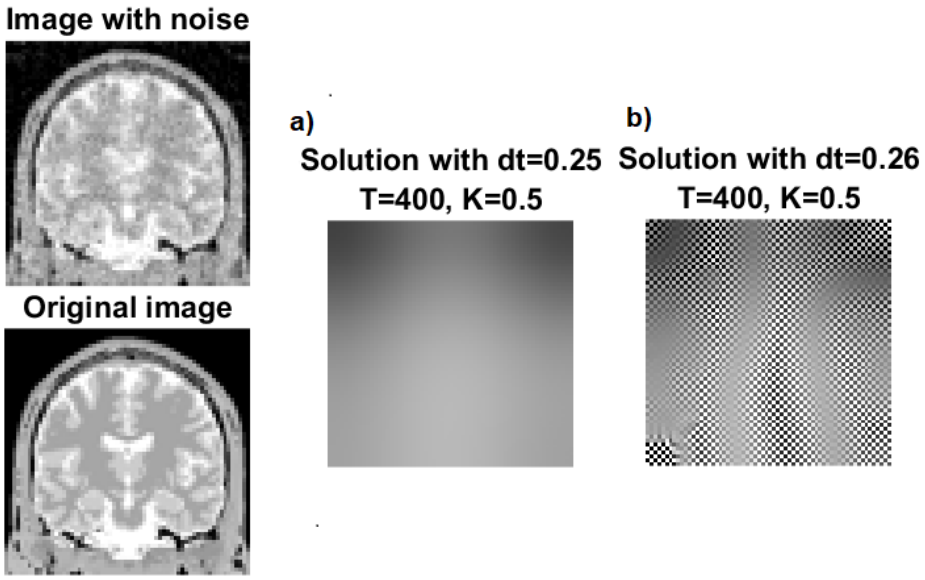

Figure 7.6: Stability for a) $\Delta t = 0.25$ and instability for b) $\Delta t = 0.26$ for the Ella phantom

We can see very clearly a chequered pattern for $\Delta t = 0.26$, meaning the solution is unstable. For $\Delta t = 0.25$ we do not see this pattern. The stability region seems to be correct for the Perona-Malik method 5.4. The results for the other Perona-Malik method were similar, so the conclusion also holds for this method. Also for a value of $K = 0.05$ the results showed this difference between $\Delta t = 0.25$ and $\Delta t = 0.26$.

For the ill-posed and well-posed method we stated that the stability interval is smaller than $0 < \Delta t < 0.25$. We tested both methods for the Shepp-Logan and Ella phantom for values of $T = 100$, $K = 0.05$ and $K = 0.5$. The results are shown in table 7.3.

	ill posed method	*well-posed method*
Shepp-Logan, $n = 200$, $K = 0.05$	$0 < \Delta t \leq 0.007$	$0 < \Delta t \leq 0.01$
Shepp-Logan, $n = 200$, $K = 0.5$	$0 < \Delta t \leq 0.14$	$0 < \Delta t \leq 0.12$
Ella, $K = 0.05$	$0 < \Delta t \leq 0.007$	$0 < \Delta t \leq 0.01$
Ella, $K = 0.5$	$0 < \Delta t \leq 0.15$	$0 < \Delta t \leq 0.12$

Table 7.3: Stability regions for the ill-posed and well-posed methods

From the table we see that the region is definitely smaller than $0 < \Delta t \leq 0.25$. It depends slightly on the test problem and the value of parameter K. These values probably change the value of $c(\cdot)$ in such a way that it influences the stability region.

The next property is the constant total pixel value. We have tested this problem for all methods, but only the results for the Perona-Malik method 5.4 are given here. The other methods all gave similar results. In table 7.4 the values are shown for TPV and APV and the corresponding images are shown in figure 7.7.
In the table we can see that the total pixel value changes slightly. Overall it stays quit constant. We can also see that the image converges to a constant value, which was given by the average pixel value for $T = 100000$. Therefore we can conclude that the statements made earlier are not exactly correct: the total pixel value stays almost constant, but the solution converges to the average pixel value.

T	0	10	20	50	100	100000
TPV $(\cdot 10^3)$	7.2414	7.2416	7.2413	7.2411	7.2412	7.2866
$APV(T = 100000)$	0.1822					

Table 7.4: Total pixel value for Shepp-Logan, $n = 200$

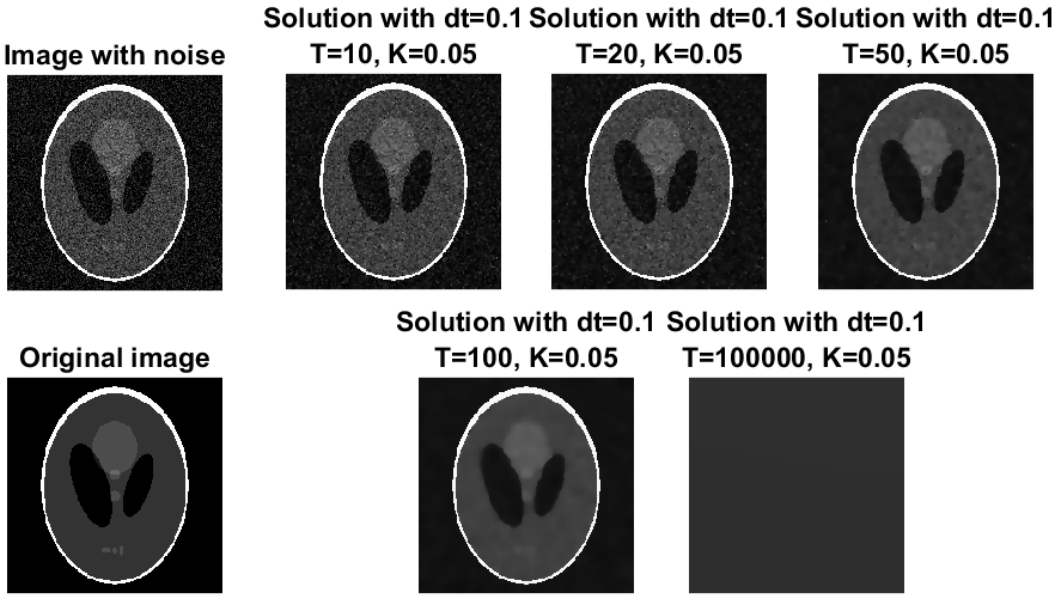

Figure 7.7: Results for $\Delta t = 0.1$ and various number of time steps T for the Shepp-Logan phantom

7.2. SHEPP-LOGAN: GAUSSIAN NOISE

7.2.1. FTCS IMPLEMENTATION

The parameters used for the Gaussian test problem using the FTCS implementation are $n = 200$ and $\Delta t = 0.01, 0.05, 0.1, 0.2$. The number of time steps is $T = 5, 10, 20, 50, 100, 200$. K is chosen as $K = 0.05, 0.5$ for all methods except the heat equation, which does not have a parameter K. The value N in the ill-posed equation 5.13 is set as $N = 3/2$, this will be the case for all numerical experiments.

The parameters of the best result of each method, as well as the standard deviation δ are given in table 7.5. Recall that the initial value is $\delta = 0.06$. The table is ordered by difference, starting with the smallest difference. Figure 7.8 to 7.14 give the best results for all methods.

method	Δt	T	K	*standard deviation* δ
Perona-Malik 5.4	0.05	100	0.05	0.0592
Perona-Malik 5.3	0.2	200	0.05	0.0592
Perona-Malik with Gaussian kernel 5.4	0.05	100	0.05	0.0594
Perona-Malik with Gaussian kernel 5.3	0.2	200	0.05	0.0594
Well-posed method 5.12	0.01	5	0.05	0.0595
Ill-posed method 5.13	0.01	5	0.05	0.0595
Heat equation	0.01	20	-	0.0597

Table 7.5: Parameters of each method's best result ordered by difference

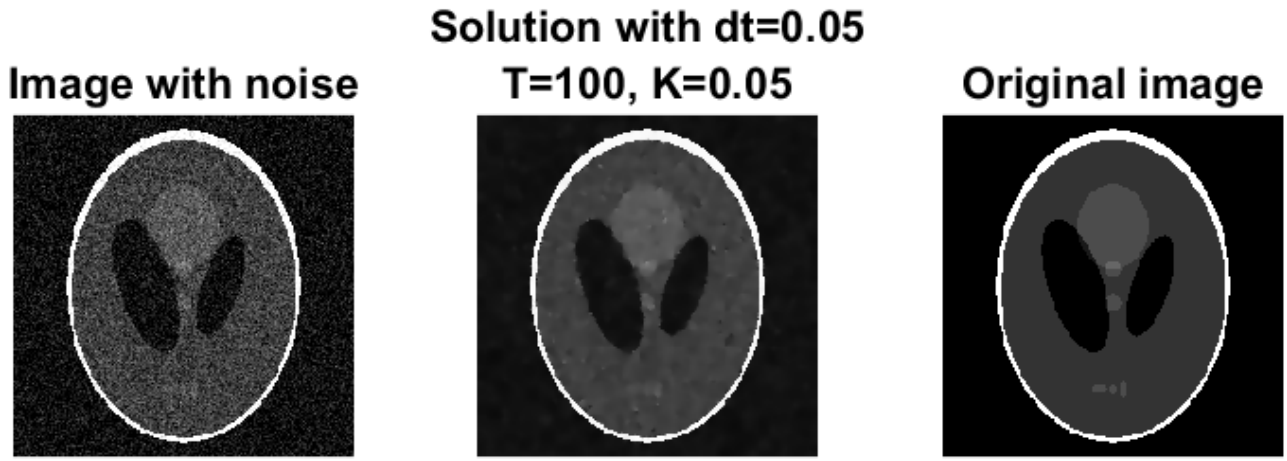

Figure 7.8: Gaussian noise: best result of Perona-Malik 5.4, $\Delta t = 0.05$, $T = 100$, $K = 0.05$

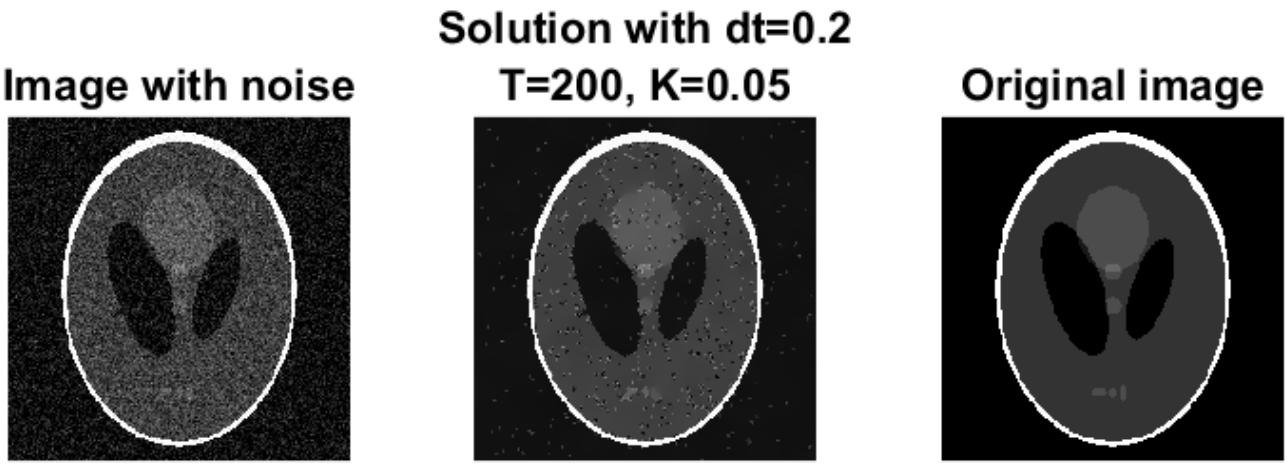

Figure 7.9: Gaussian noise: best result of Perona-Malik 5.3, $\Delta t = 0.2$, $T = 200$, $K = 0.05$

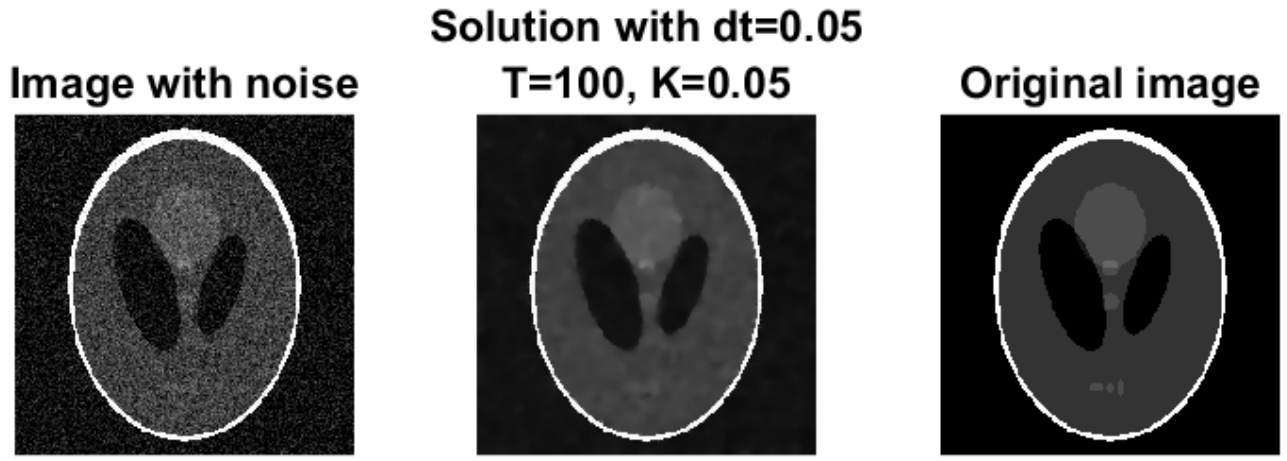

Figure 7.10: Gaussian noise: best result of Perona-Malik with Gaussian kernel 5.4, $\Delta t = 0.05$, $T = 100$, $K = 0.05$

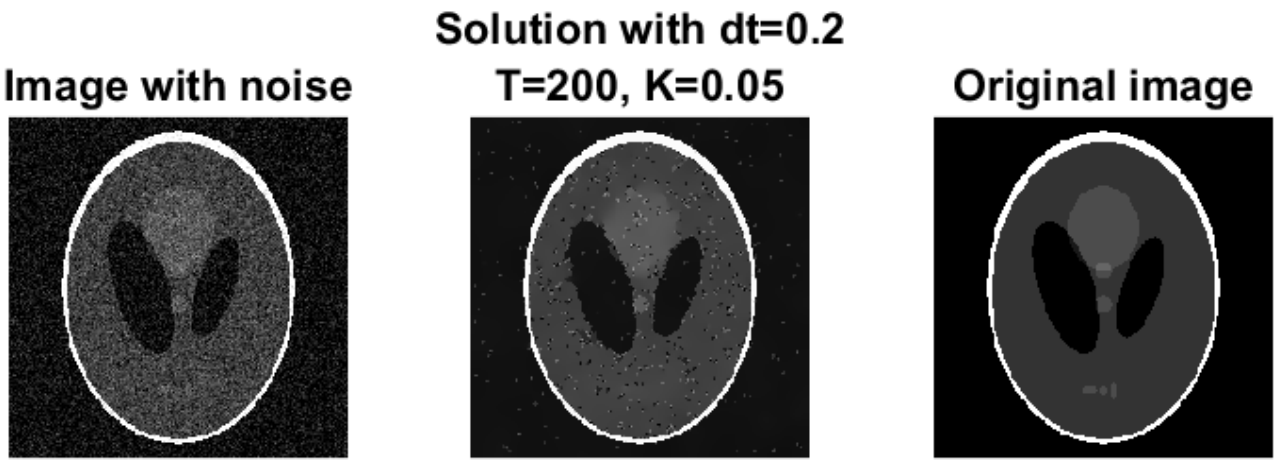

Figure 7.11: Gaussian noise: best result of Perona-Malik with Gaussian kernel 5.3, $\Delta t = 0.2$, $T = 200$, $K = 0.05$

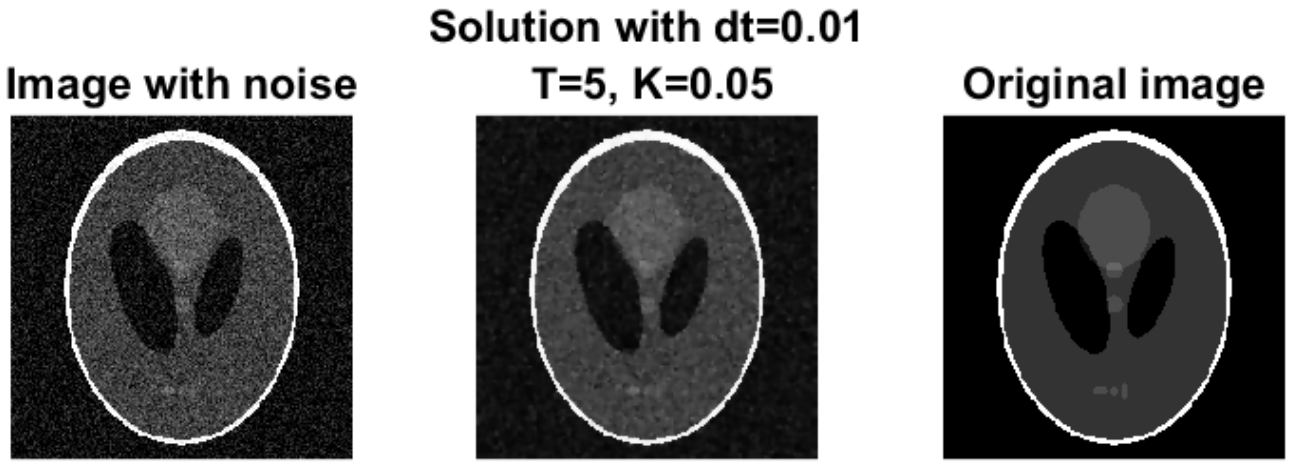

Figure 7.12: Gaussian noise: best result of the well-posed method 5.12, $\Delta t = 0.01$, $T = 5$, $K = 0.05$

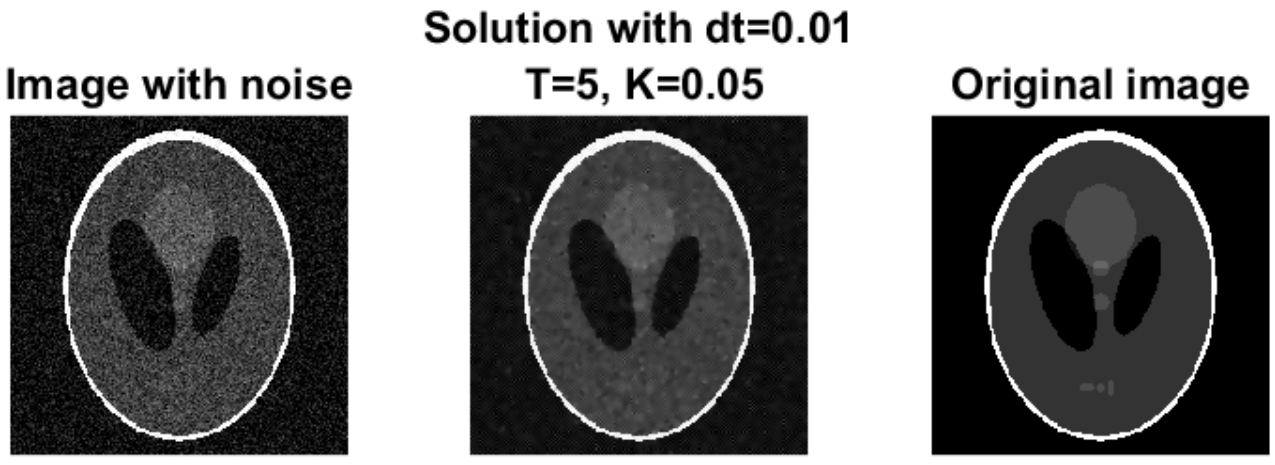

Figure 7.13: Gaussian noise: best result of the ill-posed method 5.13, $\Delta t = 0.01$, $T = 5$, $K = 0.05$

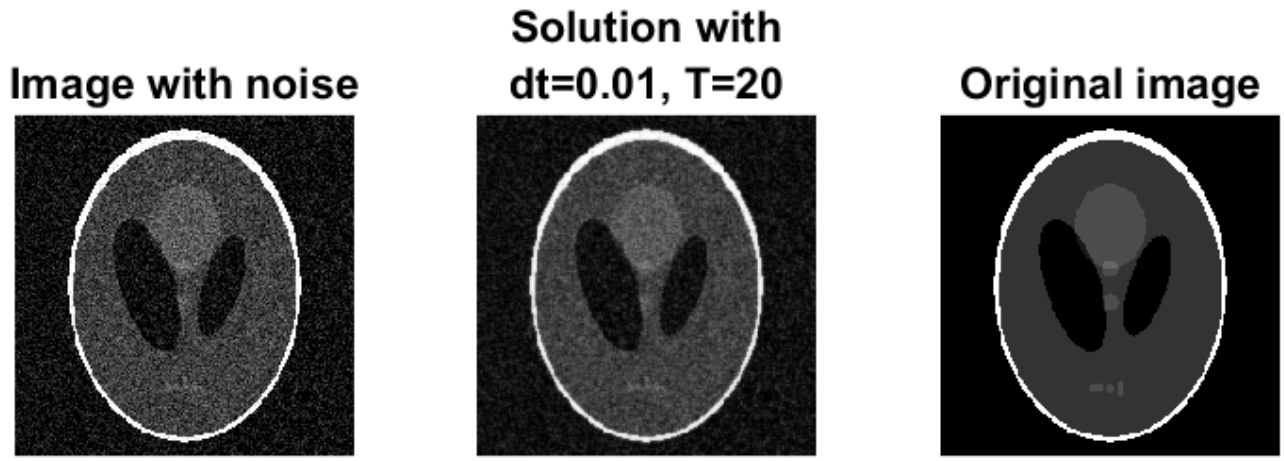

Figure 7.14: Gaussian noise: best result of linear diffusion filtering, $\Delta t = 0.01$, $T = 20$

Based on the values of δ the Perona-Malik methods have the best results compared to the other methods. By adding a Gaussian kernel to these Perona-Malik methods we obtain similar results. This is as expected, since the Perona-Malik methods were developed to have better results than the already existing other methods. They were chosen to satisfy the threshold property, which would lead to edge preservation as well as intraregional blurring. Nonetheless, when looking at the result of the Perona-Malik method (5.3) with a Gaussian kernel in figure 7.11, we see that the outer edges are quite sharp, but the inner gray circles are blurry. By choosing the parameters slightly different this problem can be overcome. This can be seen in 7.15, here the parameters are $\Delta t = 0.2$, $T = 100$ and $K = 0.05$. However, with other parameter values the standard deviation δ becomes larger due to increased intraregional inhomogeneity.
Another disadvantage of the Perona-Malik method 5.3 is the fact that the filtered image has noisy speckles. We can see this in figure 7.9 and 7.11. Although these speckles have a negative effect on the visual quality of the image, they do not seem to influence δ, since otherwise this Perona-Malik method would be lower in the table.
According to table 7.5 linear diffusion filtering has the worst result, which is also not unexpected. From the literature we know that the heat equation leads to homogeneous blurring, which means that edges fade for a large number of time steps. However, the result shown in figure 7.14 has no blurry edges, which means that for the right parameters the results can be quite good.

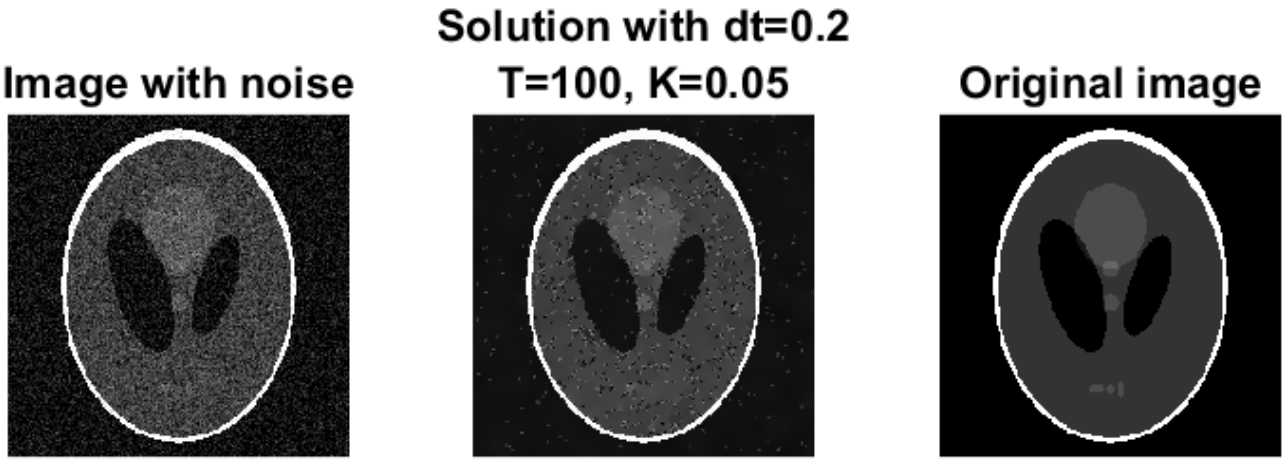

Figure 7.15: Gaussian noise: another result of Perona-Malik with Gaussian kernel 5.3, $\Delta t = 0.2$, $T = 100$, $K = 0.05$

It should also be noted that the well-posed and the ill-posed method showed instability for some of the parameters. If $K = 0.05$, the solution diverged for all tested values of Δt. If $K = 0.5$ the divergence started at $\Delta t = 0.2$. This instability may be overcome with an adaptive time step, which is discussed in the next chapter on parameter choice. The reason for this instability can be caused by two factors. The first was mentioned earlier in chapter 4 and is a too large time step Δt. The exact stability region for the ill-posed and well-posed method is unknown, but it is definitely smaller than 0.25. The other reason may be the approximation of the two functions when $|\nabla u|$ approaches zero. This approximation is given by 5.17 and 5.18. Choosing the value K too low can also cause problems for small $|\nabla u|$. So probably either a value of Δt too large or a value of K too small causes the instability.

Overall we can conclude that the parameter choice is of significant importance. Choosing parameters correctly for an inferior method like the heat equation may lead to great results, while choosing incorrect parameters for Perona-Malik may give an unwanted outcome. We can also assume that overall the Perona-Malik methods have the best results, while the heat equation the worst.

7.2.2. AOS IMPLEMENTATION

The parameters for the AOS implementation for the Gaussian test problem are different, since this implementation method is stable for all $\Delta t > 0$. However, when choosing Δt too high, splitting may become visible, leading to a less accurate outcome. In [10] a maximum value of $\Delta t = 5$ is used, which is therefore also our maximum Δt. The parameters are $n = 200$ and $\Delta t = 0.1, 0.5, 1, 2, 5$. The number of time steps is $T = 1, 2, 3, 5, 10, 50$ and again $K = 0.05, 0.5$.

Table 7.6 shows the best results for each method.

method	Δt	T	K	*standard deviation* δ
Perona-Malik with Gaussian kernel 5.4	2	2	0.05	0.0590
Perona-Malik 5.3	5	10	0.05	0.0590
Perona-Malik 5.4	0.1	50	0.05	0.0592
Well-posed method 5.12	0.1	1	0.05	0.0593
Perona-Malik with Gaussian kernel 5.3	5	3	0.05	0.0596
Ill-posed method 5.13	0.1	2	0.05	0.0596

Table 7.6: Parameters of each method's best result ordered by difference

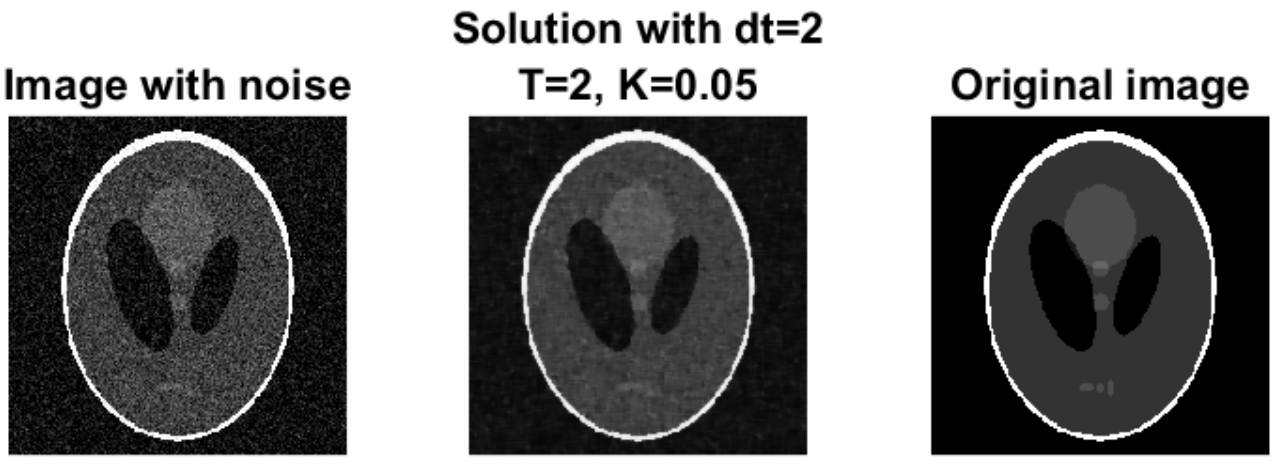

Figure 7.16: Gaussian noise: best result of Perona-Malik with Gaussian kernel 5.4, $\Delta t = 2$, $T = 2$, $K = 0.05$

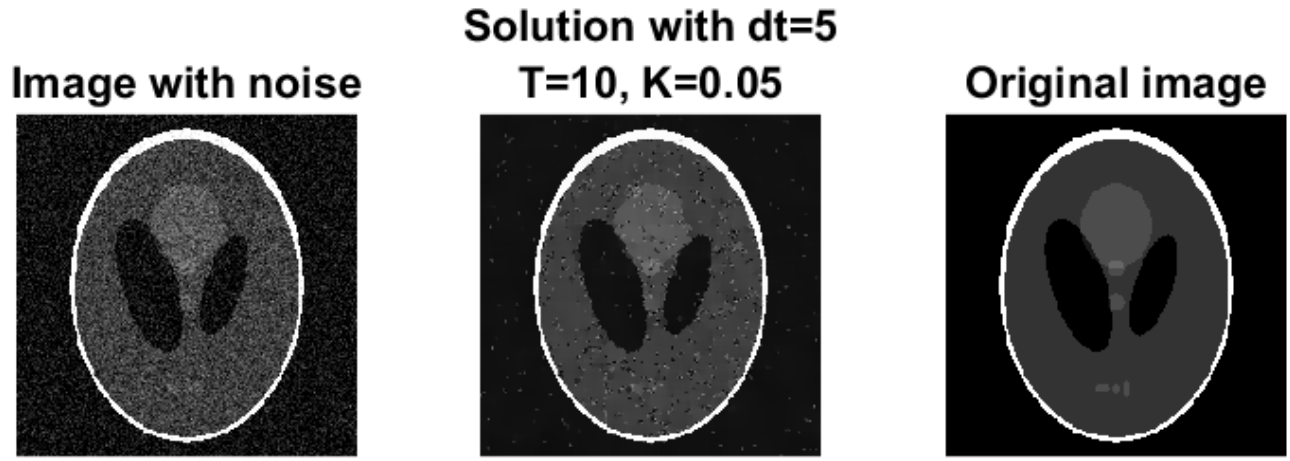

Figure 7.17: Gaussian noise: best result of Perona-Malik 5.3, $\Delta t = 5$, $T = 10$, $K = 0.05$

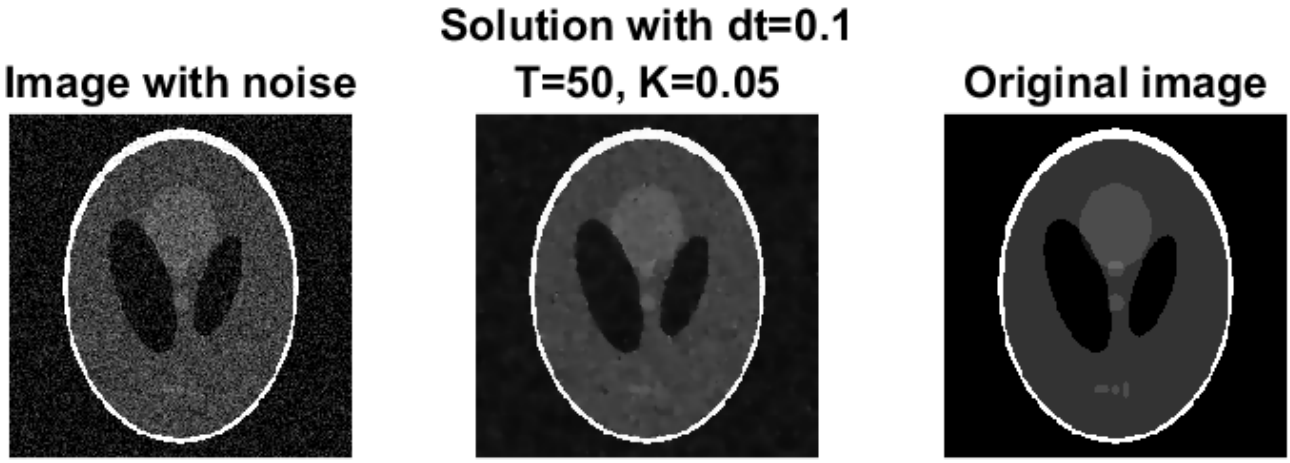

Figure 7.18: Gaussian noise: best result of Perona-Malik 5.4, $\Delta t = 0.1$, $T = 50$, $K = 0.05$

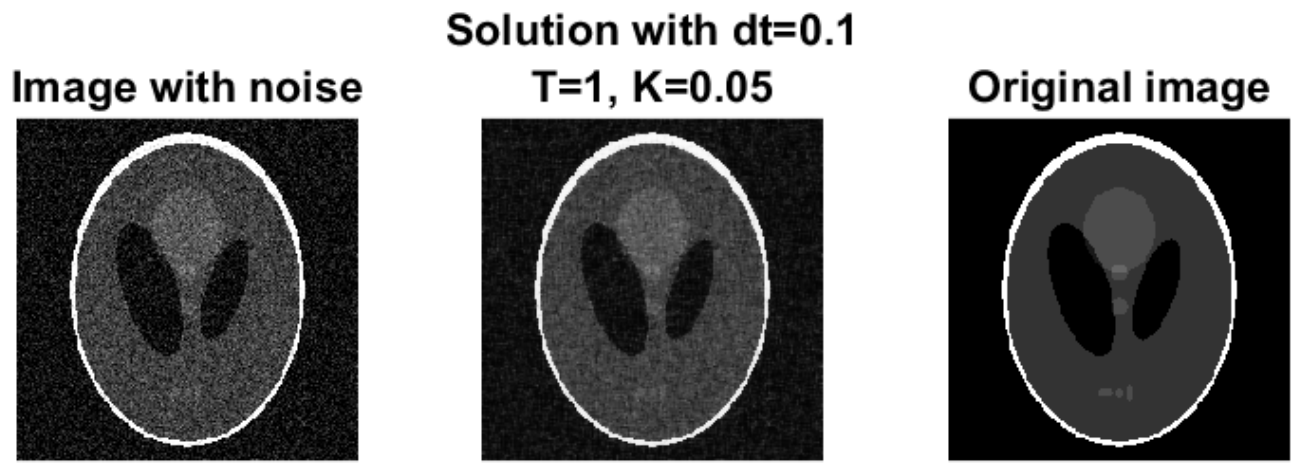

Figure 7.19: Gaussian noise: best result of the well-posed method 5.12, $\Delta t = 0.1$, $T = 1$, $K = 0.05$

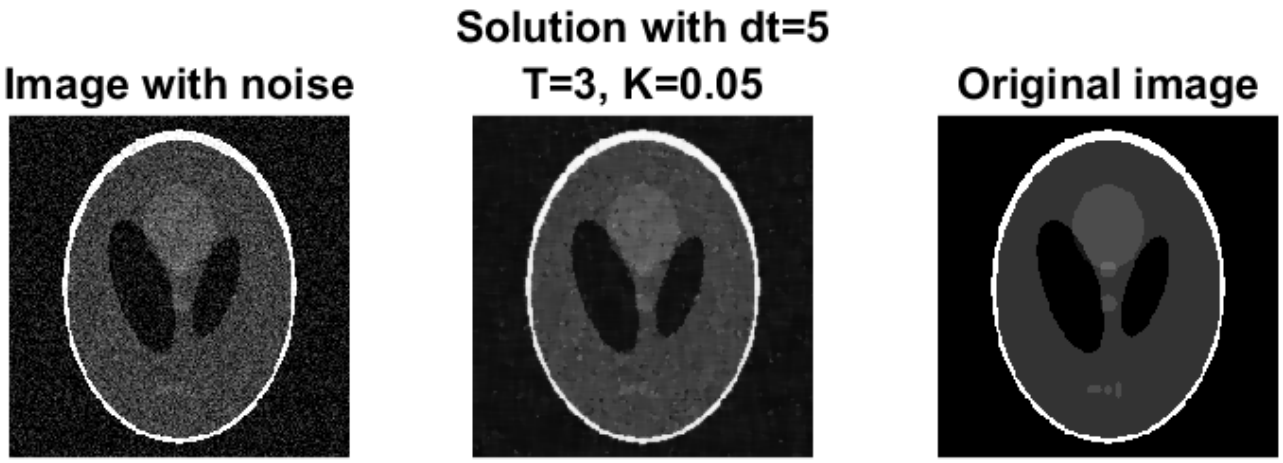

Figure 7.20: Gaussian noise: best result of Perona-Malik with Gaussian kernel 5.3, $\Delta t = 5$, $T = 3$, $K = 0.05$

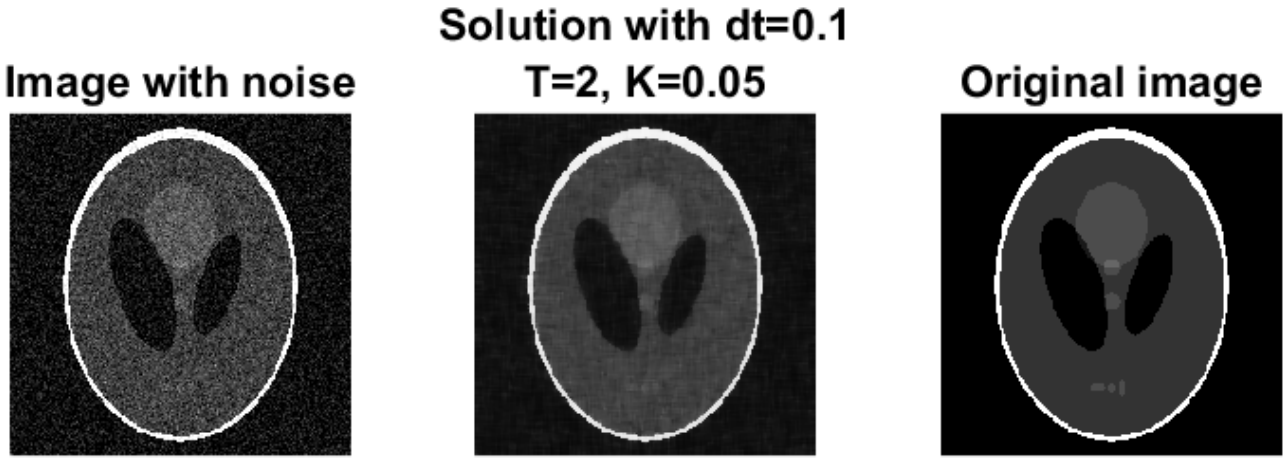

Figure 7.21: Gaussian noise: best result of the ill-posed method 5.13, $\Delta t = 0.1$, $T = 2$, $K = 0.05$

For the AOS implementation the Perona-Malik methods also seem to give the best results. However, when looking at figure 7.17 we see that the visual quality of this image is quite poor. This is due to the speckles caused by the Perona-Malik method 5.3, which we also noticed for the FTCS implementation. Again, these speckles do not seem to have an effect on the value of δ.

One of the advantages of the AOS method is the stability for all $\Delta t > 0$. In the table we see that some optimal results have been obtained with a high Δt. The best result, shown in figure 7.16, is obtained with only two time steps of size $\Delta t = 2$. This means this implementation can be more efficient than the FTCS implementation, while also generating a good visual outcome.

If we compare table 7.5 and 7.6 we observe that the standard deviations δ are fairly close. However, the two best results obtained using the AOS implementation do have a smaller δ than the best method generated with the FTCS implementation. It seems the AOS implementation leads to better results.
Also, the well-posed and ill-posed method showed no instability for the tested parameters. This is also an advantage compared to the FTCS implementation.

7.3. SHEPP-LOGAN: JOHNSON NOISE

This test problem is slighty different from the first one. First of all, the noise is more realistic, since it is a reconstructed image of the TUD/LUMC MRI scanner. The other difference is the size of the image. The image given by the simulation is only 32×32, so $n = 32$. The other parameters depend on the numerical scheme.

7.3.1. FTCS IMPLEMENTATION

For the FTCS method the parameters are $\Delta t = 0.01, 0.05, 0.1, 0.2$, $T = 5, 10, 20, 50, 100, 200$ and $K = 0.05, 0.5$. Again the results are given in a table, table 7.7, and the initial standard deviation is $\delta = 0.064$. The images are in figure 7.22 to 7.28.

method	Δt	T	K	*standard deviation* δ
Perona-Malik with Gaussian kernel 5.3	0.2	10	0.05	0.0433
Perona-Malik with Gaussian kernel 5.4	0.01	50	0.05	0.0454
Perona-Malik 5.3	0.01	10	0.5	0.0585
Perona-Malik 5.4	0.01	5	0.5	0.0592
Ill-posed method 5.13	0.01	5	0.5	0.0648
Heat equation	0.01	5	-	0.0659
Well-posed method 5.12	0.01	5	0.5	0.0678

Table 7.7: Parameters of each method's best result ordered by difference

The table for this type of noise has a somewhat different order compared to table 7.5. Combining the Perona-Malik methods with a Gaussian kernel now does lead to a better result. Also, the heat equation is not the worst method here, the worst method is now the well-posed method 5.12. However, the standard deviation δ of the ill-posed, well-posed and linear diffusion method are quite close together and all lead to results worse than the Perona-Malik methods. So the overall conclusion is still the same: Perona-Malik gives the best results. The visual perception of the images agrees with this statement. The figures 7.22 and 7.23 show very good results. Not using the Gaussian kernel has a large influence on the outcome, as can be seen in both the table and the corresponding figures. The resulting images still have noise and differ only slighty from the test problem.
Also, the result of the well-posed problem, shown in figure 7.28, seems to have a higher visual quality than Perona-Malik, but has the highest δ. Choosing other parameters for the Perona-Malik problem had results similar to this well-posed outcome, but as mentioned before: δ is higher. This means that the standard deviation δ is inconsistent with visual observation.
Another observation is that the standard deviation δ is higher than the initial standard deviation of 0.064 for the three worst methods. This means that the amount of noise is higher than in the noisy image, which is obviously not what we want.

The instability for the ill-posed and well-posed method we have seen for the first test problem are also present here. The instability starts at $\Delta t = 0.05$ for $K = 0.05$. There is no instability for $K = 0.5$. Unfortunately, both Perona-Malik methods also show instability. Perona-Malik 5.3 only for $\Delta t = 0.01$ and $K = 0.05$, but Perona-Malik 5.4 is unstable for all Δt with $K = 0.05$. Adding a Gaussian kernel to the methods eliminates the instability.

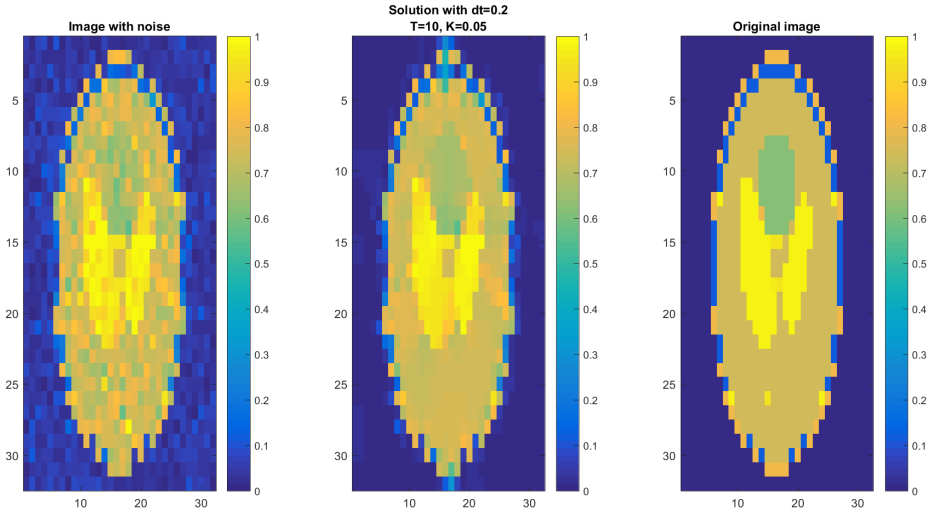

Figure 7.22: Johnson noise: best result of Perona-Malik with Gaussian kernel 5.3, $\Delta t = 0.2$, $T = 10$, $K = 0.05$

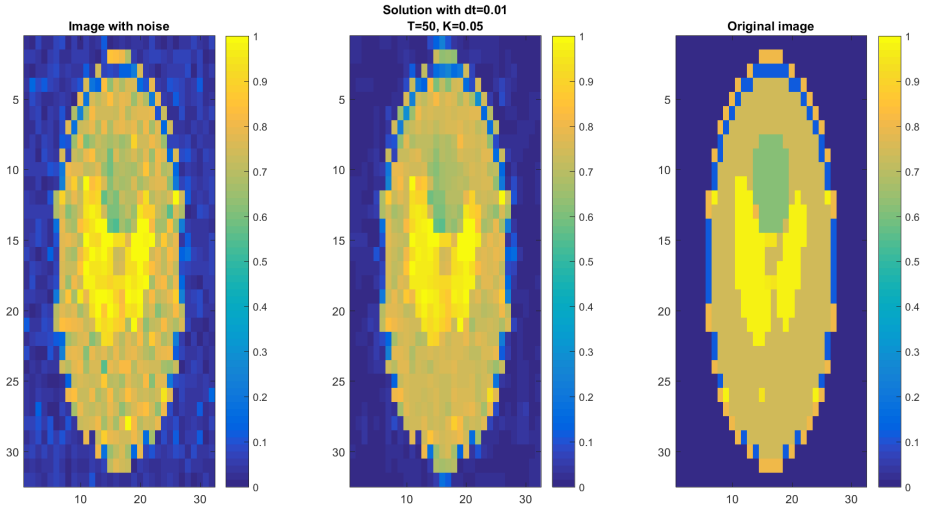

Figure 7.23: Johnson noise: best result of Perona-Malik with Gaussian kernel 5.4, $\Delta t = 0.01$, $T = 50$, $K = 0.05$

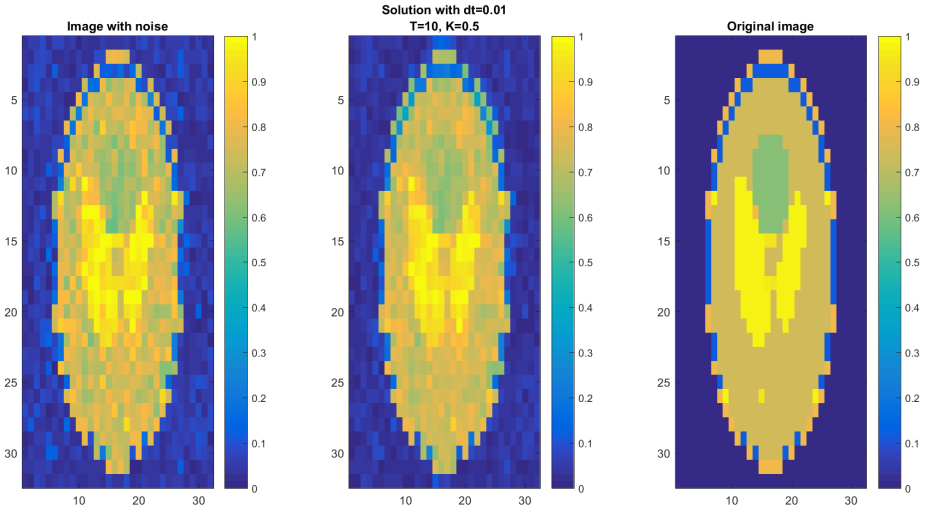

Figure 7.24: Johnson noise: best result of Perona-Malik 5.3, $\Delta t = 0.01$, $T = 10$, $K = 0.5$

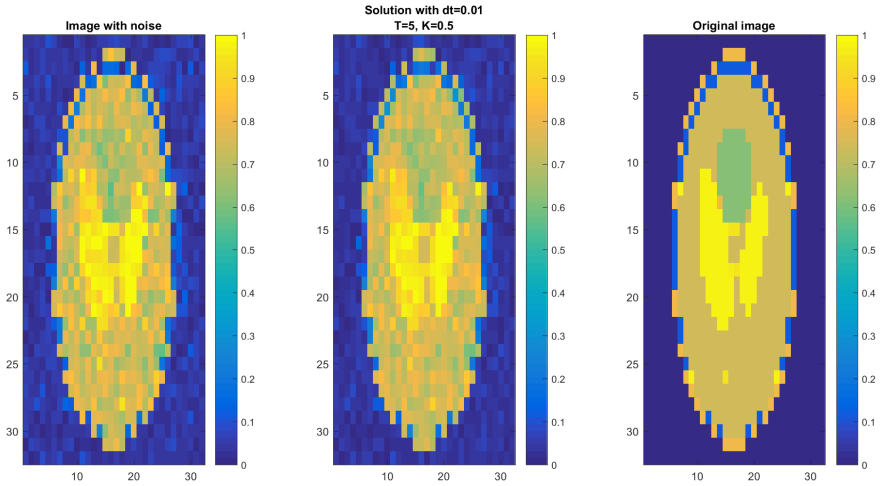

Figure 7.25: Johnson noise: best result of Perona-Malik 5.4, $\Delta t = 0.01$, $T = 5$, $K = 0.5$

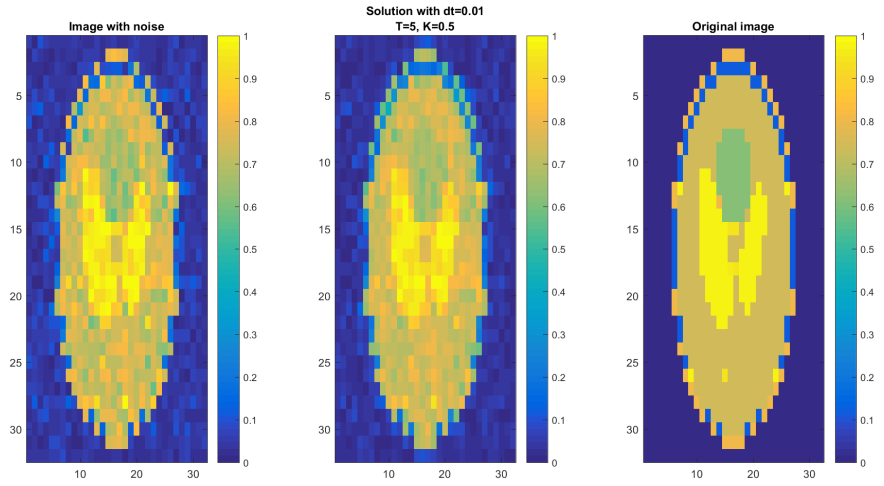

Figure 7.26: Johnson noise: best result of the ill-posed method 5.13, $\Delta t = 0.01$, $T = 5$, $K = 0.5$

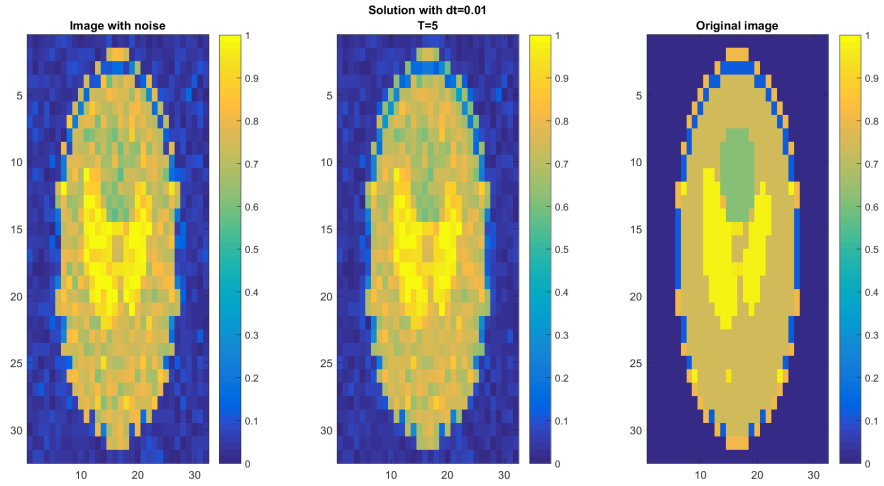

Figure 7.27: Johnson noise: best result of linear diffusion filtering, $\Delta t = 0.01$, $T = 5$

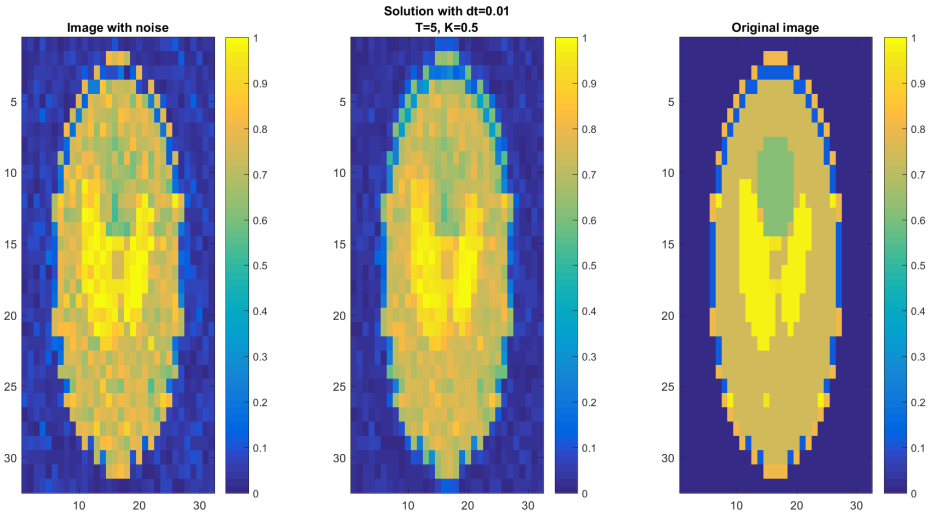

Figure 7.28: Johnson noise: best result of the well-posed method 5.12, $\Delta t = 0.01$, $T = 5$, $K = 0.5$

7.3.2. AOS IMPLEMENTATION

For the AOS method the parameters are still $\Delta t = 0.1, 0.5, 1, 2, 5$, $T = 1, 2, 3, 5, 10, 20, 50$, and $K = 0.05, 0.5$. The results are given in table 7.8 ordered by standard deviation δ. The images are in figure 7.29 to 7.34.

method	Δt	T	K	*standard deviation* δ
Perona-Malik 5.3	0.1	5	0.05	0.0536
Perona-Malik 5.4	0.1	3	0.05	0.0570
Perona-Malik with Gaussian kernel 5.3	0.5	5	0.05	0.0848
Perona-Malik with Gaussian kernel 5.4	0.1	2	0.05	0.0883
Ill-posed method 5.13	0.1	2	0.5	0.1139
Well-posed method 5.12	0.1	2	0.5	0.1139

Table 7.8: Parameters of each method's best result ordered by difference

There are a few differences with the results from the FTCS implementation. First of all, the order of the table is slightly different. For the FTCS implementation the results were better for the Perona-Malik methods in combination with the Gaussian kernel. For the AOS implementation Perona-Malik showed better outcomes without the Gaussian kernel.
Another difference is the standard deviation δ. The standard deviation for the Perona-Malik methods are around the average δ we saw for the FTCS method. This means that it is still better to use Perona-Malik with a Gaussian kernel in combination with FTCS implementation. The other four values in the table of AOS implementation are much higher than even the worst value in the FTCS implementation table. This means that for these methods more noise is present in the filtered image than in the noisy image. This is confirmed by the images: only image 7.29 and 7.30 have a reasonable visual quality, the other four are all much blurrier than any of the figures for the FTCS implementation.
Even though the AOS method makes it possible to use higher Δt, the optimal parameters did not include these large values for Δt. Only the Perona-Malik with a Gaussian kernel 5.3 used a value of Δt which is higher than the stability range for the FTCS method, but this value was still only $\Delta t = 0.5$.
A positive outcome for the AOS method is the fact that no instability has been observed for the ill-posed and well-posed methods. Also Perona-Malik 5.3 did not show instability.

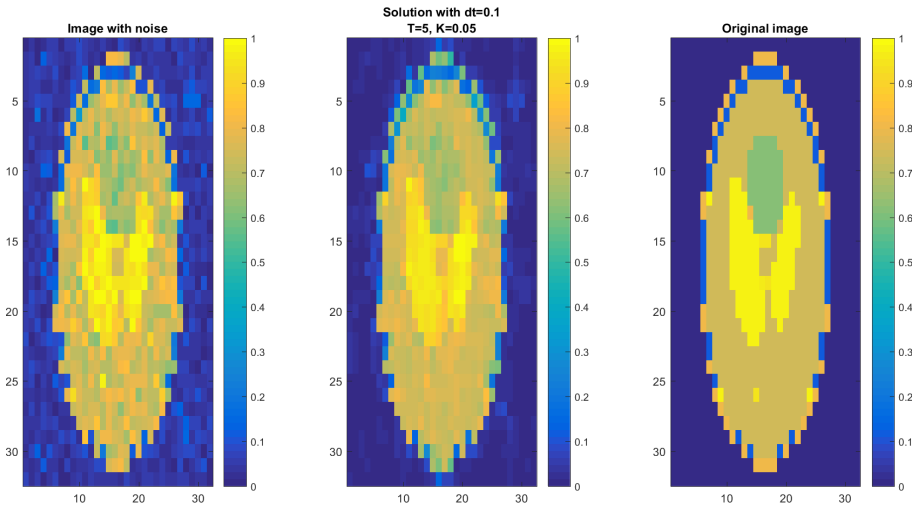

Figure 7.29: Johnson noise: best result of Perona-Malik 5.3, $\Delta t = 0.1$, $T = 5$, $K = 0.05$

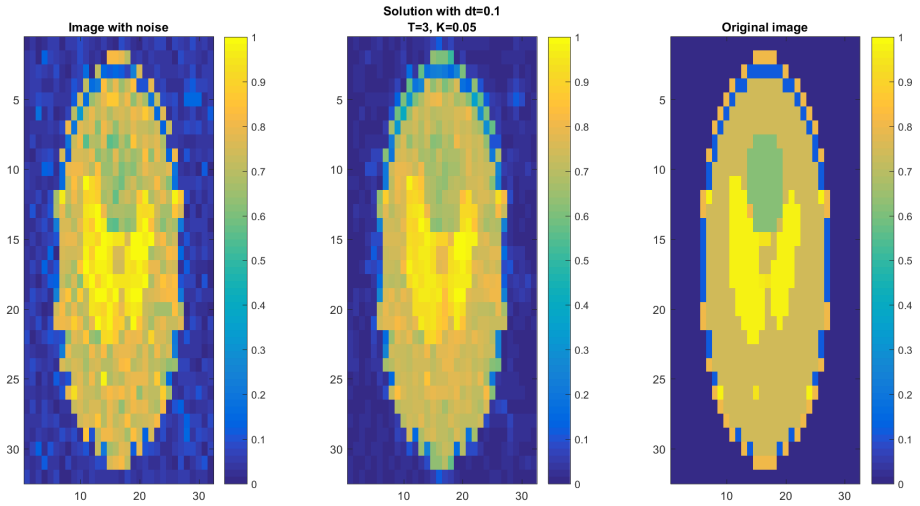

Figure 7.30: Johnson noise: best result of Perona-Malik 5.4, $\Delta t = 0.1$, $T = 3$, $K = 0.05$

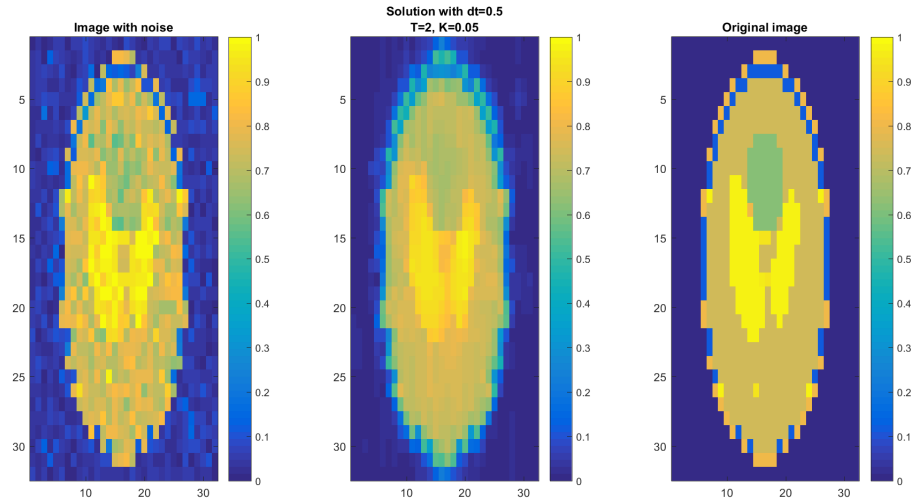

Figure 7.31: Johnson noise: best result of Perona-Malik with Gaussian kernel 5.3, $\Delta t = 0.5$, $T = 5$, $K = 0.05$

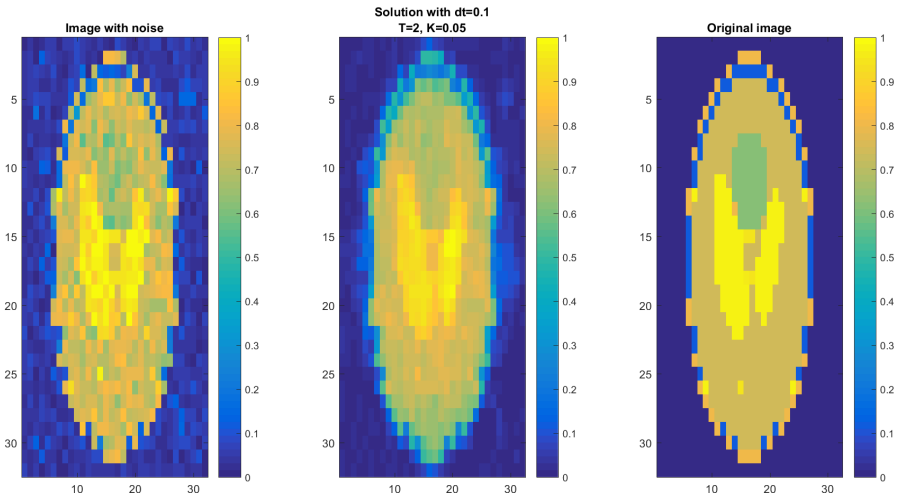

Figure 7.32: Johnson noise: best result of Perona-Malik with Gaussian kernel 5.4, $\Delta t = 0.1$, $T = 2$, $K = 0.05$

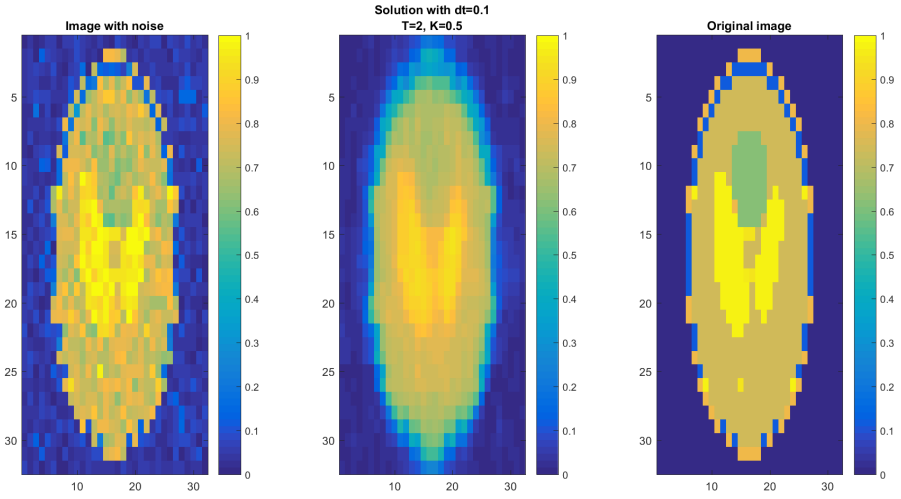

Figure 7.33: Johnson noise: best result of the ill-posed method 5.13, $\Delta t = 0.1$, $T = 2$, $K = 0.5$

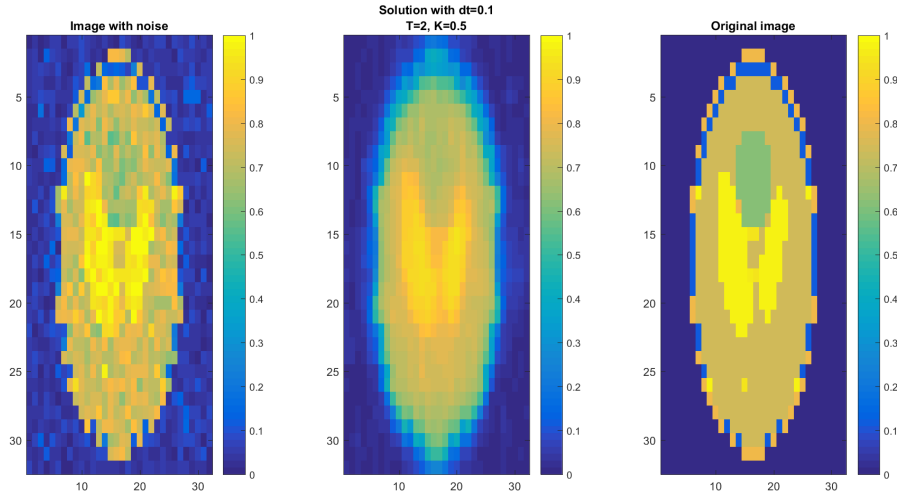

Figure 7.34: Johnson noise: best result of the well-posed method 5.12, $\Delta t = 0.1$, $T = 2$, $K = 0.5$

7.4. ELLA: JOHNSON NOISE

This test problem uses a different phantom: the Ella phantom. The noise is of the same type as the second test problem, namely Johnson noise, which is the expected noise of the TUD/LUMC MRI scanner. The Ella phantom is 64×64, so $n = 64$.

7.4.1. FTCS IMPLEMENTATION

The other parameters are similar to the parameters mentioned before. So $\Delta t = 0.01, 0.05, 0.1, 0.2$, $T = 5, 10, 20, 50, 100, 200$ and $K = 0.05, 0.5$. The results are given in table 7.9 with initial standard deviation of $\delta = 0.02$. The images are in figure 7.35 to 7.41.

method	Δt	T	K	*standard deviation* δ
Perona-Malik with Gaussian kernel 5.4	0.05	5	0.05	0.0189
Perona-Malik 5.4	0.01	50	0.05	0.0190
Perona-Malik with Gaussian kernel 5.3	0.01	50	0.05	0.0190
Perona-Malik 5.3	0.01	10	0.5	0.0191
Ill-posed method 5.13	0.01	5	0.5	0.0195
Heat equation	0.01	5	-	0.0195
Well-posed method 5.12	0.01	5	0.5	0.0199

Table 7.9: Parameters of each method's best result ordered by difference

The order of the table is again what we expect: the Perona-Malik methods lead to the best results. This is confirmed by the figures: the first three figures especially seem to show an improvement compared to the test problem.
Again the ill-posed and well-posed method were unstable. The ill-posed method was unstable for all Δt when $K = 0.05$ and for $\Delta t = 0.2$ when $K = 0.5$. The well-posed method did slightly better and was unstable for $\Delta t \geq 0.05$ when $K = 0.05$ and $\Delta t = 2$ when $K = 0.5$.

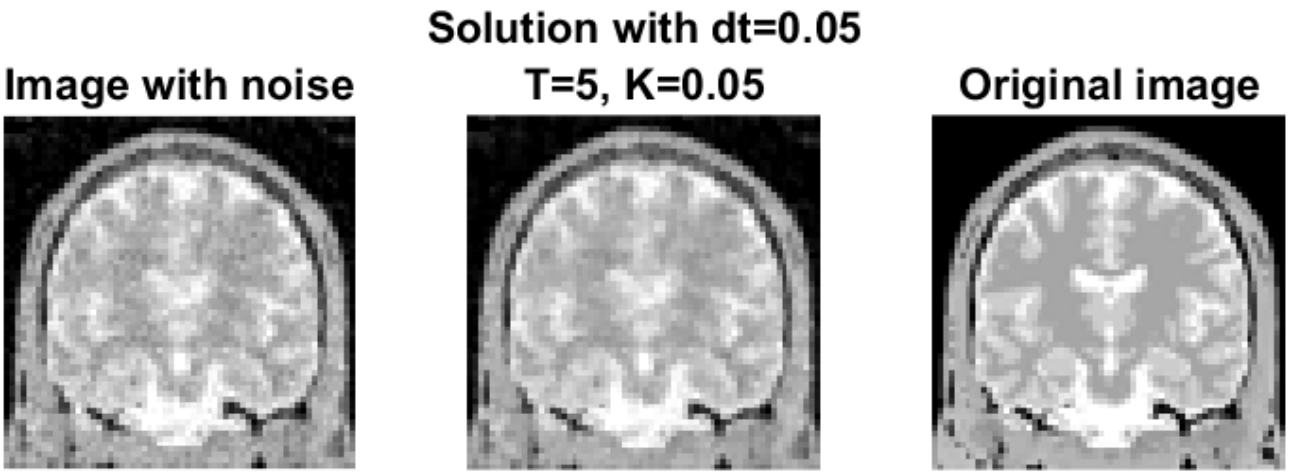

Figure 7.35: Johnson noise: best result of Perona-Malik with Gaussian kernel 5.4, $\Delta t = 0.05$, $T = 5$, $K = 0.05$

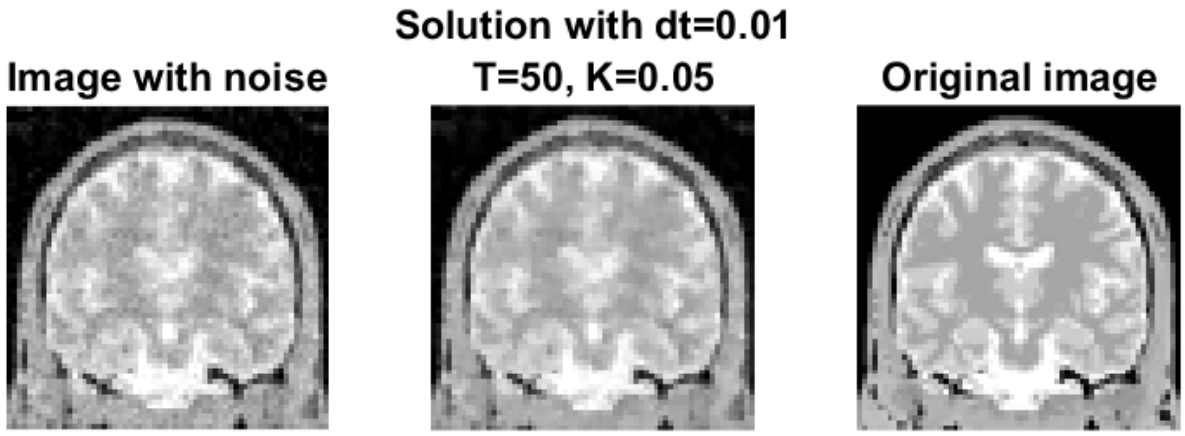

Figure 7.36: Johnson noise: best result of Perona-Malik 5.4, $\Delta t = 0.01$, $T = 50$, $K = 0.05$

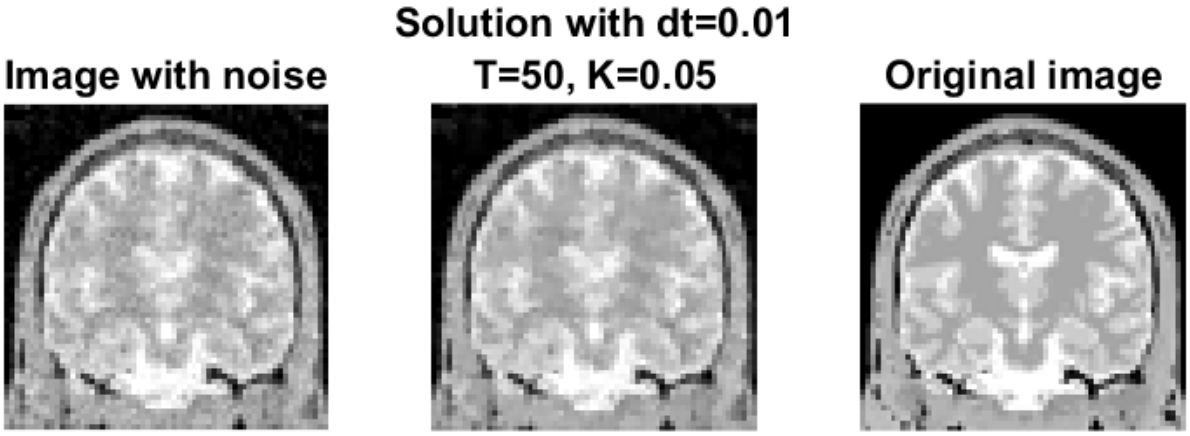

Figure 7.37: Johnson noise: best result of Perona-Malik with Gaussian kernel 5.3, $\Delta t = 0.01$, $T = 50$, $K = 0.05$

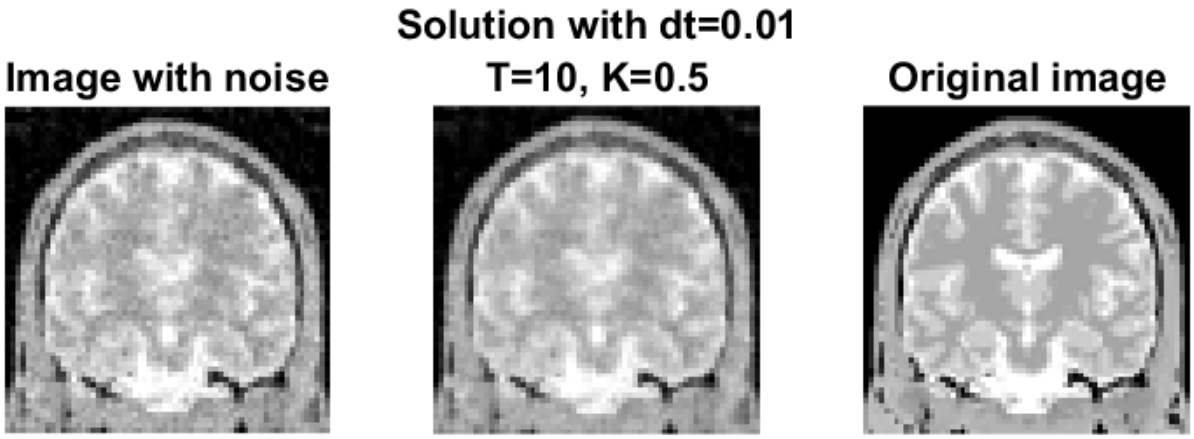

Figure 7.38: Johnson noise: best result of Perona-Malik 5.3, $\Delta t = 0.01$, $T = 10$, $K = 0.5$

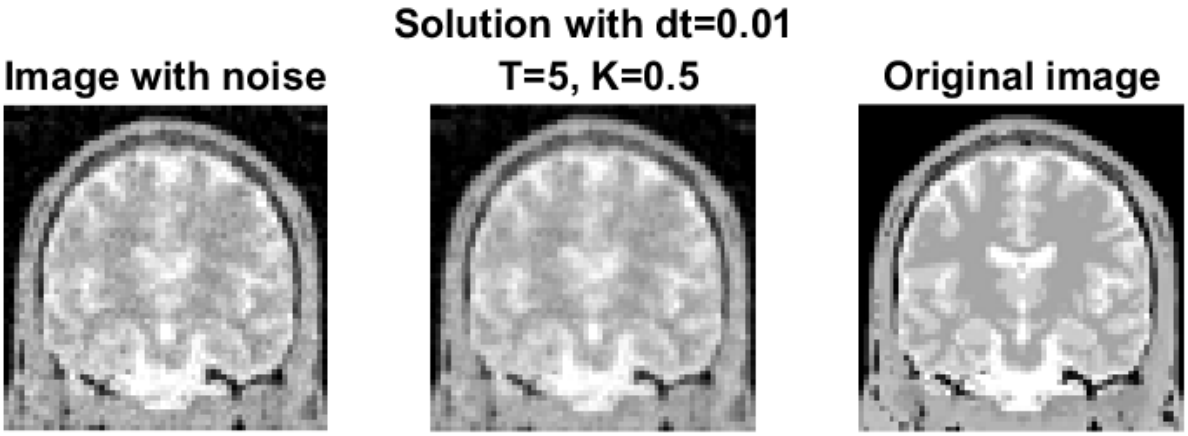

Figure 7.39: Johnson noise: best result of the ill-posed method 5.13, $\Delta t = 0.01$, $T = 5$, $K = 0.5$

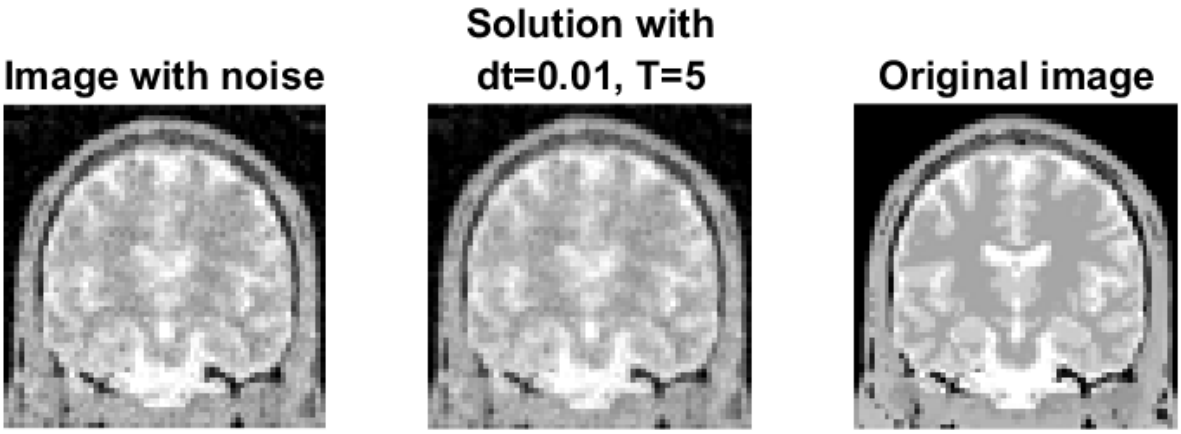

Figure 7.40: Johnson noise: best result of linear diffusion filtering, $\Delta t = 0.01$, $T = 5$

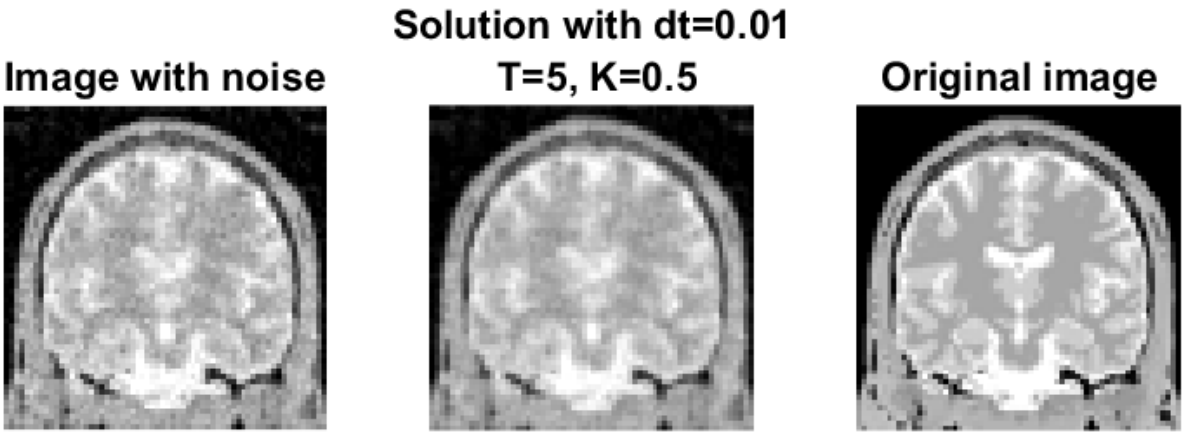

Figure 7.41: Johnson noise: best result of the well-posed method 5.12, $\Delta t = 0.01$, $T = 5$, $K = 0.5$

7.4.2. AOS IMPLEMENTATION

The parameters for the AOS methods are again $\Delta t = 0.1, 0.5, 1, 2, 5$, $T = 1, 2, 3, 5, 10, 20, 50$, and $K = 0.05, 0.5$. The results are given in table 7.10. The corresponding images are in figure 7.35 to 7.41.

method	Δt	T	K	*standard deviation* δ
Perona-Malik 5.4	0.5	2	0.05	0.0185
Perona-Malik 5.3	5	1	0.05	0.0186
Perona-Malik with Gaussian kernel 5.3	1	1	0.05	0.0192
Perona-Malik with Gaussian kernel 5.4	0.5	1	0.05	0.0193
Ill-posed method 5.13	0.1	1	0.5	0.0198
Well-posed method 5.12	0.1	1	0.5	0.0205

Table 7.10: Parameters of each method's best result ordered by difference

The Perona-Malik methods without the addition of a Gaussian kernel are again in the upper part of the table. This has been the case for all AOS implementated test problems.
And similar to the first test problem the standard deviation δ for these Perona-Malik results is lower than for the best result achieved with the FTCS implementation. The figures agree with this statement and show a significant improvement compared to the noise image.
In this case the best four test problems use a value of Δt which would have been unstable for the FTCS method. The best two even lead to better results than the best result of the FTCS method. This unconditional stability has proved to be an advantage here.
Also, no instability for all methods and parameters has been seen, which is also an improvement.

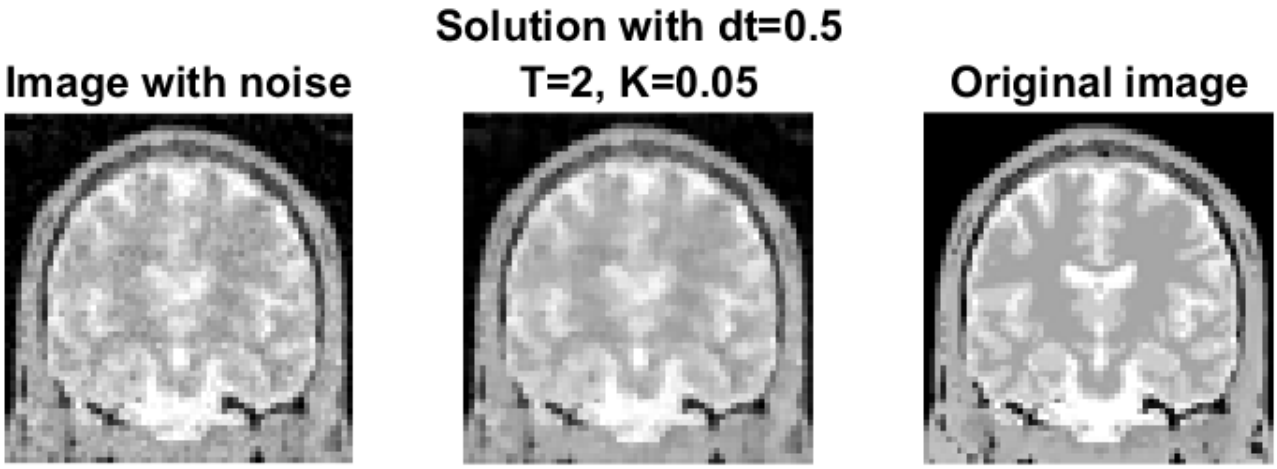

Figure 7.42: Johnson noise: best result of Perona-Malik 5.4, $\Delta t = 0.5$, $T = 2$, $K = 0.05$

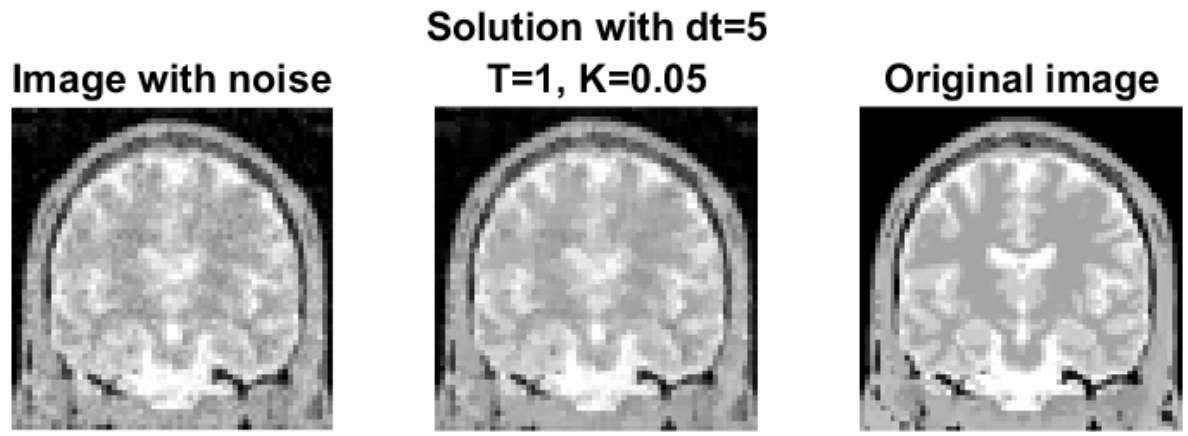

Figure 7.43: Johnson noise: best result of Perona-Malik 5.3, $\Delta t = 5$, $T = 1$, $K = 0.05$

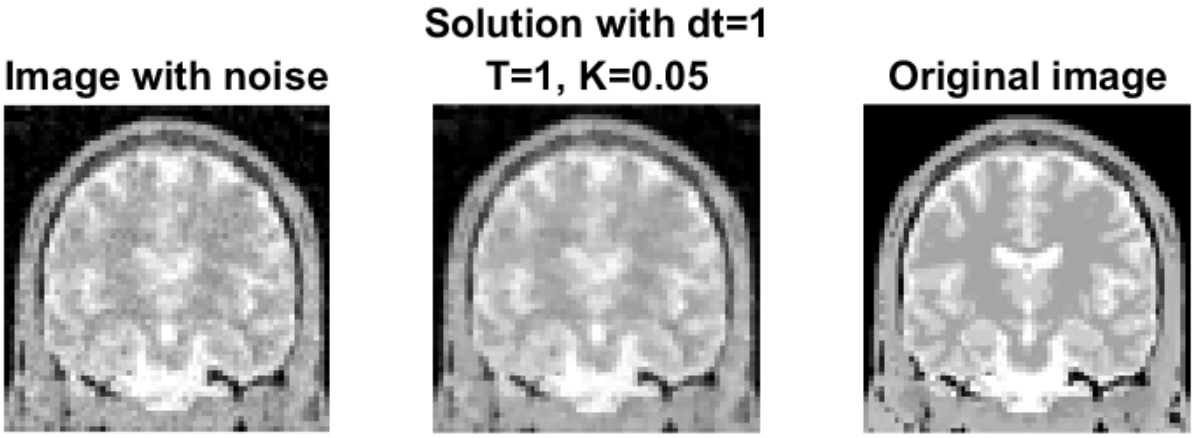

Figure 7.44: Johnson noise: best result of Perona-Malik with Gaussian kernel 5.3, $\Delta t = 1$, $T = 1$, $K = 0.05$

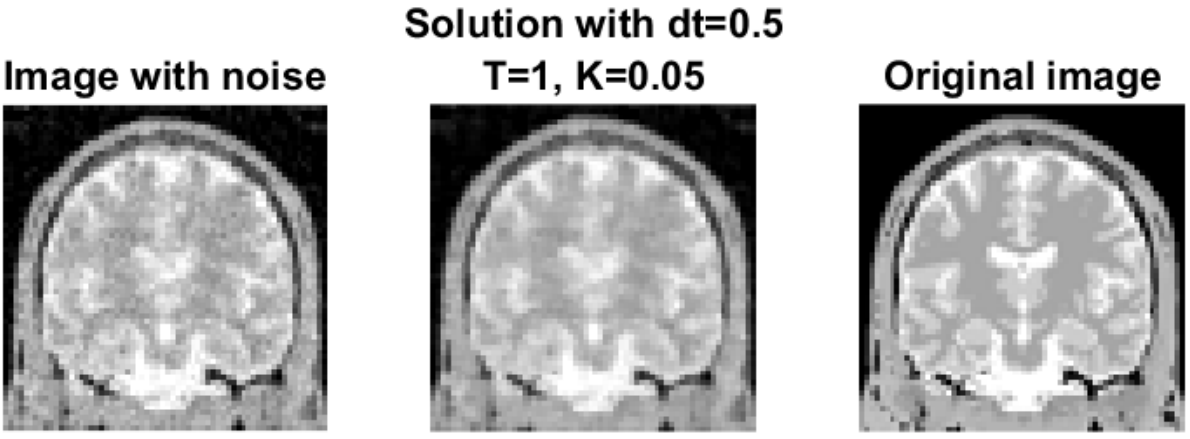

Figure 7.45: Johnson noise: best result of Perona-Malik with Gaussian kernel 5.4, $\Delta t = 0.5$, $T = 1$, $K = 0.05$

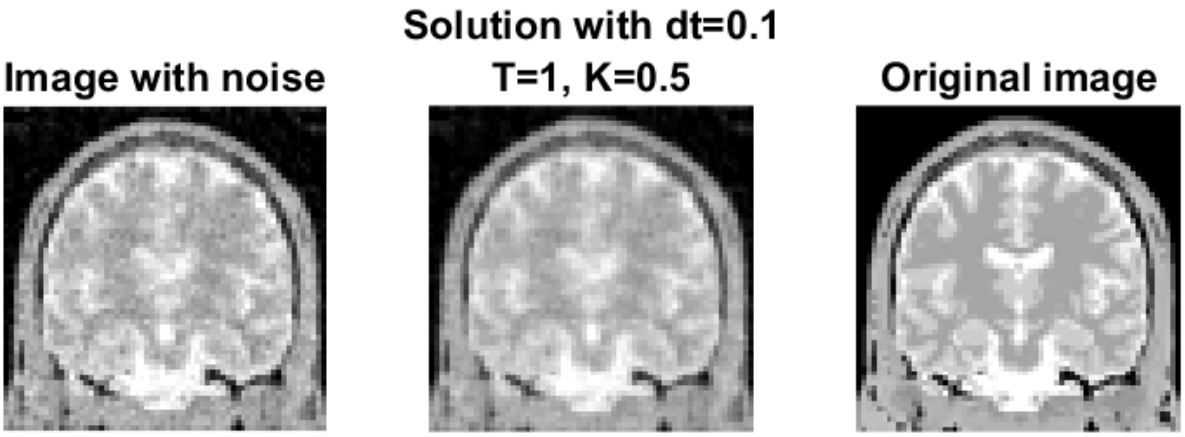

Figure 7.46: Johnson noise: best result of the ill-posed method 5.13, $\Delta t = 0.1$, $T = 1$, $K = 0.5$

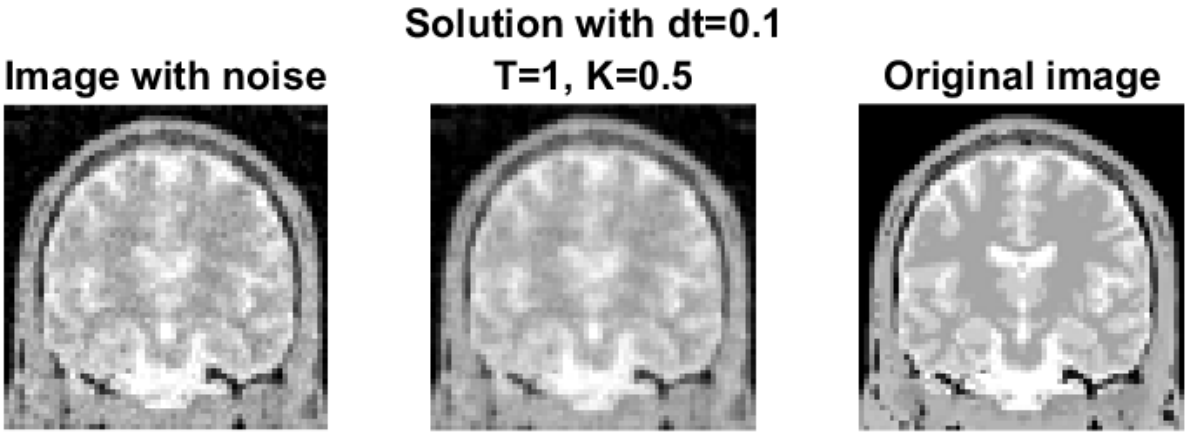

Figure 7.47: Johnson noise: best result of the well-posed method 5.12, $\Delta t = 0.1$, $T = 1$, $K = 0.5$

7.5. SUMMARY OF RESULTS

For all test problems and both numerical schemes the best result has always been obtained when using one of the Perona-Malik equations, either with or without the addition of a Gaussian kernel. Mostly the Perona-Malik method 5.4 gave the best outcome.
When comparing the numerical methods we can conclude that in the first and third test problem the AOS method gave better results, often with a smaller value of Δt and T. However, the results were only slightly better for these two test problems and the results for the second test problem were worse than the FTCS implemented outcomes. In addition to that, the running time was slightly less for the FTCS method. This running time is not a problem for a single image, but does make a difference when testing many variables for multiple test problems. Therefore, the FTCS implementation has been used for all test problems in the other chapters on numerical experiments (chapter 9 and 11)

However, the FTCS implementation does have a problem with the ill-posed and well-posed method. In all test problems instability appeared for these methods, especially for low values of K. This instability was not observed for the AOS implementation. Luckily, this instability may be overcome by the use of an adaptive time step, which will be discussed in the next chapters.

Another instability problem arised in the second test problem for both Perona-Malik methods. They showed instability for either a small Δt or even all Δt in combination with a small $K = 0.05$. This instability was not seen when these methods were combined with a Gaussian kernel, so this may be the solution to the problem.

Lastly something can be said about parameter K. All best results have been obtained with a value of $K = 0.05$. In most of the test problems even all results had this value for K.

8

CHOICE OF PARAMETERS

In the implementation of the methods three parameters are very important and have a lot of influence on the results. One of these parameters is part of the diffusion coefficient: parameter K. In the Perona-Malik methods this parameter is part of the original function, in the other two methods the parameter is added to the function to prevent problems for $|\nabla u|$ close to zero. The other two parameters come from the numerical schemes: size of a time step Δt and the total number of time steps T.

8.1. CHOICE OF K

The gradient threshold parameter or contrast parameter K describes the balance between blurring and preserving edges. When K is close to zero, almost no effect will occur, since the diffusion coefficient goes to zero. When K is high, the diffusion coefficient approaches 1, which means that the method would be similar to the heat equation and would lead to blurring. This balance between blurring and edge preservation we have seen before. Recall the threshold property:

$$\phi'(|\nabla u|) = \begin{cases} > 0, & |\nabla u| < \tau \\ < 0, & |\nabla u| \geq \tau \end{cases} \tag{8.1}$$

We can state that K is similar to this threshold value τ. To maintain edges we need $|\nabla u| \geq K$. In each iteration some blurring occurs, which means that the average value of $|\nabla u|$ decreases. Since $K \leq |\nabla u|$ we need K to decrease in each iteration as well.
In [14] three possible methods are mentioned to estimate parameter K. All methods reduce K in each iteration.

1. the first method uses a 'noise estimator', which computes a histogram of all values of $|\nabla u|$ throughout the image and sets K equal to 90% of its integral. Since it was unclear what was meant by the integral of a histogram, a different estimation based on the noise estimator was used in the implementation. K was chosen as 90% of the average absolute gradient of u throughout the image.

2. the second method was defined by Black et al. ([15]) and estimates K as $K = 1.4826MAD(\nabla u)$, with MAD the median absolute deviation given by MAD = median($|\nabla u -$ median($|\nabla u|$)|). Note that median($|\nabla u|$) is a scalar and ∇u a vector, so this notation means that for each value in ∇u the value median($|\nabla u|$) is subtracted.

3. the third method was described by Voci et al. ([16]) and uses the p-norm of the image to obtain a value for K. They define K as $K = \sigma|u|_p/n^2$, with σ a constant proportional to the average image density and $|u|_p$ the p-norm given by $|u|_p = \left(\sum_{i,j}|u|^p\right)^{1/p}$. In the article they find a value of K in the order of magnitude of 10^{-3}. This order of magnitude can be used to find an estimate of σ, since σ is not specified clearly in the article. The value of p is also not mentioned in the article.

8.2. CHOICE OF TIME STEP Δt

The second parameter is the step size Δt. We know that the AOS implementation is stable for all values of $\Delta t > 0$. The FTCS method does have a limitation: it is only stable for $0 < \Delta t \leq 0.25$.
Finding a correct value for Δt can be difficult. When the image is changing rapidly a smaller time step is beneficial, while at slow changes over time the time step can be much larger ([17]). Therefore an adaptive time step can be a solution. In [17] the following time step was chosen:

$$\Delta t^{(i)} = \frac{1}{\alpha}\left(a + b\exp\left(-\max\left(\frac{|\mathrm{Re}(\partial u^{(i)}/\partial t|)}{\mathrm{Re}(u^{(i)})}\right)\right)\right) \tag{8.2}$$

with $|\mathrm{Re}(\partial u^{(i)}/\partial t|)/\mathrm{Re}(u^{(i)})$ the fraction of change of the image at iteration i. The parameters a and b control the timestep. In the article this adaptive time step is used for the FTCS method with $0 < \Delta t \leq 0.25$. They take $\alpha = 4$ and therefore need $a + b \leq 1$, which is the case for their choice of $a = 0.25$ and $b = 0.75$.
We see that for large changes of the image the value of the exponent will approach zero, leading to a time step Δt of $a/\alpha = 0.0625$. When the image changes slowly, the exponent will go to 1, resulting in a time step Δt of $(a+b)/\alpha = 0.25$. This means that for initial iterations Δt will be around 0.0625 and will increase to a value approaching 0.25.

For the well-posed and ill-posed methods in particular, we need a different approach of Δt. We have seen in the results that the problem can become unstable for some parameters. It might be possible to overcome this problem with an adaptive time step. This time step should decrease when the image seems to diverge from the solution. A possibility for a time step with this behaviour can be:

$$\Delta t^k = q \cdot \Delta t^{k-1}, \qquad \text{if } \sum_{i,j} u_{i,j}^k > 1.0001 \sum_{i,j} u_{i,j}^{k-1}, k > 0 \tag{8.3}$$

with $0 < q < 1$.
This time step is based on the total pixel value of the image $\sum_{i,j} u_{i,j}$. It is known that the total pixel value of the image should stay constant in each iteration k. Therefore the total pixel value at time $t = 0$, the noisy image, should also be the total pixel value of the optimal solution obtained with diffusion filtering. When the total pixel value increases in an iteration, it means that the image is diverging from this optimal solution. Decreasing Δt bij a factor q can hopefully overcome this deviation. We do need to keep in mind that the total pixel value can also increase due to numerical errors, therefore the value 1.0001 has been chosen. It is sufficiently high that the difference is due to instability and not due to numerical error.

8.3. OPTIMAL NUMBER OF TIME STEPS T

Another important parameter is the number of time steps T. Since in each iteration one neighbour in each direction is used to determine a new pixel value, the value T gives the number of neighbours in each direction integrated in the new pixel value.

In this section we want to determine when to stop the diffusion. So from now on the number of time steps T at which the diffusion is stopped is named the stopping time S. This parameter is crucial in determining when the optimal solution is reached. Stopping too soon leads to an image with leftover noise, but stopping too late causes the image to lose its accuracy due to blurring.

In [18] a method was proposed to determine the stopping time S. First they mentioned a method based on the median absolute deviation (MAD) of the difference between the original solution and the solution at time t. The stopping criterium was given by:

$$\begin{aligned} S_{MAD} =& \arg\min_t \text{MAD}(u(t) - u_{original}) \\ =& \arg\min_t \text{median}\left(|(u(t) - u_{original}) - \text{median}(|(u(t) - u_{original})|)|\right) \end{aligned} \tag{8.4}$$

.
This stopping criterium can only be used when the original image is known, which is usually not the case. However, they noticed that the plot of the MAD coincides with the plot of the correlation coefficient $\text{corr}(u(0) - u(t), u(t))$, as can be seen in figure 8.1. Since this correlation coefficient is independent of the original image, it might be a suitable option for a stopping criterium.
In the article the stopping criterium was defined as:

$$S_{corr} = \arg\min_t \text{corr}(u(0) - u(t), u(t)) \tag{8.5}$$

The conclusion in [18] was that the S_{MAD} is a good estimator of the optimal stopping time. The obtained results were very close to the solutions with the best visual quality. The other stopping time, S_{corr}, underestimated the optimal stopping time: to obtain the visually optimal result the diffusion should have stopped later than at S_{corr}.
Both stopping times S_{MAD} and S_{corr} will be tested on our own test problems to verify these outcomes.

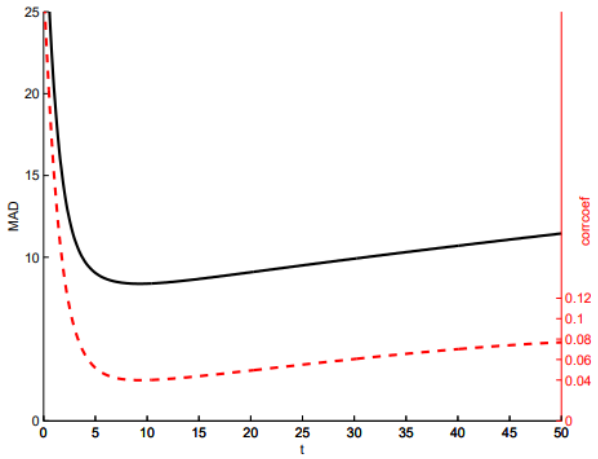

Figure 8.1: Plot of MAD (solid line) and correlation coefficient (dashed line) [18]

Another possibility for a stopping time is based on the difference between the filtered image and the noisy image $|u(t) - u(0)|$ and $|\nabla u(t)|$. In every iteration the difference between the image $u(t)$ and the initial noisy image $u(0)$ becomes larger, while the gradient of u decreases. A balance between these values can lead to a correct stopping time.

Choose stopping time S_λ as:

$$S_\lambda = \arg\min_t |u(t) - u(0)|^2 + \lambda |\nabla u(t)|^2 \tag{8.6}$$

with $\lambda > 0$. This parameter λ may be difficult to determine.
Optimal values for λ will be investigated in the following chapter on numerical experiments of the parameters. Both previously mentioned stopping times, S_{MAD} and S_{corr}, will also be tested and compared. Hopefully, these experiments will lead to an optimal formula for a stopping time.

9

NUMERICAL EXPERIMENTS: CHOICE OF PARAMETERS

Since previously the conclusion was made that the FTCS implementation in combination with the Perona-Malik method 5.4 lead to good results, these methods are used to compare the results for all parameters. The only exception is the adaptive time step for the well-posed and ill-posed method. Obviously these methods are used here, but they are still implemented with FTCS.

9.1. PARAMETER K

First it is important to gain a better understanding of the behaviour of parameter K. Therefore the optimal value K has been calculated for many parameters. The parameters were tested on two test problems: the Shepp-Logan phantom with Gaussian noise with a standard deviation of 0.06 and the Ella phantom with Johnson noise. The parameters for the first test problem are $n = 64, 200, 400$, $\Delta t = 0.05, 0.1, 0.2$ and $T = 1, 5, 10, 20, 50, 100, 200$. Since the Ella phantom is only available for $n = 64$, this is the only value for n tested for this test problem, Δt and T have the same options.
To determine K_{opt}, the standard deviation δ has been used as before:

$$\delta = \frac{1}{n}||u_{original} - u_{final}||_2 \tag{9.1}$$

It was expected that many factors influence K. The following figure 9.1 shows the influence of the image size n, the influence of the initial image and the influence of Δt on K. The first two images are both for a time step of $\Delta t = 0.05$. These results are for the Shepp-Logan phantom.

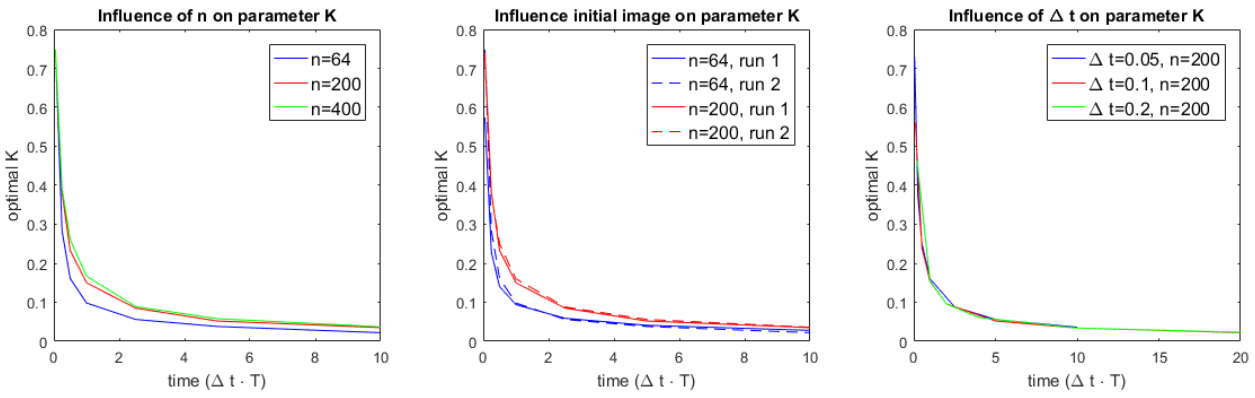

Figure 9.1: Influence of a) n, b) initial image and c) Δt on optimal K

From this image we can conclude that the behaviour of K is similar for all plots. For small values of $\Delta t \cdot T$ the optimal K descends rapidly from a value of around 0.8 to 0.1 and remains between 0 and 0.1 for the higher values of $\Delta t \cdot T$. However, we can see from the first figure that the size of the image n has an influence on the exact shape of the plot. It may seem like a small difference, but a slight change in K has a large effect on the outcome. The second figure shows the influence of the initial image on the optimal K. This was tested by calculating the value of K_{opt} for different runs of the program. This means that the noise remained Gaussian with a standard deviation of 0.06, but the initial images were not exactly the same. As can be seen from the figure, this has a slight influence on the value of K_{opt}. The third variable was Δt. From the third image we can conclude that it does not matter what time step is chosen. The optimal K is depending on the product $\Delta t \cdot T$.

A comparison between the K_{opt} values of the Shepp-Logan test problem (also with $n = 64$) and the Ella test problem are showed in figure 9.2. These results were obtained with a time step of $\Delta t = 0.05$. We can see that the behaviour of the plot is still the same, but the difference is quite large between the two test problems.

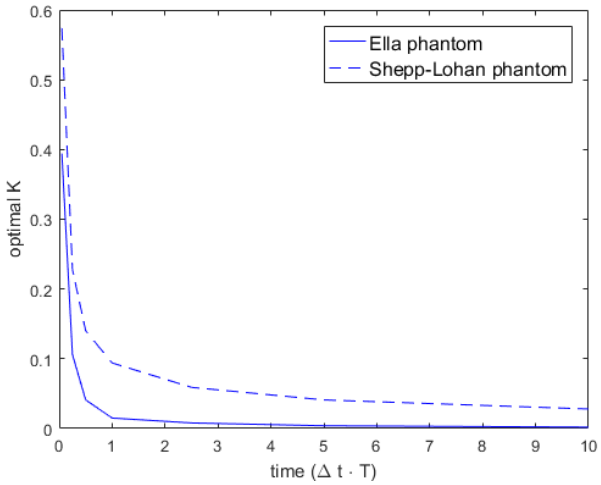

Figure 9.2: Comparison of the Shepp-Logan phantom (n=64) and Ella phantom

Based on both figures, we must unfortunately conclude that it is hard to predict a correct value for K. However, in the previous chapter we mentioned three ways of finding a reasonable value for K ([14]). These methods have been compared with the above shown results.

9.1.1. 'NOISE ESTIMATOR' METHOD

The first method was a so-called 'noise estimator'. This estimator is computed by calculating a histogram of all values of $|\nabla u|$ throughout the image and setting K equal to 90% of its integral. A slightly different estimation has been used to estimate this noise estimator. K is chosen as 90% of the average absolute gradient of u throughout the image. Since K changes and decreases for each iteration, the average value K_{avg} has been calculated and compared with the known optimal values. The results for both test problems and various values of n are in figure 9.3. These results have again been obtained with $\Delta t = 0.05$

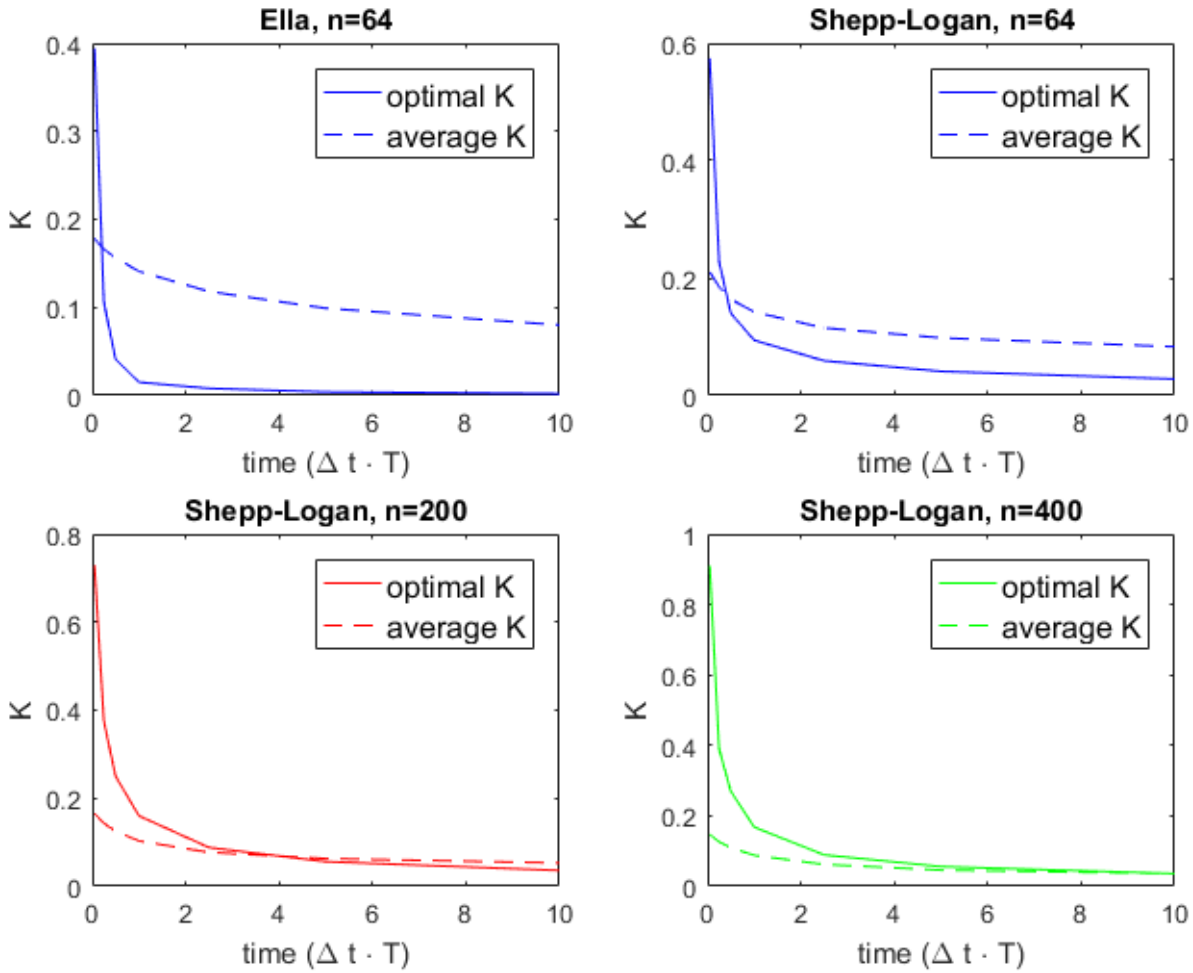

Figure 9.3: Comparison of K_{opt} and K_{avg} for the Ella and Shepp-Logan test problem

The behaviour of the average K obtained with the noise estimator method is not exactly similar to that of the optimal K. It is decreasing as $\Delta t \cdot T$ increases, but the initial decrease is much less steep than for the optimal K. In all four plots, the dotted line of K_{avg} starts out lower than the continuous line of K_{opt}. At some point these lines cross and from then on K_{avg} is higher than K_{opt}. This point of intersection is around $\Delta t \cdot T = 0.05$ for the $n = 64$ Shepp-Logan and $n = 64$ Ella phantom. It is later, at around $\Delta t \cdot T = 4$, for the $n = 200$ Shepp-Logan phantom and much later, at around $\Delta t \cdot T = 10$, for the $n = 400$ Shepp-Logan phantom.

The point of intersection is very important, since for values of $\Delta t \cdot T$ around this point, the use of the 'noise estimator' leads to a result similar to the optimal result. This can be seen in image 9.4. This figure shows the outcomes for the optimal K (9.4a) and for K obtained with the noise estimator method (9.4b).
Unfortunately, if Δt and T are chosen in such a way that $\Delta t \cdot T$ is not close to the point of intersection, the results are visually very poor compared to the optimal solution. So the method has proved to be useful only for the correct values of Δt and T. The only assumption we can make is that for low values of n we want $\Delta t \cdot T$ to be small and for higher values of n $\Delta t \cdot T$ has to be high as well.

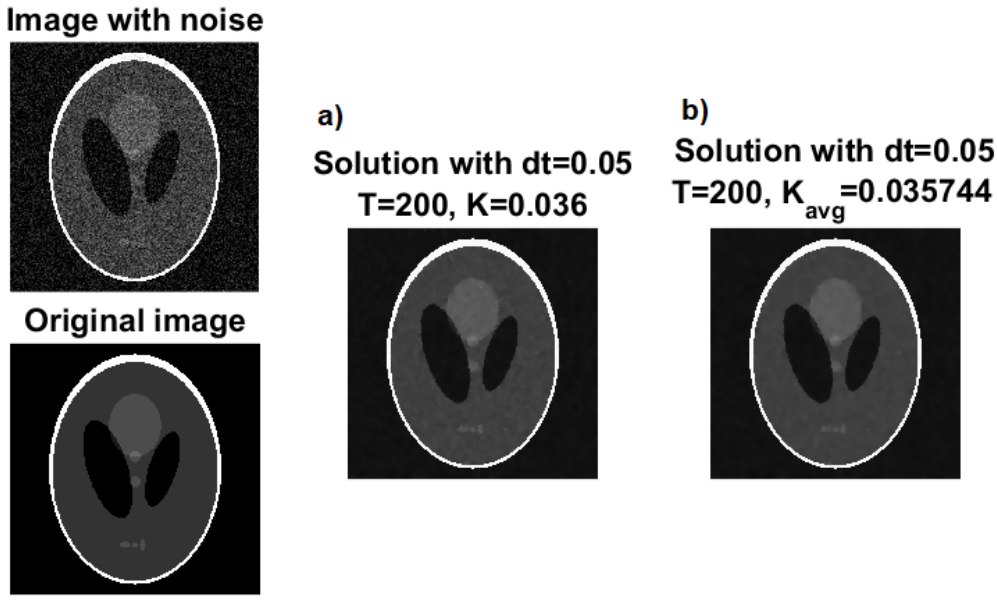

Figure 9.4: The results for $n = 400$, $\Delta t = 0.05$ and $T = 200$: a) optimal b) noise estimator method

9.1.2. MAD METHOD

The second option mentioned in [14] estimates K as $K = 1.4826MAD(\nabla u)$, with MAD the median absolute deviation given by MAD = median($|\nabla u -$ median($|\nabla u|$)$|$). Recall that median($|\nabla u|$) is a scalar and ∇u a vector, so it states that for each value in ∇u the value median($|\nabla u|$) is subtracted. This method also causes K to decrease in every iteration, so again the average value of K was used to compare it to the optimal K. The results for Shepp-Logan ($n = 64, 200, 400$) and Ella are shown in figure 9.5, all with $\Delta t = 0.05$.

We can see that this average K shows a similar pattern as the average K obtained with the noise estimator method. The only difference is that the values are much smaller. This has an immediate effect on the intersection point. Due to the lower values of K_{avg}, the intersection point is much later. Only for the Ella phantom the intersection is around a similar value. This means that in order to generate an optimal result the value of $\Delta t \cdot T$ has to be quite high. Since the FTCS method is only stable for $0 < \Delta t \leq 0.25$, this leads to high number of time steps T, resulting in a longer calculation time. We can conclude that this method of finding K is inferior to the noise estimator method.

9.1.3. P-NORM METHOD

The third method we discussed uses the p-norm of the image to obtain a value for K. K is defined as $K = \sigma|u|_p/n^2$, with σ a constant proportional to the average image density and $|u|_p$ the p-norm given by $|u|_p = \left(\sum_{i,j}|u|^p\right)^{1/p}$. In the article a value of K in the order of magnitude of 10^{-3} is found. Since σ is not specified clearly, we use this order of magnitude to find an estimate of σ. It was also unspecified what value of p would be useful.

For the numerical experiments the same options for n and T were used and the time step was chosen as $\Delta t = 0.05$. Since it was unclear how to choose σ and p, both were tested for different values. To obtain results in the region of 10^{-3} a value of $\sigma = 10 \cdot u_{avg}$ was needed for $p = 2$ and $\sigma = 100 \cdot u_{avg}$ for $p = 3$, with u_{avg} denoting the average pixel density. However, even though article [16] states that the value of K decreases, this was hardly the case for all parameters. The value remained almost constant for all iterations. It means that this method is basically similar to manually choosing a constant, but in a more unpredictable way. This method is therefore the worst of the tested methods.

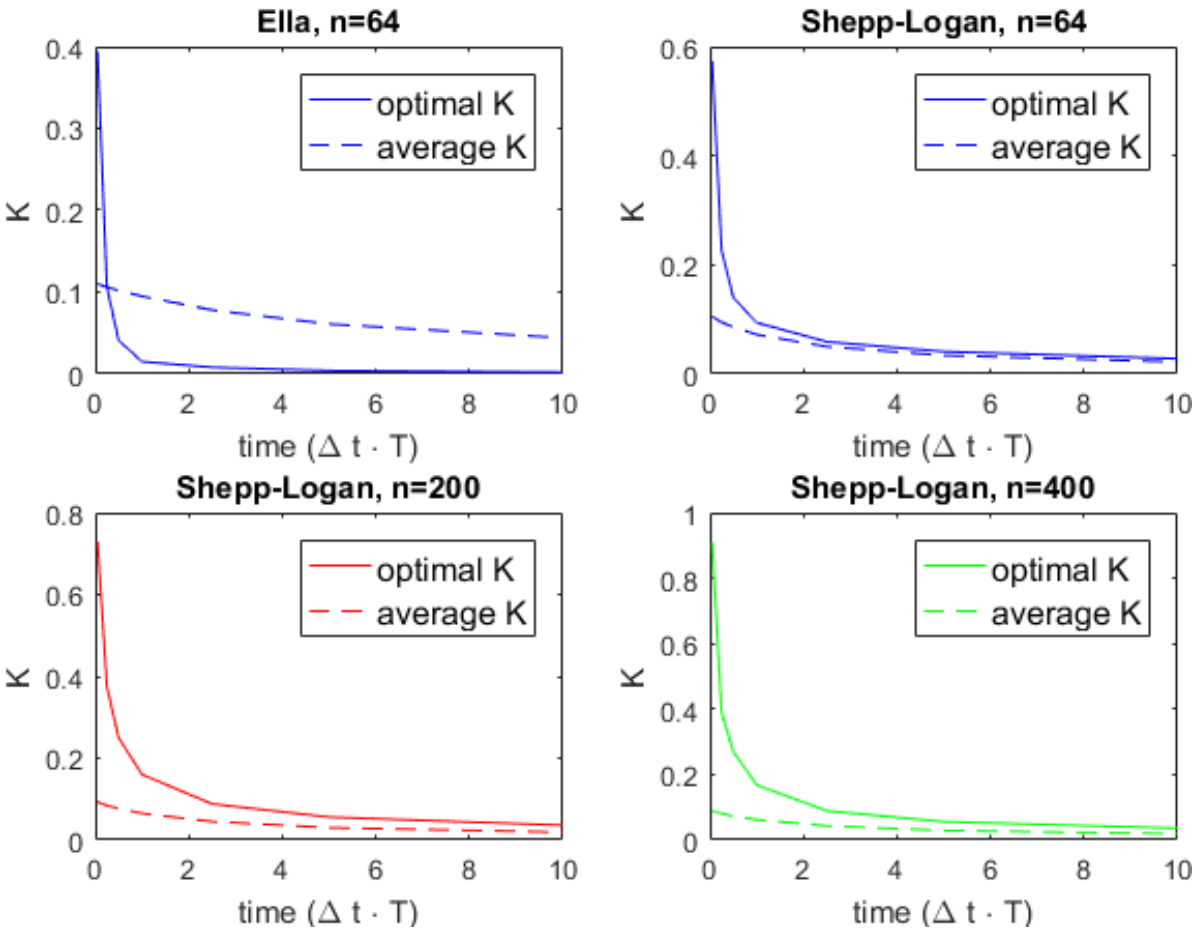

Figure 9.5: Comparison of K_{opt} and K_{avg} for the Ella and Shepp-Logan test problem

9.2. TIME STEP Δt

In the theoretical part a method with an increasing time step was discussed for the Perona-Malik method with FTCS implementation and a decreasing time step for the ill-posed and well-posed methods. Both will be discussed in this section.

9.2.1. PERONA-MALIK

Before looking into the method we try to understand the behaviour of Δt and investigate the influences of different parameters. We want to know how the size of the image n affects the optimal Δt and what happens for different values of K. We also compare the behaviour for the Shepp-Logan phantom ($n = 64$) with the behaviour for the Ella phantom. These results are shown in figure 9.6.

We see that n has a significant influence on the optimal Δt. During the initial time iterations the optimal value for Δt is the maximum time step for stability, $\Delta t = 0.25$, and after a certain number of iterations this optimal Δt starts to decrease. For higher values of n, this decrease starts at a higher number of iterations. This means that for the same number of iterations the optimal value for Δt is higher for a more detailed image than for an image with a low n.
Parameter K affects optimal Δt even more. For higher values of K ($K = 0.5$), the decrease starts immediately at the first iteration and this decrease is much steeper than for lower values of K. For larger K the influence of n is similar, but the lines for $n = 64$ and $n = 400$ are closer together than for lower K.

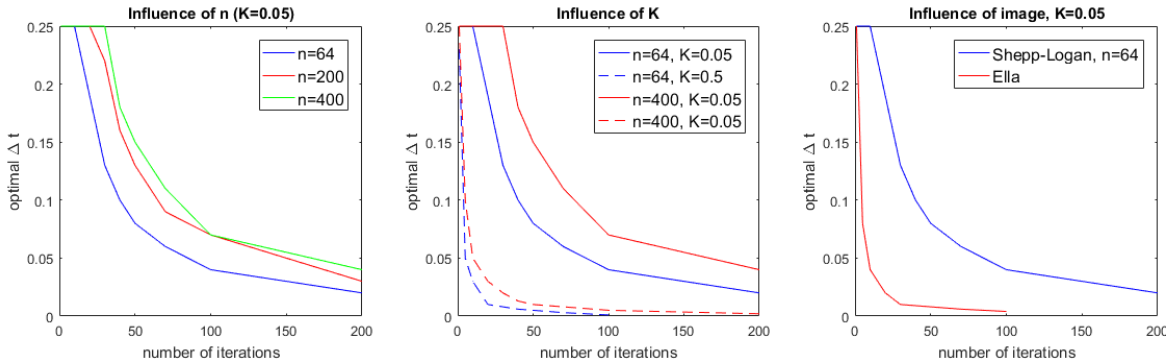

Figure 9.6: Influence of a) n, b) parameter K and c) image on optimal Δt

The third image shows the difference between the two phantoms. Even though both had similar values for K and n, the curve of Ella decreases much sooner and steeper than the Shapp-Logan curve. This behaviour is similar to the Shepp-Logan phantom with a high K, shown in the middle image.

We can conclude that many factors play a role in finding the optimal Δt. However, the behaviour of the optimal Δt is similar for all plots. It starts at the maximum value of 0.25 and decreases after some number of iterations to a value close to 0. This is contradictory to the theory described in the literature. The article ([17]) stated that for a rapidly changing image a smaller time step is beneficial, while at slow changes over time the time step can be much larger. Based on this statement the following adaptive time step was chosen, given by:

$$\Delta t^{(i)} = \frac{1}{\alpha}\left(a + b\exp\left(-\max\left(\frac{|\mathrm{Re}(\partial u^{(i)}/\partial t|)}{\mathrm{Re}(u^{(i)})}\right)\right)\right) \tag{9.2}$$

with $|\mathrm{Re}(\partial u^{(i)}/\partial t|)/\mathrm{Re}(u^{(i)})$ the fraction of change of the image at iteration i. In the article they take $\alpha = 4$, $a = 0.25$ and $b = 0.75$.

This method is based on the fact that for large changes of the image the value of the exponent will approach zero, leading to a time step Δt of $a/\alpha = 0.0625$, and for slower changes, the exponent will go to 1, resulting in a time step Δt of $(a+b)/\alpha = 0.25$. This means that for initial iterations Δt will be around 0.0625 and will increase to a value approaching 0.25, which is exactly the opposite of the behaviour seen in figure 9.6.

Therefore this equation has been modified to give rise to a decreasing function, starting at $\Delta t = 0.25$ and decreasing to a value close to 0. The equation is now given by:

$$\Delta t^{(i)}_{mod} = \frac{1}{\alpha}\left(a - b\exp\left(-\max\left(\frac{|\mathrm{Re}(\partial u^{(i)}/\partial t|)}{\mathrm{Re}(u^{(i)})}\right)\right)\right) \tag{9.3}$$

The constants have been chosen differently also: $\alpha = 4$, $a = 1$ and $b = 1$.

The behaviour of the optimal Δt, the original method and the modified method are given in figure 9.7. We see that in all images the original method is almost the opposite of the optimal Δt. However, the curve for the modified method seems to resemble the optimal Δt better. In the left image we can see that the modified method curve is below the optimal curve. This is what we want, since the optimal curve shows the optimal Δt when using a constant time step. For example, for 50 iterations the optimal Δt is given by 0.15. This means that every iteration used a constant value of $\Delta t = 0.15$ to obtain the optimal solution. The modified method has a changing time step, so to get the best result we want the average time step for 50 iterations to be 0.15. Since it starts at a high value of 0.25, we need the curve to be below the optimal curve to get an average value close to 0.15.

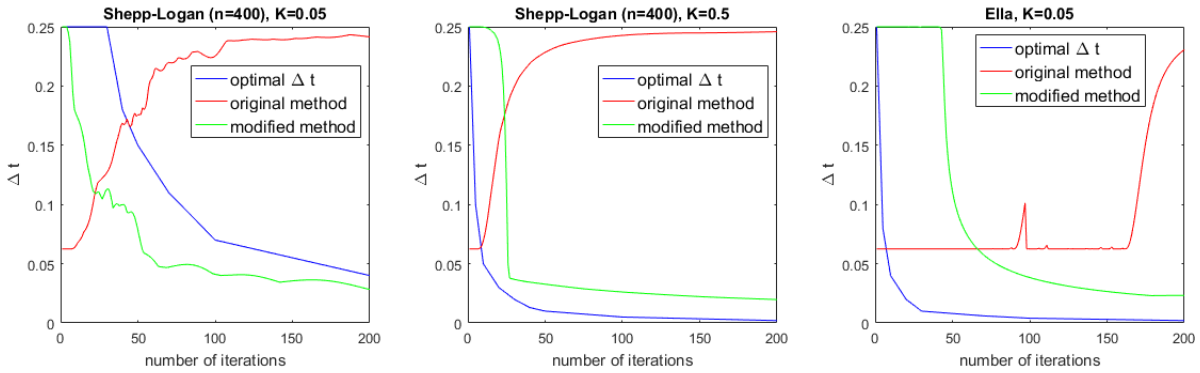

Figure 9.7: Comparison of optimal Δt, original method and modified method for the Shepp-Logan and Ella test problem

The other two images show that the modified method is above the optimal curve. This is not ideal, since this means that the average time step is higher than the optimal Δt, leading to too much blurring.

The following three figures show the optimal outcome and the outcomes when using the original method and the modified method for Shepp-Logan ($K = 0.05$, figure 9.8, and $K = 0.5$, figure 9.9) and Ella ($K = 0.05$, figure 9.10). As expected we see in the first figure that for both a low and a high number of iterations T the modified method leads to better results (figure c) and f) are better than respectively b) and e)). This was expected since the modified method stayed below the optimal curve.

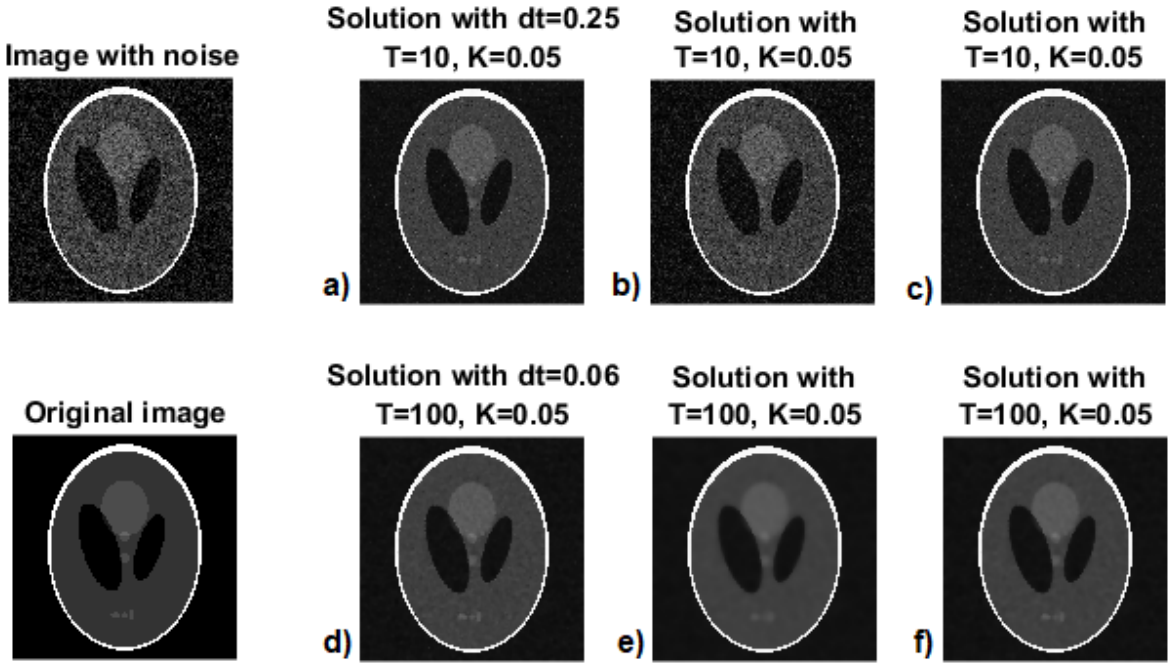

Figure 9.8: The best results obtained for $K = 0.05$ and a) $T = 10$, optimal Δt b) $T = 10$, original method c) $T = 10$, modified method, d) $T = 100$, optimal Δt, e) $T = 100$, original method, f) $T = 100$, modified method

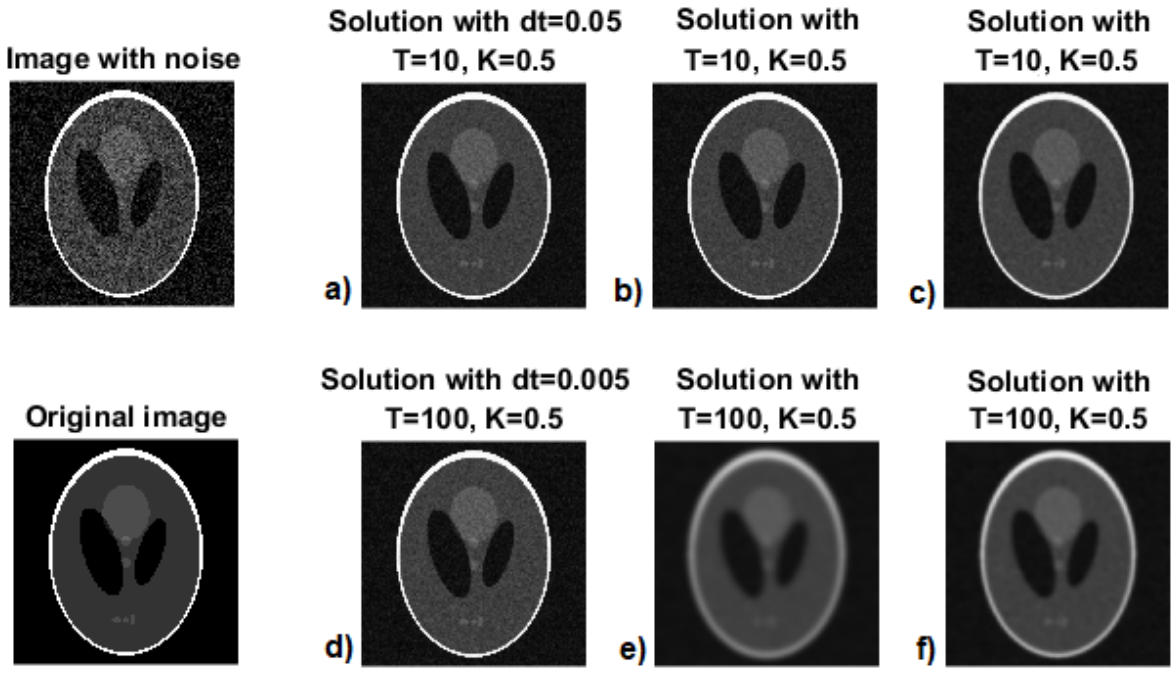

Figure 9.9: The best results obtained for $K = 0.5$ and a) $T = 10$, optimal Δt b) $T = 10$, original method c) $T = 10$, modified method, d) $T = 100$, optimal Δt, e) $T = 100$, original method, f) $T = 100$, modified method

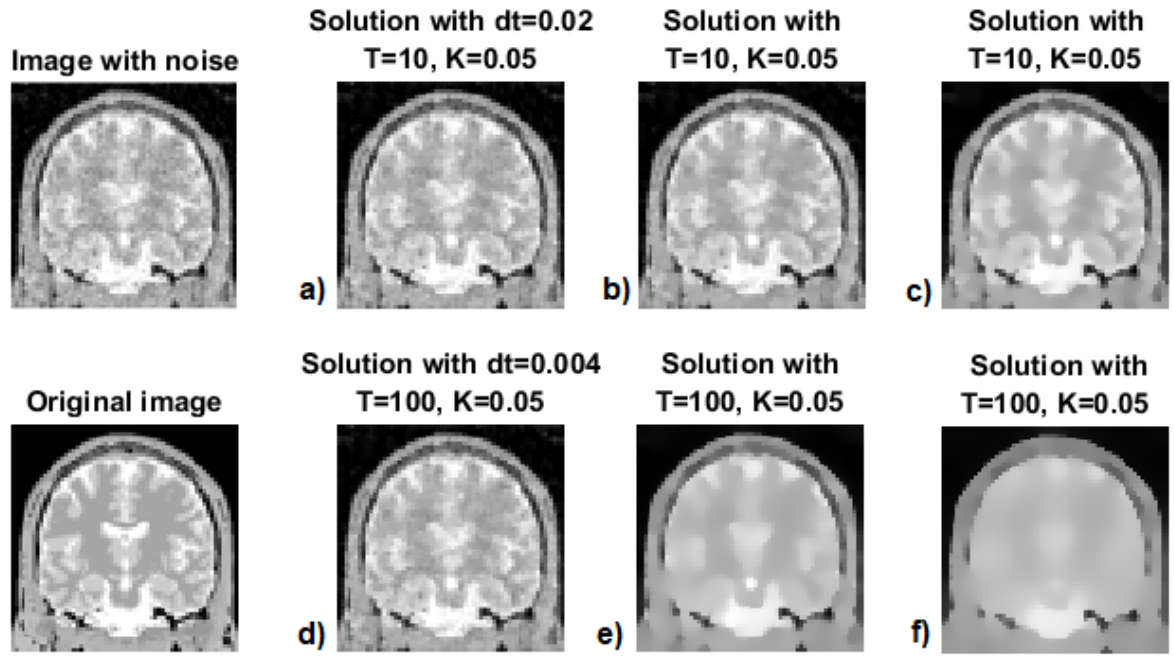

Figure 9.10: The best results obtained for $K = 0.05$ and a) $T = 10$, optimal Δt b) $T = 10$, original method c) $T = 10$, modified method, d) $T = 100$, optimal Δt, e) $T = 100$, original method, f) $T = 100$, modified method

The second figure shows that for $T = 10$ the original method leads to a slightly better result than the modified method. This is because the modified method did not have enough time to decrease, but stayed around a value of $\Delta t = 0.25$, leading to a blurrier outcome. For higher values of T the opposite is true: the original method used $\Delta t = 0.25$ for many iterations, causing blurring, while the modified method had a lower average Δt. However, both methods are much blurrier than the optimal result, due to an overall higher Δt.
The third image shows the Ella phantom for $K = 0.05$. As predicted the modified method shows too much blurring and is for both low and high values of T not a good option for estimating Δt. Even though the original method leads to better results, it is still a poor method for higher T, as can be seen in figure 9.10e.

Overall the modified method seems to be the better option for more detailed images (higher n), in that case the visual quality was high for both $K = 0.05$ and $K = 0.5$. Unfortunately, the method showed too much blurring for images with a low n, especially for many iterations. The original method had slightly better results for low n, but still of poor visual quality.

9.2.2. ILL-POSED AND WELL-POSED METHOD

For the ill-posed and well-posed method an adaptive time step was introduced to overcome the instability seen for some parameters (chapter 8). It was given as:

$$\Delta t^k = q \cdot \Delta t^{k-1}, \qquad \text{if } \sum_{i,j} u_{i,j}^k > 1.0001 \sum_{i,j} u_{i,j}^{k-1}, k > 0 \tag{9.4}$$

with $0 < q < 1$. This equation causes the time step Δt to decrease with factor q when the total pixel value increases only slightly. This condition has been chosen, since an increase in the average pixel value means deviation from the ideal solution.
The factor q is important for the outcome. Choosing q too small or too large can also cause instability. The stability interval has been determined for the Shepp-Logan phantom for $n = 64$, $n = 200$, $n = 400$, $n = 800$ and the Ella phantom with a starting time step of $\Delta t^{k=1} = 0.05$. The results are in the following table.

test problem	*stability for*
Shepp-Logan, $n = 800$	$0.23 \leq q \leq 0.25$
Shepp-Logan, $n = 400$	$0.05 \leq q \leq 0.5$
Shepp-Logan, $n = 200$	$0.05 \leq q \leq 0.5$
Shepp-Logan, $n = 64$	$0.05 \leq q \leq 0.95$
Ella	$0.01 \leq q \leq 0.95$

Table 9.1: Stability interval for several test problems

In the table we can observe that the size of the image n is important for the interval. For a value of $n = 800$ the interval is the smallest, only $0.23 \leq q \leq 0.25$. For the smallest images almost all values of q lead to a stable problem. The table only shows the results for a starting time step of $\Delta t^{k=1} = 0.05$. For a higher starting time of $\Delta t^{k=1} = 0.2$ the problem remains unstable for all q, but the outcomes are most improved for values $0.2 \leq q \leq 0.3$.
Within the stability interval the outcomes can be very different depending on q. The smallest stable value for q leads to the best results, while the largest stable q gives a much blurrier result. Two examples of this fact are shown in figure 9.11 and 9.12.

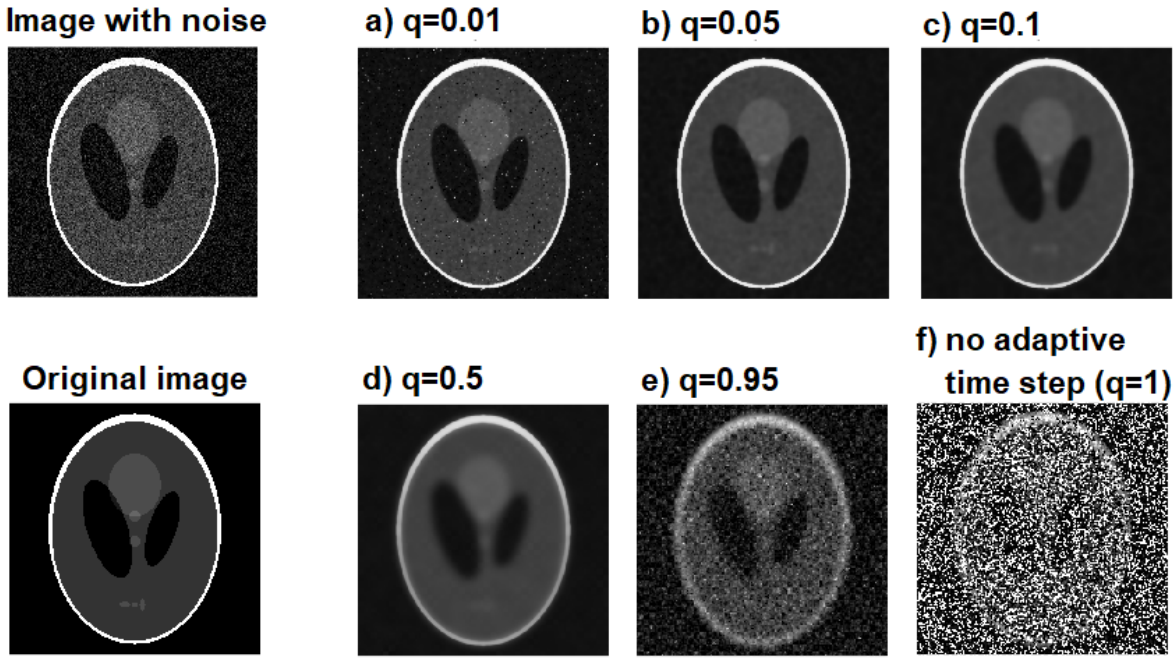

Figure 9.11: Results of the adaptive time step with a) $q = 0.01$ b) $q = 0.05$ c) $q = 0.1$ d) $q = 0.5$ e) $q = 0.95$ f) no adaptive time step ($q = 1$)

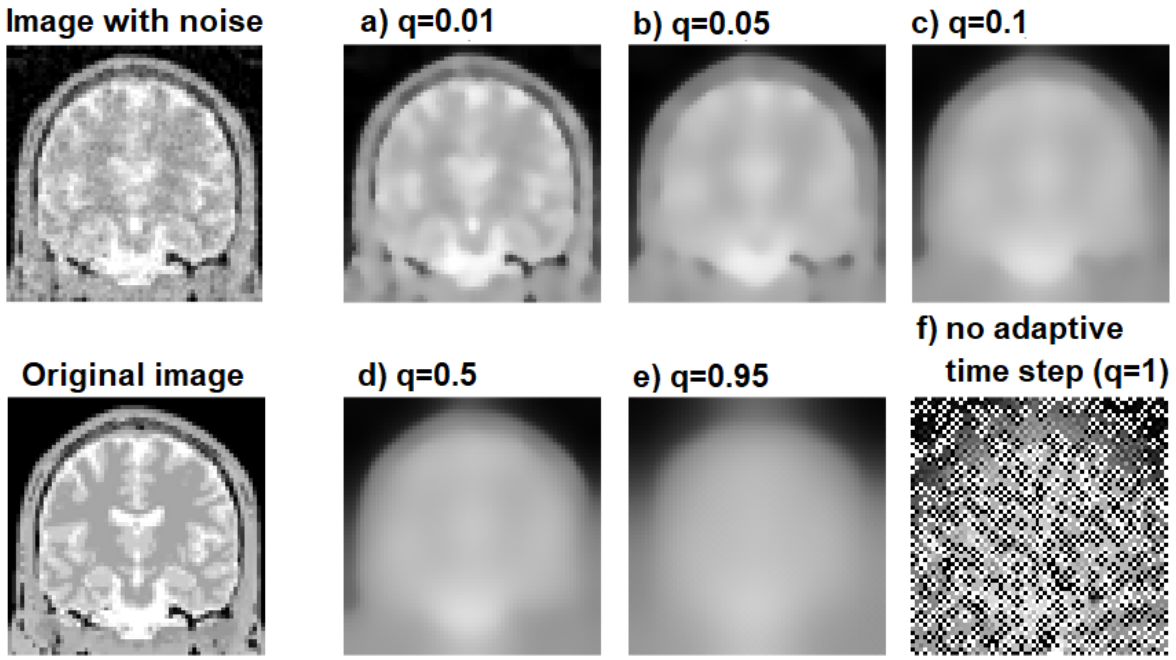

Figure 9.12: Results of the adaptive time step with a) $q = 0.01$ b) $q = 0.05$ c) $q = 0.1$ d) $q = 0.5$ e) $q = 0.95$ f) no adaptive time step ($q = 1$)

The first figure shows some outcomes for the Shepp-Logan phantom, $n = 400$, for various values of q. Obviously all outcomes are better than the unstable solution for a constant time step. However, image a) and e) give unstable results, agreeing with the table. Image b) gives the best result. A similar pattern can be observed in the second figure of Ella. Here only e) gives an unstable result, again corresponding to the table, and a) leads to the best result.
Since the size of the interval can be different for different n and starting time step sizes Δt^{k+1}, it is difficult to find the minimal stable q. According to the table a value of $q = 0.25$ is the safest option, since this always leads to stable results. Unfortunately, from figure 9.12 we can see that a value of $q = 0.25$ would lead to poor results. However, for a small initial time step size most images lead to a stable result for a much smaller value $q = 0.05$, so this value can be chosen for a small Δt^1.
The overall conclusion is that this adaptive time step definitely leads to better results than a constant Δt. The only difficulty is finding the optimal q.

9.3. NUMBER OF TIME STEPS T

The optimal number of time steps T is named the stopping time S. This stopping time is very important for the outcome: if S is too small, noise remains in the image, but if S is too large the outcome will be unclear due to too much diffusion.
To determine when we want the filtering to stop we looked into the visually optimal S for both phantoms and many parameters. These values are given in the three tables below (9.2, 9.3 and 9.4) in the colums S_{visual}.

In the theoretical part three methods were discussed. The first one used the median absolute deviation (MAD) and was given by:

$$\begin{aligned} S_{MAD} =& \arg\min_t \text{MAD}(u(t) - u_{original}) \\ =& \arg\min_t \text{median}\left(|(u(t) - u_{original}) - \text{median}(|(u(t) - u_{original})|)|\right) \end{aligned} \tag{9.5}$$

.
A big disadvantage of this stopping time is the fact that the original image is needed. For the test problems the original is known, so we can calculate this stopping time. However, usually the original is not known, which makes the stopping time useless. The results for the calculated stopping times are also in the tables in the column S_{MAD}.
The second method was based on the fact that the plot of the MAD coincides with the plot of the correlation coefficient $\text{corr}(u(0) - u(t), u(t))$. This correlation coefficient is independent of the original image and is defined as:

$$S_{corr} = \arg\min_t \text{corr}(u(0) - u(t), u(t)) \tag{9.6}$$

The results for this method are also given in the tables in the column S_{corr}.

The last method was the λ-method, which defined the stopping time as:

$$S_\lambda = \arg\min_t |u(t) - u(0)|^2 + \lambda |\nabla u(t)|^2 \tag{9.7}$$

with $\lambda > 0$.
For all visual optima given in the tables the corresponding λ was determined. This is the value λ for which 9.7 reached its minimum at exactly the visual optimum. These values can be found in the tables in the colums λ_{visual}.

Ella, $K = 0.05$				
Δt	S_{MAD}	S_{corr}	S_{visual}	λ_{visual}
$\Delta t = 0.01$	69	41	70	0.038
$\Delta t = 0.05$	14	8	15	0.036
$\Delta t = 0.1$	7	4	7	0.036
$\Delta t = 0.2$	3	3	5	0.034
Ella, $K = 0.5$				
Δt	S_{MAD}	S_{corr}	S_{visual}	λ_{visual}
$\Delta t = 0.01$	6	25	25	0.030
$\Delta t = 0.05$	1	5	5	0.027
$\Delta t = 0.1$	1	3	2	0.019
$\Delta t = 0.2$	1	1	1	0.010

Table 9.2: Stopping times for Ella

Shepp-Logan, $n = 64$, $K = 0.05$				
Δt	S_{MAD}	S_{corr}	S_{visual}	λ_{visual}
$\Delta t = 0.01$	461	236	350	0.040
$\Delta t = 0.05$	92	53	70	0.040
$\Delta t = 0.1$	46	30	35	0.038
$\Delta t = 0.2$	23	13	20	0.042
Shepp-Logan, $n = 64$, $K = 0.5$				
Δt	S_{MAD}	S_{corr}	S_{visual}	λ_{visual}
$\Delta t = 0.01$	35	22	25	0.040
$\Delta t = 0.05$	6	5	5	0.032
$\Delta t = 0.1$	3	3	3	0.020
$\Delta t = 0.2$	1	1	2	0.012

Table 9.3: Stopping times for Shepp-Logan, $n = 64$

Shepp-Logan, $n = 400$, $K = 0.05$				
Δt	S_{MAD}	S_{corr}	S_{visual}	λ_{visual}
$\Delta t = 0.01$	913	631	850	0.016
$\Delta t = 0.05$	181	126	180	0.015
$\Delta t = 0.1$	90	63	90	0.018
$\Delta t = 0.2$	46	31	45	0.018
Shepp-Logan, $n = 400$, $K = 0.5$				
Δt	S_{MAD}	S_{corr}	S_{visual}	λ_{visual}
$\Delta t = 0.01$	111	86	90	0.020
$\Delta t = 0.05$	21	16	18	0.022
$\Delta t = 0.1$	10	7	8	0.024
$\Delta t = 0.2$	5	1	4	0.022

Table 9.4: Stopping times for Shepp-Logan, $n = 400$

In the tables we see that S_{MAD} and S_{corr} are often completely different. They also differ quite a lot with the optimal stopping time S_{visual}. S_{corr} is always much lower than the optimum and S_{MAD} is either too low or too high. It seems like both these methods will not give the result we want.

We can make the following assumptions about the behaviour of λ_{visual}. We see that the values in the first table are almost similar to the values in the second table. Both tables belong to a test problem with $n = 64$. It might mean that the values for λ are similar for different images with the same n.
Unfortunately the value λ_{visual} is very different for the various parameters. Sometimes it does stay around the same value for different Δt (all problems with $K = 0.05$ and Shepp-Logan $K = 0.5$), but it can also decrease as Δt increases. Nevertheless, we can conclude that λ_{visual} is depending on many factors and no optimal value for λ can be easily determined.

Even though all tested methods do not lead to the optimal result, we might find a better stopping time by using a combination of the methods. The MAD-method uses the original image, which is in practice not available, so we do not want to use this method. The other two methods are only depending on the noisy image and the filtered image, so both can always be used.
Since we want to use the λ-method, we need a value for λ. From the table we see that λ_{visual} is in the range of $0.01 - 0.04$. Therefore we decided to use the method with a value of $\lambda = 0.02$. This value is often around the correct value and sometimes too high or too low. The stopping time obtained with the λ-method with $\lambda = 0.02$ is named S_λ. The new stopping time S_{new} is chosen as:

$$S_{new} = [(4 \cdot \min(S_{corr}, S_\lambda))/3]_{floor} \tag{9.8}$$

This stopping time is based on the fact that the optimal stopping time S_{visual} is always higher than or equal to S_{corr}. If S_λ is lower than S_{corr}, it is therefore also lower than the optimal. If S_λ is higher than S_{corr}, the minimum of the two is still lower than S_{visual}. By multiplying with a factor 4/3 the stopping time S_{new} is higher than this minimum, which is what we want. Since the division can lead to a noninteger, we round the value down to the lowest integer, which is denoted as $[\cdot]_{floor}$.
The values for S_{corr}, S_λ, S_{visual} and new stopping time S_{new} are given in table 9.5. The images obtained using stopping time S_{new} for different Δt are given in figures 9.13, 9.14 and 9.15 for Ella ($K = 0.05$), Shepp-Logan ($n = 400$, $K = 0.05$) and Shepp-Logan ($n = 400$, $K = 0.5$), respectively.

In the table we observe that S_{new} is always similar to the maximum of S_λ and S_{corr} or it is higher than this maximum. We can see that taking the maximum of S_λ and S_{corr} gives a similar result or a result closer to the optimal S_{visual}. Therefore taking $\max(S_{corr}, S_\lambda)$ as a stopping time might seem a better option. However, S_{corr} can change dramatically per program run, leading to a much higher value of S_{corr}. This means it is not wise to have a stopping time depending on the maximum value, since this value can be very different from the optimal stopping time.

Even though S_{new} is not always leading to the optimal stopping time, it is still a very good estimator. This is confirmed by the figures. The resulting solutions are all of high visual quality, which is the ultimate goal. In the following chapters segmentation is explained and tested on these outcomes to visualize the difference between the noisy image and the results even better.

	Ella, $K = 0.05$				Ella, $K = 0.5$			
Δt	S_λ	S_{corr}	S_{visual}	S_{new}	S_λ	S_{corr}	S_{visual}	S_{new}
$\Delta t = 0.01$	31	41	70	41	19	25	25	25
$\Delta t = 0.05$	6	8	15	8	4	5	5	5
$\Delta t = 0.1$	3	4	7	4	2	3	2	2
$\Delta t = 0.2$	2	3	5	2	2	1	1	1
	Shepp-Logan, $n = 64$, $K = 0.05$				Shepp-Logan, $n = 64$, $K = 0.5$			
Δt	S_λ	S_{corr}	S_{visual}	S_{new}	S_λ	S_{corr}	S_{visual}	S_{new}
$\Delta t = 0.01$	177	236	350	236	17	22	25	22
$\Delta t = 0.05$	40	53	70	53	4	5	5	5
$\Delta t = 0.1$	23	30	35	30	2	3	3	2
$\Delta t = 0.2$	10	13	20	13	2	1	2	1
	Shepp-Logan, $n = 400$, $K = 0.05$				Shepp-Logan, $n = 400$, $K = 0.5$			
Δt	S_λ	S_{corr}	S_{visual}	S_{new}	S_λ	S_{corr}	S_{visual}	S_{new}
$\Delta t = 0.01$	841	631	850	841	88	86	90	114
$\Delta t = 0.05$	168	126	180	168	18	16	18	21
$\Delta t = 0.1$	84	63	90	80	9	7	8	9
$\Delta t = 0.2$	41	31	45	41	2	1	4	1

Table 9.5: Stopping times for several test problems

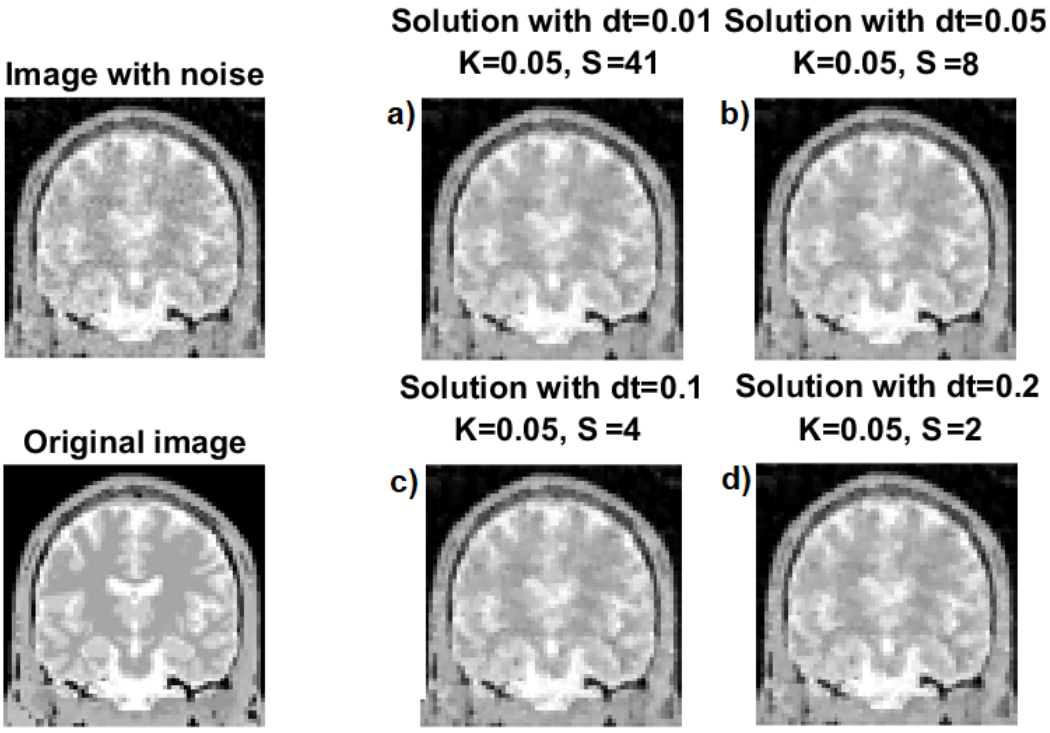

Figure 9.13: Result for Ella, $K = 0.05$, for stopping time S_{new} (in image S)

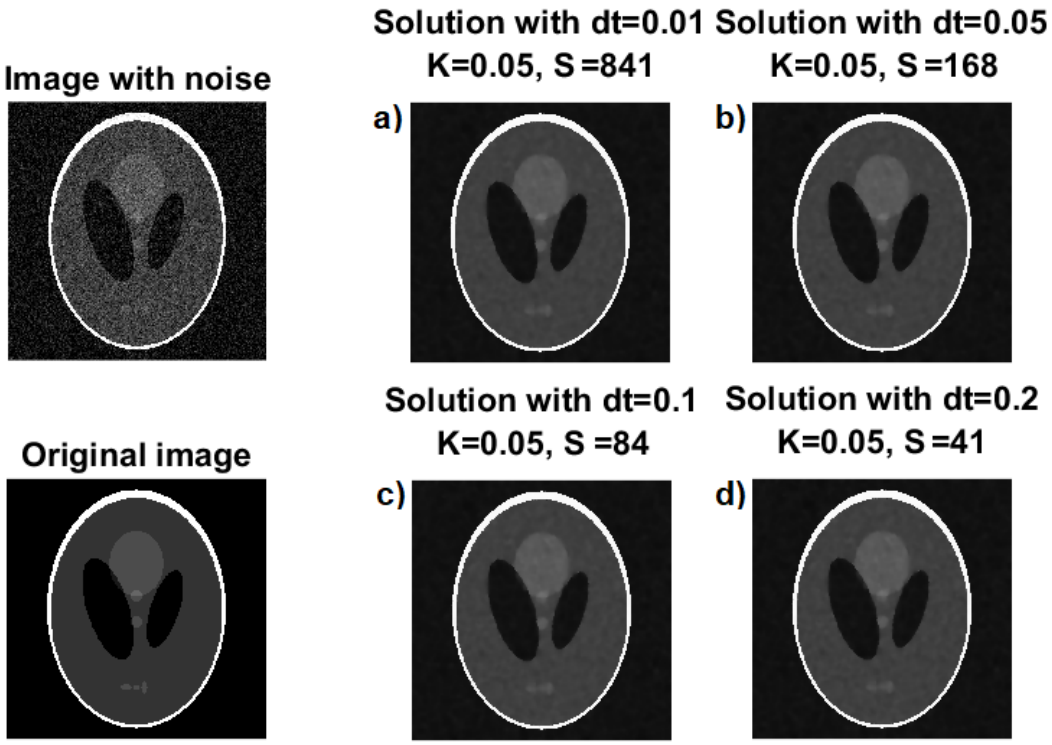

Figure 9.14: Result for Shepp-Logan, $n = 400$, $K = 0.05$, for stopping time S_{new} (in image S)

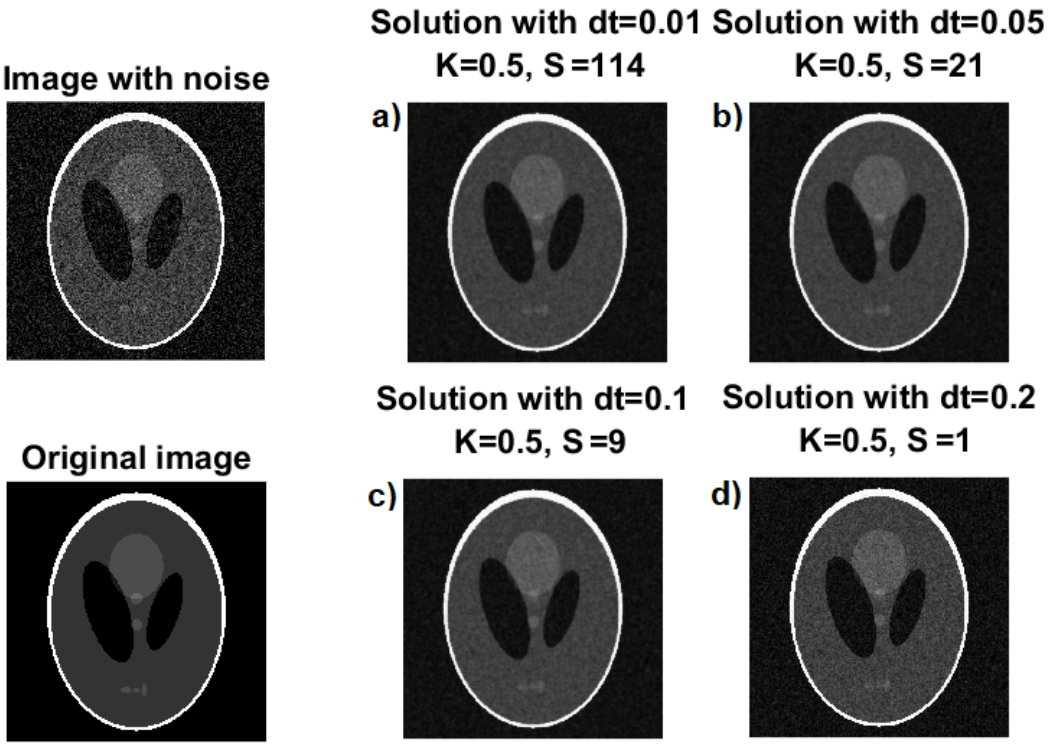

Figure 9.15: Result for Shepp-Logan, $n = 400$, $K = 0.5$, for stopping time S_{new} (in image S)

9.4. SUMMARY OF RESULTS

9.4.1. PARAMETER K

To determine which method is the most useful for finding K we need to look at the value $\Delta t \cdot T$. We have seen in the figures that the optimal K intersects with the average K of one of the methods at some value for $\Delta t \cdot T$. Where this intersection point is, is depending on many factors. One of these factors is the value of n. For low values of n this intersection is at a much lower value of $\Delta t \cdot T$ than for larger values of n. For the first method, the noise estimator method, the intersection point was in the region of $\Delta t \cdot T = 0.5$ for small n and in the region of $\Delta t \cdot T = 10$ for high n.
For the second method, the MAD method, the average value of K was much lower than for the first method. This leads to a much later intersection between this K_{avg} and K_{opt}. Therefore a much higher $\Delta t \cdot T$ is necessary for a correct estimation. This causes the calculation time to be much higher than for the first method, which makes this method less suitable for the estimation of K.
The third method, the p-norm method, was the worst of the tested methods. First of all, the method depends on two parameters, σ and p, and both were unknown. This made it hard to find the correct estimation. But the biggest disadvantage was that even though the article stated that this method leads to a decreasing value of K, the value remained almost constant. This means that it is basically similar to manually choosing a constant, but in a more difficult manner due to the unknown values of σ and p.

9.4.2. TIME STEP Δt

For the Perona-Malik method we looked at the behaviour of Δt. We noticed that the method discussed in the theory showed a behaviour opposite to the behaviour of the optimal Δt. The optimal value for Δt starts for a low number of iterations at the maximal possibility for stability, $\Delta t = 0.25$, and slowly decreases to a value of zero for a larger number of iterations. The method starts at a value of $\Delta t = 0.0625$ for a low number of iterations and increases to the value of $\Delta t = 0.25$.
To create a method with the same behaviour as the optimal value of Δt, the method mentioned in the theory was modified slightly. This new method showed good results for the Shepp-Logan phantom. For the Ella phantom both methods did not estimate Δt correctly and lead to blurry results. Overall the new method seems like the better option. This new method is given by:

$$\Delta t_{mod}^{(i)} = \frac{1}{\alpha}\left(a - b\exp\left(-\max\left(\frac{|\mathrm{Re}(\partial u^{(i)}/\partial t|)}{\mathrm{Re}(u^{(i)})}\right)\right)\right) \tag{9.9}$$

In an earlier chapter we noticed that the ill-posed and well-posed method were unstable for certain combinations of parameters. Therefore an adaptive time step was introduced for these methods, which decreases Δt when the solution starts to diverge. This time step is depending on a parameter q. This q should be chosen as low as possible, but if q is too low the solution becomes unstable. Since this stability domain is also depending on many factors, finding the optimal value of q can be difficult. However, for the correct value of q the results are very good and there exists no more instability.

9.4.3. NUMBER OF TIME STEPS T

The number of time steps at which the diffusion is ended is named the stopping time S. Three estimators for S have been tested and compared. One of these methods used the median absolute deviation (MAD) of $u(t) - u_{original}$. The biggest disadvantage of this method is that the original image is needed, which is usually not the case. Therefore this method is not very useful in practice. The second method, S_{corr}, uses the correlation coefficient of $u(0) - u(t)$ with $u(t)$ and is only depending on the noisy image and the filtered image. This method showed promising results. However, this stopping time usually stopped too early, leading to a suboptimal image.

The third method discussed in the theoretical chapter, chapter 8, is depending on parameter λ. Unfortunately this λ was varying quite a lot. The average λ was around a value of 0.02. Using this λ could lead to a good result, but also caused under- and overdiffusion of the image.
A new stopping time S_{new} was introduced. This stopping time is a combination of the second and the third method and leads to either better or equally good results. Therefore this new stopping time S_{new} seems to be the better option. It is given by:

$$S_{new} = [(4 \cdot \min(S_{corr}, S_{\lambda}))/3]_{floor} \tag{9.10}$$

9.4.4. RECOMMENDATION

After the investigation of the behaviour of these three parameters, the conclusion is that all these values are influencing each other. Therefore it is hard to come up with an optimal choice for K, Δt and S at the same time. Based on all the results discussed in this chapter, we would recommend to choose a contant value for K and Δt and decide on a stopping time to find the best solution. Having a varying K, Δt and stopping time would only make it harder to find the optimal outcome. So choose for example $K = 0.05$ and $\Delta t = 0.1$ and use stopping time S_{new} to obtain the filtered result. This should lead to good outcomes.

10

SEGMENTATION

The overall goal of the methods discussed in the previous chapters is to remove noise in MRI images to acquire a final image in which different structures can be identified. A correct identification of structures is important, since this can lead to the detection of abnormalities. The differentiation between structures can be achieved with image segmentation methods. These methods partition an image into several regions, each region hopefully coinciding with a different structure. This chapter will discuss a segmentation method called the seeded region growing method. A modified version of this method is used on some of the previously obtained images to make a better comparison between the results.

10.1. SEEDED REGION GROWING

In [19] the seeded region growing method is described. The method starts with a number of seeds, which have been grouped into n sets, $A_1, A_2, \ldots, A_n$. In each step of the algorithm one pixel is added to one of the sets. To determine which pixel is added to a set, let T be the set of all so far unallocated pixels which border at least one region:

$$T = \left\{ x \notin \bigcup_{i=1}^{n} A_i | N(x) \cap \bigcup_{i=1}^{n} A_i \neq \emptyset \right\} \tag{10.1}$$

with $N(x)$ the set of immediate neighbours of pixel x. In the article the neighbours are chosen as those being 8-connected to pixel x.
If x neighbours only one region, then $i(x)$ is the index of this neighbouring region. A measure for the difference between x and the adjoining region $A_{i(x)}$ is given by:

$$\delta_{SRG}(x) = |u(x) - \underset{y \in A_{i(x)}}{\text{mean}}(u(y))| \tag{10.2}$$

with $u(x)$ the gray value of pixel x.
If x borders multiple regions, then $i(x)$ is the index of the adjoining region with the smallest $\delta_{SRG}(x)$. These pixels can be classified as boundary pixels and added to a set of already-known boundary pixels B.
The last step is to take $z \in T$ such that

$$\delta_{SRG}(z) = \min_{x \in T} \{\delta_{SRG}(x)\} \tag{10.3}$$

and add z to $A_{i(z)}$. The process is repeated until all pixels have been allocated.

The method used to compare the obtained results is a modified version of this seeded region growing method. First of all, we are only interested in the region belonging to one specific seed point. Therefore there is only one region A. The set T is now the set of all unallocated pixels which border this one set A:

$$T = \{x \notin A | N(x) \cap A \neq \emptyset\} \tag{10.4}$$

with $N(x)$ the neighbours of pixel x. In the used algorithm the neighbours of pixel x are considered the four pixels on the left/right and above/below x, not the 8-connected pixels.

The same measure $\delta_{SRG}(x)$ is used:

$$\delta_{SRG}(x) = |u(x) - \underset{y \in A}{\text{mean}}(u(y))| \tag{10.5}$$

The pixel $x \in T$ with the smallest $\delta_{SRG}(x)$ is added to the region. However, the process is not repeated until all pixels have been allocated, but until the difference between each neighbouring pixel value and the region mean have reached a certain threshold τ_{SRG}. This threshold value τ_{SRG} has a lot of influence on the outcome of the region. To find the correct value of τ_{SRG} a few test runs can be executed on different images and known structures. The results are discussed in the next chapter on numerical experiments.

11

NUMERICAL EXPERIMENTS: SEGMENTATION

As mentioned in the previous chapter seeded region growing has been used to segment the images. This method uses a threshold τ_{SRG} to be able to make a distinction between regions. This τ_{SRG} is important to segment the image correctly: if it is too low the region will be too small, and if it is too high the region will contain multiple structures.

First we want to investigate how to determine a correct value for τ_{SRG}. The segmentation method has been tested for different test problems. Most images have been obtained by using FTCS implementation and the stopping time S_{new}, described in chapter 9 by 9.8, for $\Delta t = 0.05$ and $K = 0.05$. Two images were given and are images of a normal MRI and an MRI of a hydrocephalus head (figure 11.1).
Manually the optimal values of τ_{SRG} have been determined for six test problems and they can be found in the following table (table 11.1).

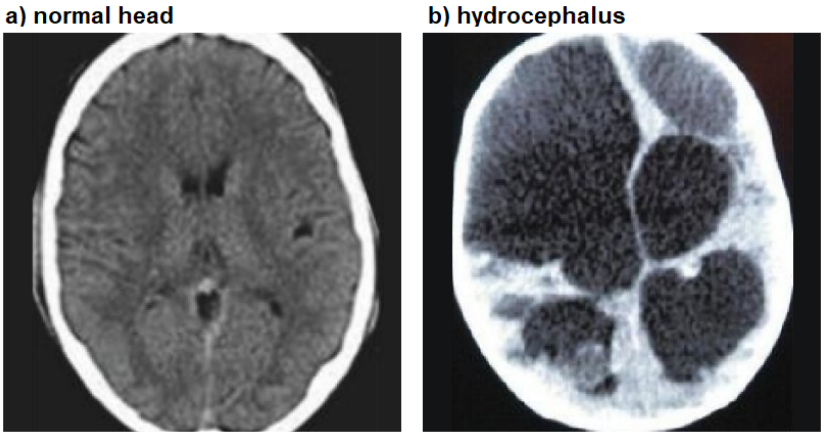

Figure 11.1: Image of a) normal head and b) hydrocephalus

test problem	*SL*, = 400	*SL*, = 200	*SL*, $n = 64$	*Ella*	*Normal head*	*Hydrocephalus*
optimal threshold	0.030	0.050	0.040	0.075	0.030	0.080
mean($\|\nabla u\|$)	0.020	0.036	0.124	0.123	0.032	0.059
median($\|\nabla u\|$)	0.005	0.007	0.034	0.099	0.028	0.049

Table 11.1: Median for different test problems

To understand how to find the optimal value for τ_{SRG} we need to look into some image properies. The normal head given in figure 11.1a has some structures, which are very close in pixel value to their surrounding tissue. To make a distinction between the structure and its surroundings we need a low value of τ_{SRG}. In the second image the structures are much darker than their surrounding tissue, therefore τ_{SRG} can be higher. We can conclude that the difference in pixel value has a high influence on the optimal τ_{SRG}. It seems logical to assume a low average pixel gradient leads to a low optimal value, but this is not always the case.
Recall the Shepp-Logan phantom. This phantom consists of many large circles. Within these circles the gradient is probably low, leading to a low average pixel difference. However, it is not wanted that within these regions a distinction is made, which will happen if the threshold τ_{SRG} is too low. Therefore having a linear relation between the average pixel value as a parameter for τ_{SRG} does not seem to be a good assumption.

We decided to look into other possibilities for parameters using the gradient of the pixels, given by $|\nabla u|$.
Histograms of the gradient distribution for all test images were made. For values between $0-0.1$ they are given by figure 11.2. The average pixel values (mean($|\nabla u|$)) were calculated as well as the values of median($|\nabla u|$) and they can all be found in the table above (11.1). In order to get values which were truly corresponding to the image, the decision was made to only look at the middle part of the images. By doing so, the black background does not have as much influence on the values. The middle part was obtained by only taking the values of the inner $\frac{3}{5}n \times \frac{3}{5}n$ into account. This means that the outer $\frac{1}{5}n$ had been taken away in all directions.

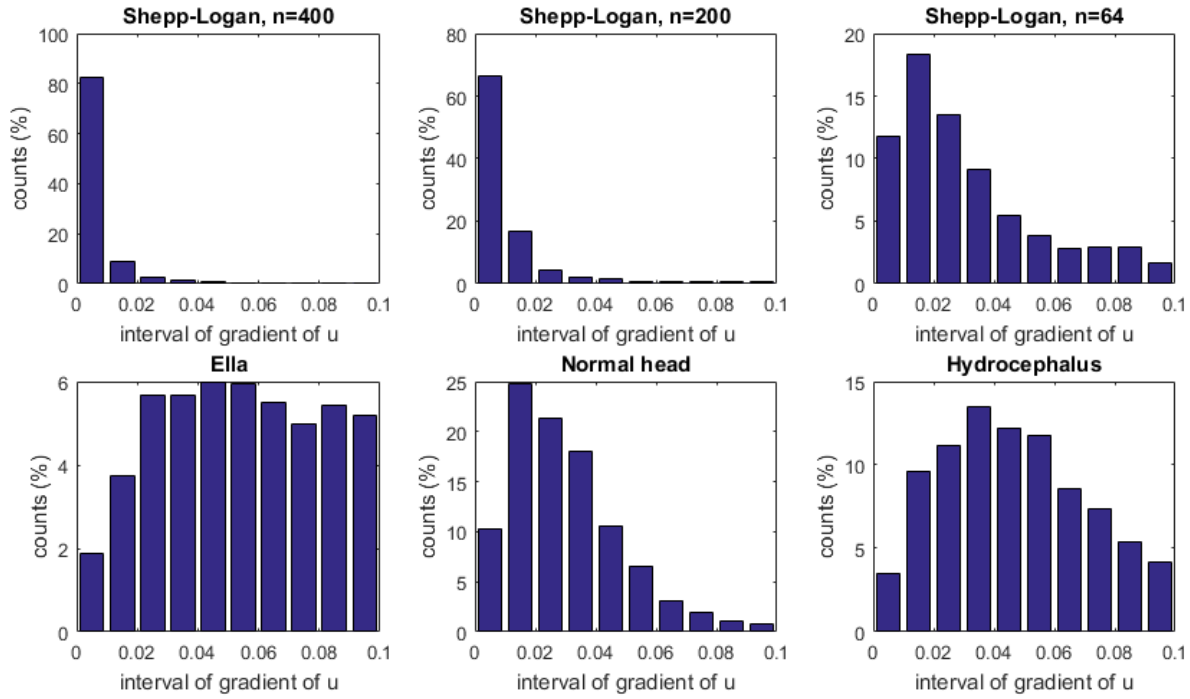

Figure 11.2: Histograms for $\nabla u = 0-0.1$ for a) Shepp-Logan, $n = 400$ b) Shepp-Logan, $n = 200$ c) Shepp-Logan, $n = 64$ d) Ella e) Normal head f) Hydrocephalus

From table 11.1 we can immediately see that there is no linear relation between the average of $|\nabla u|$ and the threshold. There is also no linear relation between median($|\nabla u|$) and the threshold. However, the median is the stronger of the two averages, since it is corresponding to the behaviour of the histograms. And to know where the bulk of the pixel difference values is located seems of more importance. Therefore we look further into the relation between the median and the threshold τ_{SRG}.
The values can be interpreted as follows. The distribution for the highly detailed Shepp-Logan phantoms are oriented to the far left, stating that most of their pixel differences are very small (in the region of $0-0.01$). The median values for these two test problems are also very small, lower than 0.01. This can be caused by the fact that the phantom consists of many large regions. Hopefully the image has very small pixel gradients inside these regions, leading to a very low median and a far left distribution. Having a very low threshold will cause segmentation within these large regions, which is suboptimal. Therefore we need τ_{SRG} to be low, but slightly higher than is expected from the low median.
The orientation of the normal head distribution and the less detailed Shepp-Logan is more to the right. The median is also higher, it is now 0.028 and 0.035 respectively. This means that the image does not have many very small gradient values (< 0.01), meaning that probably no large homogeneous structures are present. But the gradient values are sufficiently low ($0.01-0.04$) to assume that different structures with almost equal gray tones are present. In that case we want a lower threshold τ_{SRG}.
The remaining two test problems have a median higher than 0.045 and their distributions are also further to the right. This means that the structures in these regions are either not homogeneous or have surrounding tissue of a very different gray tone. In both cases we need the threshold value to be higher.
This interpretation seems logical and is in accordance with the optimal threshold values. However, no conclusions can be made about the median to obtain exact values for τ_{SRG}.

The following images show the segmentation results for all test problems. All images show three different structures, resulting from three different seed points. For Ella, results for thresholds other than the optimal τ_{SRG} are also shown.

We see in the first image that the correct value of τ_{SRG} leads to a very good segmentation, close to the segmentation of the original. However, changing the threshold only slightly causes undersegmentation (11.3c) or oversegmentation (11.3d).
The results for the Shepp-Logan phantom with a high number of pixels ($n = 400$) are almost similar to segmentation of the original. Unfortunately this is not the case for the other two Shepp-Logan images. Both have a problem with the structure marked with a red cross. The value of τ_{SRG} has to be lower to segment this part correctly. However, lowering the threshold causes the blue region to be much smaller, which is also problematic.
The normal and hydrocephalus head both show a very good partitioning with their optimal threshold τ_{SRG}.

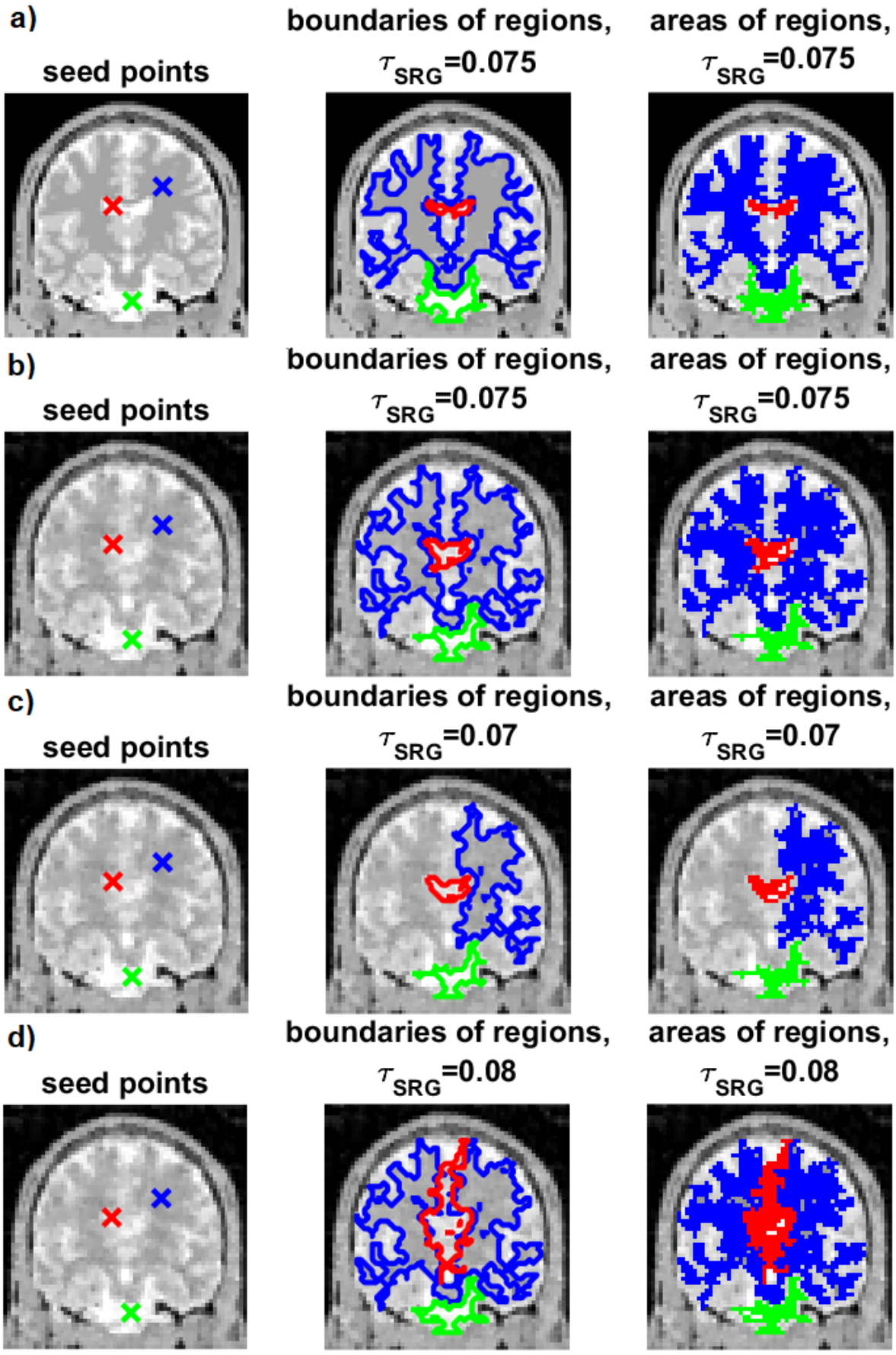

Figure 11.3: Segmentation results of Ella, a) original and image obtained with stopping method S b) optimal $\tau_{SRG} = 0.075$ c) $\tau_{SRG} = 0.07$ d) $\tau_{SRG} = 0.08$

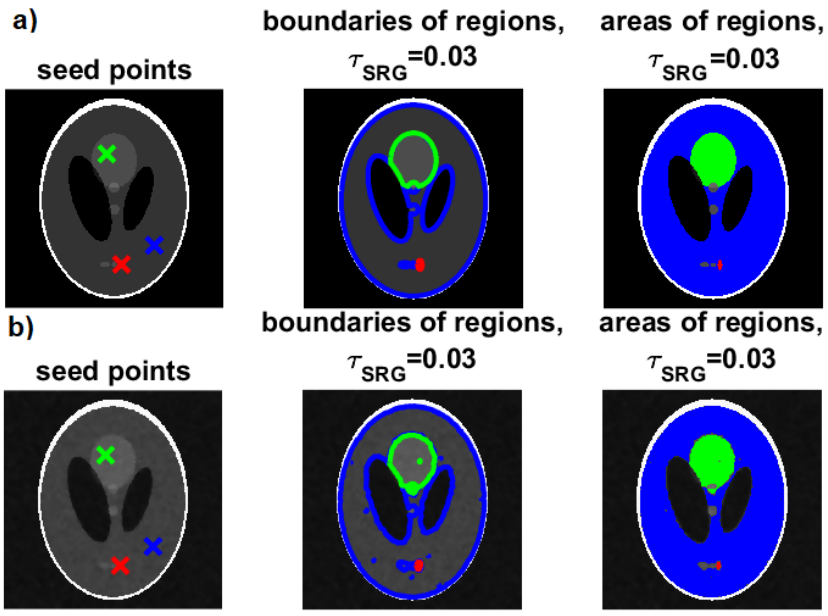

Figure 11.4: Segmentation results of Shepp-Logan, $n = 400$, for a) original b) image obtained with stopping method S

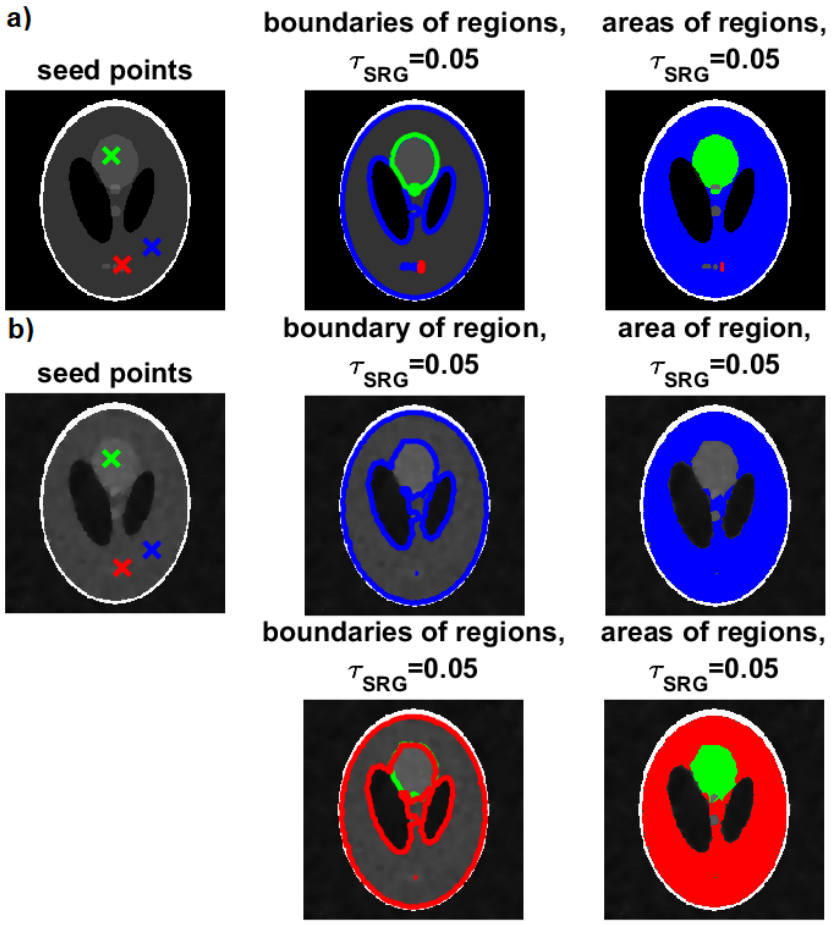

Figure 11.5: Segmentation results of Shepp-Logan, $n = 200$, for a) original b) image obtained with stopping method S

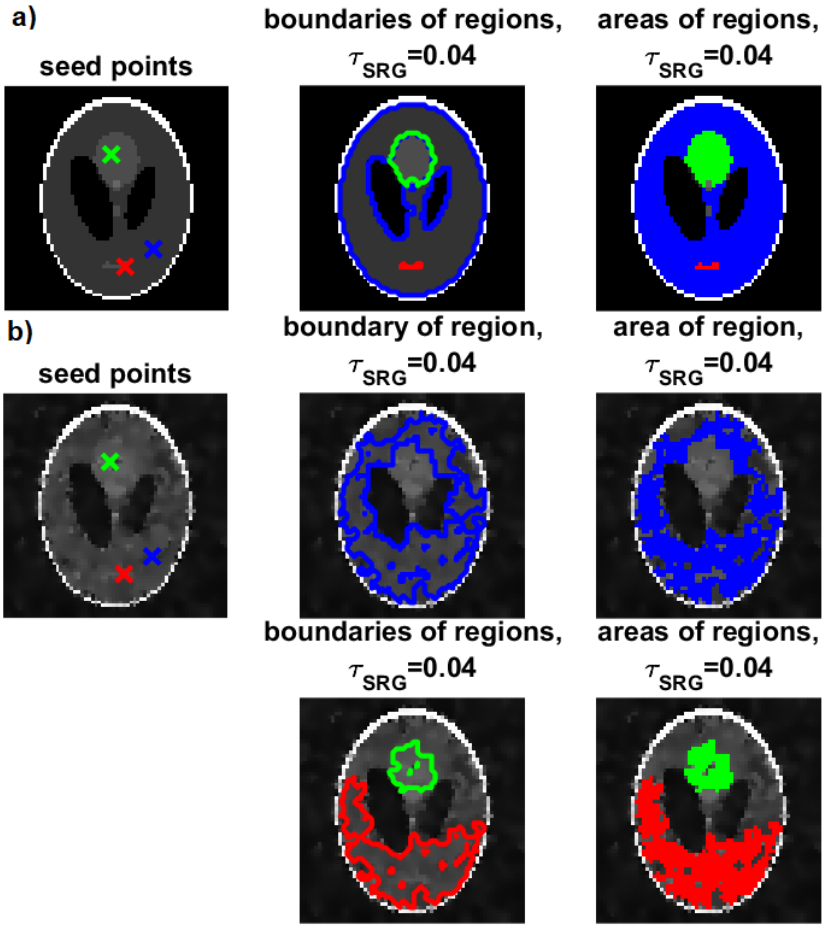

Figure 11.6: Segmentation results of Shepp-Logan, $n = 64$, for a) original b) image obtained with stopping method S

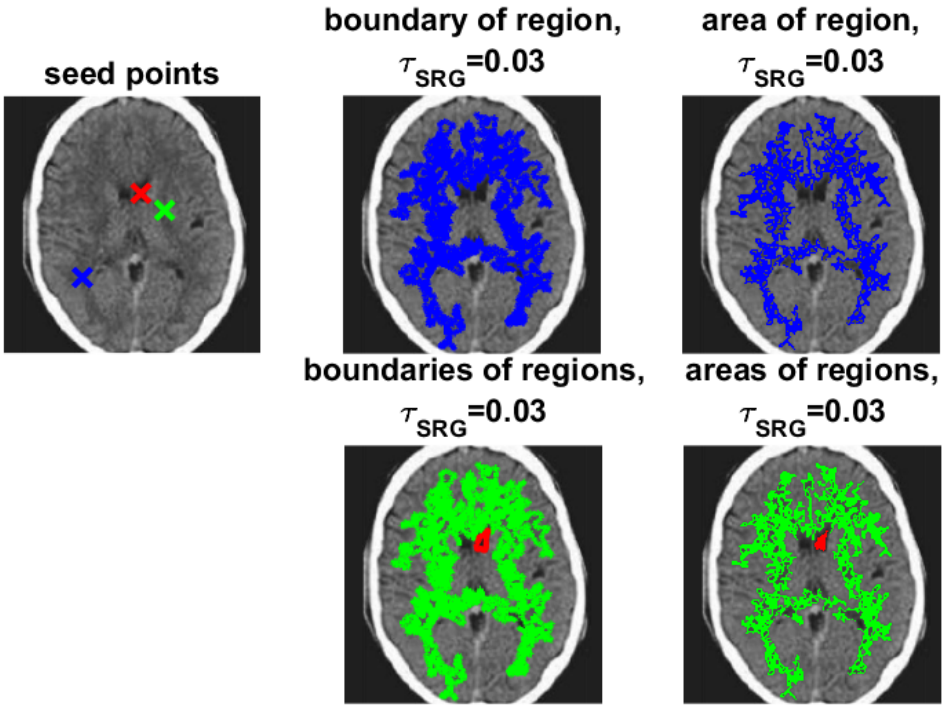

Figure 11.7: Segmentation results of the normal head image

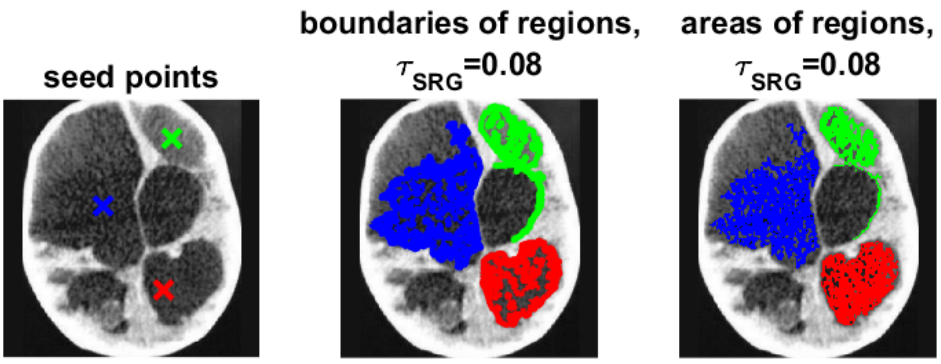

Figure 11.8: Segmentation results of the hydrocephalus image

12

CONCLUSIONS AND FURTHER RESEARCH

The main goal of this thesis consisted of three subgoals. The first was to give an overview of diffusion filtering methods and to implement these methods for comparison while using two numerical methods. First the most basic linear diffusion filtering method, also known as the heat equation, was discussed. Since the biggest disadvantage of this method is the homogeneous blurring, other nonlinear diffusion filtering methods were invented. By using diffusion functions instead of a constant the blurring of edges can be eliminated and edge sharpening can be achieved. An example of such anisotropic diffusion filtering methods are the Perona-Malik methods. These two methods satisfy an important threshold property, which leads to intraregional blurring and edge preservation. However, a downside of these methods is their ill-posedness. Fortunately, this ill-posedness can be reduced by adding a Gaussian kernel to the methods. It is also possible to choose a well-posed anisotropic diffusion filtering method, even though it does not satisfy the threshold property. All these methods were implemented with both the FTCS implementation as well as the AOS implementation. The advantage of using the AOS method is its unconditional stability for Δt, while the FTCS method is only stable for $0 < \Delta t \leq 0.25$.
The results of the implementation of all these methods lead to a few conclusions. In all test problems the Perona-Malik methods lead to the best results. Sometimes the addition of a Gaussian kernel improved the results even more. This is exactly as predicted by the literature. The AOS implementation usually leads to slightly better results compared to the FTCS method and showed less instability. The ill-posed and well-posed method often showed instability when implemented with the FTCS method. This was probably due to the fact that the diffusion coefficient function $c(\cdot)$ was altered to deal with low values of $|\nabla u|$. Even though the AOS method had many advantages, its running time was longer than the FTCS method.

The second goal was to find methods to determine the optimal values for the gradient threshold parameter K, the time step size Δt and the stopping time S. Several options were tried in combination with a Perona-Malik method and the FTCS implementation. For parameter K we found that the noise estimator method showed the best results. However, it depends on $\Delta t \cdot T$ how well K is estimated.
For Δt an adaptive time step was used and a modification of a known method proved to be a reasonable estimator. Also, the instability for the ill-posed and well-posed method was reduced by the use of another adaptive time step.
The final parameter stopping time S had several options. One of these options depended on the original image, which is never known in practice. Two other options only depended on the noisy image and the filtered image. Both were compared with the visual optimal result and a new stopping time was introduced as a combination of the two. This stopping time lead to good results.

The final goal consisted of segmentation of some test problems. The test problems were either obtained with the new stopping time S or were two given images of a normal head MRI and a hydrocephalus MRI. The region growing method showed a good partition of all images, but was also depending on a parameter τ_{SRG}. Unfortunately, no good relation was found between properties of this image and the optimal threshold.

12.1. FURTHER RESEARCH

Of course, there is still a lot unknown about how to use diffusion filtering methods optimally. Therefore, we propose some ideas for further research.

- One of the goals of the thesis was to gain a better understanding of the parameters. Unfortunately still a lot is unknown. The parameters K, Δt and S have all been investigated and it can be concluded that these values are all influencing one another and are also influened by other factors. Therefore it is hard to find the exact optimal value. More methods can be studied or even invented to determine the optimal parameters. This would greatly improve the use of diffusion filtering, since less manual input would be necessary.
Also for the segmentation method the way of finding the threshold value τ_{SRG} can be improved. It proved difficult to find a relation between the image and this value.

- Both numerical methods showed promising results, but the AOS implementation was usually the best. Unfortunately this method is much slower than the FTCS method. A more efficient implementation of the AOS method can make it more useable.

- It is also a possibility to look at other numerical schemes. In this thesis the FTCS method with the Euler forward method was used. In [20] the Euler forward method is compared to the Euler backward method. The advantage of the Euler backward method is that this method is stable for all step sizes and it can therefore lead to good results using less time steps. In the article the results are comparable to those of the Euler forward method. In the future other schemes can be tried also.

- Since there are many other image processing techniques, it may be interesting to combine anisotropic diffusion filtering with these methods. Segmentation is one of these image processing techniques, which we have already tested in this thesis. Another option is the combination with the level-set method. This method facilitates the numerical analysis of time-varying surfaces and shapes. More information on the level-set method can be found in [21]. Combining this method with diffusion filtering methods may lead to new insights and techniques and is therefore a good starting point for future research.

- In this thesis the methods were tested on images obtained with simulations. Obviously the ultimate goal is to use these methods on real images obtained by a low-field MRI scanner, preferably the TUD/LUMC scanner. Hopefully this will be possible in the near future.

BIBLIOGRAPHY

[1] Weickert, J. (1998). Anisotropic Diffusion in Image Processing B.G. Teubner Stuttgart

[2] Perona P., Malik, J. (1990). Scale-Space and Edge Detection Using Anisotropic Diffusion IEEE Transactions on Pattern Analysis and Machine Intelligence: vol. 12, no. 7, pp. 629-639

[3] Fessler, J.A. (2010). Model-based Image Reconstruction for MRI IEEE Signal Process Magazine: vol. 27, no. 4, pp. 81-89

[4] Guerquin-Kern, M. (2011). A fast Wavelet-Based Reconstruction for Magnetic Resonance Imaging IEEE Transactions on Medical Imaging: vol. 30, no. 9, pp. 1649-1660

[5] De Leeuw den Bouter, M.L. (2017). Image Reconstruction in Low-Field MRI: a Super-Resolution Approach TU Delft, Master Thesis

[6] Vuik, C., Vermolen, F.J., Van Gijzen, M.B. (2015). Numerical Methods for Ordinary Differential Equations VSSD

[7] You, Y., Kaveh, M., Xu, W., Tannenbaum, A. (1994). Analysis and Design of Anisotropic Diffusion for Image Processing IEEE Image Processing: vol. 2, pp. 497-501

[8] You, Y., Xu, W., Tannenbaum, A., Kaveh, M. (1996). Behavioral Analysis of Anisotropic Diffusion in Image Processing IEEE Transactions on Image Processing: vol. 5, no. 11, pp. 1539-1553

[9] Catté, F., Lions, P.-L., Morel, J.-M., Coll, T. (1992). Image Selective Smoothing and Edge Detection by Nonlinear Diffusion SIAM Journal on Numerical Analysis: vol. 29, no. 1, pp. 182-193

[10] Weickert, J. (2001). Applications of Nonlinear Diffusion in Image Processing and Computer Vision Acta. Math. Univ. Comenianae: vol. 70, pp. 33-50

[11] Grochulla, M. (2007). Additive Operator Splitting Lecture slides, Saarland University
http://www.mia.uni-saarland.de/Teaching/NAIA07/naia07_e1_slides.pdf

[12] Ralli, J. (2014). PDE Based Image Diffusion and AOS PhD thesis, University of Granada

[13] Thomas, L.H. (1949). Elliptic Problems in Linear Differential Equations over a Network Watson Sci. Comput. Lab Report, Columbia University, New York.

[14] Tsiotsios, C., Petrou, M. (2013). On the choice of the parameters for anisotropic diffusion in image processing Pattern Recognition: vol. 46, no. 5, pp. 1369-1381

[15] Black, J., Sapiro, G., Marimont, D.H., Heeger, D. (1998). Robust Anisotropic Diffusion IEEE Transactions on Image Processing, vol. 7, no. 3, pp. 421-432

[16] Voci, F., Eiho, S., Sugimoto, N., Sekiguchi, H. (2004). Estimating the Gradient Threshold in the Perona-Malik Equation IEEE Signal Processing, pp. 39-46

[17] Bernandes,R., Maduro, C., Serranho, P. (2010). Improved adaptive complex diffusion despeckling filter Optics Express, vol. 18, no. 23, pp. 24048-24059

[18] Mrázek, P., Navara, M. (2003). Selection of Optimal Stopping Time for Nonlinear Diffusion Filtering International Journal of Computer Vision, vol. 52, no. 2-3, pp. 189-203

[19] Adams, R., Bischof, L. (1994). Seeded Region Growing International IEEE Transactions and Pattern Analysis and Machine Intelligence, vol. 16, no. 6, pp. 641-647

[20] Barash, D. (2001). One-Step Deblurring and Denoising Color Images Using Partial Differential Equations Hewlett Packard Labs Technical Reports

[21] Sethian, J.A. (2008). Level Set Methods and Fast Marching Methods Cambridge University Press