Exponential fitting for stripe noise reduction from dental x-ray images

Anssi Koskinen

Supervisor: Samuli Siltanen
Advisor: Henrik Lohman

University of Helsinki

August 25, 2020

Abstract

The applied mathematical field of inverse problems studies how to recover unknown function from a set of possibly incomplete and noisy observations. One example of real-life inverse problem is image destriping, which is the process of removing stripes from images. The stripe noise is a very common phenomenon in various of fields such as satellite remote sensing or in dental x-ray imaging.

In this thesis we study methods to remove the stripe noise from dental x-ray images. The stripes in the images are consequence of the geometry of our measurement and the sensor. In the x-ray imaging, the x-rays are sent on certain intensity through the measurable object and then the remaining intensity is measured using the x-ray detector. The detectors used in this thesis convert the remaining x-rays directly into electrical signals, which are then measured and finally processed into an image. We notice that the gained values behave according to an exponential model and use this knowledge to transform this into a nonlinear fitting problem. We study two linearization methods and three iterative methods. We examine the performance of the correction algorithms with both simulated and real stripe images.

The results of the experiments show that although some of the fitting methods give better results in the least squares sense, the exponential prior leaves some visible line artefacts. This suggests that the methods can be further improved by applying suitable regularization method. We believe that this study is a good baseline for a better correction method.

Preface

I would like to thank professor Samuli Siltanen from the University of Helsinki for supervising this thesis. This master's thesis was done at Direct Conversion Ltd. Oy and I would like to thank the company for giving me an opportunity to work there as a thesis worker from 14.5.2019-31.1.2020. I would like to thank my advisor Henrik Lohman for providing the topic of the thesis and for giving me guidance and feedback during my employment. Additionally, I would like to thank Mikael Platan and my colleagues at the research team for assisting me.

Contents

1 Introduction

1.1 Inverse problems and noise reduction

Inverse problems rise from the need to evaluate parameters that we cannot directly observe. Inverse problems can be seen as a reverse of forward problems. Consider an equation

$$m = A(f),$$

where m is the measurement, f denotes some unknown function and A some operator describing the relation between the unknown and the measurement. Here, whereas the forward problem would be determining m given A and f, the respective inverse problem is determining f, given m and A.

The main issue with inverse problems can be easily seen in the following example: consider a forward problem of computing the sum of two numbers a and b. It is easy to see that the corresponding inverse problem "given the sum of two numbers, determine the input values a and b" has infinitely many solutions. This is an example of an ill-posed problem in the Hadamard's sense. Hadamard's conditions for well-posed problem are

1. *Existence.* The solution exists.
2. *Uniqueness.* There is at most one solution.
3. *Stability.* The solution depends continuously from the data.

The take away from this example is that to guarantee well-posedness, some kind of additional knowledge, which we call prior knowledge, is sometimes required.

1.2 Basics of x-ray imaging

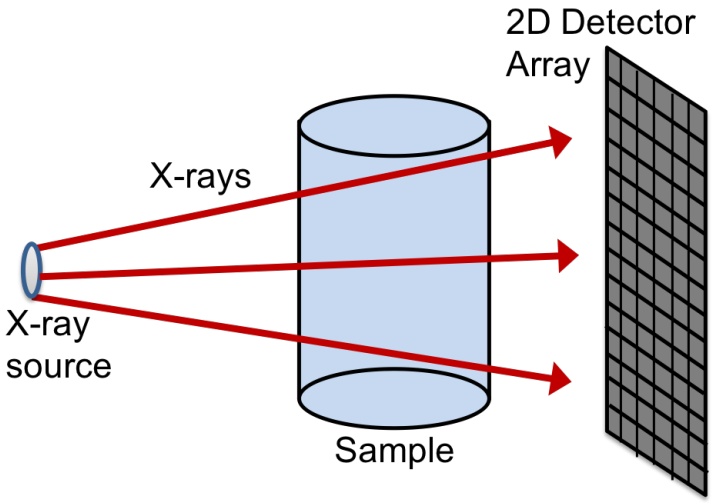

Figure 1: Example of the x-ray imaging experimental setup [1].

In the x-ray imaging, the x-rays are sent on certain intensity through the measurable object and then the remaining intensity is measured using the x-ray detector. The idea is that different materials absorb x-rays with different efficiencies, letting different amounts of x-rays to reach the detector. In mathematical way we can describe the measurable object with some unknown function $f : B \to \mathbb{R}_{\geq 0}$ with compact support, where $B \subset \mathbb{R}^2$ is the imaging area. The function at a point $\mathbf{x} = (x_1, x_2) \in B$ corresponds the intensity loss of the x-rays at that point. We assume that the x-ray source is located at the point $\mathbf{x}_0$ and the ray arrives on a detector pixel at point $\mathbf{x}_1$. We denote the initial intensity and the measured intensity by $I(\mathbf{x}_0) = I_0$ and $I(\mathbf{x}_1) = I_1$ respectively. The relative loss in the intensity I of a narrow x-ray along the line $\mathbf{s} = \mathbf{s}(t), t \in [0, 1]$ from $\mathbf{s}(0) = \mathbf{x}_0$ to $\mathbf{s}(1) = \mathbf{x}_1$ is given by the formula

$$\frac{\Delta I(\mathbf{s})}{I(\mathbf{s})} = -f(\mathbf{s})\Delta\mathbf{s}, \tag{1.1}$$

By taking the line integral along the path $\mathbf{s}$, we get

$$\int_{\mathbf{s}} f(\mathbf{s})d\mathbf{s} = -\int_{\mathbf{s}} \frac{1}{I(\mathbf{s})}\frac{d}{d\mathbf{s}}I(\mathbf{s})d\mathbf{s} = -\left[\log I(\mathbf{s}(t))\right]_0^1 = \log I_0 - \log I_1. \tag{1.2}$$

By rearranging the terms we the equation known as *Beer-Lambert law for monochromatic x-rays*:

$$I_1 = I_0 \exp\left(-\int_{\mathbf{s}} f(\mathbf{s})d\mathbf{s}\right). \tag{1.3}$$

It is common to represent this equation of the form

$$I_1 = I_0 e^{-\mu \Delta \mathbf{s}}, \tag{1.4}$$

where $\mu := \int_s f(s)$ is known as the attenuation coefficient.

In computed tomography, the inverse problem is to recover the function f of an object of interest given the initial intensity I_0, the measured intensity I_1 and the path $\mathbf{s}$. To satisfy the Hadamard's conditions, several measurements from multiple angles of view is required. Additionally, in dental imaging, we want to minimize the x-ray doses exposed to the patient. Creating a suitable reconstruction algorithm while balancing between these kinds of requirements is a very interesting subject in and on itself. For more information we refer to book *Linear and nonlinear inverse problems with practical applications* by Jennifer L. Mueller and Samuli Siltanen [2].

1.3 The problem description

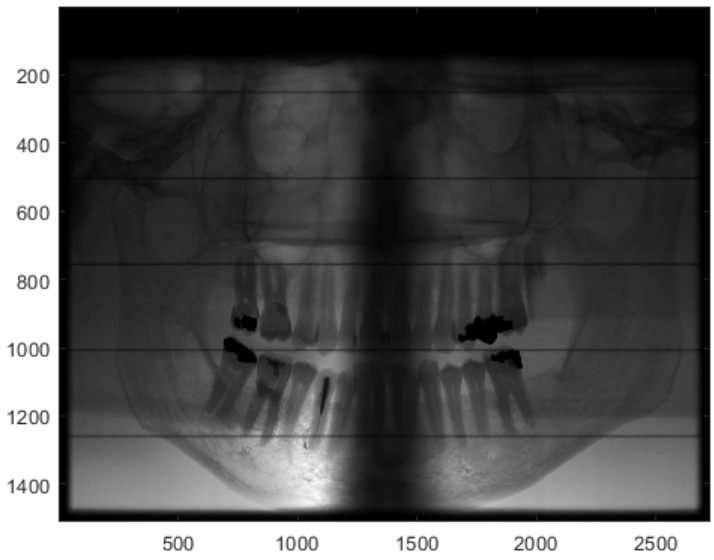

Figure 2: Example image with stripe noise

Noise reduction is other example of an inverse problem. In noise reduction, the wanted signal f is influenced by a partially known process A, giving the noisy measurement of the signal m. The goal in noise reduction is to reverse the effects of that process. In this thesis we go through methods for reducing the so-called stripe noise from the dental x-ray images. The stripe noise is a very

common phenomenon in various of fields such as satellite remote sensing [3], or in our case, dental x-ray imaging.

In recent years, several destriping methods have been proposed. These methods can roughly be divided into three categories: filtering-based, optimization-based and statistics-based methods. In the filtering-based methods, the stripe noise is reduced by constructing filter on a transformed domain, such as the Fourier transform [4] and wavelet analysis [5]. These methods perform very well when the stripe noise is periodic, however, the filters may also affect the structural details with the same frequencies as the stripes.

In the optimization-based methods, the destriping is regarded as an ill-posed inverse problem. For example, Yong Chen et al. proposed a denoising algorithm for remote sensed images using total variation and group sparsity constraint [6], and Min Wang et al. proposed a unidirectional total variation and second order total variation model for destriping [7]. While these methods perform quite well most of the time, they do not consider the characteristics of the stripes themselves. This means that these methods may obtain favorable results for specified images, and might not work in generic case.

Statistics-based methods are based on the statistical properties of the detector response. Typical examples for statistics-based methods are moment matching [8] and histogram matching [9]. These methods attempt to remove the stripes by matching the mean and standard deviation or histogram of an uncalibrated signal to the reference signal. The method we use in this thesis belongs to this category.

The stripes in the images are consequence of the geometry of our measurement and the sensor itself. The detectors used in this thesis convert the remaining x-rays directly into electrical signals using the material cadmium telluride (CdTe), which are then measured and finally processed into an image. The detectors are built by lining up rectangular hybrids of CdTe such that a gap is left between the panels. Since the gap area doesn't measure anything, it leaves stripes along the measurement direction. Additionally, the response of the pixels near the gaps is nonlinear, and might also change during the course of time, thus making the calibration difficult. We notice that the gained values behave according to the exponential model, which we will show in the section 2.2. We use this prior knowledge to try to improve the existing gap correction method.

2 Materials and Methods

2.1 The image observation model

We represent the relation between the noisy image $I_n \in \mathbb{R}^{n\times m}$, the gain matrix $g \in \mathbb{R}^{n\times m}$ and the true image $I \in \mathbb{R}^{n\times m}$ as

$$I_n(i,j) = g(i,j)\left[I(i,j) + \varepsilon(i.j)\right], \tag{2.1}$$

where ε denotes all the additional noise which we assume to be Gaussian. In this thesis, we are only interested in removing the effects of the gain matrix g, hence we consider pixel (i,j) healthy, if $g(i,j) = 1$. We also omit the processing of the gap areas, where we assume that $g(i,j) = 0$. The gain matrix near the gap edges behave according to exponential function

$$g(i,j) = 1 + d_0 A^{|i-i_0|}, \tag{2.2}$$

where $d_0 \in (-1,1)$ is the gain-loss at the edge-most row i_0, and $A \in (0,1)$ is the rate of which the gain normalizes as we get further from the edge. Since

$$g(i,j) \to 1, \text{ as } |i-i_0| \to \infty, \tag{2.3}$$

we assume that there exists $k \in \mathbb{N}$ s.t.

$$g(i,j) = 1, \text{ when } |i-i_0| \geq k. \tag{2.4}$$

For simplicity, we consider only areas below the gaps. Now, for edge row i_0, (2.2) and (2.4) gives

$$g(i,j) = 1 + d_0 A^{i-i_0}, \text{ where } i_0 \leq i \leq i_0 + k - 1. \tag{2.5}$$

Then, we denote

$$\begin{aligned} y_{1,j} &:= g(i_0, j) \\ y_{2,j} &:= g(i_0+1, j) \\ &\vdots \\ y_{k,j} &:= g(i_0+k-1, j). \end{aligned}$$

Now (2.5) becomes

$$y_{i,j} = 1 + d_0 A^{i-1}, \text{ where } 1 \leq i \leq k. \tag{2.6}$$

Finally, due to the reference methods used, we will have a lot of deviation between the estimated gain values. For that reason, since the gain function (2.5) does not depend on the column j, we first estimate the gain value on each row separately. The estimation is done by taking median row-wise. By setting $y_i := median\left(\{y_{i,j} : 1 \leq j \leq m\}\right)$, (2.6) transforms into

$$y_i = 1 + d_0 A^{i-1}, \text{ where } 1 \leq i \leq k \tag{2.7}$$

2.2 Fitting the exponential function

2.2.1 Linearization methods

Common way to fit the exponential model is to linearize the model. One way to do this is to use logarithm. Subtracting one and taking the logarithm on (2.7) yields a linear model

$$y_i^l := \ln(y_i - 1) = \ln(d_0) + \ln(A)(i-1).$$

Now we have, in matrix notation:

$$\mathbf{y} := \begin{bmatrix} y_1^l \\ y_2^l \\ \vdots \\ y_k^l \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ \vdots & \vdots \\ 1 & k-1 \end{bmatrix} \begin{bmatrix} \ln(d_0) \\ \ln(A) \end{bmatrix} =: \mathbf{X}\beta$$

We solve this by minimizing residual sum of squares

$$\hat{\beta} = \arg\min_{\beta} \left\{ \|\mathbf{y} - \mathbf{X}\beta\|^2 \right\}. \tag{2.8}$$

The minimizer $\hat{\beta}$ satisfies

$$\begin{aligned} \left. \frac{\partial \|\mathbf{y} - \mathbf{X}\beta\|^2}{\partial \boldsymbol{\beta}} \right|_{\hat{\beta}} &= 0. \\ \implies \mathbf{X}^T(\mathbf{X}\hat{\beta} - \mathbf{y}) &= 0 \\ \implies \mathbf{X}^T\mathbf{X}\hat{\beta} - \mathbf{X}^T\mathbf{y} &= 0 \end{aligned}$$

The solution of the minimization problem (2.8) is therefore

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}. \tag{2.9}$$

Other way for linearizing the gain function is to use numerical integration [10]. We assume that y_i is a discrete sample of continuous function $f : \mathbb{R} \to \mathbb{R}$ such that

$$f(i) = 1 + d_0 A^{i-1}. \tag{2.10}$$

We estimate the integral of f from 1 to $t \leq k$, $t \in \mathbb{N}$ as follows

$$\int_1^t f(i)di \approx \sum_{i=1}^{t} S_i : \begin{cases} S_1 = 0 \\ S_i = \frac{1}{2}(y_i - y_{i-1})), \text{ when } i > 1. \end{cases} \tag{2.11}$$

Applying (2.11) gives

$$\sum_{i=1}^{t} S_i = t - 1 + \frac{d_0}{\ln(A)} A^{t-1} - \frac{d_0}{\ln(A)}.$$

Multiplying both sides by $\ln(A)$ yields

$$\ln(A)\sum_{i=1}^{t} S_i = \ln(A)(t-1) + d_0 A^{t-1} - d_0.$$

Applying the measurement (2.7) and rearranging the terms gives

$$y_t - y_1 = \ln(A)(\sum_{i=1}^{t} S_i - t + 1).$$

We have, in matrix notation

$$\mathbf{y} := \begin{bmatrix} y_2 - y_1 \\ y_3 - y_1 \\ \vdots \\ y_k - y_1 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{2} S_i - 1 \\ \sum_{i=1}^{3} S_i - 2 \\ \vdots \\ \sum_{i=1}^{k} S_i - k + 1 \end{bmatrix} \ln(A) =: \mathbf{X}\ln(A). \tag{2.12}$$

The least squares estimate of A is now

$$\hat{A} = \exp\left[(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}\right]. \tag{2.13}$$

Now applying (2.13) to (2.7) yields

$$\mathbf{y} := \begin{bmatrix} y_1 - 1 \\ y_2 - 1 \\ y_3 - 1 \\ \vdots \\ y_k - 1 \end{bmatrix} = \begin{bmatrix} 1 \\ \hat{A} \\ \hat{A}^2 \\ \vdots \\ \hat{A}^{k-1} \end{bmatrix} d_0 =: \mathbf{X}d_0.$$

And the least squares estimate of d_0 is

$$\hat{d_0} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}. \tag{2.14}$$

2.2.2 Non-linear system

Next we look at iterative least squares methods. In these methods we minimize the residual sum of squares iteratively through a sequence of updates to parameter values. From (2.7), we have

$$f(i, \beta) := 1 + d_0 A^{i-1}, \text{ where } \beta := [d_0, A]^T.$$

We set

$$\mathbf{y} := \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix} \text{ and } \mathbf{f}(\beta) := \begin{bmatrix} f(1,\beta) \\ f(2,\beta) \\ \vdots \\ f(k,\beta) \end{bmatrix} \tag{2.15}$$

Like in (2.8), we search the solution by minimizing the least squares

$$\hat{\beta} = \arg\min_{\beta}\{\|\mathbf{y} - \mathbf{f}(\beta)\|^2\}$$

We denote

$$\mathbf{r}(\beta) := \mathbf{y} - \mathbf{f}(\beta), \text{ and } l(\beta) := \|\mathbf{r}\|^2. \tag{2.16}$$

To optimize (2.16), we want to generate a sequence of parameters $\beta_1, \beta_2, \ldots$ from an initial guess β_0 such that

$$l(\beta_0) \geq l(\beta_1) \geq l(\beta_2) \geq \ldots \tag{2.17}$$

2.2.3 The gradient descent

Let $s \in \mathbb{R}^{2\times 1}$ and l be infinitely differentiable. Then the Taylor series of l is

$$l(\beta + s) = l(\beta) + [\nabla l(\beta)]^T s + \frac{1}{2} s^T \nabla^2 l(\beta) s + \ldots, \tag{2.18}$$

where $\nabla l(\beta)$ denotes the gradient vector given by

$$\nabla l(\beta) = 2\left[\frac{\partial \mathbf{r}(\beta)}{\partial \beta}\right]^T \mathbf{r}(\beta) \tag{2.19}$$

and $\nabla^2 l(\beta)$ the matrix of second derivatives, so called Hessian matrix computed by

$$\nabla^2 l(\beta) = 2\left(\left[\frac{\partial \mathbf{r}(\beta)}{\partial \beta}\right]^T \frac{\partial \mathbf{r}(\beta)}{\partial \beta} + [\mathbf{r}(\beta)]^T \frac{\partial^2 \mathbf{r}(\beta)}{\partial^2 \beta}\right) \tag{2.20}$$

We set

$$J_{\mathbf{r}} = \frac{\partial \mathbf{r}(\beta)}{\partial \beta} \text{ and } Q_{\mathbf{r}} = [\mathbf{r}(\beta)]^T \frac{\partial^2 \mathbf{r}(\beta)}{\partial^2 \beta}. \tag{2.21}$$

Now (2.19) and (2.20) can be written in matrix notation as

$$\nabla l(\beta) = 2J_{\mathbf{r}}^T \mathbf{r}(\beta). \tag{2.22}$$

and

$$\nabla^2 l(\beta) = 2\left(J_{\mathbf{r}}^T J_{\mathbf{r}} + Q_{\mathbf{r}}\right) \tag{2.23}$$

respectively.

The gradient descent method, also known as the steepest descent method, was introduced by Cauchy in 1847 [2]. In gradient descent, we assume that the function l around β is linear. Then, if $\|s\|$ is small, the Taylor series (2.18) becomes

$$l(\beta + s) = l(\beta) + [\nabla l(\beta)]^T s. \tag{2.24}$$

We set s such that the function l decreases the fastest. The rate of change of l at the point β in a direction represented by a unit vector u is given by

$$D_u l(\beta) = \lim_{h\to 0} \frac{l(\beta + hu) - l(\beta)}{h} = \nabla l(\beta) \cdot u.$$

By the chain rule, we have

$$D_u l(\beta) = \nabla l(\beta) \cdot u,$$

where $\cdot$ denotes the dot product. To find the direction where l can be reduced the fastest, we minimize $D_u l(\beta)$ with respect to u. Since

$$\nabla l(\beta) \cdot u = \|\nabla l(\beta)\| \, \|u\| \cos(\theta).$$

where θ is the angle between $\nabla l(\beta)$ and u, and the minimum value of $\cos(\theta)$ is -1 when $\theta = \pi$, the $D_u l(\beta)$ is minimized when u is at the opposite direction of the gradient $\nabla l(\beta)$. Hence, for each $t \geq 1$, we set

$$\beta_{t+1} = \beta_t - \gamma_t \nabla l(\beta), \ \gamma_t > 0, \tag{2.25}$$

where γ_t determines the step length at iteration step t. Inserting (2.25) to (2.24) gives

$$l(\beta_{t+1}) = l(\beta_t - \gamma_t \nabla l(\beta)) = l(\beta_t) - \underbrace{\gamma_t \left[\nabla l(\beta)\right]^T \nabla l(\beta)}_{\geq 0} \leq l(\beta_t).$$

Thus; if the step lengths γ_t are small enough, the sequence of parameters $\beta_0, \beta_1, \beta_2, \ldots$ defined in (2.25) satisfy (2.17).

In the gradient descent, the step length γ_t is chosen by the so called line search rule

$$\gamma_t = \underset{\gamma \geq 0}{\arg\min}\, l(\beta_t - \gamma \nabla l(\beta_t)). \tag{2.26}$$

However, this method is known to converge slowly, therefore we use method introduced by Barzilai and Borwein in 1988. In Barzilai-Borwein, the step length γ_t is given by the rule

$$\gamma_t = \frac{\|\beta_t - \beta_{t-1}\|^2}{(\beta_t - \beta_{t-1})^T (\nabla l(\beta_t) - \nabla l(\beta_{t-1}))}. \tag{2.27}$$

2.2.4 The Gauss–Newton algorithm

The Newton method follows from the second-order Taylor approximation

$$l(\beta + s) \approx l(\beta) + \left[\nabla l(\beta)\right]^T s + \frac{1}{2} s^T \nabla^2 l(\beta) s. \tag{2.28}$$

Since (2.28) is quadratic function of s its minimum can be found by setting derivative to zero. We have

$$\frac{d}{ds}\left(l(\beta) + \left[\nabla l(\beta)\right]^T s + \frac{1}{2} s^T \nabla^2 l(\beta) s \right) = 0$$

$$\implies \nabla l(\beta) + \nabla^2 l(\beta) s = 0,$$

$$s = -\left[\nabla^2 l(\beta)\right]^{-1} \nabla l(\beta). \tag{2.29}$$

The Hessian matrix in (2.29) is not always positive definite; thus; its inverse might not exist. In Gaussian-Newton method we approximate the Hessian matrix by setting

$$Q_{\mathbf{r}} = 0.$$

Now (2.23) can be written as

$$\nabla^2 l(\beta) = 2J_{\mathbf{r}}^T J_{\mathbf{r}}. \tag{2.30}$$

By substituting (2.30) and (2.22) into (2.29) we obtain

$$s = -\left[J_{\mathbf{r}}^T J_{\mathbf{r}}\right]^{-1} J_{\mathbf{r}}^T \mathbf{r}(\beta)$$

and therefore we set

$$\beta_{t+1} = \beta_t - \left[J_{\mathbf{r}}^T J_{\mathbf{r}}\right]^{-1} J_{\mathbf{r}}^T \mathbf{r}(\beta). \tag{2.31}$$

The Gaussian-Newton method is known to converge faster than the gradient descent when the Hessian approximation(2.30) is sufficiently accurate. However, the convergence of the Gauss-Newton method is not always guaranteed. The approximation

$$|Q_{\mathbf{r}}| \ll |J_{\mathbf{r}}^T J_{\mathbf{r}}| \tag{2.32}$$

that needs to hold in order to set $Q_{\mathbf{r}}$ to zero is valid when the values in $\mathbf{r}(\beta)$ are small in magnitude, or when the values $\frac{\partial^2 \mathbf{r}(\beta)}{\partial^2 \beta}$ are relatively small.

2.2.5 The Levenberg-Marquardt-Fletcher algorithm

The last fitting method we are going to look at is modified Marquardt subroutine by R. Fletcher.[11]. The Levenberg-Marquardt algorithm varies the parameter updates between the gradient descent update and Gauss-Newton update:

$$\beta_{t+1} = \beta_t - \left[J_{\mathbf{r}}^T J_{\mathbf{r}} + \lambda_t \mathbf{I}\right]^{-1} J_{\mathbf{r}}^T \mathbf{r}(\beta_t), \tag{2.33}$$

where $\lambda_t \geq 0$ is the damping parameter and $\mathbf{I}$ is the identity matrix. The damping parameter λ_t serves for scaling purposes. For $\lambda = 0$, the method transforms into the Gauss-Newton method (2.31), while for $\lambda \to \infty$ the method approaches the gradient descent method (2.25).

We set the initial damping parameter value λ_0 to be relatively large so that that first updates are small steps in the gradient descent direction. If any iteration results in worse approximation ($(\beta_{t+1}) > \mathbf{r}(\beta_t)$), we set $\lambda_{t+1} = \nu\lambda_t$, where $2 \leq \nu \leq 10$, to get closer to gradient descent, otherwise we set $\lambda_{t+1} = \lambda_t/\nu$ so that as we approach the minimum, the Levenberg-Marquardt update gets closer and closer to the Gauss-Newton update.

Levenberg's algorithm has a disadvantage that if λ is large, the value of $\left[J_{\mathbf{r}}^T J_{\mathbf{r}} + \lambda_t \mathbf{I}\right]^{-1}$ is not used at all, making the method converge slowly in the

direction of a small gradient. Fletcher improved the Levenberg-Marquardt algorithm by replacing the identity matrix I in (2.33) by diagonal matrix of the Hessian estimation (2.31):

$$\beta_{t+1} = \beta_t - \left[J_{\mathbf{r}}^T J_{\mathbf{r}} + \lambda_t diag(J_{\mathbf{r}}^T J_{\mathbf{r}})\right]^{-1} J_{\mathbf{r}}^T \mathbf{r}(\beta_t), \tag{2.34}$$

scaling each component of the gradient according to the curvature, allowing larger movement along the directions where the gradient is smaller.

2.3 Creating the reference image

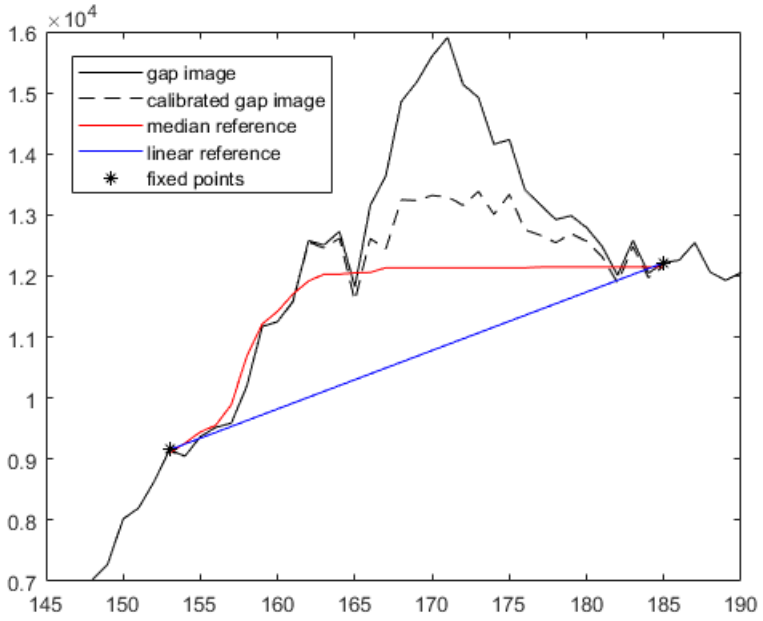

Figure 3: Demonstration of the reference methods used on one row.

Before fitting the gain function, we need to have an estimate of I which we call reference image I_r. Of course, a nearly perfect reference image I_r would give a very accurate estimation of the gain function g. However, it would also make the rest of the correction method redundant since we could just use the reference image as our solution. Therefore we examine how the fitting method behaves under coarse reference images. We compare two simple methods: linear interpolation and median filtering.

Let x_1 and x_2 denote the uppermost and the lowest gap row respectively. In the linear interpolation method, for each column, we choose two reference points taken k rows up and below the gap area, and replace the in between area

by linearly interpolated values. In the median filtering method we replace these pixels by median filtered values. Let $l := 2k + x_2 - x_1 + 1$; that is, the height of the gap area. In the linear interpolation method, for each column $j \in [1, m]$ and $i \in [0, l-1]$, we set

$$I_r(x_1 - k + i, j) = I_r(x_1 - k, j) + i\frac{I_n(x_2 + k, j) - I_r(x_1 - k, j)}{l - 1}. \tag{2.35}$$

In the median filtering method, for each column $j \in [1, m]$ and $i \in [x_1 - k, x_2 + k]$, we set

$$I_r(i, j) = median(\{I_n(i - M, j), \ldots, I_n(i + M, j)\}), \tag{2.36}$$

where $M \in \mathbb{N}$ determines the size of the filtering window.

2.4 The correction algorithm

The correction algorithm goes as follows:

1. Using the input image I_n, compute the reference image I_r using either median filtering or linear interpolation.
2. Create the gain measurement y_i as described in the section (2.1.2).
3. Estimate the gains by fitting the exponential model using the fitting methods described in the section (2.2).
4. Use the estimated gain function $\hat{g}$ to correct the edge gain affected areas of the image.
5. Apply the existing gap correction method for the gap area.

Overall goal of this thesis is to find the best possible method to estimate the gain function g.

2.5 Quality measures

In this section we define the quality measures used in the result section in this thesis. First we define the relative square error for the fitting evaluation, then we define the measures PSNR, SSIM and HaarPSI used in the image quality evaluation.

2.5.1 RSE

The relative square error, RSE, is the residual sum of squares normalized with the square sum of the real value of the signal, given the square error as a percentage of the real value. Given the real signal $y \in \mathbb{R}^n$ and the estimation $x \in \mathbb{R}^n$, the RSE is defined as

$$RSE(x, y) = \frac{\|x - y\|_2}{\|y\|_2} * 100\%. \tag{2.37}$$

2.5.2 PSNR

Peak signal-to-noise ratio (PSNR) is the ratio between the maximum possible value of the signal and the level of noise. PSNR is commonly defined via mean squared error (MSE). Given a noise-free $n \times m$ image y and the estimated image x, the MSE is defined as:

$$MSE(x, y) = \frac{1}{nm} \sum_{i=1}^{m} \sum_{j=1}^{n} [y(i, j) - x(i, j)]^2 . \tag{2.38}$$

Because of the very wide dynamic range of the 16 bit images, we express the PSNR in terms of the logarithmic scale as follows:

$$PSNR = 10 \log_{10} \left(\frac{MAX_x^2}{MSE} \right). \tag{2.39}$$

Here, the MAX_f represents the maximum possible pixel value of the image, e.g in 16 bit images, this is $2^{16} - 1 = 65535$. It is worth noting that in the absence of noise, PSNR is undefined.

2.5.3 SSIM

The structural similarity (SSIM) index developed by Wang et. al [12] compares three features: luminance, contrast and structure. SSIM is built such that each of these three components are relatively independent, for example changes in the luminance and/or contrast will not affect the structure of images. The system diagram of the SSIM measurement system can be seen in the Figure 4.

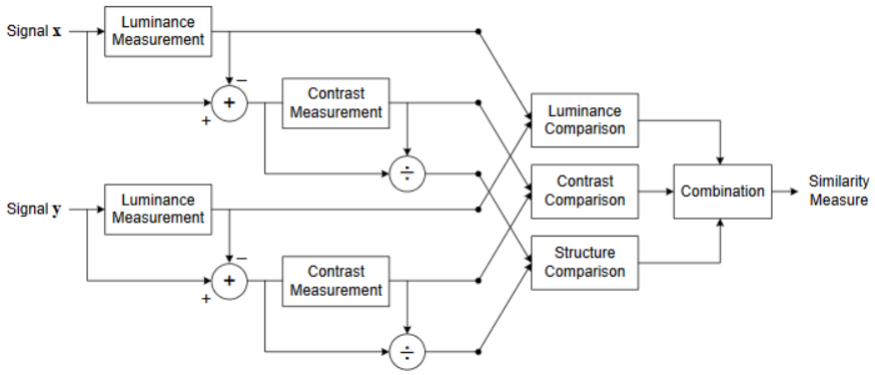

Figure 4: Diagram of the structural similarity (SSIM) measurement system

First, the luminance of each signal is compared. For image x we evaluate this using the mean intensity

$$\mu_x = \frac{1}{nm} \sum_{i=1}^{m} \sum_{j=1}^{n} x_{i,j}. \tag{2.40}$$

The luminance comparison function $l(x, y)$ is then a function of μ_x and μ_y.

Then, we estimate the signal contrast. First, we remove the mean intensity from the signal. The resulting image $x - \mu_x$ corresponds to projection of image x onto the hyper plane defined by

$$\sum_{i=1}^{m}\sum_{j=1}^{n} x_{i,j} = 0. \tag{2.41}$$

The signal contrast is now estimated using the standard deviation. An unbiased estimate is given by

$$\sigma_x = \left(\frac{1}{nm-1} \sum_{i=1}^{m}\sum_{j=1}^{n} (x_{i,j} - \mu_x)^2 \right)^{\frac{1}{2}}. \tag{2.42}$$

The contrast comparison function $c(x, y)$ is then the comparison of σ_x and σ_y.

To estimate the structure of the signal, we first normalize the image by its standard deviation, so that the resulting two images have unit standard deviation. The structure function $s(x, y)$ is quantified using these normalized signals $(x - \mu_x)/\sigma_x$ and $(y - \mu_y)/\sigma_y$.

Finally, the three functions $l(x, y), c(x, y), s(x, y)$ are combined using some combination function $f(\cdot)$ to form an overall similarity measure

$$S(x, y) = f(l(x, y), c(x, y), s(x, y)), \tag{2.43}$$

The similarity measure (2.43) should also satisfy the following conditions:

1. Symmetry: $S(x, y) = S(y, x)$;
2. Boundedness: $S(x, y) \leq 1$;
3. Unique maximum: $S(x, y) = 1$ if and only if $x = y$.

To satisfy these three conditions, we define functions (2.40) and (2.42) as follows:

$$l(x, y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c1}, \tag{2.44}$$

$$c(x, y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2}, \tag{2.45}$$

Here; the constants c_1 andc_2 are chosen in order to avoid division by zero. More specifically, we set

$$c_1 = (k_1 L)^2, \text{ and } c_2 = (k_2 L)^2, \tag{2.46}$$

where $k_1 \ll 1$ and $k_2 \ll 1$ are small constants and L is the dynamic range of the pixel values.

Structure comparison is associated with normalized signals $(x - \mu_x)/\sigma_x$ and $(y - \mu_y)/\sigma_y$. We use correlation between these two to measure the structural similarity. Note, that correlation between the normalized signals is equivalent

to the correlation coefficients of x and y. Thus, we define structure comparison function as

$$s(x,y) = \frac{\sigma_{xy} + c_3}{\sigma_x \sigma_y + c_3}, \tag{2.47}$$

where σ_{xy} can be estimated as

$$\sigma_{xy} = \frac{1}{nm-1} \sum_{i=1}^{m} \sum_{j=1}^{n} (x - \mu_x)(y - \mu_y) \tag{2.48}$$

As in the luminance and contrast measures, the constant c_3 is chosen in order to avoid division by zero.

The general form of SSIM is defined as

$$SSIM(x,y) = \left[l(x,y)^{\alpha} \cdot c(x,y)^{\beta} \cdot s(x,y)^{\gamma} \right], \tag{2.49}$$

where $\alpha > 0$, $\beta > 0$ and $\gamma > 0$ are parameters used to adjust the relative importance of the three components. In this thesis, we use the following simplified form of this measure by setting $\alpha = \beta = \gamma = 1$ and $c_3 = c_2/2$:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}. \tag{2.50}$$

2.5.4 HaarPSI

The Haar Wavelet-based Perceptual Similarity Index [13] is a similarity measure of two digital images to the value in the interval $[0, 1]$, that is

$$HaarPSI : \ell(\mathbb{Z}^2) \times \ell(\mathbb{Z}^2) \to [0,1]. \tag{2.51}$$

The output value of HaarPSI aims to express the perceptual similarity of two images with respect to human eye, such that the HaarPSI of two identical images is 1 and HaarPSI of two completely different images is 0.

For measuring the local similarity, HaarPSI uses the coefficients of a discrete wavelet transform. The wavelet chosen for HaarPSI is the so-called Haar wavelet proposed by Alfred Haar in 1910, which is known for its simplicity and computational efficiency. Lets briefly recall the main ideas about Haar wavelets. The Haar scaling function is defined by

$$\phi(x) = \begin{cases} 1, & \text{if } 0 \le x \le 1 \\ 0, & \text{elsewhere} \end{cases} \tag{2.52}$$

Suppose that $j \in \mathbb{N}$. The space of step functions at level j, denoted by V_j is defined to be the space spanned by the set

$$\left\{ \ldots, \phi(2^j x - 1), \phi(2^j x), \phi(2^j x + 1), \phi(2^j x + 2), \ldots \right\} \tag{2.53}$$

over real numbers. Now V_j is the space of piece-wise constant functions of finite support whose discontinuities are contained in the set

$$\left\{\ldots, -\frac{1}{2^j}, 0, \frac{1}{2^j}, \frac{2}{2^j}, \ldots\right\}. \tag{2.54}$$

Following proposition holds by the definition (2.52) and (2.53)

$$\begin{aligned} f(x) \in V_0 \text{ if and only if } f(2^j x) \in V_j, \\ f(x) \in V_j \text{ if and only if } f(2^{-j} x) \in V_0. \end{aligned} \tag{2.55}$$

We also have

$$\left\|\phi(2^j x - k)\right\|_2^2 = \int_{-\infty}^{\infty} \left(\phi(2^j x - k)\right)^2 dx = \frac{1}{2^j} \int_k^{k+1} 1 dx = \frac{1}{2^j}, \tag{2.56}$$

and using the L^2 inner product we have

$$\langle \phi(x-k), \phi(x-j) \rangle = \int_{-\infty}^{\infty} \phi(x-k)\phi(x-j)dx = 0, \tag{2.57}$$

since $\phi(x-k)$ and $\phi(x-j)$ are disjoint when $j \neq k$. We conclude that the set of functions

$$\left\{2^{\frac{2}{j}} \phi(2^j x - k),\ k \in \mathbb{Z}\right\} \tag{2.58}$$

is an orthogonal basis of V_j.

The Haar mother wavelet is defined by

$$\psi(x) = \phi(2x) - \phi(2x-1). \tag{2.59}$$

By definition, the Haar mother wavelet ψ is a member of V_1. Also it holds that

$$\int_{-\infty}^{\infty} \left(\phi(2x) - \phi(2x-1)\right) \phi(x) dx = \int_0^{1/2} 1 dx - \int_{1/2}^1 1 dx = 0, \tag{2.60}$$

moreover, if $k \neq 0$, then the support of $\psi(x)$ and $\phi(x-k)$ does not overlap and so we have $\int \psi(x)\phi(x-k)dx = 0$. Therefore ψ belongs to V_1 and it is orthogonal to V_0. By above deduction, it follows that any function of the form

$$f_1 = \sum_{k \in \mathbb{Z}} a_k \phi(2x-k) \in V_1, \tag{2.61}$$

is orthogonal to each $\phi(x-l),\ l \in \mathbb{Z}$, i.e; is orthogonal to space V_0 if and only if

$$a_1 = -a_0,\ a3 = -a_2, \ldots \tag{2.62}$$

hence we have

$$f_1 = \sum_{k \in \mathbb{Z}} a_{2k} \left(\phi(2x-2k) - \phi(2x-2k-1)\right) = \sum_{k \in \mathbb{Z}} a_{2k} \psi(x-k). \tag{2.63}$$

We know have shown that an arbitrary function f_1 in V_1 is orthogonal to V_0 if and only if it is of the form $\sum_{k\in\mathbb{Z}} a_{2k}\psi(x-k)$. Now, let W_j be the space of functions of the form

$$\sum_{k\in\mathbb{Z}} a_k\psi(2^j-k),\ a_k\in\mathbb{R}, \tag{2.64}$$

where we assume that only finite number of a_k are nonzero. By (2.63), we have

$$V_1 = V_0 \oplus W_0, \tag{2.65}$$

in other words, function in V_1 can be decomposed uniquely as a sum of a function in V_0 and a function in W_0. By the similar deduction and the induction principle, we can state a more general result

$$\begin{aligned} V_{j+1} &= V_j \oplus W_j \\ &= V_{j-1} \oplus W_{j-1} \oplus W_j \\ &= \dots \\ &= V_0 \oplus W_0 \oplus W_1 \oplus \cdots \oplus W_j \end{aligned} \tag{2.66}$$

Above result means that each $f \in V_{j+1}$ can be decomposed uniquely as a sum

$$f = f_0 + w_0 + w_1 + \cdots + w_j, \tag{2.67}$$

where $f_0 \in V_0$ and $w_l \in W_l$ for all $0 \le l \le j$.

Basic idea behind wavelet decomposition is to approximate the signal f in space V_j spanned by (2.58) and then decompose it according to (2.67). We want to choose the scale $j \in \mathbb{N}$ large enough so that the approximation $f_j \in V_j$ captures most of the important features of f. Suppose

$$f(x) \approx f_j(x) = \sum_{k\in\mathbb{Z}} a_k^j \phi(2^j x - k) \in V_j. \tag{2.68}$$

Then by (2.67), f_j can be decomposed as

$$f_j = f_{j-1} + w_{j-1}, \tag{2.69}$$

where

$$w_{j-1} = \sum_{k\in\mathbb{Z}} b_k^{j-1}\psi(2^{j-1}x-k) \in W_{j-1} \tag{2.70}$$

$$f_{j-1} = \sum_{k\in\mathbb{Z}} a_k^{j-1}\phi(2^{j-1}x-k) \in V_{j-1}. \tag{2.71}$$

Here, the scalars a_k^{j-1} and b_k^{j-1} are called approximation and detail coefficients respectively, which are the coefficients of a discrete wavelet transform mentioned in the beginning of this section. Next we are going to define the Haar filters used to compute the discrete Haar wavelet transform. For more information about the Haar wavelet transform and its implementation, we refer to the book *Ten lectures on wavelets* by Ingrid Daubechies [14].

The one dimensional Haar wavelet filters for level 1 are given by

$$h_1^{1D} = \frac{1}{\sqrt{2}} \cdot [1,1]^T \tag{2.72}$$

and

$$g_1^{1D} = \frac{1}{\sqrt{2}} \cdot [-1,1]^T \tag{2.73}$$

where h_1^{1D} denotes the low-pass scaling filter which and g_1^{1D} the corresponding high-pass wavelet filter. One-dimensional filters h_j^{1D} and $g_j^{1D}$for $j > 1$ are given by

$$g_j^{1D} = h_1^{1D} * \left(g_{j-1}^{1D}\right)_{\uparrow 2}, \tag{2.74}$$

and

$$h_j^{1D} = h_1^{1D} * \left(h_{j-1}^{1D}\right)_{\uparrow 2}, \tag{2.75}$$

where $\uparrow 2$ denotes the dyadic up-sampling operator which lengthens the given vector by substituting zero values:

$$\left(\begin{bmatrix} x_1 \\ x_2 \\ \vdots \end{bmatrix}\right)_{\uparrow 2} = \begin{bmatrix} 0 \\ x_1 \\ 0 \\ x_2 \\ 0 \\ \vdots \end{bmatrix} \tag{2.76}$$

and $*$ denotes the one-dimensional convolution operator with zero boundary conditions.

The two dimensional Haar wavelet system is spanned by the functions

$$\phi(x)\psi(y),\ \psi(x)\phi(y) \text{ and } \psi(x)\psi(y) \tag{2.77}$$

giving us the horizontal-, vertical- and diagonal details, and by

$$\phi(x)\phi(y) \tag{2.78}$$

which gives a coarse approximation of the given image. For any scale $j \in \mathbb{N}$ the horizontal features are given by the filter

$$g_j^{(1)} = g_j^{1D} \left[h_j^{1D}\right]^T \tag{2.79}$$

and the vertical features are obtained by the filter

$$g_j^{(2)} = h_j^{1D} \left[g_j^{1D}\right]^T. \tag{2.80}$$

Let $f_1, f_2 \in \ell(\mathbb{Z}^2)$ be two gray-scale images and let the scalar values $a, b \in \mathbb{R}$ assess the local features of f_1 and f_2 respectively. We measure the similarity of those two measures with function

$$S(a,b,C) = \frac{2ab + C}{a^2 + b^2 + C}, \tag{2.81}$$

where a constant $C > 0$. In HaarPSI, the scalar values a, b are based on the first two levels of discrete Haar wavelet transform. We compute

$$LS^{(k)}_{f_1,f_2}[x] = \frac{1}{2}\sum_{j=1}^{2} S\left(|\left(g_j^{(k)} * f_1\right)[x]|, |\left(g_j^{(k)} * f_2\right)[x]|, C\right), \tag{2.82}$$

where $C > 0$ is constant, $k \in \{1, 2\}$ selects either horizontal or vertical filters and $*$ denotes the two-dimensional convolution operator.

To correctly predict the perceptual similarity experienced by human viewers, the following logistic function given by parameter $\alpha > 0$ is applied:

$$l_\alpha(x) = \frac{1}{1 + e^{-\alpha x}}. \tag{2.83}$$

Hence, the Haar wavelet based local similarity map HS is defined as

$$HS^{(k)}_{f_1,f_2}[x] = l_\alpha\left(LS^{(k)}_{f_1,f_2}[x]\right). \tag{2.84}$$

Finally, the Haar-wavelet based perceptual similarity index for grey-scale images f_1, f_2 is defined as a weighted average of the local similarity map $HS^{(k)}_{f_1,f_2}$. The weights are given by the third scale of a discrete Haar wavelet transform, that is,

$$W_f^{(k)}[x] = |(g_3^{(k)} * f)[x]|. \tag{2.85}$$

Now the the Haar-wavelet based perceptual similarity index is defined by

$$HaarPSI_{f_1,f_2} = l_\alpha^{-1}\left(\frac{\sum_x \sum_{k=1}^{2} HS^{(k)}_{f_1,f_2}[x] \cdot W^{(k)}_{f_1,f_2}[x]}{\sum_x \sum_{k=1}^{2} W^{(k)}_{f_1,f_2}[x]}\right) \tag{2.86}$$

where

$$W^{(k)}_{f_1,f_2}[x] = \max\left\{W^{(k)}_{f_1,}[x], W^{(k)}_{f_2}[x]\right\}. \tag{2.87}$$

2.6 Implementation

2.6.1 Creating the reference image

Before going through the results, let's recall all the methods we will use in this thesis and how we are going to implement them. We described the relation between the noisy image $I_n \in \mathbb{R}^{n\times m}$, the gain function $g \in \mathbb{R}^{n\times m}$, the real image $I \in \mathbb{R}^{n\times m}$ and additional noise $\varepsilon \in \mathbb{R}^{n\times m}$ as

$$I_n(i,j) = g(i,j)\left[I(i,j) + \varepsilon(i.j)\right].$$

We defined two methods which we will use to estimate the term I called the reference image I_r. In the implementation of these two methods, we assume that we know the uppermost and the undermost rows of the gap, denoted x_1 and x_2 respectively.

In the first method, called the linear interpolation method, for each column j, we compute a straight line between the two reference points taken $k \in \mathbb{N}$ rows above and below the gap and use them as the reference image values.

Algorithm 1 Linear interpolation

Input:
Image I, gap rows x_1 and x_2, number of reference rows k
Output:
Reference image I_r
Initialize:
$I_r \leftarrow I$
$l \leftarrow 2k + x_2 - x_1 + 1$
for $j = 1$ to m **do**
  for $i = 0$ to $l - 1$ **do**
    $I_r(x_1 - k + i, j) \leftarrow I(x_1 - k, j) + i \left(\frac{I(x_2+k,j) - I(x_1-k,j)}{l-1} \right)$
  end for
end for

In the median filtering method, for each column j, we replace each pixel with median of M preceding and following pixels. Since all the tested gaps were far enough from the border, we were able to determine M such that $M > k$, $x_1 - k - M > 0$ and $x_2 + k + M > m$. Here, the first condition is to ensure that the filtering method can remove the stripe and the last two conditions allows us to avoid the boundary issues.

Algorithm 2 Median filtering

Input:
Image I, gap rows x_1 and x_2, number of reference rows k,
size of the filtering window M
Output:
Reference image I_r
Initialize:
$I_r \leftarrow I$
for $j = 1$ to m **do**
  for $i = x_1 - k$ to $x_2 + k$ **do**
    $I_r(i, j) \leftarrow median(\{I(i - M, j), \ldots, I(i + M, j)\})$
  end for
end for

2.6.2 Creating the measure

After applying the reference image, we have an approximation

$$I_n(i, j) = g(i, j) \left[I(i, j) + \varepsilon(i, j) \right] \approx g(i, j) I_r(i, j),$$

from which we get

$$g(i,j) \approx \frac{I_n(i,j)}{I_r(i,j)}.$$

We assumed that $g(i,j)$ is constant column-wise so we will take median to create the measurement vector $\mathbf{y}$.

Algorithm 3 Creating the measurement vector

1: **Input:**
   Image I, reference image I_r
2: **Output:**
   Vector containing the gain function approximations $\mathbf{y}$
3: **Initialize:**
   $\mathbf{y} \leftarrow \mathbf{1} \in \mathbb{R}^m$
4: **for** $i = 1$ to n **do**
5: $\quad y_i \leftarrow median\left(\left\{\frac{I(i,1)}{I_r(i,1)}, \ldots, \frac{I(i,m)}{I_r(i,m)}\right\}\right)$
6: **end for**

We stated that near the edges the gain function g behaves according to an exponential function

$$g(i,j) = 1 + d_0 A^{|i-i_0|},$$

where i_0 is the edge-most row above or below the gap. It is worth to note, that parameter values d_0 and A may differ for the lower and the upper edge-most row, so they have to be processed separately. Without the loss on generality, we made assumption that i_0 is the edge-most row below the gap. Now we have

$$y_i \approx g(i,j) = 1 + d_0 A^{i-i_0}, i_0 \leq i \leq i_0 + k - 1$$

After re-indexing so that $y_{i_0} = y_1$ we finally get the measurement vector $\mathbf{y} \in \mathbb{R}^k$

$$y_i \approx 1 + d_0 A^{i-1}, 1 \leq i \leq k.$$

2.6.3 Linearization methods

Now let's recall the fitting methods. Logarithm method uses the logarithm to transform our equation into a linear one:

$$y_i^l := \ln(y_i - 1) = \ln(d_0) + \ln(A)(i-1).$$

We will solve in the least squares sense.

Algorithm 4 The Logarithm method

1: **Input:**

Measurement vector $\mathbf{y}$

2: **Output:**

Vector β containing the estimations of the parameters d_0 and A.

3: **Initialize:**

$\mathbf{y}^l \leftarrow \mathbf{1} \in \mathbb{R}^k$

$\mathbf{X} \leftarrow \mathbf{1} \in \mathbb{R}^{k \times 2}$

4: **for** $i = 1$ to k **do**

5: $y_i^l \leftarrow \ln(y_i - 1)$

6: $X(i, 2) \leftarrow i - 1$

7: **end for**

8: $\beta \leftarrow \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}^l$

9: $\beta \leftarrow \exp(\beta)$

In the second linearization method, we assumed that y_i is a discreet sample of continuous function, in other words,

$$y_i \approx 1 + d_0 A^{i-1} =: f(i).$$

We manipulated this equation by taking the integral both sides. After some manipulations we got

$$y_t - y_1 = \ln(A)\left(\int_1^t f(i)di - t + 1\right),$$

for $1 \leq t \leq k$. We estimate the integral with Simpson's rule

$$\int_1^t f(i)di \approx \sum_{i=1}^{t} S_i : \begin{cases} S_1 = 0 \\ S_i = \frac{1}{2}(y_i - y_{i-1})), \text{ when } i > 1. \end{cases}$$

and now the parameters d_0 and A can be solved separately in the least squares sense.

Algorithm 5 The integral method

Input:

Measurement vector $\mathbf{y}$

Output:

Vector β containing the estimations of the parameters d_0 and A.

Initialize:

$\mathbf{y}^d \leftarrow \mathbf{0} \in \mathbb{R}^k$

$\mathbf{X} \leftarrow \mathbf{0} \in \mathbb{R}^k$

$S \leftarrow 0$

for $i = 2$ to k **do**

  $y_i^d \leftarrow y_i - y_1$

  $S \leftarrow S + \frac{1}{2}(y_i - y_{i-1})$

  $X_i \leftarrow S - i + 1$

end for

$A \leftarrow \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}^d$

$A \leftarrow \exp(A)$

for $i = 1$ to k **do**

  $y \leftarrow y_i - 1$

  $X_i \leftarrow A^{i-1}$

end for

$d_0 \leftarrow \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$

$\beta \leftarrow [d_0, A]$

2.6.4 Iterative methods

Then we introduced three different non-linear least square methods. We have the measurement

$$y_i \approx 1 + d_0 A^{i-1} =: f(i, \beta)$$

and want to find $\beta = [d_0, A]^T$ that minimizes

$$l(\beta) := \|\mathbf{r}\|_2^2 := \|\mathbf{f}(\beta) - \mathbf{y}\|_2^2 .$$

We denoted the Jacobian matrix as

$$\mathbf{J}_r(\beta) := \frac{\partial \mathbf{r}(\beta)}{\partial \beta}.$$

First, we introduced the gradient descent method, which is based on the fact that the function $l(\beta)$ decreases the fastest, if we take suitable sized steps from β in the direction of the negative gradient. In other words, we say that if

$$\beta_{t+1} = \beta_t - \gamma_t \nabla l(\beta_t)$$

for $\gamma_t \in \mathbb{R}_+$ small enough, then $l(\beta_t) \geq l(\beta_{t+1})$. Due to it's faster convergence rate we use the Barzilai-Borwein method to determine the step size γ_t, that is, we set

$$\gamma_t = \frac{\|\beta_t - \beta_{t-1}\|^2}{(\beta_t - \beta_{t-1})^T (\nabla l(\beta_t) - \nabla l(\beta_{t-1}))}.$$

We also noted that the gradient can be written in terms of the Jacobian matrix $\mathbf{J_r}$ as

$$\nabla l(\beta_t) = 2\mathbf{J}_\mathbf{r}^T \mathbf{r}(\beta_t)$$

Algorithm 6 The Barzilai-Borwein method

Input:
Measurement vector $\mathbf{y}$, tolerance T, initial value of β, max iterations M, initial step size γ

Output:
Vector β containing the estimations of the parameters d_0 and A.

Initialize:
$i \leftarrow 1$
$\mathbf{J} \leftarrow \mathbf{J_r}(\beta)$
$\mathbf{r} \leftarrow \mathbf{f}(\beta) - \mathbf{y}$
$g \leftarrow 2\mathbf{J}^T\mathbf{r}$

while $\|\mathbf{r}\|_2^2 > T$ or $i \leq M$ **do**
  $\beta_{prev} \leftarrow \beta$
  $\beta \leftarrow \beta - \gamma g$
  $\mathbf{r} \leftarrow \mathbf{f}(\beta) - \mathbf{y}$
  $\mathbf{J} \leftarrow \mathbf{J_r}(\beta)$
  $g_{prev} \leftarrow g$
  $g \leftarrow 2\mathbf{J}^T\mathbf{r}$
  $\gamma \leftarrow \frac{\|\beta - \beta_{prev}\|_2^2}{(\beta - \beta_{prev})^T (g - g_{prev})}$
  $i \leftarrow i + 1$
end while

The second non-linear method we introduced was The Gaussian-Newton method which is a modification of the Newton's method. Unlike the Newton's method, the Gauss-Newton algorithm does not require the computation of the second derivatives. Instead, we set

$$\nabla^2 l(\beta) = 2\mathbf{J}_r^T \mathbf{J}_r(\beta)$$

so that the iteration step can be written as

$$\beta_{t+1} = \beta_t - \left[\mathbf{J}_\mathbf{r}^T \mathbf{J}_\mathbf{r}\right]^{-1} \mathbf{J}_\mathbf{r}^T \mathbf{r}(\beta)$$

Algorithm 7 The Gaussian-Newton method

Input:

Measurement vector $\mathbf{y}$, initial value of β, tolerance T, max iterations M

Output:

Vector β containing the estimations of the parameters d_0 and A.

Initialize:

$i \leftarrow 1$

$\mathbf{r} \leftarrow \mathbf{f}(\beta) - \mathbf{y}$

while $\|\mathbf{r}\|_2^2 > T$ or $i \leq M$ **do**

  $\mathbf{J} \leftarrow \mathbf{J}_{\mathbf{r}}(\beta)$

  $s \leftarrow \left[\mathbf{J}^T\mathbf{J}\right]^{-1}\mathbf{J}^T\mathbf{r}$

  $\beta \leftarrow \beta - s$

  $\mathbf{r} \leftarrow \mathbf{f}(\beta) - \mathbf{y}$

  $i \leftarrow i + 1$

end while

The last method we introduced was the Levenberg-Marquardt-Fletcher(LMF) algorithm which varies the parameter updates between the previous two. In the Levenberg-Marquardt algorithm, set

$$\beta_{t+1} = \beta_t - \left[\mathbf{J}_{\mathbf{r}}^T + \lambda_t\mathbf{I}\right]^{-1}\mathbf{J}_{\mathbf{r}}^T\mathbf{r}(\beta_t).$$

Fletcher's modification of this algorithm replaces the identity matrix by diagonal matrix of the Hessian estimation to allow larger movement along the directions where the gradient is smaller. Hence, we set

$$\beta_{t+1} = \beta_t - \left[\mathbf{J}_{\mathbf{r}}^T\mathbf{J}_{\mathbf{r}} + \lambda_t diag\left(\mathbf{J}_{\mathbf{r}}^T\mathbf{J}_{\mathbf{r}}\right)\right]^{-1}\mathbf{J}_{\mathbf{r}}^T\mathbf{r}(\beta).$$

The damping parameter $\lambda_t \geq 0$ serves for scaling purposes. We noted that when $\lambda = 0$, the method transforms into the Gauss-Newton method, while for $\lambda \to \infty$ the method approaches the gradient descent method. We set λ_0 to be relatively large so that first updates are small steps in the gradient descent direction. We set $\lambda_{t+1} = \nu\lambda_t$, if $(\mathbf{r}(\beta_{t+1}) > \mathbf{r}(\beta_t))$, otherwise, we set $\lambda_{t+1} = \lambda_t/\nu$, for $2 \leq \nu \leq 10$.

Algorithm 8 The LMF-method

Input:

Measurement vector $\mathbf{y}$, initial value of β, initial value of λ, tolerance T, max iterations M

Output:

Vector β containing the estimations of the parameters d_0 and A.

Initialize:

$i \leftarrow 1$

$\mathbf{r} \leftarrow \mathbf{f}(\beta) - \mathbf{y}$

while $\|\mathbf{r}\|_2^2 > T$ or $i \leq M$ **do**

  $\mathbf{J} \leftarrow \mathbf{J}_{\mathbf{r}}(\beta)$

  $s \leftarrow \left[\mathbf{J}^T\mathbf{J} + \lambda diag\left(\mathbf{J}^T\mathbf{J}\right)\right]^{-1}\mathbf{J}^T\mathbf{r}$

  $\beta \leftarrow \beta - s$

  $\mathbf{r}_{prev} \leftarrow \mathbf{r}$

  $\mathbf{r} \leftarrow \mathbf{f}(\beta) - \mathbf{y}$

  if $\mathbf{r} > \mathbf{r}_{prev}$ **then**

    $\lambda \leftarrow \nu\lambda$

  else

    $\lambda \leftarrow \lambda/\nu$

  end if

  $i \leftarrow i + 1$

end while

3 Results

3.1 Preliminary tests

3.1.1 Reference image method evaluation

To find out which reference image generation method to use, we used 12 images. For each image, we planted 6 simulated gaps such that the parameter values of the exponential function were randomized. Then, we applied the reference methods for each of the gap areas separately. Since the areas above and below the gaps can be treated separately, we have a 144 testing areas.

Since our fitting methods minimize the residual sum of squares, we evaluated the reference methods using the relative least squares. The methods are evaluated based on how well the methods estimate the real image as well as how well they, together with row-wise median (see: (2.7)), estimate the gains.

Reference Method	Mean (%)	Variance (%)	Max (%)
Linear	7.15	0.49	50.63
Median	5.36	0.20	37.13

Table 1: Relative square errors of the reference image vs the real image

In the Table 1, we have the mean, variance and maximum of the relative square errors of the linear and the median estimations. As we can see, the median reference method seems to give much better results than the linear method.

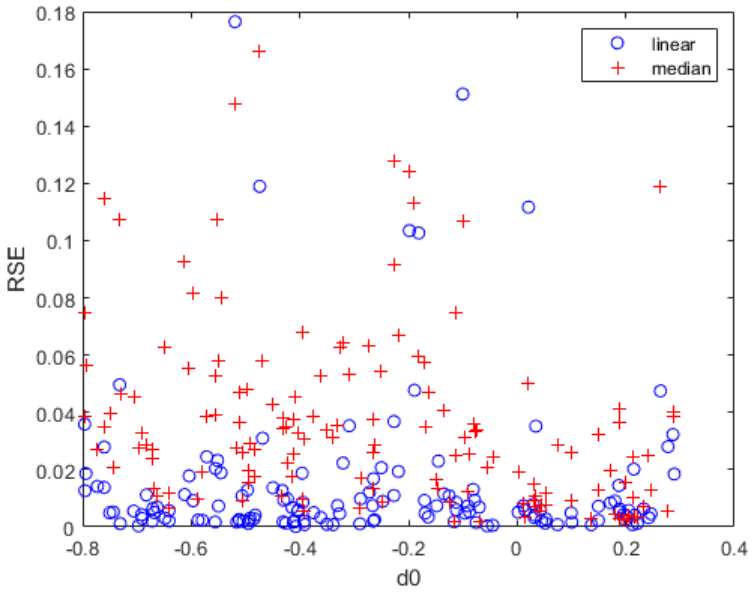

Figure 5: Relative square errors of linear and median gain estimates computed using the row-wise median

.

In (2.7), we suggested to take row-wise median to make use of the fact that the gain function is constant row-wise. The Figure 5 shows us the relation between the simulated d_0 and the relative square error after applying the median row-wise. Now, the linear interpolation seems more stable compared to the median filtering.

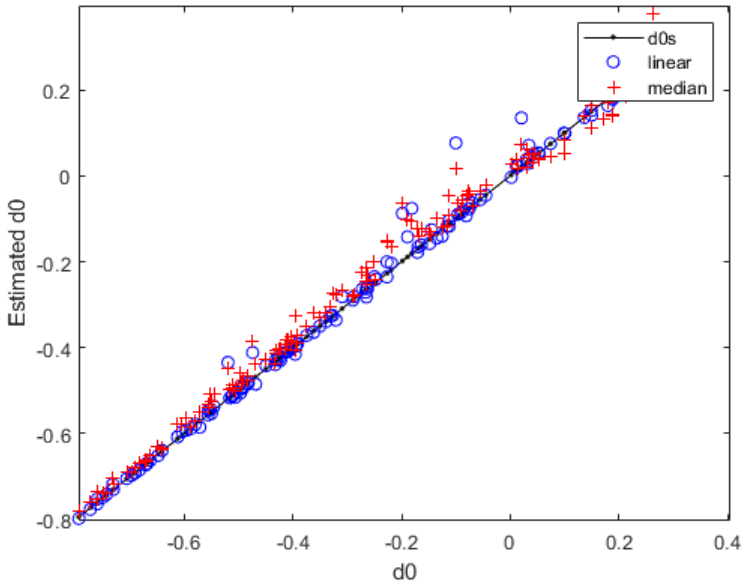

Figure 6: Linear and median gain estimate of the edge-most row (d_0) comparison

.

Note that the gain function $g \to 1$ as we go further from the gap. This means that the edge-most row has the most noise. Hence, the last thing in this section we want to examine is the comparison of the real d_0 and estimated d_0 in the Figure 6. From here, it is easy to see that both methods give great estimation of the gain of the edge-most row.

In conclusion, since the linear estimation gave slightly better results after taking the row-wise median, we chose to use the linear interpolation as the reference method in the simulation and real data tests.

3.1.2 Exponential fitting

We tested different fitting methods on a simulated sample of randomly generated exponential function corrupted by a white Gaussian noise with zero mean and variance $\sigma = 0.1$. For this test, 200 exponential function samples with a length of 15 were generated. Then, we repeated the test, but now added a spike to randomly selected entry of the measurement vector by multiplying it by a factor of 1.5. This was to simulate possible sudden detail changes which in combination with reference method can cause a spike to the measurement vector. Example of a simulated gain function can be seen in the Figure 7. The point of these tests was to pick one linearization and one iterative optimization method for the actual correction method tests.

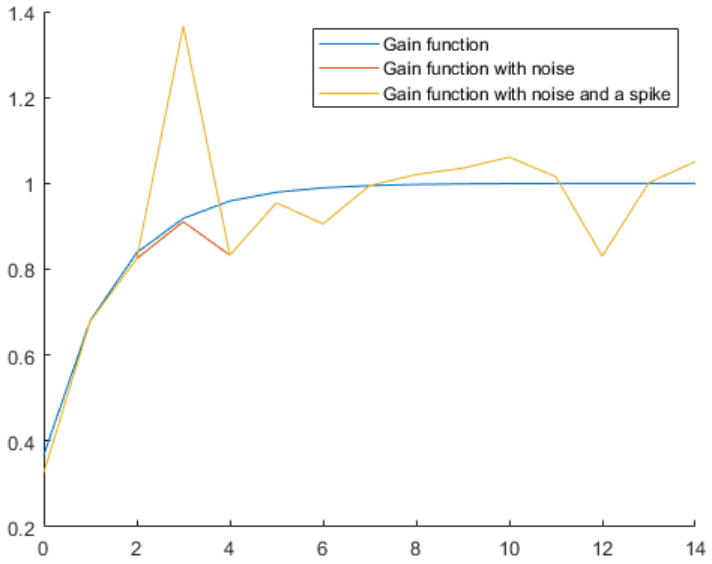

Figure 7: Example of a simulated gain function used in the preliminary fitting tests

.

	Perfect		
Correction Method	Mean (%)	Variance (%)	Avg Time (s)
Logarithm	∼0	∼0	0.0001
Integral	0.1	∼0	0.0001
Gauss-Newton	∼0	∼0	0.0003
LMF	0	∼0	0.0007

Table 2: Relative square errors and computation time of the perfect measurement fitting

Table 2 shows the fitting results from the perfect measurements. The purpose of this test is to see the possible computational errors in each method. During this and the next test we noticed that the Barzilai-Borwein failed to converge as soon as any noise was introduced. In the other methods, the error in the best case scenario is basically non-existent.

	Noised, $\sigma = 0.1$		
Correction Method	Mean (%)	Variance (%)	Avg Time (s)
Logarithm	7.97	0.22	0.0001
Integral	3.34	0.03	0.0001
Gauss-Newton	3.48	0.03	0.0371
LMF	3.43	0.03	0.0012

Table 3: Relative square errors and computation time of the noised measurement fitting

Table 3 shows the effect of the Gaussian noise to each method. We notice that mean and variance of the relative square error of the logarithm method is over two times larger than the other methods. Additionally, we can notice that the Gauss-Newton is significantly slower than the other methods.

	Noised and spiked, $\sigma = 0.1$		
Method	Mean (%)	Variance (%)	Avg Time (s)
Logarithm	8.38	2.4	0.0001
Integral	5.21	1.1	0.0001
Gauss-Newton	5.21	1.0	0.0371
LMF	5.21	1.0	0.0012

Table 4: Relative square errors and computation time of the the simulated measurement with Gaussian noise and random spike

Table 4 shows the relative errors after we add one random spike. The results in this tests are very similar to the results in the Table 5: the logarithm method is significantly worse than every other method and the Gauss-Newton is clearly the slowest.

In conclusion, we had that the Barzilai-Borwein didn't converge after applying any noise, hence we omitted it from these tests entirely. Also, since the logarithm performed poorly and the Gauss-Newton was slow compared to other methods, we chose to leave out these methods from the further tests.

3.2 Correction method tests

3.2.1 Simulated data

In this test we used the same planted simulated gaps as in the reference image test. By the results of the preliminary tests, we computed the gain estimates using linear reference method and then fitted the gain function using the integral and LMF methods. Since the linear reference with row-wise median gave quite good estimates already, we also tested how well the fitting improves that estimation.

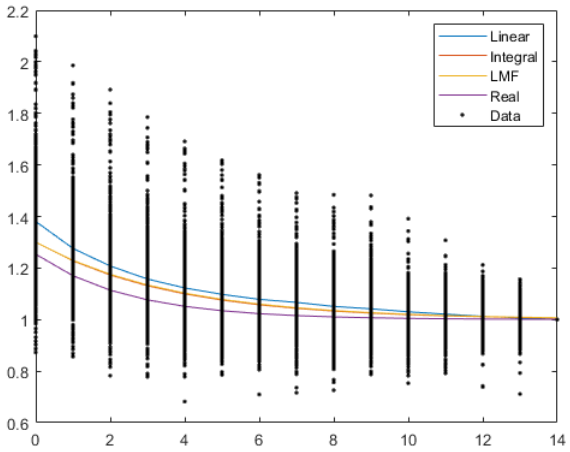

Figure 8: Fitting example from the data collected using the linear reference

.

In the Figure 8 we have an example of data fitting. As we can see, all the methods seem to give relatively good estimations of the gain function even if the data itself is quite scattered.

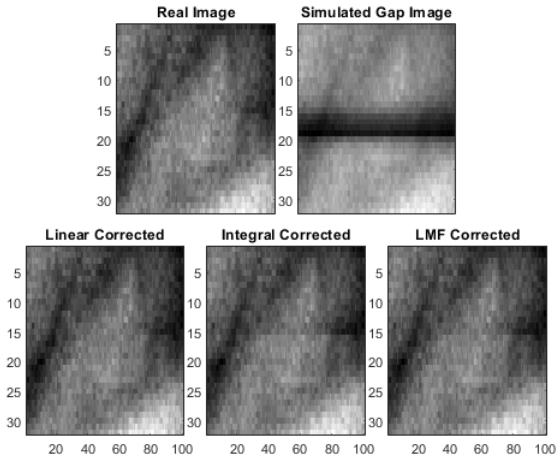

Figure 9: Zoomed in area of a failed correction

.

Before going through the quality measures, we can immediately notice that little artefacts in the corrected images as can be seen in the Figure 9. It seems

that even after correction, the effect of the gain function is still somewhat visible on the edge-most rows. The explanation for this can be seen in the Figure 8: most of the remaining error is still on the edge-most rows. Other thing to note is that the straight linear correction looks slightly better than the corrections using the fitted gain function.

	Above the gap			Below the Gap		
Method	PSNR	SSIM	HaarPSI	PSNR	SSIM	HaarPSI
Uncorrected	128.2	0.9696	0.51721	125.4	0.9539	0.4007
Linear	140.2	0.9984	**0.9747**	143.1	0.9994	**0.9858**
Integral	140.5	0.9983	0.8809	143.9	0.9994	0.9278
LMF	**141.6**	**0.9988**	0.9353	**144.7**	**0.9995**	0.9476

Table 5: Results of the image area in Figure 9

Table 5 shows the quality measures of the image area which can be seen in the Figure 9. We notice that the LMF method gives the best results in terms of PSNR and SSIM, although just slightly. On the other hand the linear estimation gives the best result in terms of HaarPSI.

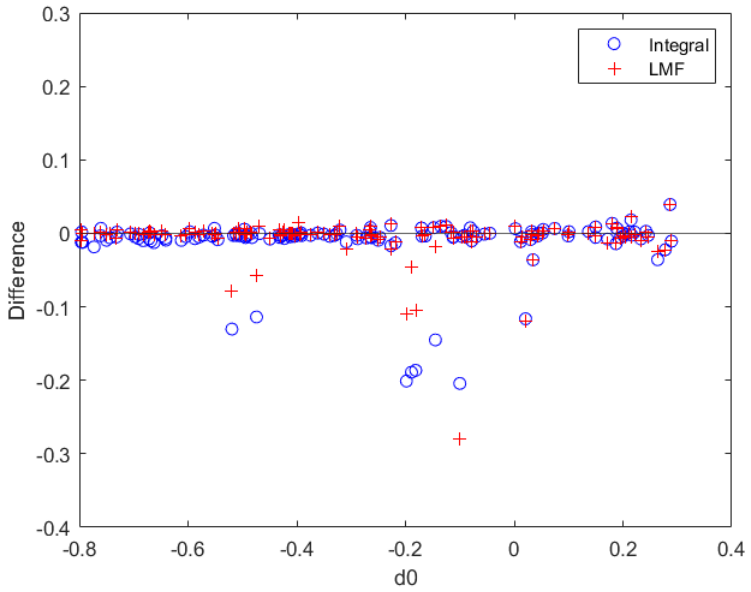

Figure 10: The difference between the real parameter value d_0 and its estimation $\hat{d}_0$

.

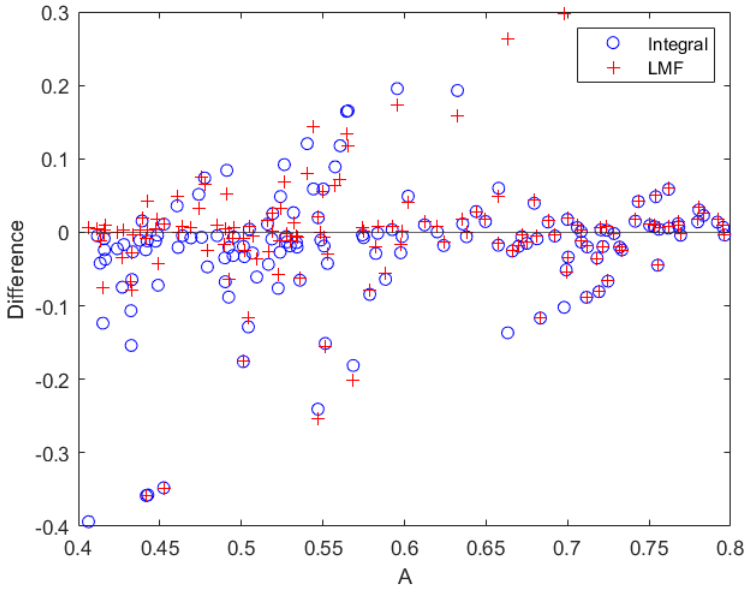

Figure 11: The difference between the real parameter value A and its estimation $\hat{A}$

.

In the Figure 10 we have plotted the relation between the d_0 and the difference between the estimation and the real d_0. Respectively, in the Figure 11 we have plotted the relation between the A and the difference between the estimation and the real A. It seems that both methods give good estimations of the parameter d_0, whereas estimations of the parameter A have a lot more deviation.

	Simulated Data		
Method	PSNR	SSIM	HaarPSI
Uncorrected	124.5	0.9330	0.5056
Linear	149.8	0.9990	**0.9877**
Integral	150.5	0.9986	0.9601
LMF	**153.2**	**0.9994**	0.9781

Table 6: Mean results of the simulated data test

Table 6 supports our conclusion from the Table 5: LMF seems to be a little better in terms of PSNR and SSIM, but in terms of HaarPSI, the straight linear estimation seems to perform the best.

3.2.2 Real data

Finally, the whole correction algorithm is evaluated on a real gap data. We use various doses to generate 10 frame images of a skull phantom using an old sensor. The corrected images are compared against their respective calibrated images.

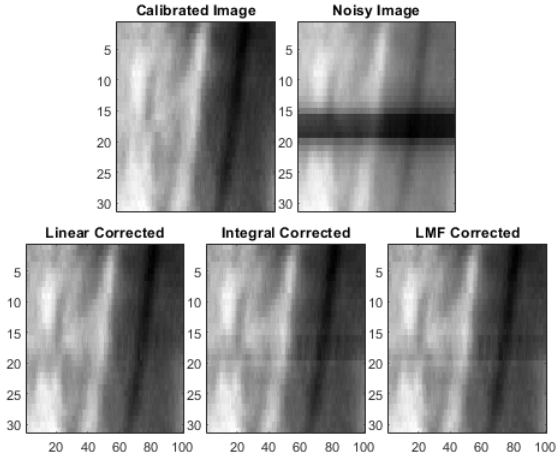

Figure 12: Zoomed in area of a correction of the real image.

.

In Figure 12 we have an example of one image area which is corrected with different correction methods. Like we saw in the simulated data tests, we again have the line artefacts in the corrected images. In particular, now the line artefacts can be seen even more clearly, especially from the images which use fitted gain function to correct the gain area.

	Above the gap			Below the gap		
Method	PSNR	SSIM	HaarPSI	PSNR	SSIM	HaarPSI
Uncorrected	113.9	0.9222	0.3437	108.6	0.8936	0.3341
Linear	**124.1**	**0.9959**	**0.9426**	**124.5**	**0.9966**	**0.9475**
Integral	123.9	0.9957	0.9088	124.2	0.9963	0.8918
LMF	123.9	0.9958	0.9161	124.2	0.9964	0.9044

Table 7: Test results of the image area in Figure 12

From the Table 7 we can see the quality measures of the area showed in the Figure 12. We notice that in this image, the linear method was the best method in terms of every evaluation criteria we used. Additionally, it seems that only the HaarPSI seems to catch that there is still some line artefact left in the image.

	Real Data			
Method	Fit RSE (%)	PSNR	SSIM	HaarPSI
Uncorrected	-	126.0	0.9093	0.3729
Linear	**4.45**	**137.272**	**0.9974**	**0.9695**
Integral	4.64	137.123	0.9972	0.9235
LMF	4.56	137.270	0.9973	0.9359

Table 8 supports the conclusion: linear method is clearly the best correction method in terms of PSNR, SSIM and HaarPSI. Furthermore, all the methods have similar PSNR and SSIM even though the difference is clear visually. This suggests that the HaarPSI is the best quality measure in terms of this application.

4 Discussion

4.1 Preliminary tests

The median filtering turned out to give the best reconstruction of the image, as can be seen in the table 1. However, after applying the median row-wise, as suggested in (2.7), the linear estimation gave slightly better estimation of the gain function than the median filtering, as can be seen in the Figure 5. More importantly, the linear estimation turned out to be very good estimation of the gain function overall.

In the exponential fitting test, the Barzilai-Borwein method failed miserably. The method got relatively close for one parameter, but after that, it failed to balance the step sizes and diverged to infinity. This was probably due to the small parameter values (both $|d_0|, |A| < 1$) and the vastly different growth rates along the parameter directions[15].

From the methods which did give a solution, the logarithm was clearly the weakest. One of the reasons for this may be because measured values were really close to one, hence when the noise is applied, the logarithm is sometimes taken from a negative number. This causes the result to be represented as a complex number and converting this back to a real number in Matlab, may have resulted in added inaccuracies.

According to this experiment, the Integral method, the Gauss-Newton and the LMF are all suitable candidates. As said before, the gradient descent starts to diverge after a couple of iteration, hence, LMF behaves like a Gauss-Newton almost immediately. Still, as we can see in the Table 4, adding a spike makes those couple gradient steps make the LMF-method significantly faster than the Gauss-Newton method. For that reason, we chose the integration and LMF method for the image correction tests.

4.2 Correction method tests

As can be seen in the Figure 8, all of the tested methods gave quite good and similar fitting result. In particular, the fitting methods seem to give very good estimations of d_0 most of the time. On the other hand the estimations of the parameter A seems to have a little more deviation.

As we can see in the Table 6, LMF-method seems to perform the best according to the first three criteria. However, according to HaarPSI, the straight linear estimation without any fitting appears to produce better quality images. The reason for this can be seen in the Figure 8. Here, the edge-most gain value gets slightly over valued. Since fitted function must follows the exponential function, the parameter A must be slightly under valued to ensure that the square error is minimized. Therefore, the fitted reconstruction is not as smooth as the straight linear reconstruction. The problem with this can be seen in the Figure 9. Here, one can barely see that the linear reconstructed image is just a little bit darker than the real image, whereas the fitted reconstructions have one row which is clearly worse than the real image, leaving some line artefacts.

Hence, although the fitted reconstructions are better in the least squares sense, they sometimes look worse, which is consistent with the fact that HaarPSI is constructed with the human perception in mind.

In the real data tests, we stumble upon similar results, although a bit more clearly. In the Figure 12, we can see that both fitting methods over correct the edge most pixel, leaving a visible stripe on the edge most row. In these tests the straight linear reconstruction method was the best according to every evaluation criteria. This further suggests that forcing the gain function estimate to be exponential does not improve the solution from the straight row-wise median of linear reference estimated gains.

4.3 Conclusion

We tested two different reference image generation methods and five different gain function estimation methods, four of which was based on exponential fitting. We found that although the LMF and integral fitting methods sometimes gave a better gain estimation in the least squares sense, the shape of the overall remaining error in these methods left some visible line artefacts, and thus gave visibly worse overall reconstructions. Still, this study is a good baseline for a better gap correction method. The results suggest that some kind of regularization method to spread our correction more evenly could improve the overall correction. However, our search for a suitable regularization method was unfruitful. Other direction we could go to would be a method where we alternate between solving the gain function and the image, following the idea proposed in the article *Stripe Noise Removal of Remote Sensing Images by Total Variation Regularization and Group Sparsity Constraint* by Yong Chen et al. [6].

References

[1] K Aditya Mohan, Robert M Panas, and Jefferson A Cuadra. "SABER: A Systems Approach to Blur Estimation and Reduction in X-ray Imaging". In: *arXiv preprint arXiv:1905.03935* (2019).

[2] Jennifer L Mueller and Samuli Siltanen. *Linear and nonlinear inverse problems with practical applications*. Vol. 10. Siam, 2012.

[3] Hervé Carfantan and Jérôme Idier. "Statistical linear destriping of satellite-based pushbroom-type images". In: *IEEE transactions on geoscience and remote sensing* 48.4 (2009), pp. 1860–1871.

[4] Jinsong Chen et al. "Destriping CMODIS data by power filtering". In: *IEEE Transactions on Geoscience and remote sensing* 41.9 (2003), pp. 2119–2124.

[5] Jinsong Chen et al. "Oblique striping removal in remote sensing imagery based on wavelet transform". In: *International Journal of Remote Sensing* 27.8 (2006), pp. 1717–1723.

[6] Yong Chen et al. "Stripe noise removal of remote sensing images by total variation regularization and group sparsity constraint". In: *Remote Sensing* 9.6 (2017), p. 559.

[7] Min Wang et al. "A unidirectional total variation and second-order total variation model for destriping of remote sensing images". In: *Mathematical Problems in Engineering* 2017 (2017).

[8] Lixin Sun et al. "Automatic destriping of Hyperion imagery based on spectral moment matching". In: *Canadian Journal of Remote Sensing* 34.sup1 (2008), S68–S81.

[9] Michael Wegener. "Destriping multiple sensor imagery by improved histogram matching". In: *International Journal of Remote Sensing* 11.5 (1990), pp. 859–875.

[10] Jean Jacquelin. *Regressions and integral equations.(April 2009), 16–17 pages*. 2009.

[11] R. Fletcher. *Modified Marquardt Subroutine for Non-Linear Least Squares.* Rpt. AERE-R 6799, Harwell.

[12] Zhou Wang et al. "Image quality assessment: from error visibility to structural similarity". In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612.

[13] Rafael Reisenhofer et al. "A Haar wavelet-based perceptual similarity index for image quality assessment". In: *Signal Processing: Image Communication* 61 (Feb. 2018), pp. 33–43. ISSN: 0923-5965. DOI: 10.1016/j.image.2017.11.001. URL: http://dx.doi.org/10.1016/j.image.2017.11.001.

[14] Ingrid Daubechies. *Ten lectures on wavelets*. Vol. 61. Siam, 1992.

[15] K Irene Snyder and Wesley E Snyder. *Determination of exponential parameters*. Tech. rep. North Carolina State University. Center for Communications and Signal Processing, 1991.