Automatic and efficient tomographic reconstruction algorithms

Lagerwerf, M.J.

Automatic and Efficient Tomographic Reconstruction Algorithms

Marinus Jan Lagerwerf

Automatic and Efficient Tomographic Reconstruction Algorithms

Proefschrift

ter verkrijging van

de graad van doctor aan de Universiteit Leiden,

op gezag van rector magnificus prof.dr.ir. H. Bijl,

volgens besluit van het college voor promoties

te verdedigen op dinsdag 5 oktober 2021

klokke 10:00 uur

door

Marinus Jan Lagerwerf

geboren te Wageningen

in 1990

Promotor:
Prof. dr. K. J. Batenburg

Copromotor:
dr. W. J. Palenstijn

Samenstelling promotiecommissie

Voorzitter:
Prof. dr. F.A. van der Duijn Schouten

Secretaris:
Prof. dr. S. J. Edixhoven

Overige leden:

Prof. dr. C.B. Schönlieb	University of Cambridge
Dr. C. Brune	Universiteit Twente
Prof. dr. R.H. Bisseling	Universiteit Utrecht

The research presented in this dissertation was carried out at Centrum Wiskunde & Informatica (CWI) in Amsterdam.

Financial support was provided by The Netherlands Organisation for Scientific Research (NWO), project number 639.073.506.

Contents

Chapter 1

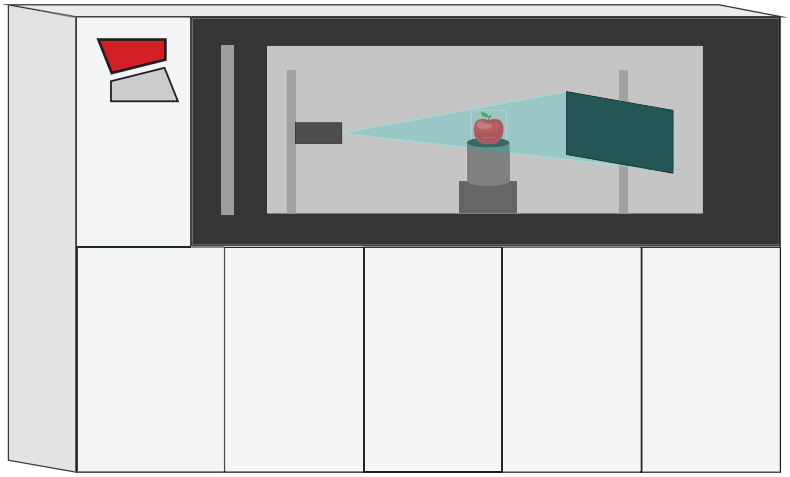
Introduction

Tomography deals with imaging the interior of an object without destroying it. It is a useful tool in many applications in science, industry and medicine. In tomography a penetrating wave is used to measure projection images of an object along different directions. These projection images are then used to determine the interior of the object, through a *tomographic reconstruction method* [KS01; Nat01; Her09].

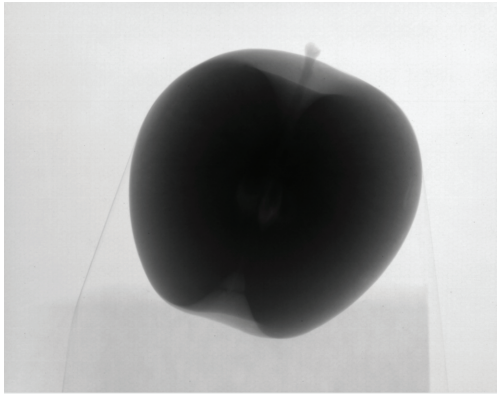
A popular type of tomography is *computed tomography* (CT), where X-rays are used as the penetrating waves to measure the projection images. Within CT imaging there many different types of scanners, such as medical CT scanners, μ-CT scanners (laboratory setup) and synchotron facilities. All these scanners follow a similar principle. A *source* generates X-rays which penetrate the measured object. Inside this object the X-rays are attenuated through interaction with the object. The amount of attenuation depends on the material properties of the object and the energy distribution of the X-rays. After interacting with the object the intensity of the X-ray beams is measured with a *detector* forming a projection image, see **Figure 1.1**a. This process is repeated for several source positions and detector positions[1] and all the resulting projection images combine into the *measured projection data*. This measured projection data is then used as the input for a CT specific reconstruction method that computes the interior of the object **Figure 1.1**b.

The problem of computing the interior of the object from the measured projection data is called the *reconstruction problem*. The difficulty of the reconstruction problem depends on the amount of information available and the uncertainty in the measured projection data, *e.g.*, the number of projection images and the noise

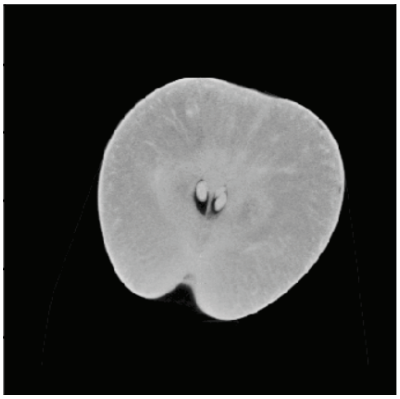
[1] In some applications the source and detector are fixed and the object rotates. This is equivalent to rotating the source and detector position.



(a) Illustration of a CT scanner.



(b) Projection image.



(c) 2D slice of a 3D reconstruction.

Figure 1.1: Examples of a CT scanner, a projection image and a reconstruction. (a) Illustration of a CT scanner. The source in the CT scanner generates X-rays penetrating the apple and the detector measures the intensity of the X-ray beam after interacting with the apple. (b) These measurements form a projection image. (c) From a collection of projection images a reconstruction can be computed showing the interior of the apple.

levels in the projection images. Consequently, the effectiveness of a reconstruction method depends on the amount of information and uncertainty in the measured projection data and how the reconstruction method handles this (lack of) information and uncertainty. For example, direct inversion reconstruction methods are

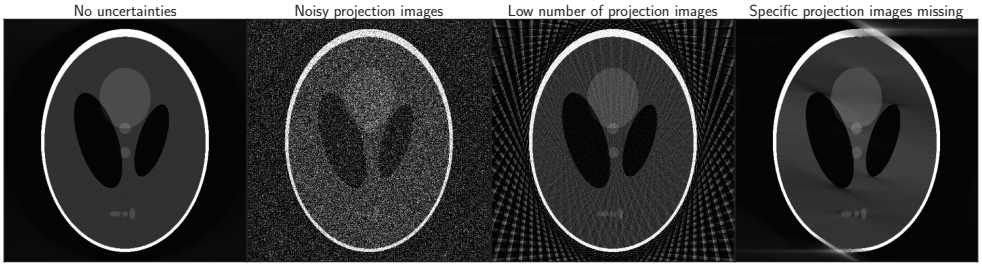


Figure 1.2: Reconstructions using a direct inversion reconstruction method (Filtered backprojection) with a different amount of information or uncertainties in the measured projection data. From left to right: 360 projection images with no noise (close to ground truth), 360 projection images with high noise levels (uncertain data), 36 projection images with no noise (insufficient data), the first 300 projection images from the left case (insufficient data).

derived with the assumption that there is enough information and no noise in the measured projection data and they may fail to compute accurate reconstructions when these conditions are not met. This is illustrated in **Figure 1.2**.

CT imaging is used in a broad spectrum of applications, such as industrial quality control [GUV11], materials sciences [Die+14; Bul+16] and medical imaging [For+02; GKT17]. However, the CT scans in these applications are typically processed *offline*, *i.e.*, the reconstructions are computed after scans are acquired. If instead the reconstruction can be computed while the scan is acquired, *i.e.*, in real-time or *online*, the operator of the CT scan can react to insights gained from the reconstruction. For example, practical problems related to the acquisition parameters — *e.g.*, misaligned source or detector, incorrect center of rotation, or incorrect field of view — could be fixed while scanning. In industrial quality control manual inspection could be replaced by real-time scanning and automatic removal from the assembly line. Furthermore, dynamic processes could be scanned and followed as they occur. Consider for example a dynamic process where external parameters such as pressure or heat are applied to an object. By following these processes in real-time the operator can adjust the parameters when specific events occur, such as the small cracks due to pressure or overheating.

In real-time CT imaging, there are limitations on the scanning time — *i.e.*, the scans should be acquired real-time — and on the reconstruction process — *i.e.*, computing the reconstructions should be real-time. This means for the scanning process that the exposure time per projection image should be low (this will lead to high noise levels) and the number of projection images should be small (this leads to

limited information). Consequently, this means for the reconstruction method that it should be able to compute an accurate reconstruction from projection data with limited amount of information and uncertain data in a short amount of time. This example shows that depending on the restrictions introduced by the application a different type of reconstruction method will be most effective.

The research presented in this thesis follows a plug-and-play strategy, *i.e.*, we focus on practical limitations of existing reconstruction methods and develop new strategies to make these methods more effective or easier to use. This strategy is suggested in [PSV09], because they observe that many promising reconstruction methods are not used in practice due to limited consideration on how to effectively apply these new reconstruction methods in practice.

Chapter 2 and **Chapter 3** of this thesis focus on developing mathematical frameworks to pick the correct parameter for reconstruction methods. It is often not clear how to choose the correct parameter and in some cases even the effect of the parameter choice is not clear. This means that in practice the parameter choice becomes a process of trial-and-error requiring manual tuning of the parameters and a good understanding of the reconstruction method. We develop a framework for streamlining the process of picking the correct regularization parameter for variational methods in **Chapter 2**. The idea of this framework is to efficiently compute approximations of reconstructions through pixel-wise interpolation for a broad range of regularization parameters. These approximations are then used to (visually) determine the correct regularization parameter. In **Chapter 3** we formulate an optimization problem which can be used to automatically compute a filter that is adapted to the measured projection data for the FDK algorithm. We show that these computed filters achieve similar performance as optimally smoothed standard filters.

In **Chapter 4** and **Chapter 5** we focus on developing reconstruction methods for 3D CT imaging applications where both reconstruction time and scanning time are a constraint, such as the examples discussed before. The challenge with the combination of these restrictions lies in balancing the reconstruction time and the reconstruction accuracy. This is because reconstruction methods that can accurately reconstruct measured projection data containing noise or a low number of projection data generally do not satisfy the reconstruction time constraints and vice versa. Therefore, we adapt existing filtered-backprojection reconstruction methods — which are known for their short reconstruction times — to improve their reconstruction accuracy for noisy projection data and data with a low number of projection images, while maintaining their computational efficiency. Specifically, we expand upon the Neural Network filtered-backprojection (NN-FBP) algorithm [PB13]. The NN-FBP algorithm is an adaptation of the standard FBP

algorithm where a machine learning component is added to greatly improve the reconstruction accuracy of the algorithm. In **Chapter 4** we extend the NN-FBP algorithm to the FDK algorithm and show that this is possible for any linear FBP-type method. In **Chapter 5** we fit the NN-FBP algorithm to a real-time quasi-3D reconstruction framework (RECAST3D) [Buu+18] and replace the supervised learning strategy with a semi-supervised learning strategy proposed in [HPB20]. This leads to the Noise2Filter (N2F) algorithm, a reconstruction method that can be trained on the fly and can compute arbitrarily oriented 2D slices of a 3D volume in real-time.

In this thesis we will use simulated and experimental data. We consider three scanning geometries: parallel beam, fan beam and circular cone-beam. The experimental fan and cone-beam data was acquired using the custom-built and highly flexible FleX-ray CT scanner, developed by XRE NV and located at CWI [Cob+20], and the experimental parallel beam data was taken from the public TomoBank repository [De +18]. More specifically, the fuel cell data used for the TomoChallenge, which was acquired at the TOMCAT beamline at the Swiss Light Source (Paul Scherrer Institut, Switzerland).

In the remainder of this chapter we give a mathematical description — continuous and discrete — of the tomographic reconstruction problem and introduce several reconstruction methods used throughout this thesis.

1.1 Tomographic reconstruction problem

This section gives a mathematical introduction to the idealized tomographic reconstruction problem. Specifically, we introduce the continuous formulation and the scanning geometries in **Section 1.1.1**, the discrete formulation of the reconstruction problem in **Section 1.1.2**, and discuss the Radon transform and the Ray transform in **Section 1.1.3**. Note that all following chapters are based on self-contained articles, each containing a separate introduction. Therefore, we give a more general introduction here.

1.1.1 Continuous formulation

We model the scanned object as a function $f : \mathbb{R}^n \to \mathbb{R}$ in the image function space X with $n \in \{2, 3\}$ and $f(x)$ representing the attenuation coefficient of the scanned object at position $x \in \mathbb{R}^n$. The photon count $I(l)$ for (monochromatic) X-rays traversing the object along the line l (called *X-ray line* from this point onwards) can be expressed in terms of the emitted photon count I_0 at the source and the

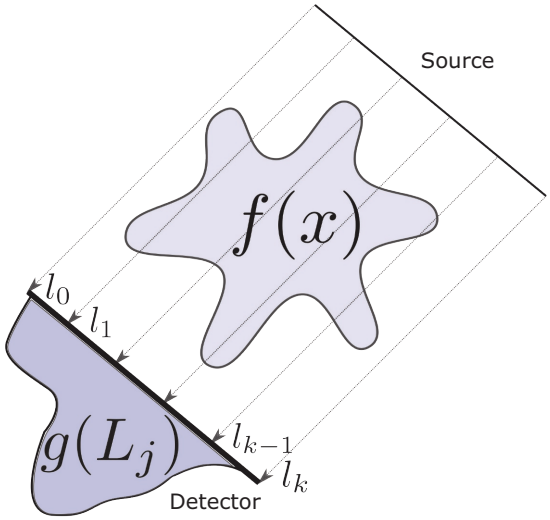


Figure 1.3: Schematic representation of a projection image for a 2D parallel beam CT scanning geometry. The dotted arrows represent the X-ray lines. We define the projection image $g(L_j)$ as the image formed by the values $g(l_i)$ with $l_i \in L_j$ the measured X-ray lines for a fixed source and detector position.

attenuation coefficient function f using the Beer-Lambert law:

$$I(l) = I_0 e^{-\int_l f(x)dx}, \tag{1.1}$$

where we assume that f is bounded with compact support.

We can simplify (1.1) by rearranging the terms and taking the logarithm:

$$-\log\left(\tfrac{I(l)}{I_0}\right) = \int_l f(x)dx. \tag{1.2}$$

This is the linearized photon count along the X-ray line l, which we will denote by $g(l)$. Representing the projection data by this function $g \in Y$ enables us to formulate the linear forward operator $K : X \to Y$, with Y the projection data function space. More specifically,

$$g(l) = -\log\left(\tfrac{I(l)}{I_0}\right), \qquad (Kf)(l) = \int_l f(x)dx. \tag{1.3}$$

Up till now we have only considered one X-ray line l, however, the projection data is known for a (possibly infinite) set of measured X-ray lines l.

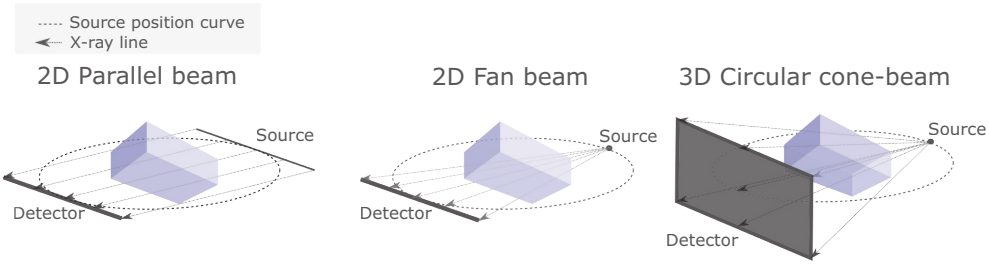


Figure 1.4: Different scanning geometries. In the 2D parallel beam and 2D fan beam scanning geometries, each projection image is a 1D image and the reconstruction is a 2D slice of the measured object, whereas in the 3D circular cone-beam geometry each projection image is a 2D image and the reconstruction is a 3D image of the measured object.

Let us define the set of measured X-ray lines for a fixed source and detector position as L_j. The projection image j is the image formed by the linearized photon counts $g(l_i)$ along the X-ray lines $l_i \in L_j$, which we denote (with a small abuse of notation) by $g(L_j)$. In a similar fashion we define L as the set of all measured X-ray lines for all considered source and detector positions and $g(L)$ then forms the projection data.

We can now formulate the tomographic reconstruction problem. Given the forward operator $K : X \to Y$ and projection data $g \in Y$ find a function $f \in X$ that satisfies:

$$g(l) = (Kf)(l), \qquad \text{for all } l \in L. \tag{1.4}$$

Note that this is an idealized version of the reconstruction problem as we assume there are no practical problems, such as measurement noise, photon scattering or misalignment of the source and the detector.

The set L is determined by the position and properties of the detector — *e.g.*, the shape, number of detector pixels and the physical size of the detector — and the position and properties of the X-ray source — *e.g.*, multiple X-ray sources emitting parallel beams, or a point source emitting X-rays in all directions. These properties form the *scanning geometry*. Common examples of scanning geometries are parallel beam, fan beam for 2D tomographic problems and circular cone-beam for 3D tomographic problems. Schematic representations of these scanning geometries are given in **Figure 1.4** for a fixed source and detector position.

An essential operator for the tomographic reconstruction problem is the adjoint $K^* : Y \to X$, or *backprojection operator*, of the forward operator K. This operator

is defined through the following condition:

$$\langle Kf, g\rangle_Y = \langle f, K^* g\rangle_X, \qquad \text{for all } f \in X, g \in Y. \tag{1.5}$$

The backprojection operator is used in many theoretical derivations and reconstruction algorithms [Nat01]. In **Section 1.1.3** we derive the explicit forms for the backprojection operators.

1.1.2 Discrete formulation

In practice only a finite number of X-ray lines can be measured, *i.e.*, the number of elements of the set L is finite. Therefore, it is natural to consider the projection data as a finite dimensional vector $\mathbf{y} \in \mathbb{R}^M$ with each element relating to the linearized measured photon count along an X-ray line l, and M the number of X-ray lines in L. For the discretization of the scanned object we assume that our object is contained in a rectangular box and discretize this box in a number of elements, called *pixels* or *voxels*, for 2D or 3D objects, respectively. We define a vector $\mathbf{x} \in \mathbb{R}^N$, where the elements of the vector correspond to the attenuation coefficient on the elements of the discretized box and let N denote the number of elements in this box.

A natural way of relating $\mathbf{y}_i$ — the i-th element of $\mathbf{y}$ — to the vector $\mathbf{x}$ is by approximating a line integral through the discretized box over the line l_i by a weighted sum over the elements of $\mathbf{x}$. More specifically, given the weight vector $w_i \in \mathbb{R}^N$ with elements $w_{ij} \in \mathbb{R}$, we have the relation:

$$\mathbf{y}_i = \sum_{j=1}^{N} w_{ij}\mathbf{x}_j = w_i^T \mathbf{x}. \tag{1.6}$$

The implementation choice of approximating the line integral leads to different weight vectors $w_i \in \mathbb{R}^N$. Moreover, the approximation can be adapted to attain desirable numerical properties or achieve better performance [Aar+15].

If we consider the matrix $W \in \mathbb{R}^{M\times N}$ with rows w_i, we can relate the vector $\mathbf{x}$ to $\mathbf{y}$ and formulate the discrete tomographic reconstruction problem: given measured projection data $\mathbf{y} \in \mathbb{R}^M$ find a reconstruction $\mathbf{x} \in \mathbb{R}^N$ such that the following holds approximately:

$$W\mathbf{x} = \mathbf{y}. \tag{1.7}$$

This matrix W is often referred to as the *projection matrix*.

Analogously to the continuous formulation, the adjoint[2] of W is the backprojection operator W^T, which is essential in many reconstruction methods.

[2] As W is real-valued, the adjoint is equivalent to the transpose.

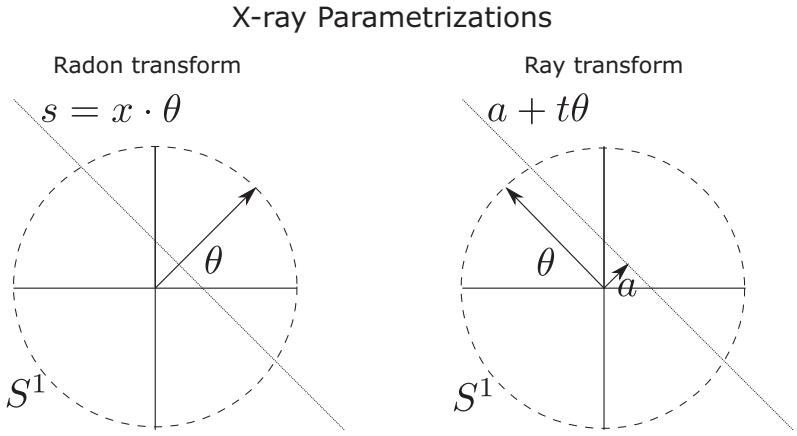


Figure 1.5: Illustrations of the X-ray parametrizations for the Radon transform and the Ray transform in $\mathbb{R}^2$.

Although we did not consider practical problems in these idealized formulations of the reconstruction problems, they are often present in reality. These practical problems introduce additional uncertainties to the reconstruction problem. Specifically, the reconstruction problem might not have a solution, it might have multiple solutions, or the solution varies heavily with respect to changes in the projection data. Therefore, it is important to develop reconstruction methods that take these challenges into consideration.

1.1.3 The Radon and Ray transform

In this section we focus on the explicit expressions for the Radon and the Ray transform and derive the adjoint for both. Note that most of this section goes beyond the scope of the main chapters of this thesis.

The Radon transform integrates a function f on $\mathbb{R}^n$ over hyperplanes. More specifically, given a hyperplane $\{x \in \mathbb{R}^n | s = x \cdot \theta\}$, with $\theta \in S^{n-1}, s \in \mathbb{R}$ and S^{n-1} the unit sphere in $\mathbb{R}^n$, the Radon transform is:

$$(Rf)(\theta, s) = \int_{\mathbb{R}^n} \delta(s - x \cdot \theta) f(x) dx \tag{1.8}$$

Since hyperplanes in $\mathbb{R}^2$ are just lines we can conclude that the Radon transform is in fact identical to the forward operator K from (1.3) for $\mathbb{R}^2$. For $\mathbb{R}^3$, however, any hyperplane is a plane, showing us that the Radon transform does not fit the forward model for the 3D reconstruction problem.

For the Ray transform a more general line parametrization is used, see **Figure 1.5** (right). Given a direction $\theta \in S^{n-1}$ of a line in $\mathbb{R}^n$, we define the hyperplane $\theta^\perp$ in $\mathbb{R}^n$, which is the hyperplane orthogonal to the vector θ, *i.e.*,

$$\theta^\perp := \left\{ a \in \mathbb{R}^n |\ a \cdot \theta = 0, \theta \in S^{n-1} \right\}. \tag{1.9}$$

The parametrization of the line then becomes for an $a \in \theta^\perp$ and $\theta \in S^{n-1}$:

$$l(\theta, a) := \{ a + t\theta | t \in \mathbb{R} \}. \tag{1.10}$$

Note that this parametrization can be used in $\mathbb{R}^n$ for $n \geq 2$. If we subsitute this parametrization in the right-hand side of (1.4) we get the explicit expression for the Ray transform:

$$(Pf)(\theta, a) = \int_{\mathbb{R}} f(a + t\theta) dt. \tag{1.11}$$

To conclude this section, we derive the expressions for the adjoint operators related to the Radon and the Ray transform using (1.5). Following [Nat01] we take X and Y to be a Schwartz space $\mathscr{S}(\cdot)$ on the domain of f and g, respectively.

For the Radon transform we use the definition of the Dirac δ function to get:

$$\langle Rf, g \rangle_{\mathscr{S}(S^{n-1} \times \mathbb{R})} = \int_{S^{n-1}} \int_{\mathbb{R}} \int_{\mathbb{R}^n} \delta(s - x \cdot \theta) f(x) g(\theta, s) dx ds d\theta, \tag{1.12}$$

$$= \int_{\mathbb{R}^n} \int_{S^{n-1}} f(x) g(\theta, x \cdot \theta) d\theta dx = \langle R^* g, f \rangle_{\mathscr{S}(\mathbb{R}^n)}, \tag{1.13}$$

$$(R^* g)(x) = \int_{S^{n-1}} g(\theta, x \cdot \theta) d\theta. \tag{1.14}$$

In a similar fashion we derive the adjoint operator for the Ray transform. Note that in this case the domain of g differs from the Radon transform, *i.e.*, $g \in \mathscr{S}^2(S^{n-1} \times \theta^\perp)$.

In this derivation we substitute $x = a + t\theta$, which implies[3] $t = x \cdot \theta$, *i.e.*,

$$\langle Pf, g \rangle_{\mathscr{S}(\mathscr{S}(S^{n-1}\times\theta^{\perp}))} = \int_{S^{n-1}} \int_{\theta^{\perp}} \int_{\mathbb{R}} f(a+t\theta) g(\theta, a) dt da d\theta, \tag{1.15}$$

$$= \int_{\mathbb{R}^n} \int_{S^{n-1}} f(x) g(\theta, x - (x\cdot\theta)\theta) d\theta dx, \tag{1.16}$$

$$= \int_{\mathbb{R}^n} \int_{S^{n-1}} f(x) g(\theta, E_\theta(x)) d\theta dx = \langle P^* g, f \rangle_{\mathbb{R}^n}, \tag{1.17}$$

$$(P^* g)(x) = \int_{S^{n-1}} g(\theta, E_\theta(x)) d\theta, \tag{1.18}$$

with $E_\theta(x) = x - (x \cdot \theta)\theta$ the orthogonal line projection of x onto $\theta^{\perp}$.

If we compare the adjoint operators for the Radon ($n = 2$) and Ray transform, we observe similar behavior. Specifically, the adjoint operator computes the integral of the measured projection data over all X-ray lines that contain the point x. In practice this means that the adjoint, or backprojection, *smears out* the measured projection data over the reconstruction volume.

Further details about the Radon, Ray transform are given in [Nat01].

1.2 Reconstruction methods

In this thesis many different reconstruction methods are used. In this section we give an introduction to these methods. We mainly consider the discretized versions and omit detailed derivations.

Reconstruction methods can roughly be subdivided in three categories: (1) Direct methods, which are often based on an (approximate) closed form inverse of the operator K, (2) Iterative methods, which solve the tomographic reconstruction problem by using an iterative optimization scheme, (3) Machine learning methods, which use a data-driven approach to remove artifacts from reconstructions or improve existing reconstruction methods.

1.2.1 Direct methods

Direct reconstruction methods are closed form and are often designed for a particular scanning geometry, limiting their general use. Examples of direct methods are GridRec [OSu85], Katsevich [Kat03], the filtered backprojection algorithm [Nat01], and the Feldkamp-Davis-Kress algorithm [FDK84].

[3] This equality is found by taking the inner product with θ of $x = a + t\theta$ on both sides and using $a \in \theta^{\perp}$

Filtered backprojection algorithm

The filtered backprojection (FBP) algorithm is derived for the 2D tomographic reconstruction problem using properties of the Radon transform. Specifically, consider **Theorem 2.3** from [Nat01]. This theorem states that, given functions $f, V \in X$ and $g, v \in Y$ that satisfy $g = Rf$ and $V = R^*v$ — with R and R^* the Radon transform and its backprojection operator, respectively — the following holds true:

$$V * f = R^*(v * g). \tag{1.19}$$

with $*$ denoting a two-dimensional convolution over $\mathbb{R}^2$ on the left-hand side and a one-dimensional convolution over $\mathbb{R}$ along the detector width on the right-hand side.

By taking v to be the inverse Fourier transform ($\mathscr{F}^{-1}$) of the absolute value of the frequencies in the Fourier domain, *i.e.*, $v = \mathscr{F}^{-1}\{|\xi|\}$, one can show that V is the Dirac δ function. This means that for this choice of v the left-hand side of (1.19) simplifies to f and the right-hand side is the expression used for the Filtered Backprojection (FBP) algorithm. This function v is called the *ramp filter* and with an additional cut-off function it is referred to as the *Ram-Lak filter* [RL71]. Note that if g does not contain noise and is available for all possible measured X-ray lines (1.19) is an exact inversion of the Radon transform.

Similar to (1.19) we can formulate a discrete expression used in the FBP algorithm in terms of $\mathbf{x}, \mathbf{y}, W$ and the discretized ramp filter $\mathbf{h}_\mathrm{r}$:

$$\mathbf{x}_\mathrm{FBP} = W^T(\mathbf{y} * \mathbf{h}_\mathrm{r})_{1D}, \tag{1.20}$$

with the convolution $(\cdot * \cdot)_{1D}$ applied in one dimension over the detector width.

The FBP algorithm can be applied to the 2D parallel beam and fan beam scanning geometry. Moreover, it can be applied to the 3D parallel beam and 3D fan beam scanning geometries as these can be considered a stack of their respective 2D reconstruction problems.

Feldkamp-Davis-Kress algorithm

The Feldkamp-Davis-Kress (FDK) algorithm [FDK84] is a reconstruction algorithm for the circular cone-beam scanning geometry. This geometry does not satisfy the Tuy-Kirrilov condition [Tuy83] meaning that it inherently has insufficient information for unique inversion. Therefore, instead of directly inverting the Ray transform for this geometry, the authors propose considering the cone-beam projection data as a stack of fan beam data and using the FBP algorithm for fan beam data in combination with a reweighting of the data that aims to compensate

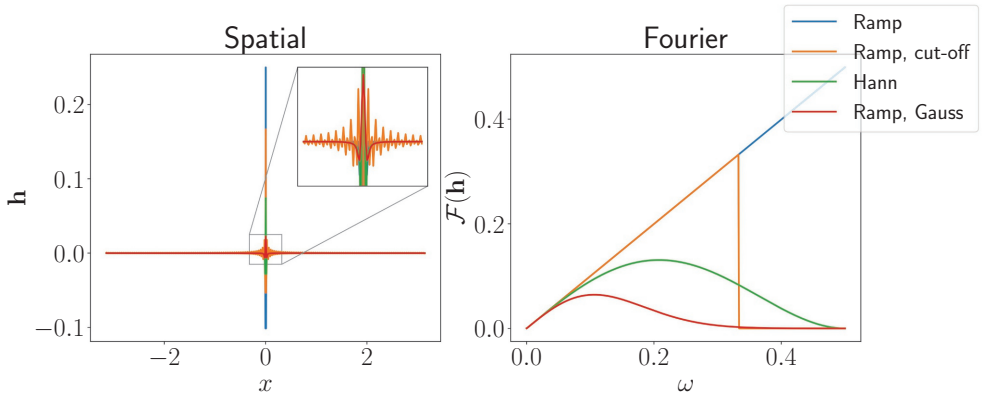


Figure 1.6: Examples of standard filters and adapted standard filters.

for the mismatch between this assumption and the actual cone-beam geometry. The expression used in the FDK algorithm in terms of $\mathbf{x}, \mathbf{y}, W$ and $\mathbf{h}_r$ is:

$$\mathbf{x}_{\text{FDK}} = W^T(r(\mathbf{y}) * \mathbf{h}_r)_{1D}. \tag{1.21}$$

with $r(\cdot)$ the reweighting operator. Note that W^T is the backprojection operator related to the 3D cone-beam geometry and not the backprojection operator related to the 2D fan beam geometry.

Filter adaptation for FBP-type methods

The ramp filter follows from a theoretical result based on the assumption that there is no noise in the measured projection data. However, this is often not the case in practice, therefore, filter adaptations to the ramp filter have been suggested, such as the Shepp-Logan filter, the Cosine filter, and the Hann filter. These filters put less emphasis on higher frequencies to reduce the noise in the reconstruction. This strategy can be taken further by applying smoothing filters to these filters — such as Gaussian or Binomial filters — or cutting the higher frequencies of by setting their contributions to zero. Examples of these standard and adapted filters are given in **Figure 1.6** and examples of reconstructions of noisy projection data is shown in **Figure 1.7**.

1.2.2 Iterative methods

Iterative methods aim to solve the tomographic reconstruction problem by refining the solution over a number of iterations. Note that these methods often do not rely

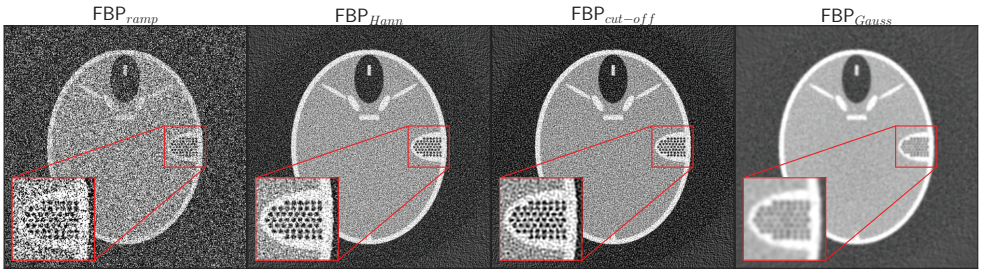


Figure 1.7: Examples of reconstructions with standard filters and adapted standard filters applied to projection data with high noise levels. We see that the filters with less emphasis on the high frequencies contain less noise at the cost of smoother edges. These reconstructions were computed within a second.

on specific properties of the forward operator except for being a linear operator, meaning that these methods can be applied to reconstruction problems with any scanning geometry. There are many different iterative methods, *e.g.*, SIRT [VV90], (S)ART [Kac37; GBH70; AK84], ICD [Wat94], and variational methods [ROF92; GHO99; BKP10; Goc16]. We highlight SIRT and variational methods as these will be used throughout the thesis.

SIRT

The SIRT algorithm [VV90] is a common iterative method for CT reconstruction. An iteration in the SIRT algorithm is defined as follows:

$$\mathbf{x} \leftarrow \mathbf{x} + \omega C W^T R(\mathbf{y} - W\mathbf{x}). \tag{1.22}$$

with $\omega \in \mathbb{R}$ an optional relaxation parameter and

$$C = \operatorname{diag}(\mathbf{c}), \qquad c_j^{-1} = \sum_i W_{ij}, \tag{1.23}$$

$$R = \operatorname{diag}(\mathbf{r}), \qquad r_i^{-1} = \sum_j W_{ij}. \tag{1.24}$$

Simple prior information, such as non-negativity, can be used to improve the results of the SIRT algorithm. The update with this non-negativity constraint becomes:

$$\mathbf{x} \leftarrow \max(\mathbf{x} + \omega C W^T R(\mathbf{y} - W\mathbf{x}), 0). \tag{1.25}$$

We will refer to SIRT with a non-negativity constraint as SIRT$^+$.

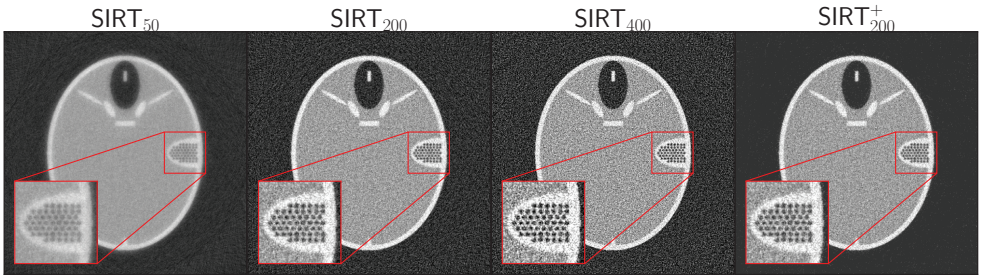


Figure 1.8: Examples of reconstructions with the SIRT algorithm applied to projection data with high noise levels. We see that fewer iterations lead to less noise, but also lower contrast and blurrier edges in comparison to higher iteration reconstructions. Moreover, we observe that the background for SIRT with non-negativity is almost noiseless. These reconstructions are more than a hundred times slower than the earlier shown FBP reconstructions.

In **Figure 1.8** we show example reconstructions of the SIRT algorithm. Here the subscript indicates the number of iterations that were computed.

Note that the SIRT algorithm is closely related to gradient descent applied to the standard least squares problem related to (1.7). By taking $C = R = Id$ instead of the above definition the iteration coincides with an gradient descent iteration applied to the least squares problem.

Variational methods

Variational methods are a class of iterative methods where the reconstruction is the solution to a minimization problem related to the tomographic reconstruction problem. A general (discrete) formulation for the minimization problem is

$$\mathbf{x}_{\mathrm{VM},\lambda} = \underset{\mathbf{x}}{\operatorname{argmin}} \left\{ \mathscr{D}(W\mathbf{x}, \mathbf{y}) + \lambda \mathscr{R}(\mathbf{x}) \right\}. \tag{1.26}$$

with $\mathscr{D}$ the data fidelity term, $\mathscr{R}$ the regularizer and $\lambda > 0$ the regularization parameter. The data fidelity measures the distance between the data and the forward projection of the reconstruction $W\mathbf{x}$. A common choice is simply the squared difference $\mathscr{D}(W\mathbf{x}, \mathbf{y}) = \|W\mathbf{x} - \mathbf{y}\|_2^2$. The regularizer promotes certain properties of the reconstruction $\mathbf{x}$, for example taking $\mathscr{R}(\mathbf{x}) = \| \, |\nabla \mathbf{x}| \, \|_2$ promotes piece-wise constant reconstructions, because $\mathscr{R}(x)$ is large for $\mathbf{x}$ with a large gradient. The regularization parameter λ balances the data fidelity and the regularizer, *i.e.*, taking λ close to zero emphasises the data fidelity and taking λ large emphasises the

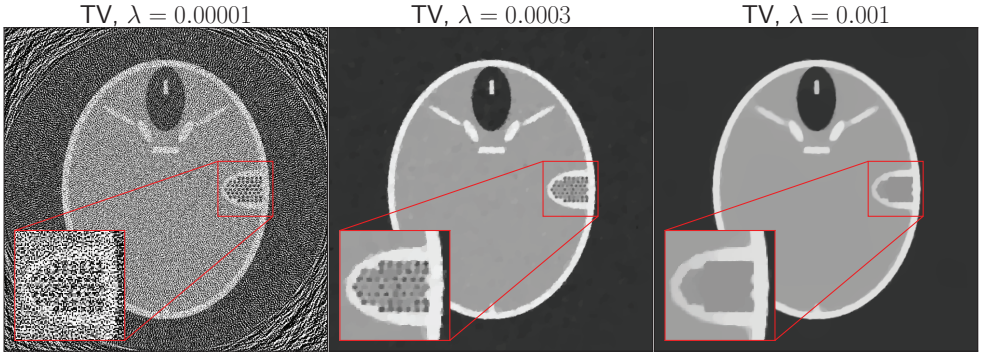


Figure 1.9: Examples of reconstructions with TV regularization applied to projection data with high noise levels. We see that taking λ too low will lead to noisy reconstructions and taking λ too high will lead to over-regularized reconstructions. These reconstructions were computed within a minute.

Method	Data fidelity	Regularizer
Tikhonov regularization	$\|W\mathbf{x}-\mathbf{y}\|_2^2$	$\|\mathbf{x}\|_2^2$
Sobolev regularization	$\|W\mathbf{x}-\mathbf{y}\|_2^2$	$\|\,\|\nabla\mathbf{x}\|\,\|_2^2$
Total Variation (TV) regularization	$\|W\mathbf{x}-\mathbf{y}\|_2^2$	$\|\,\|\nabla\mathbf{x}\|\,\|_1$

Table 1.1: Variational methods with the corresponding terms for the data fidelity and the regularizer.

regularizer. Examples of the effect of λ on the TV regularization reconstructions are shown in **Figure 1.9** for noisy projection data.

Some common variational methods with the corresponding choices for data fidelity and regularizer are given in **Table 1.1**. Depending on the choices for $\mathscr{D}$ and $\mathscr{R}$ the properties of (1.26) differ and different optimization schemes are needed. For example, gradient descent can be used for optimizing Tikhonov and Sobolev regularization, whereas for TV regularization FISTA [BT09] or PDHG [CP11] schemes are needed. The choice for regularization parameter λ depends on many different properties of the reconstruction problem and the variational method.

1.2.3 Machine learning methods

Using *machine learning* methods is an emerging approach in CT imaging [Wan+18] and has shown promising results for many applications within the development of

CT reconstruction methods [KMY17].

One well established strategy is training a network to remove the artifacts from the output of a standard reconstruction method. This is often called *post-processing* [RFB15; KMY17; PS18; Jin+17]. The promise of these methods is aided by the fact that the post-processing problem can be viewed as a classic image enhancement problem — *e.g.*, denoising, inpainting, or deblurring — for which many effective machine learning methods have already been developed [SLD17; PCC18; Zha+17]. We will introduce the general post-processing strategy below, because this strategy is used in the main chapters of this thesis.

Another strategy is incorporating machine learning components in existing reconstruction methods. Examples of these are variational networks [Kob+17; Ham+18], plug and play priors [VBW13; REM17; RS18] and learned regularizers [LÖS18; Muk+20]; all introduce a machine learning component to various variational methods. Additionally, in [SLX+16; AÖ18; WKL19] a network is proposed that learns an iterative scheme. Lastly, for direct methods the Neural Network Filtered-backprojection (NN-FBP) [PB13] and Neural Network FDK algorithms (**Chapter 4**) were developed.

Post-processing

In general the idea is to find an image-to-image mapping that can remove artifacts from a reconstruction. This mapping is found by defining a set of possible functions — *i.e.*, fixing a neural network architecture — and determining the best functions from this set by determining the best possible mapping on problems for which we know the answer — *i.e.*, using *supervised learning* [HTF09] to determine the network parameters. The network architectures used for post-processing are often *convolutional neural networks* (CNNs), which means that the set of possible functions is a concatenation of convolutions where the weights of the convolution are learned.

More specifically, given a network architecture CNN_Θ with *trainable parameters* Θ and a *training set* containing T *training pairs*, where a training pair consists of an input reconstruction with artifacts $\mathbf{x}_{i,rec}$ and a target reconstructions $\mathbf{x}_{i,target}$ without artifacts. We can train the network CNN_Θ by finding the $\Theta^\star$ that minimizes the *loss function*:

$$\sum_{i=1}^{T} \|\mathrm{CNN}_\Theta(\mathbf{x}_{i,\,rec}) - \mathbf{x}_{i,target}\|_2^2. \tag{1.27}$$

Using the trained network $\mathrm{CNN}_{\Theta^\star}$ we can remove artifacts from a reconstruction

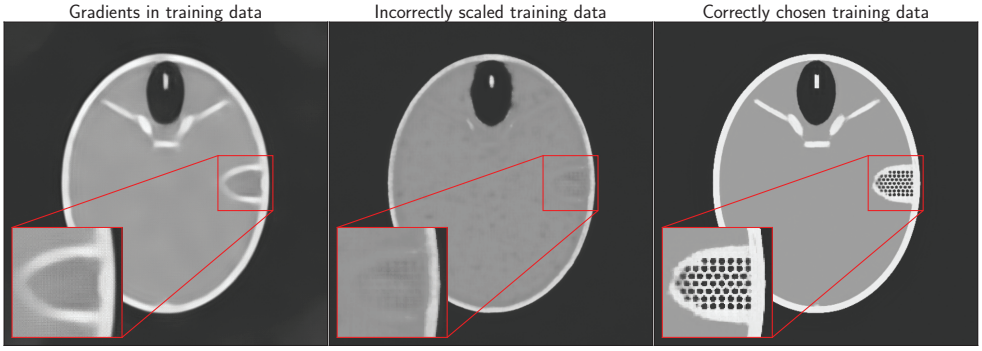


Figure 1.10: Examples of MSD networks trained on different training data applied to the same FBP reconstruction with high noise levels. All training data had the same noise levels. (Left) Training data contained gradients in the measured objects. (Middle) Training data contained piece-wise constant measured objects, but with a different scaling. (Right) Training data contained piece-wise constant objects with the correct scaling. Training a network took roughly 6 hours and computing a reconstruction took roughly a second.

$\mathbf{x}_{\text{rec}}$ similar to the input reconstructions in the training set:

$$\mathbf{x}_{\text{post-process}} = \text{CNN}_{\Theta^\star}(\mathbf{x}_{rec}). \tag{1.28}$$

One can vary the network architecture, loss function, training procedure, training data and corresponding hyper parameters and all these choices will lead to post-processing methods with different properties.

Examples of reconstructions using different types of training data are shown in **Figure 1.10**. We see that the networks are sensitive to changes in the training data.

1.2.4 The process of computing a reconstruction

In this section we will discuss the process of computing a reconstruction and the influence of reconstruction parameters on the ease of use of a reconstruction method.

From the introduction of the reconstruction methods we observe that all reconstruction methods have some kind of set of reconstruction parameters. For example: for the direct methods a filter has to be chosen, for iterative optimization schemes the number of iterations and the step-size parameter have to be set, the choice for regularization parameter is key for variational methods, and for

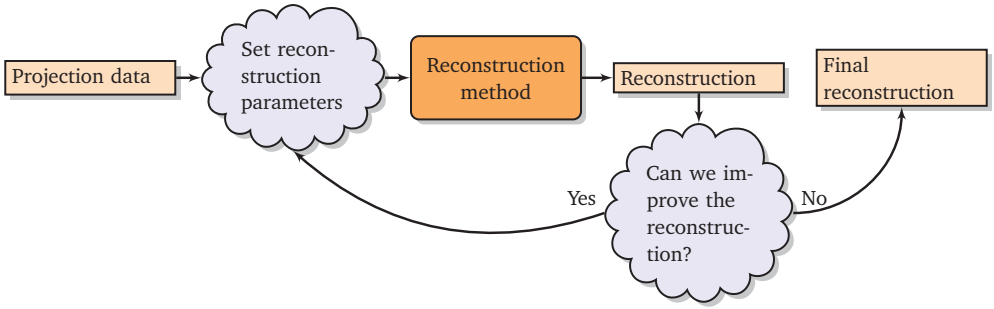


Figure 1.11: Schematic representation of the process of computing a reconstruction. The more time consuming it is to set the reconstruction parameters and compute the reconstruction, the more cumbersome it is to pick an almost-optimal set of reconstruction parameters.

machine learning methods the network architecture needs to be determined and the corresponding weights $\Theta^\star$ have to be learned for the chosen training data. Moreover, we have seen in the previous sections that the choice of these reconstruction parameters can strongly influence the accuracy of the reconstruction method. Following this reasoning we give a schematic representation of the process of computing a reconstruction in **Figure 1.11**. From this representation we can conclude that the harder it is to pick a suitable set of reconstruction parameters — *e.g.*, due to the number of possible choices or the time it takes to set the reconstruction parameters and compute a reconstruction — the more cumbersome the reconstruction process becomes.

If we now compare the reconstruction accuracy for different reconstruction methods — see **Figure 1.12** — we see that TV regularization and FBP + MSD reconstructions produce the most accurate results. However, recall from **Figure 1.9** and **Figure 1.10** that these methods are also the most time consuming methods and that the accuracy of these methods strongly depends on the choice of reconstruction parameters. Consequently, these methods are harder and more involved to use, especially for users with limited experience. This reiterates the importance of developing methods that are easy to use and perform similar to state-of-the-art methods or methods that improve the ease of use of an existing reconstruction method.

1.3 Outline of the thesis

This thesis is structured as follows: chapters 2 through 5 are based on self-contained research articles. Although these chapters have been edited slightly, they can be

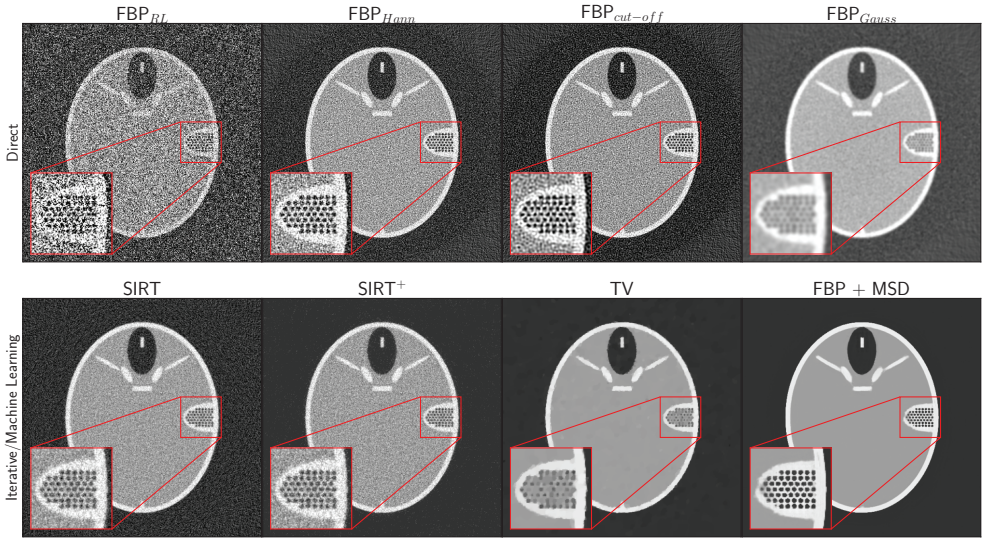


Figure 1.12: Comparison of different reconstruction methods applied to projection data with high noise levels. The reconstructions shown here are the 'optimal' reconstructions shown earlier in the section.

read independently. In **Chapter 2** we develop a framework for streamlining the process of picking the correct regularization parameter for variational methods. In **Chapter 3** we formulate an automatic algorithm for computing a data-dependent filter for the FDK algorithm. The NN-FBP algorithm is extended to the FDK algorithm in **Chapter 4** and combined with the Noise2Inverse training strategy in the RECAST3D framework in **Chapter 5**. A conclusion and outlook is given in **Chapter 6**, followed by the bibliography, a layman summary in Dutch and my Curriculum Vitae.

Chapter 2

An interpolation approach for determining regularization parameters

2.1 Introduction

Tomography is a generic 3D imaging technique for reconstructing the interior of an object from a series of its projections. Projections can be acquired using a broad variety of modalities, such as Computed Tomography (CT) [Nat01], Magnetic Resonance Imaging (MRI) [Fes10] and Electron Microscopy (EM) [Mid+01]. The resulting image reconstruction problems all have a similar mathematical problem structure: given a set of tomographic measurements and a description of the physics process, determine a reconstruction of the measured object. If many projections are available over a full angular range around the object, and if the projections have low noise, the reconstruction problem can be solved in a straightforward way by closed-form inversion techniques. For an overview see [Nat01; KS01]. In practice, however, the number of tomographic measurements is typically limited and the measurements can contain substantial noise.

This chapter is based on:

An Efficient Interpolation Approach for Exploring the Parameter Space of Regularized Tomography Algorithms. *MJ Lagerwerf, WJ Palenstijn, F Bleichrodt, KJ Batenburg.* Fundamenta Informaticae (Volume: 172), number 2, pp. 143–167, 2020.

In such limited data cases the information from the measurements and the geometry of the acquisition setup is not sufficient to solve the reconstruction problem accurately and some form of prior knowledge about the object must be incorporated in the solution process. One way of incorporating such prior knowledge in the reconstruction method is the use of regularization, where the balance between the prior knowledge and solving the original problem is usually determined by a regularization parameter [ROF92; BKP10; Goc16].

The choice for this regularization parameter depends on many properties of the problem, such as the measured data and its noise level, or the reconstruction method and its implementation. Moreover, the desired choice of regularization parameter may also be application-specific, for instance with the aim of reconstructing particular image features as sharply as possible at acceptable noise levels, creating a reconstruction that is well-suited for subsequent segmentation, etc.

Consider for example the Total Variation (TV) reconstruction algorithm implemented with a Primal-Dual Hybrid Gradient (PDHG) algorithm initially proposed by Chambolle, Pock, Bischof and Cremers, as described in [SJP12]. Given all the information about the reconstruction algorithm and its implementation, the effect of a particular choice of the regularization parameter on the reconstructed image still varies significantly for different instances of the reconstruction problems. If it is not possible to specifically define and model additional information about the object, it is common that the algorithm user computes a series of reconstructions for different choices of the regularization parameter and chooses the preferred setting by visual inspection. Although there have been strategies proposed to handle such parameter space exploration in a structural manner, such as described in [Sed+14; Pre+11], these strategies do not specifically address the key drawbacks we encounter here. The key drawbacks of such a trial-and-error method are twofold: (i) computing many reconstructions for different regularization parameter values is computationally intensive, as even computing a single reconstruction can already be computationally demanding; (ii) with only a small number of reconstruction evaluations, it is difficult to choose a regularization parameter in a consistent manner as the actual desired value may lie somewhere in between the sampling points.

For variational methods, of which TV is a well known example, there has been extensive theoretical work on how to determine the “optimal” regularization parameter, where each approach has a different concept of optimality. For example, for the Tikhonov method, explicit analytic expressions are found based on the singular value decomposition and discrepancy principle, see chapter 3 and 7 of [Sch+09], and [Vai82]. Moreover, substantial analytical results have been obtained on how the properties of a reconstruction change depending on the value of the

regularization parameters [Bur+13; Bur+16; Bri+18]. Although these results are powerful, they also make strong assumptions on the reconstruction algorithm, such as continuity of the solution with respect to the regularization parameter, full convergence of the iterative algorithm, or the availability of certain prior knowledge about the problem such as the noise level, which are not always valid in practice. Other more general strategies, such as the discrepancy principle and the L-curve criterion [Vai82; BM12; Han92; HO93; Han99], have been developed. These methods also rely on a specific definition of the "optimal" reconstruction and require many evaluations of the reconstruction algorithm. A key limitation of all mentioned approaches is that they do not take the application-specific needs into account. The criterion of optimality is based on a mathematical problem formulation without involving the particular requirements of the user.

In this chapter we propose an algorithmic approach for computationally efficient exploration of the regularization parameter space. Once a relatively small number of reconstructions have been computed for a sparse sampling of the regularization parameters, an approximation of the reconstructed image for other parameter values can be computed with very high efficiency (linear time in the number of pixels). In the case of manual selection of the regularization parameter, our approach makes it possible to present the user with a real-time interface where parameters can be adjusted on-the-fly and immediate visual feedback is obtained on the effect of parameter changes on the reconstruction. In the case of automated selection, the output of our approximation method can be used as input for any image-based quality metric that one wants to optimize for.

Accurate approximation of the output of general regularization reconstruction methods is a difficult problem. However, we found that if the output of the reconstruction algorithm is available for just a small number of regularization parameter values, a pixel-wise interpolation scheme is highly suitable for such approximations. The choice of the sampling scheme is of particular importance to the effectiveness of our approach. Through computational experiments we found that although the major changes of the reconstructed image with respect to the regularization occur in a relatively narrow region of the space of regularization parameter values, a logarithmic sampling and corresponding interpolation scheme results in relatively smooth behavior of the pixel values with respect to the regularization parameter choice.

Our experimental results on simulated data for the parallel beam tomography problem demonstrate that for three common variational reconstruction methods, our approach results in accurate approximations of the reconstructed image and that it can be used in combination with existing approaches for choosing optimal regularization parameters. We also provide results for an experimental X-ray CT

dataset. The approach is presented in such a manner that it can easily be extended to different modalities and reconstruction methods.

This chapter is structured as follows. In **Section 2.2** we introduce the problem, related notation and mathematical descriptions of the methods used in this chapter. In **Section 2.3** we discuss our proposed method and how it can be used in existing methods. Details about the implementation and experiments are discussed in **Section 2.4**. The results are shown in **Section 2.5** and in **Section 2.6** we summarize and conclude the chapter.

2.2 Notation and mathematical preliminaries

2.2.1 Problem Definition

We focus here on the two dimensional (2D) parallel beam tomography problem, which we define below, and three different types of variational methods; see **Section 2.2.2**. Our approach can be used for other tomography geometries (both 2D and 3D) and other reconstruction methods in a straightforward manner.

The 2D parallel beam tomography problem entails reconstructing a two-dimensional unknown object from its parallel beam projection data. We will consider the discrete version of this problem:

$$W\mathbf{x} = \mathbf{y}, \tag{2.1}$$

with $\mathbf{x} \in \mathbb{R}^{N^2}$ the unknown object, defined on a $N \times N$ pixel grid; $\mathbf{y} \in \mathbb{R}^{N_\theta N_d}$ the parallel beam projection data, N_θ the number of projection angles, N_d the number of detector pixels, and $W : \mathbb{R}^{N^2} \to \mathbb{R}^{N_\theta N_d}$ a discrete version of the Radon transform. A more in depth description of this problem can be found in [Nat01; KS01].

We define a reconstruction method $F : \mathbb{R}^{N_\theta N_d} \times \mathbb{R}^{N_\lambda} \to \mathbb{R}^{N^2}$ with N_λ real-valued regularization parameters for the problem (2.1) as a type of black-box operator:

$$F(\mathbf{y}, \lambda) = \mathbf{x}_F^\lambda, \tag{2.2}$$

with $\lambda \in \mathbb{R}^{N_\lambda}$ a vector containing all the regularization parameters λ_i with $i = 0, ..., N_\lambda - 1$.

This definition fits general variational methods that incorporate regularization as discussed in Section 2.2.2, but also fits the well-known Filtered Backprojection (FBP) algorithm [Nat01], where bandwidth of a low-pass filter can be considered as the regularization parameter, and the Simultaneous Iterative Reconstruction Technique (SIRT) [VV90] with the number of iterations as a regularization parameter. As we consider F as a black-box operator, we will only make use of the

result of applying F to the projection data for different values of λ, but will not make use of specific properties of the operator F.

The key contribution of this chapter is to propose a computationally efficient approach for approximating $F(\mathbf{y}, \lambda)$ for many values of λ. Having this ability, it provides a way to choose the "optimal" value of the regularization parameter with respect to any user-defined quality criterion: *'Determine $\lambda^\star$ such that $\mathbf{x}_F^{\lambda^\star}$ is the optimal solution to* (2.1).' We do not specify here what an optimal solution is, because this varies per problem, application or even user of the reconstruction method.

2.2.2 Variational methods

In this section we discuss the choices for reconstruction methods F that we consider in this chapter. The methods we consider are all variational methods and instead of solving (2.1) directly these methods solve a related minimization problem. The following problem formulation is specifically for one regularization parameter:

$$\mathbf{x}_\lambda^\star = \underset{\mathbf{x}\in\mathbb{R}^{N^2}}{\operatorname{argmin}} \{\mathscr{D}(W\mathbf{x}, \mathbf{y}) + \lambda R(\mathbf{x})\}, \tag{2.3}$$

where $\mathscr{D}$ is the *data fidelity*, R is the *regularization* term or the *regularizer* and $\lambda \in \mathbb{R}$ the *regularization parameter*.

The data fidelity term encodes the information of the original problem (2.1). It determines the distance between the input data and the solution $\mathbf{x}$. In this chapter we will only consider the least squares norm as data fidelity:

$$\mathscr{D}(W\mathbf{x}, \mathbf{y}) = \tfrac{1}{2} \parallel W\mathbf{x} - \mathbf{y} \parallel_2^2 . \tag{2.4}$$

The prior knowledge for our inverse problem is encoded in the regularization term. The idea is to define a functional that is small when the image $\mathbf{x}$ has a certain preferred property, such as smoothness or sparsity with respect to a certain set of basis functions.

We consider three types of regularizers in this chapter:

Sobolev and Total Variation regularization These regularizers penalize the gradient of the reconstruction $\mathbf{x}$ in the L^2—sense and L^1—sense, respectively. In mathematical terms:

$$R_S(\mathbf{x}) = \|\nabla\mathbf{x}\|_2^2, \qquad R_{TV}(\mathbf{x}) = \|\nabla\mathbf{x}\|_1 . \tag{2.5}$$

Note that the resulting minimization problem (2.3) now has a single scalar value $\lambda \in \mathbb{R}$ as the regularization parameter. For a more in depth discussion on these regularizers see [Goc16; ROF92].

Total Generalized Variation regularization The idea for Total Generalized Variation is that one can split the reconstruction into parts with a different order of regularity. In this chapter we will consider the version which splits it into two parts: TGV^2_λ. The priority between these parts is balanced by a minimization problem with a second regularization parameter:

$$R_{\mathrm{TGV}}(\mathbf{x},\lambda) = \underset{\mathbf{v}\in\mathbb{R}^{2\times N^2}}{\operatorname{argmin}} \{\|\nabla\mathbf{x}-\mathbf{v}\|_1 + \lambda\,\|\mathscr{E}\mathbf{v}\|_1\}, \tag{2.6}$$

here $\mathscr{E}:\mathbb{R}^{2\times N^2}\to\mathbb{R}^{4\times N^2}$ is the distributional symmetrized derivative.

The minimization problem (2.3) in this case has two regularization parameters and two objects to minimize for:

$$\mathbf{x}^\star_\lambda, \mathbf{v}^\star_\lambda = \underset{\mathbf{x}\in\mathbb{R}^{N^2},\mathbf{v}\in\mathbb{R}^{2\times N^2}}{\operatorname{argmin}} \left\{ \tfrac{1}{2}\,\| W\mathbf{x}-\mathbf{y}\|_2^2 + \lambda_1\left(\|\nabla\mathbf{x}-\mathbf{v}\|_1 + \lambda_2\,\|\mathscr{E}\mathbf{v}\|_1\right)\right\}, \tag{2.7}$$

with $\lambda=(\lambda_1,\lambda_2)$. For a more in-depth discussion on this regularizer see [BKP10].

At this point we have only described the mathematical functions to be minimized for a particular variational method. We use a Primal-Dual Hybrid Gradient (PDHG) method presented in [CP11]. Note that such an algorithm will typically be terminated before the solution has fully converged, and will therefore not reach the exact solution $\mathbf{x}^\star_\lambda$. Taking this all into consideration we define our black-box reconstruction method as follows:

$$F(\mathbf{y},\lambda) = \mathbf{x}^\lambda_{\mathrm{PDHG}_{\mathrm{VM}}}, \tag{2.8}$$

with the variational method VM $\in$ {S, TV, TGV}.

Details about the implementation and parameter choices for the PDHG algorithm are discussed in **Section 2.4**.

2.2.3 Parameter optimization methods

Two well known methods for choosing the regularization parameter for single parameter variational methods are the discrepancy principle and the L-curve criterion, which we will briefly introduce here.

Discrepancy principle For the discrepancy principle one chooses the maximal regularization parameter λ such that the projection data $\mathbf{y}$ satisfies:

$$\| WF(\mathbf{y},\lambda)-\mathbf{y}\|_2^2 \le \epsilon, \tag{2.9}$$

where the specific norm is the same as the one used in the data fidelity and ϵ is a parameter that corresponds to the expected noise level in the projection data. For more information see [Vai82; BM12].

L-curve criterion The L-curve criterion chooses the regularization parameter $\lambda^\star$ such that the log-log curve of the data fidelity and the regularizer has a maximum curvature. We define $\rho(\lambda)$ and $\eta(\lambda)$ as the logarithm of the data fidelity and the regularizer, respectively. The log-log curve can then be described as follows:

$$(\rho(\lambda), \eta(\lambda)) = (\log(\mathscr{D}(W\mathbf{x}_\lambda, \mathbf{y})), \log(R(\mathbf{x}_\lambda))), \tag{2.10}$$

Using these expressions we define the $\lambda^\star$ for the L-curve criterion as the λ for which the curvature κ attains its maximum:

$$\lambda^\star = \operatorname{argmax}_\lambda \{\kappa(\lambda)\} = \operatorname{argmax}_\lambda \left\{ \frac{\eta''\rho' - \rho''\eta'}{((\rho')^2 + (\eta')^2)^{3/2}} \right\}. \tag{2.11}$$

The idea is that for any other λ the relative increase of the data fidelity is higher than the relative decrease in the regularization term and vice versa. Therefore, this $\lambda^\star$ provides an "optimal" balance between the two terms. A more in depth discussion with theoretical analysis is given in [Han92; HO93; Han99].

We point out that both the discrepancy principle and the L-curve criterion implicitly assume that the reconstruction method only has one regularization parameter. Moreover, as will also become clear from our simulation results, the regularization parameter values obtained using both methods can be substantially different. No application-specific properties of the reconstructed image are taken into account, and these methods may not yield optimal results if the algorithm user wants to optimize for a particular type of image quality metric.

2.3 Method description

In this section we propose our method for efficiently computing a large number of reconstructions of a given set of measurements $\mathbf{y}$ while varying the regularization parameter. The idea is to evaluate the reconstruction method on a coarse regularization parameter grid and interpolate between these reconstructions in such a way that the resulting interpolation scheme is computationally efficient and accurately approximates the actual reconstructed image for a broad range of parameter values.

2.3.1 Interpolation scheme

Approximating the output of general tomographic reconstruction methods by interpolation is in general a challenging problem. However, if we fix the measurements $\mathbf{y}$ and consider the behavior of one pixel in the reconstructed image for varying λ, we observe that for many relevant reconstruction algorithms, including the

variational methods described in Section 2.2.2, the pixel value changes smoothly with the value of λ and the curve seems to be suitable for approximation using a relatively simple model. Therefore we propose to perform pixel-wise interpolation using cubic spline interpolation. The idea is that if the approximation per pixel is sufficiently accurate, combining these single-pixel approximations into an approximation of the reconstructed image will be an accurate approximation of the true reconstructed image for a given value of λ.

First of all let us consider the case with one regularization parameter, $N_\lambda = 1$, and define a coarse grid of N_{ip} interpolation points:

$$\Lambda = \{\lambda_0, \lambda_1, ..., \lambda_{N_{\text{ip}}-1}\}, \qquad \text{with,} \qquad 0 < \lambda_0 < \lambda_1 <, ..., < \lambda_{N_{\text{ip}}-1}, \tag{2.12}$$

for this regularization parameter. We evaluate the reconstruction method $F(\mathbf{y}, \lambda_i) = \mathbf{x}_F^{\lambda_i}$. An important observation is that the range of possible choices for λ can be very large (e.g. between 0 and 10^5), while the actual range where the interesting changes of pixel values take place is usually much narrower. Therefore a linear spacing between the interpolation points $\{\lambda_0, \lambda_1, ..., \lambda_{N_{\text{ip}}-1}\}$ does not cover the transitions of pixel values well. Instead, we found that choosing a scheme where the values $\log \lambda_i$ are equidistantly sampled results in more accurate capturing of the transitions[1]. This requires, however, that also the interpolation between the sampling points respects this logarithmic scale. Specifically: given a set of regularization parameters with corresponding reconstructions from the reconstruction method F, $\{(\log(\lambda_i), \mathbf{x}_F^{\lambda_i})\}_{i=0}^{N_{\text{ip}}-1}$, we can compute the cubic spline interpolation $S_p : \mathbb{R} \to \mathbb{R}$ for a pixel p,

$$S_p(\lambda) = S_p^i(\lambda), \qquad \text{with } \lambda_{i-1} \leq \lambda \leq \lambda_i, i = 1, ..., N_{\text{ip}-1}, \tag{2.13}$$

such that the following statements are satisfied:

$$S_p^i(\lambda) = a_i + b_i \log(\lambda) + c_i \log(\lambda)^2 + d_i \log(\lambda)^3, \qquad \text{with } d_i \neq 0, \tag{2.14}$$

$$S_p(\lambda) = (\mathbf{x}_F^\lambda)_p, \qquad \text{with } \lambda \in \Lambda. \tag{2.15}$$

These conditions are not sufficient to uniquely compute $S_p(\lambda)$, so we also need boundary conditions. Let us consider the regularization parameter interval Λ broad enough, such that $\mathbf{x}_F^{\lambda_0}$ will be under-regularized and $\mathbf{x}_F^{\lambda_{N_{\text{ip}}-1}}$ will be over-regularized. This means that taking a lower or higher λ, respectively, will not result in significant changes to the reconstruction. Therefore, if we assume the regularization parameter interval broad enough, *clamped* boundary conditions will

[1] Further discussion on how to determine the grid Λ is given in **Section 2.4.1**.

be satisfied on the left and right boundaries, *i.e.*,

$$\frac{dS_p(\lambda)}{d\lambda}(\lambda_0) = 0, \qquad \frac{dS_p(\lambda)}{d\lambda}(\lambda_{N_{\text{ip}}-1}) = 0. \tag{2.16}$$

If all the pixel-wise spline interpolations $S_p(\lambda)$ are computed, we can consider the cubic spline interpolation function $S_F : \mathbb{R} \to \mathbb{R}^{N^2}$ for the full object and reconstruction method F:

$$S_F(\lambda) = \begin{bmatrix} S_0(\lambda) & \cdots & S_{N-1}(\lambda) \\ \vdots & \ddots & \vdots \\ S_{(N-1)N}(\lambda) & \cdots & S_{N^2-1}(\lambda) \end{bmatrix} \qquad \text{with } \lambda \in [\lambda_0, \lambda_{N_{\text{ip}}-1}]. \tag{2.17}$$

To summarize, our proposed method for one regularization parameter λ is step-by-step described in **Algorithm 1**.

Algorithm 1 Pixel-wise spline interpolation

1: Determine λ_0 and $\lambda_{N_{\text{ip}}-1}$, s.t. $\mathbf{x}_F^{\lambda_0}$ and $\mathbf{x}_F^{\lambda_{N_{\text{ip}}-1}}$, are under- and over-regularized, respectively.
2: Define a coarse grid Λ on $[\lambda_0, \lambda_{N_{\text{ip}}-1}]$, s.t. $\log(\lambda_i)$ are equidistantly spaced.
3: **for** $i = \{0, 1, ..., N_{\text{ip}} - 1\}$ **do**
4: Compute $F(\lambda_i, \mathbf{y}) = \mathbf{x}_F^{\lambda_i}$.
5: **for** $p = \{0, 1, .., N^2 - 1\}$ **do**
6: Compute the spline interpolation $S_p(\lambda)$ for pixel p such that (2.14), (2.15) and (2.16) are satisfied.
7: Combine the $S_p(\lambda)$, as described in (2.17), to get the spline interpolation function $S_F(\lambda)$.

The proposed method can easily be extended to two regularization parameters. In this case the pixel-wise interpolation becomes a two-dimensional interpolation problem. This means that the coarse grid of regularization parameters Λ is also a two-dimensional grid for which $N_{\text{ip},1}N_{\text{ip},2}$ evaluations of the reconstruction method are needed, increasing the computational effort significantly.

2.3.2 Optimizing the regularization parameter

Once the reconstruction algorithm F has been evaluated on the coarse grid of interpolation points, we can sample the approximations of the reconstructions from the interpolation function $S_F(\lambda)$ and use these to determine the optimal regularization parameter $\lambda^\star$ according to the specific requirements of the algorithm,

user, and the application. Here we discuss four strategies for using our interpolation technique to optimize the regularization parameter.

Parameter space exploration When there is no reference image available we simply determine the "visually optimal" $\lambda^\star$ through inspection or exploration of the approximations given by the interpolation function $S_F(\lambda)$. A good general framework for strategies such as this is described in [Sed+14], here one can replace the sampling of the original function with the interpolation function $S_F(\lambda)$. Additionally, one can embed the interpolation scheme in a visual tool where the user can adjust the regularization parameter on-the-fly and receive immediate visual feedback on the approximation of the resulting reconstruction.

Quantitative measure optimization In this strategy the approximations $S_F(\lambda)$ are compared to a ground truth or a high quality reconstruction with respect to a certain quantitative measure. We can define the "optimal" regularization parameter $\lambda^\star$ as follows:

$$\lambda^\star = \operatorname*{argmin}_\lambda \{\mathrm{QM}(S_F(\lambda), \mathbf{x}_{\mathrm{ref}})\}, \tag{2.18}$$

with $\mathrm{QM} : \mathbb{R}^{N^2} \to \mathbb{R}$, a quantitative measure on the reconstruction space. A simple example of such a function is the root Mean Squared Error (rMSE). We will discuss quantitative measures further in **Section 2.4.4**.

Discrepancy principle This strategy assumes that an estimate ϵ of the noise level on the projection data is available. We define our $\lambda^\star$ as the largest λ for which $\|WS_F(\lambda) - \mathbf{y}\|_2^2 \leq \epsilon$ is true.

L-curve criterion We can compute the L-curve by plugging in the spline interpolation function $S_F(\lambda)$ in (2.10):

$$(\rho(\lambda), \eta(\lambda)) = (\log(\mathscr{D}(S_F(\lambda), \mathbf{y}), \log(R(S_F))). \tag{2.19}$$

To compute the curvature $\kappa(\lambda)$ we use numerical approximations for the gradient based on spline interpolation.

2.4 Experiments

2.4.1 Implementation

Code All methods are implemented using Python 3.6.5, Numpy 1.14.3 [WCV11], SciPy 1.1.0 [JOP+01], ODL [AKÖ17]. The tomography operators are implemented

on the GPU using the ASTRA-toolbox [Van+16], where for performance reasons the forward and backprojection are not exactly each other's adjoint. The code used for this chapter is available on GitHub [Lagc].

Reconstruction method setup To normalize the range of the regularization parameters we scale the data fidelity term and the regularization term in (2.3) to the same range. Without loss of generality we set $\lambda = \hat{\lambda}\frac{\|W\|}{\|\nabla\|}$. For Sobolev regularization this gives:

$$\mathbf{x}_\lambda^\star = \operatorname*{argmin}_{\mathbf{x}\in\mathbb{R}^{N^2}} \left\{ \frac{1}{2}\left\|W\mathbf{x}-\mathbf{y}_2^2\right\| + \hat{\lambda}\frac{\|W\|}{\|\nabla\|}\|\nabla\mathbf{x}\|_2^2 \right\}, \tag{2.20}$$

$$= \operatorname*{argmin}_{\mathbf{x}\in\mathbb{R}^{N^2}} \left\{ \frac{1}{2}\frac{\left\|W\mathbf{x}-\mathbf{y}_2^2\right\|}{\|W\|} + \hat{\lambda}\frac{\|\nabla\mathbf{x}\|_2^2}{\|\nabla\|} \right\}, \tag{2.21}$$

with $\|\cdot\|$ the operator norm. Similar scaling can be done for TV and TGV regularization. Further reference to the regularization parameter will be to this normalized parameter $\hat{\lambda}$.

As stated before we use the PDHG algorithm to compute the reconstructions. For this algorithm we need to set the step-size parameters τ and σ, the relaxation parameter θ and the number of iterations. To ensure convergence we must have, $\tau\sigma\|L_{\text{VM}}\|^2 < 1$, therefore we take the step-size parameters as follows:

$$\tau = \frac{0.1}{\|L_{\text{VM}}\|}, \qquad \sigma = \frac{0.99}{\tau\|L_{\text{VM}}\|^2}, \tag{2.22}$$

with L_{VM} the operator related to the PDHG implementation of the variational method VM, more specifically:

$$L_{\text{S}} = L_{\text{TV}} = \begin{bmatrix} W \\ \nabla \end{bmatrix}, \qquad L_{\text{TGV}} = \begin{bmatrix} W & 0 \\ \nabla & -I \\ 0 & \mathscr{E} \end{bmatrix}. \tag{2.23}$$

Lastly, we set the relaxation parameter, $\theta = 1$, and the number of iterations, $n_{\text{iter}} = 500$, if not mentioned otherwise.

Parameter grid choice To assess the accuracy of our approximations, we will compute reconstructions for all the regularization parameters on a fine grid Λ^f. For our interpolation method we use a coarse grid $\Lambda \subset \Lambda^f$.

For Λ^f we take N_s logarithmically sampled points on the interval $[\lambda_0, \lambda_{N_{\text{ip}}-1}]$ containing the endpoints, such that we have

$$\log(\lambda_i) = \frac{\log(\lambda_{N_{\text{ip}}-1}) - \log(\lambda_0)}{N_s - 1} i + \log(\lambda_0). \tag{2.24}$$

for $\lambda_i \in \Lambda^f$ and $\lambda_{N_s-1} = \lambda_{N_{\text{ip}}-1}$.

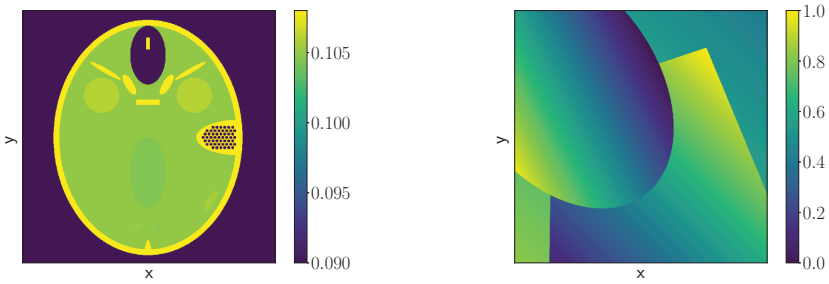
By fixing the number of points N_{ip} we use for the interpolation, we get Λ:

$$\Lambda = \left\{ \lambda_j \in \Lambda^f \mid j = k\lceil \tfrac{N_s}{N_{\text{ip}}-1} \rceil, k = 0, .., N_{\text{ip}} - 2 \right\} \cup \left\{ \lambda_{N_s-1} \right\}, \tag{2.25}$$

which means that the first $N_{\text{ip}} - 1$ points are chosen such that the exponents are equidistant, and the last point coincides with the endpoint.

2.4.2 Computer simulated data

We consider two computer simulated phantoms, shown in **Figure 2.1**, to test the performance of our method. These phantoms are defined independent of a pixel grid. The reconstructions are defined on a 1024×1024 uniform pixel grid. The projection data is defined as 2048 detector elements per projection angle. To avoid the so-called inverse crime the projection data is generated using 2048×2048 phantoms. The resulting projection data with 4096 detector elements per projection angle is rebinned to 2048 detector elements by taking the average of two neighboring elements.



(a) FORBILD head phantom

(b) Gradient objects phantom

Figure 2.1: Computer simulated phantoms. The FORBILD head phantom is presented in [LB]. Note that the range of the figure is not the actual range of the phantom; this is to visualize the low contrast objects. The gradient objects phantom is a standard phantom in the ODL package [AKÖ17].

We will consider zero mean additive Gaussian noise with the variance equal to a percentage of the maximum value of the projection data, *i.e.*,

$$\mathbf{y} = \mathbf{y}_{\text{GT}} + \delta, \qquad \delta \sim \mathcal{N}(0, V_\delta), \tag{2.26}$$

with $V_\delta = n_l \cdot \max_i \{\mathbf{y}_i\}$ the variance of the noise and $n_l \geq 0$ the noise level.

2.4.3 Experimental data

The experimental data is acquired from a low-dose scan of a pomegranate. The original scan is a 3D circular cone-beam CT scan of which we took the central detector row for all projection angles, to get a 2D circular fan-beam reconstruction problem. A detector row is 145.34 mm long and contains 1536 detector elements per projection angle and the dataset contains 500 equiangular spaced projection angles. The scans were done using the custom-built and highly flexible FleX-ray CT scanner, developed by XRE NV and located at CWI. Additionally, we use a high-dose scan of the same pomegranate with 2000 equiangular spaced projection angles, from which we compute a *gold standard reconstruction*[2], $\mathbf{x}_{GS}$, that can be used as a reference reconstruction $\mathbf{x}_{\text{ref}}$. Further details about the original scans can be found here [CLB18].

2.4.4 Quantitative measures

To test the accuracy of the approximations of our method compared to the original reconstructions or the ground truth we use two quantitative measures: relative Mean Squared Error (rMSE) and the structural similarity index (SSIM).

The rMSE is defined as follows

$$\text{rMSE}(\mathbf{x}, \mathbf{x}_{\text{ref}}) = \frac{\|\mathbf{x} - \mathbf{x}_{\text{ref}}\|_2^2}{\|\mathbf{x}_{\text{ref}}\|_2^2}, \tag{2.27}$$

which measures the distance between the object $\mathbf{x}$ and the reference object $\mathbf{x}_{\text{ref}}$ in the L^2-sense.

The SSIM measures the luminance, contrast and structure between the samples $\mathbf{x}$ and $\mathbf{x}_{\text{ref}}$. We use the implementation from ODL [AKÖ17]. We set the constants as suggested in [Wan+04], except for L, which we set equal to the dynamic range of the pixel values.[3] The mean and variance are computed with a Gaussian filter with

[2]The reconstruction method used to compute the gold standard reconstruction is a SIRT reconstruction with 300 iterations and a non-negativity constraint.

[3]The dynamic range is the difference between the maximum pixel value and the minimum pixel value of the reference reconstruction $\mathbf{x}_{\text{ref}}$.

width of 11 pixels. We chose these settings because in this case SSIM reflected our own observations of the relative quality of the reconstructions. The SSIM ranges between -1 and 1, where 1 indicates that the image and reference image are identical.

2.5 Results and discussion

The results are structured as follows: we investigate the quality of the approximations in **Section 2.5.1**, in **Section 2.5.2** we investigate the use of our approach for selecting an "optimal" regularization parameter in various scenarios for simulated data and in **Section 2.5.3** for experimental data.

2.5.1 Method validation

As stated in **Section 2.4.2** we consider two computer simulated phantoms (**Figure 2.1**). For these phantoms we will consider two reconstruction problems: a *sparse view reconstruction problem*, with 64 equidistant projection angles and no noise ($n_l = 0$); and a *noisy reconstruction problem*, with 740 projection angles and a noise level $n_l = 0.1$.

Single regularization parameter methods

In this section we will validate the proposed method for the Sobolev and TV regularization reconstruction methods. For the experiments with these methods we took $N_s = 301$ sample points on the interval $[10^{-3}, 1]$, unless mentioned otherwise.

We will mainly look at a sparse view problem with **Figure 2.1a** as phantom reconstructed with TV regularization, however, the shown results are similar for the other cases described. In **Figure 2.2** we show the rMSE and SSIM of the approximations $S_{\mathrm{TV}}(\lambda)$ with respect to the reconstructions $\mathbf{x}_{\mathrm{TV}}^{\lambda}$ for a varying number of interpolation points N_{ip}. We observe that the rMSE and SSIM lie close to 0 and 1, respectively, which indicates that the approximations are close to the reconstructions. Moreover, we see that worst approximations lie in the middle between two interpolation points and that taking more interpolation points results in better approximations. In **Figure 2.3** the approximations for several pixels p are shown. We observe that for $N_{\mathrm{ip}} = 6$ the approximations are of relatively low accuracy. Moreover, we see that the assumption on the boundary conditions (2.16) are not always valid on the right hand side of the interval, which also influences the accuracy of the approximation.

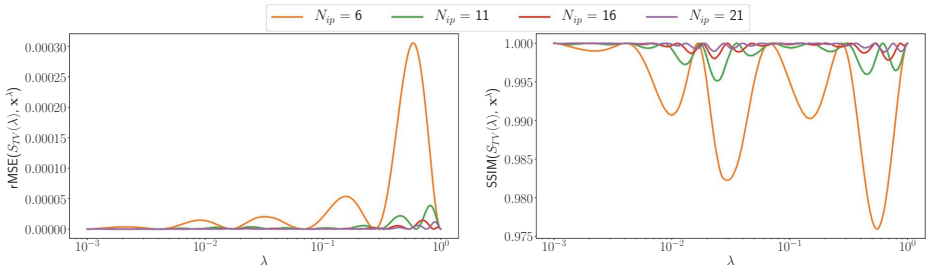


Figure 2.2: The rMSE (Left) and SSIM (Right) of the interpolated approximations with respect to the reconstructions as a function of the regularization parameter λ for varying number of interpolation points N_{ip}. Here we consider the sparse view reconstruction problem as defined in **Section 2.5.1** for the phantom shown in **Figure 2.1a** and the reconstructions are computed with the TV method.

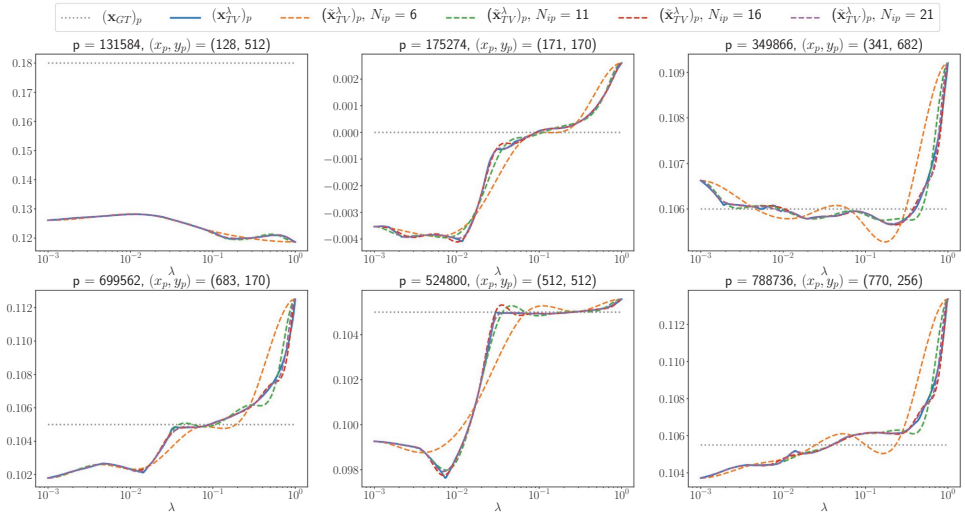


Figure 2.3: Pixel values of the phantom, the reconstruction and the interpolated approximations with varying number of interpolation points N_{ip} as a function of the regularization parameter for several pixels in the image. Here p indicates the pixel position in the vector and (x_p, y_p) indicates the pixel position in the image. Here we consider the sparse view reconstruction problem as defined in **Section 2.5.1** for the phantom shown in **Figure 2.1a** and the reconstructions are computed with the TV method.

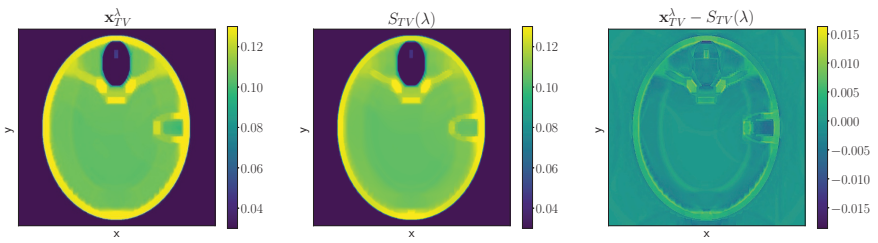


Figure 2.4: (Left) The worst case pixel-wise interpolated approximation with $N_{\text{ip}} = 6$ interpolation points of the TV reconstruction with regularization parameter $\lambda = 10^{-0.26}$ (Middle) and their difference (Right). Here we consider the sparse view reconstruction problem as defined in **Section 2.5.1** for the phantom shown in **Figure 2.1a** and the reconstructions are computed with the TV method.

In **Figure 2.4** we show the worst approximation for the full reconstruction with respect to the rMSE, the corresponding reconstruction and their difference. We observe that the approximation is worst around the high contrast parts of the object. However, the general behavior and properties of the reconstruction are accurately represented.

In **Figure 2.5** the average and the worst approximation of the rMSE and SSIM with respect to the reconstructions are shown for all the cases we stated at the beginning of this section. To avoid a positive bias the interpolation points are not taken into consideration for the statistics. Again we observe that the more interpolation points are used, the better the approximations are, and that the worst approximations are still close to the original.

Two regularization parameter method

For two regularization parameters computing reference reconstructions at high resolution to validate our methods is prohibitively expensive. Therefore, we only consider the noisy reconstruction problem as defined in **Section 2.5.1** for the phantom shown in **Figure 2.1b** and take a smaller reconstruction problem in terms of pixels and angles: 256×256, with 360 equidistant projection angles. For the regularization parameter grid we take $\lambda_1 \in [10^{-4}, 10^{2}]$ and $\lambda_2 \in [10^{-2}, 10^{2}]$ and $N_{s,1} = 181$ and $N_{s,2} = 121$.

The quantitative measures of the approximations with respect to the reconstructions are shown in the left and middle column of **Figure 2.6** and in the right column the quantitative measures of the reconstruction with respect to the ground truth. We observe lower accuracy of the approximation between the grid points of

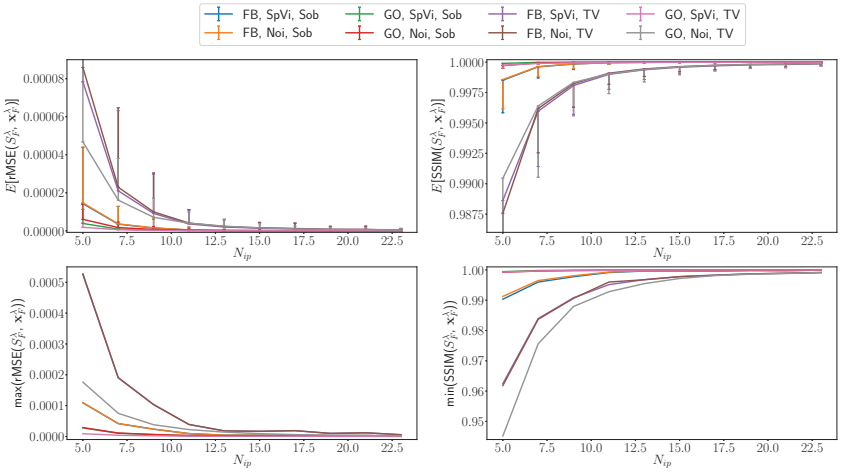


Figure 2.5: The average and standard deviation (Top row) and worst cases (Bottom row) for the rMSE (Left column) and SSIM (Right column) of the pixel-wise interpolated approximations with respect to the reconstructions for varying number of interpolation points N_{ip}, reconstruction problems and methods. To avoid cluttering of the figure the standard deviation bars are only plotted in one direction.

λ_1, as we also saw for one regularization parameter. However, the approximations do not vary in accuracy in λ_2-direction. In **Figure 2.7** we show the worst approximation, and the difference with respect to the SSIM, $\lambda_1 = 10^{-0.267}$, $\lambda_2 = 10^{-0.3}$. In the top row we show $N_{\text{ip},1} = 5$, $N_{\text{ip},2} = 5$ and in the bottom row $N_{\text{ip},1} = 10$, $N_{\text{ip},2} = 5$. We see that taking more points in the λ_1-direction results in more accurate reconstructions.

In **Table 2.1** the average and standard deviation of the rMSE and SSIM with respect to the reconstructions are shown. Again we observe that a finer grid in the λ_1 direction results in more accurate approximations. Moreover, we observe that taking a finer grid for λ_2 has a limited influence on the accuracy of the approximations.

2.5.2 Parameter optimization with simulated data

Single parameter regularization methods

In this section we use the approximations $S_F(\lambda)$ to determine the "optimal" regularization parameter $\lambda^\star$, as determined by a number of objective measures or criteria. We compare our methods to evaluating the method using only the reconstructions available at the interpolation points Λ and at the fully sampled grid Λ^f.

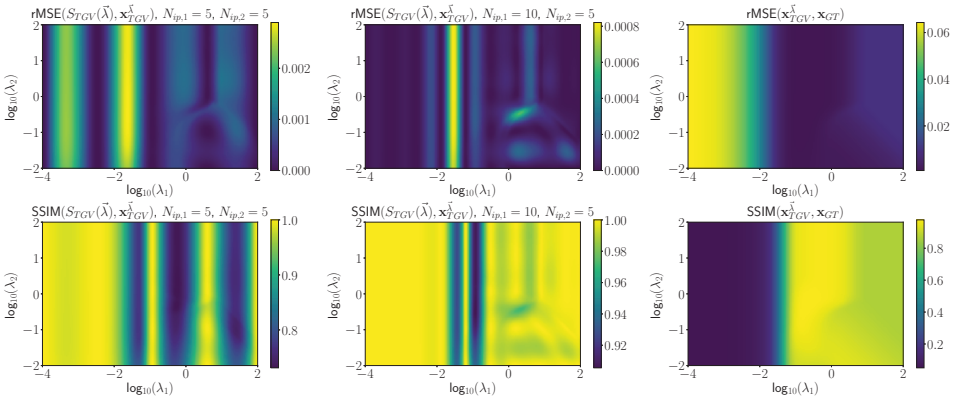


Figure 2.6: (Right and middle column) Quantitative measures of the interpolated approximations with respect to the reconstructions for varying λ_1 and λ_2. The approximations are done with $N_{\text{ip},1} = 5$, $N_{\text{ip},2} = 5$ and $N_{\text{ip},1} = 10$, $N_{\text{ip},2} = 5$ interpolation points in respectively the left and middle column. The right column shows the quantitative measures of the reconstructions with respect to the ground truth.

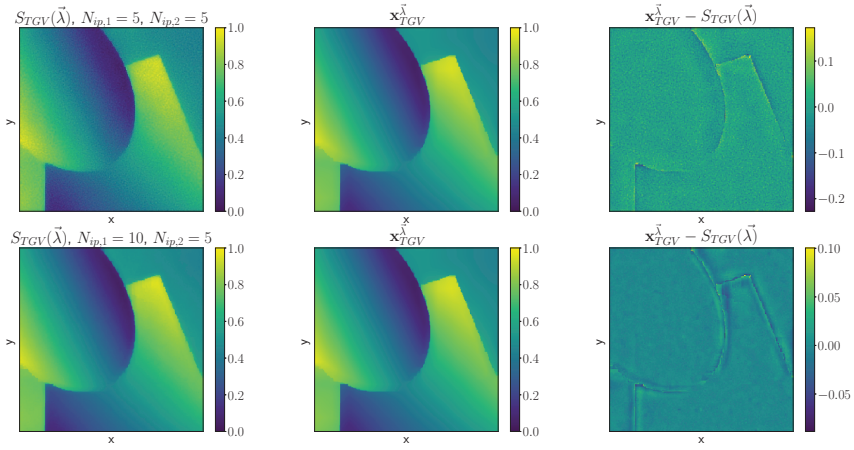


Figure 2.7: (Left column) Worst case pixel-wise interpolated approximation with $N_{\text{ip},1} = 5$, $N_{\text{ip},1} = 5$ (Top) and $N_{\text{ip},1} = 10$, $N_{\text{ip},1} = 5$ (Bottom) interpolation points of the TGV reconstruction with regularization parameters $\lambda_1 = 10^{-0.267}$, $\lambda_2 = 10^{-0.3}$ (Middle column) and their difference (Right column).

	rMSE		
	$N_{\text{ip},1} = 5$	$N_{\text{ip},1} = 10$	$N_{\text{ip},1} = 15$
$N_{\text{ip},2} = 5$	$(8.99 \pm 0.95) \cdot 10^{-4}$	$(1.05 \pm 0.04) \cdot 10^{-4}$	$(4.53 \pm 0.08) \cdot 10^{-5}$
$N_{\text{ip},2} = 10$	$(8.87 \pm 0.96) \cdot 10^{-4}$	$(8.91 \pm 0.4) \cdot 10^{-5}$	$(2.58 \pm 0.03) \cdot 10^{-5}$
$N_{\text{ip},2} = 15$	$(8.88 \pm 0.96) \cdot 10^{-4}$	$(8.85 \pm 0.4) \cdot 10^{-5}$	$(2.46 \pm 0.03) \cdot 10^{-5}$
	SSIM		
$N_{\text{ip},2} = 5$	$.907 \pm 0.12$	$.988 \pm 0.06$	$.996 \pm 0.01$
$N_{\text{ip},2} = 10$	$.908 \pm 0.12$	$.989 \pm 0.06$	$.997 \pm 0.01$
$N_{\text{ip},2} = 15$	$.908 \pm 0.12$	$.989 \pm 0.06$	$.997 \pm 0.01$

Table 2.1: The average and standard deviation rMSE and SSIM of the interpolated approximations with respect to the TGV reconstructions for a varying number of interpolation points $N_{\text{ip, 1}}$, $N_{\text{ip, 2}}$.

Parameter space exploration **Figure 2.8** shows the process of exploring the parameter space for the noisy reconstruction problem as defined in **Section 2.5.1** for the phantom shown in **Figure 2.1a** with the TV regularization reconstruction method, using 6 reconstructions ($N_{\text{ip}} = 6$). Here we determine the "visually optimal" parameter $\lambda^\star = 10^{-1.58}$ based on our visual inspection of the approximations. The top row are the 6 available reconstructions (red border), from which we see that the visually optimal parameter should lie in the interval $[10^{-1.78}, 10^{-1.17}]$. Knowing this, we take a number of interpolated approximations in this interval and compare them (the first 5 images on the bottom row, orange border). We observe that the approximation with $\lambda = 10^{-1.58}$ has a good trade-off between sharpness and artifacts in the background. The actual reconstruction is shown in the bottom right (green border). Note that without the interpolations the only information available would be the top row, meaning that one would have to choose between $\lambda = 10^{-1.78}$ and $\lambda = 10^{-1.17}$ or do additional computations.

Quantitative measure optimization In the case that there is a ground truth or high quality reconstruction available we determine the "optimal" regularization parameter with respect to a quantitative measure, by optimizing the QM curves (recall (2.18)). In **Figure 2.9** we show the resulting curves for the noisy reconstruction problem as defined in **Section 2.5.1** for the phantom shown in **Figure 2.1a** reconstructed with TV regularization. For comparison the curve computed with the actual reconstructions and direct spline interpolations through the points $\text{QM}(\mathbf{x}_{TV}^{\Lambda}, \mathbf{x}_{GT})$ are shown. We see that both interpolation methods give good approximations, although the pixel-wise interpolations are less accurate at the boundaries. This is most likely due to the assumption of clamped boundary conditions not being accurate enough.

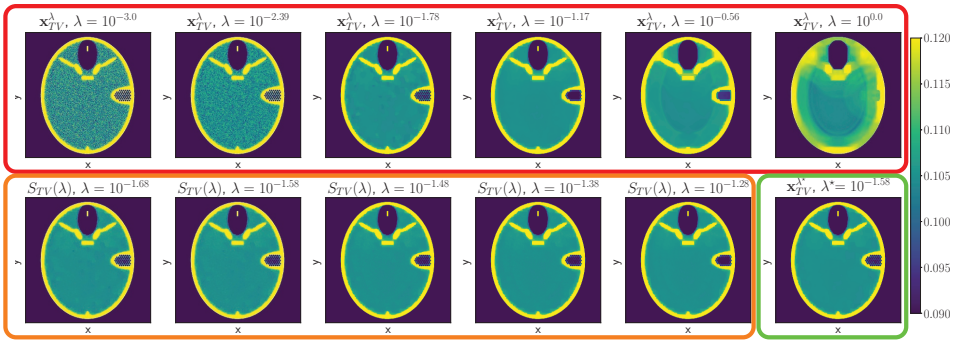


Figure 2.8: Visualization of the parameter space exploration. The top row with the red border are the reconstructions on the coarse grid Λ that are used for the interpolations. The approximations in the window of interest are shown in the bottom row with the orange border and the reconstruction with the "visually optimal" regularization parameter $\lambda^\star = 10^{-1.58}$ is shown in the bottom right with the green border. Here we consider the noisy reconstruction problem as defined in **Section 2.5.1** for the phantom shown in **Figure 2.1a** and the reconstructions are computed with the TV method.

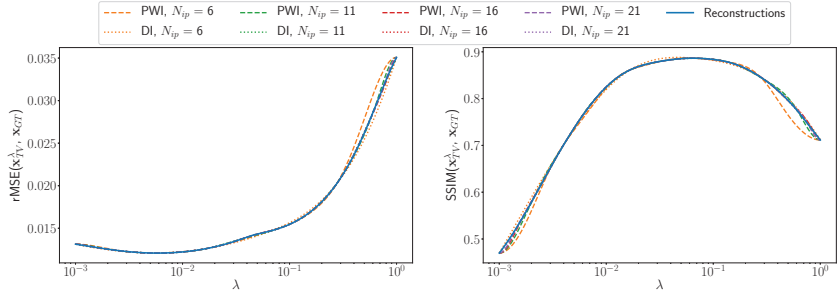


Figure 2.9: The rMSE (Left) and SSIM (Right) curves for the reconstructions and the pixel-wise interpolated (PWI) approximations with respect to the ground truth and the QM curves directly interpolated (DI) from the QM values on the coarse grid Λ for varying number of interpolation points N_{ip}. Here we consider the noisy reconstruction problem as defined in **Section 2.5.1** for the phantom shown in **Figure 2.1a** and the reconstructions are computed with the TV method.

In **Table 2.2** we show the estimated regularization parameters for the 8 different cases we considered also in the previous section. We observe that estimations of both the pixel-wise interpolation and the direct interpolation methods are close

to the "optimal" parameters. For the cases where the estimated parameter is less accurate, we inspected the curves and observed that the curve was at a plateau around the optimal value, making it more sensitive to errors. Lastly, we observe that the "optimal" regularization parameter varies depending on which quantitative measure is used.

	Sobolev regularization				TV regularization			
Case: Noisy	FORBILD		Gradient objects		FORBILD		Gradient objects	
Method	rMSE	SSIM†	rMSE	SSIM†	rMSE	SSIM	rMSE	SSIM
$\log_{10}(\lambda^\star)$	-2.02	-0.06	-0.9	0.05	-2.24	-1.19	-0.63	-0.36
$\log_{10}(\lambda^\star_{\text{PWI}})$, $N_{\text{ip}} = 6$	-2.03	-0.39	-0.88	-0.02	-2.25	-1.25	-0.6	-0.59
$\log_{10}(\lambda^\star_{\text{PWI}})$, $N_{\text{ip}} = 11$	-2.02	-0.15	-0.9	0.04	-2.23	-1.21	-0.63	-0.15
$\log_{10}(\lambda^\star_{\text{PWI}})$, $N_{\text{ip}} = 16$	-2.02	-0.05	-0.9	0.05	-2.24	-1.19	-0.62	-0.35
$\log_{10}(\lambda^\star_{\text{PWI}})$, $N_{\text{ip}} = 21$	-2.02	-0.06	-0.9	0.05	-2.24	-1.19	-0.63	-0.35
$\log_{10}(\lambda^\star_{\text{DI}})$, $N_{\text{ip}} = 6$	-2.02	-0.05	-0.89	0.07	-2.27	-1.33	-0.84	-0.84
$\log_{10}(\lambda^\star_{\text{DI}})$, $N_{\text{ip}} = 11$	-2.02	-0.06	-0.9	0.05	-2.24	-1.16	-0.61	-0.39
$\log_{10}(\lambda^\star_{\text{DI}})$, $N_{\text{ip}} = 16$	-2.02	-0.06	-0.9	0.05	-2.24	-1.19	-0.63	-0.35
$\log_{10}(\lambda^\star_{\text{DI}})$, $N_{\text{ip}} = 21$	-2.02	-0.06	-0.9	0.05	-2.24	-1.19	-0.63	-0.35

Table 2.2: Estimated and "optimal" regularization parameters based on quantitative measure optimization for two reconstruction methods, two phantoms and the rMSE and SSIM. We only show the noisy reconstruction problem, the sparse view problems have similar results. The estimations of the regularization parameters are done based on the pixel-wise interpolations (PWI) and the direct interpolation (DI) of the QM curves. For the cases denoted with a † the range of the regularization parameter is changed to $[10^{-2}, 10^{1}]$ to ensure that the "optimal" parameter is within the considered range.

Discrepancy principle & L-curve criterion To ensure capturing the desired behavior we take a larger interval for the regularization parameter, $[10^{-4}, 10^{2}]$, and scale the sampling points accordingly, $N_s = 601$. In **Figure 2.10** we show show results for reconstructions, pixel-wise interpolations and direct interpolations. The top row shows the values of the data-fidelity as a function of the regularization parameter λ. The results shown are for the noisy reconstruction problem as defined in **Section 2.5.1** for the phantom shown in **Figure 2.1b**. Here we observe that the significant changes of the functions are in a relative small window of λ. This results in bad approximations if there is no reconstruction available in this window. Moreover, we see that the noise level ϵ intersects the data-fidelity curve at a plateau, which might result in inaccurate estimates for the regularization parameter (see the second to last column of **Table 2.3**).

In the bottom row of **Figure 2.10** we show the L-curve and its curvature. Here we observe that inaccuracies in the initial approximations result in inaccuracies

in the L-curve, and even more in its curvature. In the extreme case we observe a change of sign of a peak for the pixel-wise interpolations with $N_{\text{ip}} = 6$. However, even with these inaccuracies, the approximations follow the general behavior of the reference curve and through visual inspection of the approximations and curves one can stil determine the optimal parameter.

The "optimal" parameters for discrepancy principle and the L-curve criterion for the 8 cases considered are shown in **Table 2.3**. We observe more volatility in the "optimal" parameters compared to the results in **Table 2.2**, which coincides with the observations in **Figure 2.10**. Additionally, for two cases the direct interpolations resulted in negative values for the TV term resulting in an undefined L-curve. Lastly, we again see that the optimal value of the regularization parameter varies depending on the used method.

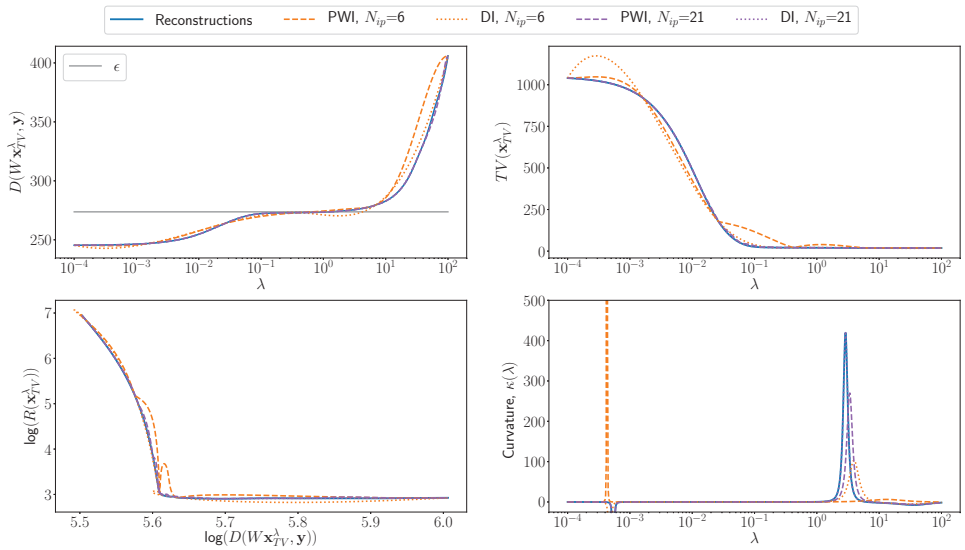


Figure 2.10: (Top) The data-fidelity (Left) and TV functional (Right) as a function of the regularization parameter λ. (Bottom) The L-curve (Left) and its curvature (Right). These curves are computed with the reconstructions, the pixel-wise interpolated (PWI) approximations and through direct interpolation (DI). Here we consider the noisy reconstruction problem as defined in **Section 2.5.1** for the phantom shown in **Figure 2.1b** and the reconstructions are computed with the TV method.

Case: Noisy	Sobolev regularization				TV regularization			
	FORBILD		Gradient objects		FORBILD		Gradient objects	
Method	DP	LC	DP	LC	DP	LC	DP	LC
$\log_{10}(\lambda^\star)$	-2.33	0.13	-0.58	0.83	-2.95	-0.64	0.09	0.46
$\log_{10}(\lambda^\star_{\text{PWI}})$, $N_{\text{ip}} = 6$	-2.34	0.07	-0.58	0.82	-3.01	-0.56	-0.18	$1.12^\dagger$
$\log_{10}(\lambda^\star_{\text{PWI}})$, $N_{\text{ip}} = 11$	-2.33	0.13	-0.58	$.82^\dagger$	-2.96	-0.61	0.09	0.72
$\log_{10}(\lambda^\star_{\text{PWI}})$, $N_{\text{ip}} = 16$	-2.33	0.13	-0.58	0.83	-2.95	-0.63	0.09	0.58
$\log_{10}(\lambda^\star_{\text{PWI}})$, $N_{\text{ip}} = 21$	-2.33	0.13	-0.58	0.84	-2.95	-0.63	0.09	0.51
$\log_{10}(\lambda^\star_{\text{DI}})$, $N_{\text{ip}} = 6$	-2.39	-	-0.62	-	-2.89	-0.55	0.63	0.6
$\log_{10}(\lambda^\star_{\text{DI}})$, $N_{\text{ip}} = 11$	-2.33	0.14	-0.58	0.83	-2.97	-0.62	0.05	0.45
$\log_{10}(\lambda^\star_{\text{DI}})$, $N_{\text{ip}} = 16$	-2.33	0.13	-0.58	0.83	-2.95	-0.64	0.09	0.46
$\log_{10}(\lambda^\star_{\text{DI}})$, $N_{\text{ip}} = 21$	-2.33	0.13	-0.58	0.83	-2.95	-0.63	0.09	0.46

Table 2.3: Estimated and "optimal" regularization parameters based on discrepancy principle (DP) and L-curve criterion (LC) for two reconstruction methods and two phantoms. We only show the noisy reconstruction problem, because these methods are not feasible for the sparse view problem. The estimations of the regularization parameters are done based on the pixel-wise interpolations (PWI) and the direct interpolation (DI) of the QM curves. For the cases denoted with a † the maximum curvature is at another value of λ, however, through closer inspection of the curve (in a similar manner as for **Figure 2.10**) this parameter is chosen.

Two regularization parameter method

Taking into consideration the observations from **Section 2.5.1** we use the same settings for the reconstruction problem and we take $N_{\text{ip},1} = 10$, $N_{\text{ip},2} = 5$ interpolation points for our interpolation scheme.

Quantitative measure optimization In **Figure 2.11** we show the quantitative measures of the approximations (left column) and the reconstructions (right column) with respect to the ground truth. The "optimal" regularization parameters and their respective quantitative measures determined from these figures are given in the caption. We observe that the found "optima" all lie in a plateau region which has relatively small differences in the quantitative measures. This indicates that the proposed method arrives at similar results as the original method.

Parameter space exploration Upon visual inspection of the reconstructions we concluded that the "visually optimal" set of parameters lies in $\log_{10}(\lambda_1) \in [-0.5, 0.2]$ and $\log_{10}(\lambda_2) \in [0.07, 1.1]$ (red border in **Figure 2.12**). Inspection of the approximations (orange border) in this interval resulted the optimal parameter set $(\lambda_1, \lambda_2) = (10^{-0.33}, 10^{0.07})$. We observe minimal differences between the approximation and the actual reconstruction (green border) for this "visually optimal" set of parameters.

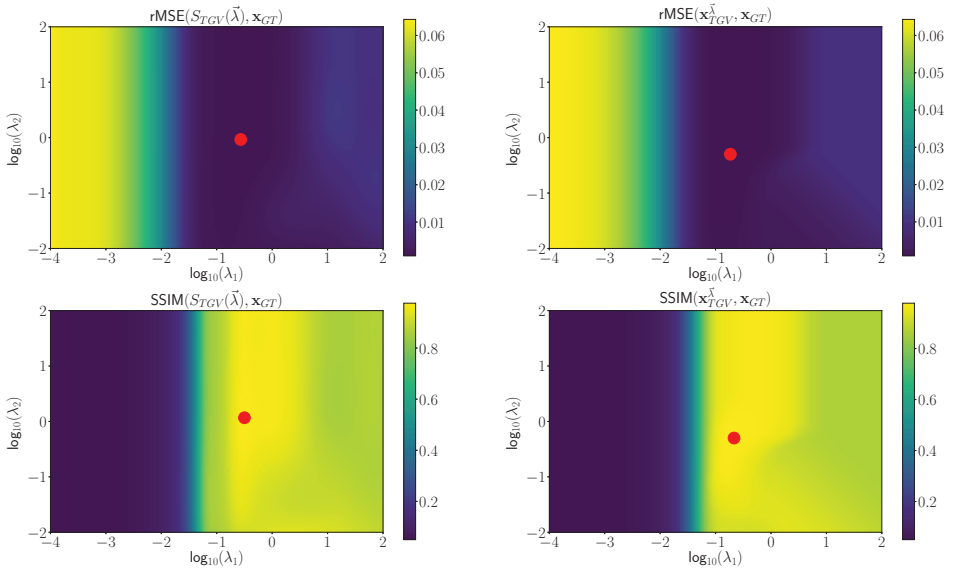


Figure 2.11: Quantitative measure figures for the TGV reconstruction method with respect to the ground truth. The red dot indicates the extremum of the figure.
(Top left) rMSE of the approximations with $\lambda^\star = (10^{-0.57}, 10^{-0.033})$ and min(rMSE) $= 9.1 \cdot 10^{-4}$.
(Top right) rMSE of the reconstructions with $\lambda^\star = (10^{-0.73}, 10^{-0.30})$ and min(rMSE) $= 8.8 \cdot 10^{-4}$.
(Bottom left) SSIM of the approximations with $\lambda^\star = (10^{-0.5}, 10^{0.067})$ and max(SSIM) $= 0.9755$.
(Bottom right) SSIM of the reconstructions with $\lambda^\star = (10^{-0.67}, 10^{-0.30})$ and max(SSIM) $= 0.9779$.

2.5.3 Parameter optimization with experimental data

In this section we show results for our method on experimental fan beam CT data. For this case we took the TV regularization reconstruction method implemented with the PDHG algorithm, using $N_{\text{iter}} = 2500$ iterations, $\tau = \frac{0.1}{\|L_{\text{TV}}\|}$, regularization parameter range $[10^{-4}, 1]$, $N_s = 401$ sample points and $N_{\text{ip}} = 11$ interpolation points.

The parameter space exploration for the experimental data and the TV reconstruction method is shown in **Figure 2.13**. Again only a selection of the approximations and initial reconstructions is shown for a clearer visualization. In **Figure 2.14** we show in the top row the TV reconstruction with the "visually

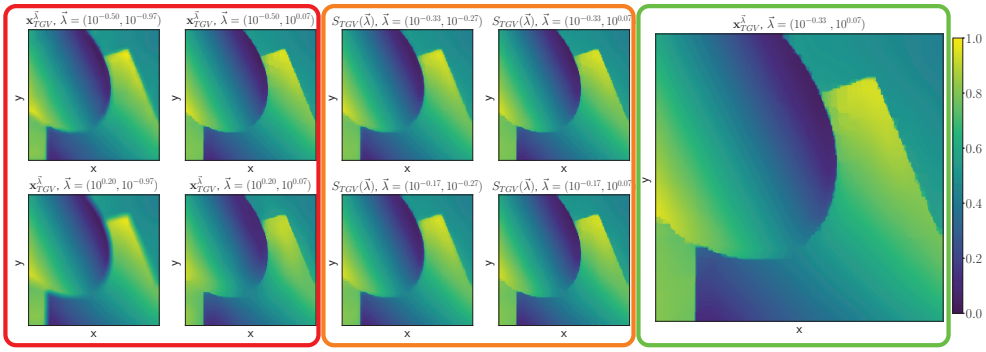


Figure 2.12: Partial visualization of the parameter space exploration for the TGV reconstruction method. The first and second column (red border), show several close to "visually optimal" reconstructions on the coarse grid Λ. The third and fourth column (orange border) partially show the further exploration through the approximations with in the top right the "visually optimal" approximation and the image on the right (green border) shows the reconstruction with this "visually optimal" regularization parameter.

optimal" regularization parameter, a FBP reconstruction and the gold standard reconstruction. We can conclude from the FBP reconstruction[4] that the noise in the data is quite severe and that it would be surprising if any reconstruction method can retrieve the small features at the boundary of the pomegranate and inside the seeds (observed in the gold standard reconstruction). Taking this into consideration the choice of regularization parameter results in an adequate TV reconstruction. We observe in the difference between the gold standard reconstruction and the TV reconstruction (Bottom right in **Figure 2.14**) the loss of the smaller details and a general loss of contrast, which is a known property of the TV method. Lastly, we consider the quantitative measures shown in **Table 2.4**. These results confirm the earlier conclusions; the pixel-wise approximation is very good and the chosen regularization parameter results in an adequate TV reconstruction.

2.6 Conclusion

In this chapter we have proposed an algorithmic approach for computationally efficient exploration of the regularization parameter space, based on a pixel-wise

[4]The FBP reconstruction is done with a Ram-Lak filter, with no windowing to reduce the noise in the input data [Nat01].

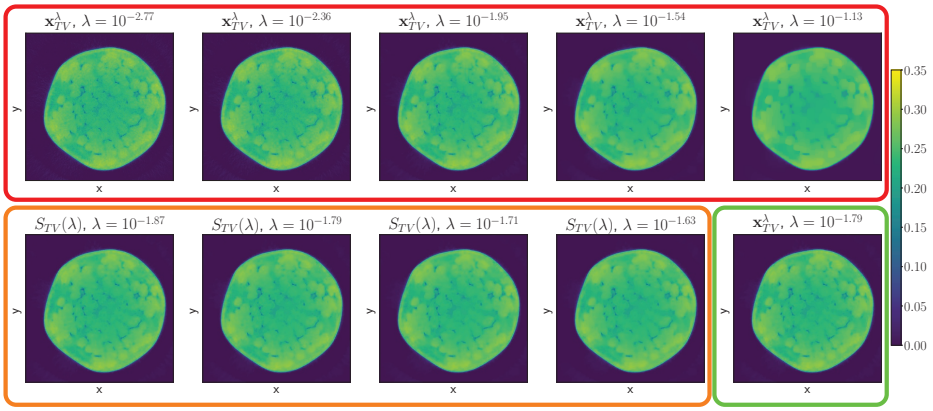


Figure 2.13: Partial visualization of the parameter space exploration. The top row (red border) shows several reconstructions on the coarse grid Λ. The images on the bottom row (orange border) show the further exploration of the parameter space through the use of approximations and on the bottom right (green border) the reconstruction with the "visually optimal" regularization parameter is shown. The difference between the approximation of the reconstruction is shown in **Figure 2.14**.

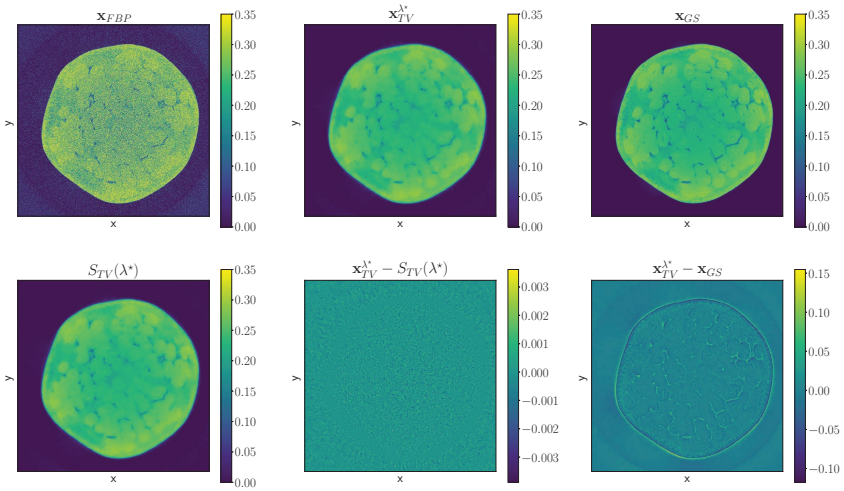


Figure 2.14: (Top left) FBP method reconstruction, (Top middle) TV method reconstruction, (Top right) Gold standard reconstructions. (Bottom left) Pixel-wise approximation to the TV reconstruction, (Bottom middle) Difference between TV approximation and reconstruction, (Bottom right) Difference between the TV and gold standard reconstruction.

Metric	$\mathbf{x} = S_{TV}(\lambda^\star)$, $\mathbf{x}_{\text{ref}} = \mathbf{x}_{TV}^{\lambda^\star}$	$\mathbf{x} = \mathbf{x}_{TV}^{\lambda^\star}$, $\mathbf{x}_{\text{ref}} = \mathbf{x}_{GS}$	$\mathbf{x} = \mathbf{x}_{\text{FBP}}$, $\mathbf{x}_{\text{ref}} = \mathbf{x}_{GS}$
rMSE$(\mathbf{x}, \mathbf{x}_{\text{ref}})$	$1.6328 \cdot 10^{-6}$	$2.2750 \cdot 10^{-3}$	$8.5259 \cdot 10^{-2}$
SSIM$(\mathbf{x}, \mathbf{x}_{\text{ref}})$	0.9988	0.6986	0.0186

Table 2.4: Quantitative measures for the experimental data results. Here $\mathbf{x}_{\text{FBP}}$ indicates the FBP reconstruction [Nat01] with a Ram-Lak filter.

interpolation scheme. Given a relatively small number of reconstructions on a sparsely sampled parameter grid, our method can be used to quickly compute an approximation to a reconstruction for any regularization parameter within the sampled range.

We have shown for three common variational reconstruction methods, Sobolev, TV and TGV regularization, that our method produces accurate approximations for simulated and experimental data. Moreover, we have shown that the approximations can be used in existing parameter optimization methods.

To conclude, our method enables developing computationally efficient tools that provide real-time visualization of the regularization parameter space, and automated parameter selection based on existing optimization criteria.

Chapter 3

Automated FDK-Filter selection for cone-beam Computed Tomography

3.1 Introduction

Research environments in academia nowadays have cone-beam (micro-)CT systems that are used for imaging the 3D interior structure of highly diverse objects. These systems may be shared by many users, each studying their own type of objects and their own questions they would like to answer based on the interior structure. As an example, one can think of a natural history department where various fossils, meteor fragments, plant remains, insects, and a variety of other objects are all scanned using the same system. Similarly, industrial research labs use micro-CT to analyze their products ranging from detergents to dairy products and packaging materials, all using the same CT system. For each new scan, the settings of the scan (number of angles, dose, energy level, etc.) are chosen by the user, often based on how much time is available or the dose sensitivity of the sample.

The Feldkamp-Davis-Kress algorithm (FDK) is the most common reconstruction method used in laboratory circular cone-beam CT systems. It is well known that

This chapter is based on:

Automated FDK-Filter Selection for Cone-Beam CT in Research Environments. *MJ Lagerwerf, WJ Palenstijn, H Kohr, KJ Batenburg.* IEEE Transactions on Computational Imaging (Volume: 6), pp. 739–748, 2020.

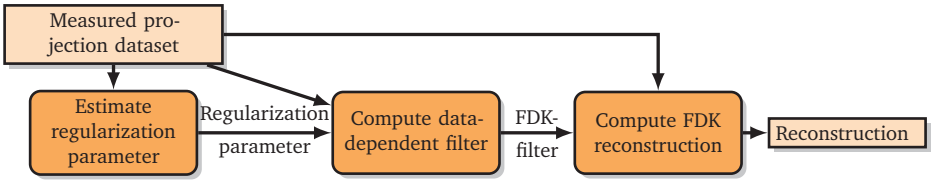


Figure 3.1: Schematic view of the proposed approach. Given a measured projection dataset with a certain geometrical setup, we estimate a regularization parameter and compute an FDK-filter that yields accurate results for common automated tasks (such as segmentation by global thresholding, porosity quantification).

optimizing the filter, also referred to as filter kernel, in the FDK algorithm to the characteristics of the scan (number of angles, dose, cone angle, etc.) can improve the accuracy of the FDK algorithm [Hsi+09; Rus17] (see Ch. 3.4.2 and 35.4.3.3, respectively). For high-throughput CT systems designed for a specific application (e.g. medical CT-scanners, dental CBCT scanners) the scanner comes with a set of proprietary pre-optimized filter [Com; Pla] that are chosen through a predefined protocol or by the user. In contrast, the broad variety in scans made in research scanners (many different objects with many different scan settings) require the user to manually select the parameters of the filter on a case-by-case basis, requiring specific expertise and time-consuming intervention from the user, or otherwise resulting in sub-optimal image quality.

Several studies have been made on how to compute such filters in an automated way, based on the geometrical parameters of the scanning process. In [GMD06; Nie+12] the authors exploit the tomosynthesis geometry to compute an acquisition-dependent filter. Alternatively, one can use the fact that the backprojection and filtering step are interchangeable in the FBP algorithm — for the parallel beam geometry — to optimize filters to approximate an iterative reconstruction method [BP12; Zen12; PB13], or to fit towards a specific scanner [Kun+07].

For the parallel beam geometry, a more general strategy for determining filters is proposed in [PB14], where a filter is computed that minimizes the residual error of the FBP reconstruction in the least squares sense. So far, this general concept has not been introduced in cone-beam tomography as the algorithm for computing the filter does not scale well to the 3D case of cone-beam tomography, where the full 3D volume must be taken into account.

In this chapter we present a computationally efficient and automated method to compute an FDK-filter for a given measured projection dataset that is optimal with respect to an objectively defined quality criterion based on the ℓ^2-norm of the difference between the measured projection data and the computed projections

of the reconstructed volume (**Figure 3.1**). Since this criterion is often referred to as the minimum residual, we will refer to our filters as Minimum Residual (MR) filters. We show that for a variety of objects, scan settings (number of angles and noise levels), and tasks (porosity quantification, threshold-based segmentation), the MR filters computed by our approach yield accurate results in terms of several different metrics (e.g. MAE, SSIM, MTF, which we will define later). In contrast, using the same manually tuned filter in all scenarios only yields accurate results for some of the cases, regardless of the particular choice of filter.

This chapter is structured as follows. In **Section 3.2** we introduce our method and describe how it allows for fully-automatic and efficient computation of the MR filter. In **Section 3.3** we describe how a set of experiments was carried out to investigate the behavior of our method under various scanning conditions, using both simulated and real experimental data. The results of these experiments are presented in **Section 3.4**. Conclusions are drawn in **Section 3.5**.

3.2 Method

3.2.1 Filter optimization problem

The 3D tomographic reconstruction problem can be modeled by a system of linear equations

$$W\mathbf{x} = \mathbf{y}, \tag{3.1}$$

where $\mathbf{x} \in \mathbb{R}^n$ is a vector containing the voxel gray values, $\mathbf{y} \in \mathbb{R}^m$ is a vector containing the measured projection data, and $W \in \mathbb{R}^{m \times n}$ is a discretized version of the *forward model*, i.e. the Radon transform for parallel beam tomography and the cone-beam transform for cone-beam tomography. In this chapter we focus exclusively on the circular cone-beam geometry, where the object rotates with respect to a point source and a planar detector, acquiring 2D cone-beam projections. For the sake of simplicity we assume that the volume consists of $n = N \times N \times N$ voxels and the detector consists of $2N \times N$ pixels. We denote the number of angles with N_a, so we have $m = N_a \times 2N \times N$.

The FDK algorithm [FDK84] is an extension of the well-known Filtered Backprojection algorithm that approximately solves (3.1) for the circular cone-beam geometry. For each projection angle, it applies a *reweighting* step, that corrects for some of the geometrical properties of the cone-beam transform, a *filtering* step, that filters the projections line-by-line by convolving the data with a *filter*, and a *backprojection* step that transfers the filtered projection into the image volume

domain. Using the notation of (3.1), the FDK algorithm is given by

$$\mathrm{FDK}(\mathbf{y}, \mathbf{h}) = W^T(\mathbf{h} * r(\mathbf{y}))_{1D}, \tag{3.2}$$

with W^T the transpose of W, known as the *backprojection operator*, $\mathbf{h} \in \mathbb{R}^{2N}$ a one-dimensional filter, r the reweighting operator, and $((\mathbf{h} * r(\mathbf{y}))_{1D})$ the discrete convolution between the data and the filter. While the standard Ram-Lak filter corresponds to the analytical derivation of the reconstruction problem, a variety of filters are used in practice for reaching a trade-off between artifacts, noise, sharpness of the reconstruction, and other application-specific image properties. The key contribution of this chapter is to propose a computationally efficient numerical algorithm for computing a filter for a specific combination of scanned object, geometrical parameters of the cone-beam acquisition, number of angles, and noise level. The aim is to devise an approach that provides decent quality results across a broad range of scenarios, such that the same automated approach can be used to compute FDK reconstructions, yielding high quality results in all cases.

A problem in automatically optimizing the FDK filter is that without access to a high quality reference image of the scanned object, defining reliable quality metrics is not straightforward. To solve this problem, we introduce a criterion that is not based directly on the reconstructed image, but instead on the *consistency* of the FDK-image with respect to the measured projection data $\mathbf{y}$, measured by simulating the projections of the FDK reconstruction and comparing these to the measured projections. Specifically, we select the filter as the minimizer of this cost function:

$$\mathbf{h}^\star = \operatorname*{argmin}_{\mathbf{h}} \|W(\mathrm{FDK}(\mathbf{y}, \mathbf{h})) - \mathbf{y}\|_2^2 + \lambda \|\mathbf{h}\|_2^2 \tag{3.3}$$

The first term corresponds to the *residual* of the FDK-reconstruction for a given filter $\mathbf{h}$, computed by applying the cone-beam transform to the FDK-reconstruction and comparing to the measured data $\mathbf{y}$. This term will be low if the filter results in an FDK-reconstruction that is consistent with the measured projections. Using a detector of size $2N \times N$ ensures that the convolution of the projected object with the filter is fully supported on the detector discretization domain and the influence of all filter coefficients on the reconstructed image is taken properly into account. The second term is a Tikhonov-type regularization term that penalizes filters with coefficients that have large absolute value. The regularization parameter λ determines the relative weight of this term. Note that this objective function only incorporates data specific information and general regularization, it does not include task or problem specific assumptions. In computing these filters we

are minimizing the residual, therefore we will refer to these computed filters as Minimum Residual (MR) filters.

We also evaluated more sophisticated regularizers, such as a Tikhonov-type term with the reconstructed image or the gradient of the reconstructed image, but we have not included them in this chapter as they lead to similar filters with similar performance as the proposed method.

We point out that the FDK algorithm is a *bilinear* operator with respect to the projection data $\mathbf{y}$ and the filter $\mathbf{h}$. This implies that for fixed projection data $\mathbf{y}$, the output FDK$(\mathbf{y}, \mathbf{h})$ of the FDK algorithm can be considered as a matrix-vector product $F_{\mathbf{y}}\mathbf{h}$. Consequently, the minimization problem in (3.3) is a linear least-squares problem for which the solution corresponds to the solution of the normal equations[1]:

$$(F_{\mathbf{y}}^T W^T W F_{\mathbf{y}} + \lambda I_{2N})\mathbf{h} = F_{\mathbf{y}}^T W^T \mathbf{y}, \tag{3.4}$$

with $I_{2N} \in \mathbb{R}^{2N \times 2N}$ the identity matrix. In the next subsection we will discuss how (3.4) can be solved accurately and efficiently.

3.2.2 Computational aspects

Note that the matrix inverse problem in (3.4) is small enough to solve directly once an explicit representation is available of the matrix $M = F_{\mathbf{y}}^T W^T W F_{\mathbf{y}}$. This matrix is square and its size equals the number of entries in the FDK-filter $\mathbf{h}$. However, computing the matrix M explicitly is not straightforward as it involves much larger matrices W and $F_{\mathbf{y}}$ and their transposes, which are too large to be represented explicitly. Instead, we compute the columns M_j of the matrix M individually, by evaluating the following expression $2N$ times:

$$M_j = M\mathbf{e}_j = (F_{\mathbf{y}}^T W^T W F_{\mathbf{y}})\mathbf{e}_j, \tag{3.5}$$

with $\mathbf{e}_j \in \mathbb{R}^{2N}$ a unit vector with all entries equal to zero except for the j^{th} element.

The computation for each element M_j involves a forward and backprojection, which can still impose a high computational load if the filter has many coefficients. Therefore, to reduce the number of filter coefficients, the filter is represented with respect to a small set of basis functions as $\mathbf{h} = E\mathbf{h}_e$ on an exponentially binned grid similar to [PB14]. As a result, we have $N_e \approx \log(N)$, with N_e the number of elements of $\mathbf{h}_e$. Details about this approximation are discussed in **Appendix 3.6.1**.

The algorithm to compute a MR filter is summarized in **Algorithm 2**. Lastly, we observe that the computational effort of computing a MR filter is roughly $2N_e$ forward and backward projections and lies mainly in the computation of the matrix M.

[1] Consider the first order optimality conditions of (3.3) and rearrange the terms to get (3.4).

Algorithm 2 Computing a MR filter

1: **for** $j = \{0, 1, 2, .., N_e - 1\}$ **do**
2: $M_j = \left(E^T F_{\mathbf{y}}^T W^T W F_{\mathbf{y}} E\right) \mathbf{e}_j$
3: Compute: $\mathbf{h}_e^\star = \left(M + \lambda I_{N_e}\right)^{-1} E^T F_{\mathbf{y}}^T W^T \mathbf{y}$

3.2.3 Regularization parameter

The final component of our automated approach for computing the FDK-filter is to determine a suitable value for the regularization parameter λ. Finding an optimal value for this parameter is in general not possible as it requires knowledge of the ground truth as well as detailed modelling of the application-specific quality criteria. We therefore aim for a computationally efficient heuristic that yields decent results across a broad range of imaging scenarios.

Our strategy involves three consecutive components:

- **Computing a low-noise reference reconstruction at strongly reduced spatial resolution**. We compute a low-resolution reconstruction, where the volume as well as the projection data are down-sized by a factor of 4 in each dimension (so, 4×4×4 for the volume and 4×4 for each projection). In this small reconstruction problem the signal-to-noise ratio is much higher and the number of angles is much larger relative to the size of the volume compared to the full problem. To compute the reference reconstruction, 200 iterations of the iterative SIRT algorithm are used, with a nonnegativity constraint applied in each iteration.

- **Optimizing the regularization parameter for the low-resolution problem using the noise characteristics of the full problem**. We subsample the high resolution data by taking every 4^{th} pixel in the detector width and height and compute the matrix M_{LR} with this subsampled low resolution data and compute reconstructions $\mathbf{x}_{LR}^\lambda$ for several values of λ. Note that by subsampling we do not reduce the noise levels, making the noise characteristics of the low resolution problem similar to the original high resolution problem. We choose the regularization parameter for which the difference with the reference reconstruction is minimal in the ℓ^1-norm. This heuristic choice of norm works well for our experiments. Other norms could also be used.

  Regularization parameters can vary strongly per problem. Therefore, we scale the parameter to account for the influence of the operators W, $F_{\mathbf{y}}$ and

E:

$$\lambda = \tilde{\lambda} \left\| W_{LR} F_{\mathbf{y},LR} E_{LR} \right\| = \tilde{\lambda} \sqrt{\|M_{LR}\|}. \tag{3.6}$$

Moreover, we consider a two step process for optimizing the parameter. First, we optimize over a broad and coarse grid, more specifically a logarithmically scaled grid spanning from 10^{-6} to 10 with 8 grid points. Second, we optimize over a fine grid around the optimal parameter from the first grid.

- **Scaling the regularization parameter to the full-resolution case**. The regularization parameter λ computed for the low-resolution problem cannot be used directly in the high-resolution problem. To account for the scaling differences between the problems, we apply the following conversion formula (*cf.* (3.6)):

$$\lambda_{HR} = \lambda_{LR} \frac{\left\| W F_{\mathbf{y}} E \right\|}{\left\| W_{LR} F_{\mathbf{y}_{LR}} E_{LR} \right\|} = \lambda_{LR} \frac{\sqrt{\|M\|}}{\sqrt{\|M_{LR}\|}}. \tag{3.7}$$

Once the regularization parameter λ and corresponding filter $\mathbf{h}$ have been computed, this filter can be used in the FDK algorithm to compute a reconstruction of the high-resolution data.

3.3 Experiments

We performed a series of experiments to assess the properties of our proposed MR filters. The goal of our experiments is twofold; (1) Compare the accuracy of the FDK results for MR filters to manually selected filters for the experiments, (2) Investigate the capability of MR filters to automatically adapt to a variety of objects, scan settings and tasks.

To achieve this we consider the following four scenarios:

- **Reconstruction problems with common data deficiencies.** We vary the noise levels, number of projection angles in the different input data and cone angle and compare the results from the computed filters to the manually tuned filters.

- **Task specific reconstruction problems.** We compute reconstructions of a *foam phantom* similar to the one used in [PBS18] and use these to do segmentations and compute the pore size distribution of this phantom. For the segmentations we use Otsu's global thresholding method [Ots79]. Details about the computation of the pore size distribution are given in **Appendix 3.6.4**.

- **Filter comparison and analysis.** We compare the computed filters and their respective *Modulation Transfer Functions* (MTF) to the manually selected filters. Details about the implementation of the MTF are given in **Appendix 3.6.5**.

- **Examples with experimental data.** We show results for two experimental datasets using the computed filters and compare the reconstructions to manually selected filters and the gold standard reconstruction (see **Section 3.3.1**).

The results are compared to several manually selected filters and analytic FDK filters. As manually selected filters we consider the family of filters which combine a low-pass filter with an analytic filter, *e.g.*, the Shepp-Logan filter and a Gaussian filter. The set of selected filters are chosen such that they are close to the optimal quality metrics for each of the conducted experiments. We point out that a manually selected filter from this set might actually yield sub-optimal results for experiments other than it was selected for, which is exactly why an automated and deterministic approach for computing filters can be beneficial.

Details on how the filters are combined, the definitions of the low-pass filters and how these filters are selected are given in **Appendix 3.6.2**.

3.3.1 Data

Simulated data

In **Figure 3.2** we show the FORBILD head phantom [LB], which is used in the simulated data experiments. Note that with the chosen scanning conditions we focus on the high contrast details and we do not expect to resolve the low contrast objects in this head phantom.

For our simulated data experiments we take $N = 1024$, which means that a reconstruction and the data are respectively defined on a 1024^3 equidistant voxel grid and a 2048×1024 equidistant detector grid per projection angle. To limit the influence of the inverse crime, we generate[2] a phantom with $N = 1536$, and forward project this phantom to a data space with detector size 3072×1536. Then interpolate the data per projection angle to a 2048×1024 detector grid and use this as input data. We set the source radius to 10 times the physical size of the phantom, resulting in a cone angle of 5.7 degrees. Poisson noise is applied before linearizing the data, *i.e.*,

$$\mathbf{y} = -\log\left(\frac{I_{\text{noise}}}{I_0}\right), \qquad I_{\text{noise}} \sim \text{Pois}(I), \tag{3.8}$$

[2]We also generated phantoms with higher resolutions, but did not observe noticeable differences. Hence we chose for a sampling factor of 1.5, to limit computational and memory constraints.

with I the noise-free photon count and I_0 the emitted photon count. Higher I_0 implies a higher dose and therefore less noise in the data.

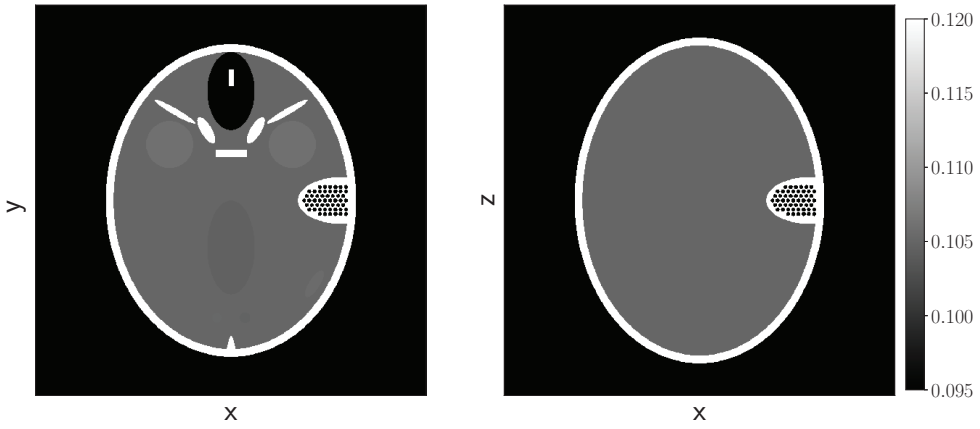


Figure 3.2: Two-dimensional slices $z = 0$ (Left) and $y = 0$ (Right) of the FORBILD head phantom [LB]. Note that the gray value scaling is different from the reconstructions further in the chapter. This is done to highlight the low contrast objects in the center, top, and bottom of the phantom. The phantom is continuously defined and sampled on a chosen grid.

Experimental data

We acquired an experimental dataset of a pomegranate using the custom-built and highly flexible FleX-ray CT scanner, developed by XRE NV and located at CWI. This scanner has a flat panel detector with 1943×1535 pixels and a physical size of 145.34×114.82 mm. The datasets contain 500 equidistantly spaced projections over a full circle. The distance from the center of rotation to the detector was set to 109 mm and the source radius to 590 mm. The scans were performed with a tube voltage of 70 kV. The high-dose scan was collected with a tube power of 45 W and an exposure time of 500 ms. The low-dose scan was collected with a tube power of 20 W and an exposure time of 100 ms. These datasets are available at Zenodo [CLB18].

In **Figure 3.3** the *gold standard* reconstruction of the experimental data is shown, obtained by computing a SIRT reconstruction with 300 iterations of the high-dose dataset with 500 equidistant projection angles, where we set the voxels in the background below a certain threshold equal to zero.

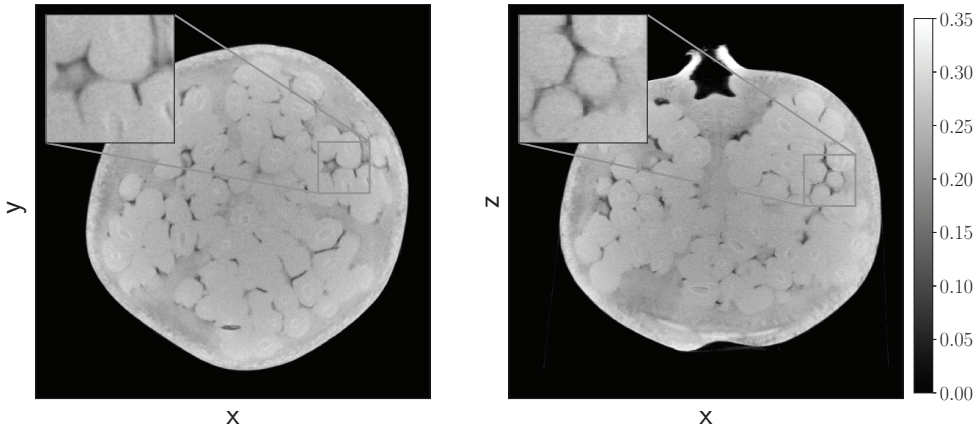


Figure 3.3: The $z = 0$ (Left) and $y = z$ (Right) slice of the gold standard reconstruction of the pomegranate dataset. The projection data is acquired using the FleX-ray scanner located at the CWI.

3.3.2 Quantitative measures

To quantify the accuracy of the reconstructions we consider two measures, the mean absolute error (MAE) and the structural similarity index (SSIM). The MAE and the SSIM compare the reconstructed image $\mathbf{x}_r$ to the phantom image $\mathbf{x}_p$. The MAE is defined as

$$\mathrm{MAE}(\mathbf{x}_r, \mathbf{x}_p) = \frac{\left\|\mathbf{x}_r - \mathbf{x}_p\right\|_1}{\left\|\mathbf{x}_p\right\|_1}, \tag{3.9}$$

The SSIM [Wan+04] is implemented based on the scikit-image 0.13.1 [Wal+14] package, where all the constants are set to default and the filter is uniform with a width of 19 pixels.

For experimental data there is no ground truth image available. Therefore, we will use the high quality gold standard reconstruction as reference image $\mathbf{x}_p$.

Lastly, we are only interested in how the methods perform on the object itself and not on the background. Therefore, we only consider the reconstructed object and roughly $0.2N$ pixels away from the object. Here the position of the object is determined in the ground truth or gold standard reconstruction.

3.3.3 Implementation

All the methods are implemented using Python 3.6.2, Numpy 1.12.1 [WCV11], ODL [AKÖ17] and PyFFTW 0.10.4 [FJ05], and the forward- and backprojection

are implemented on the GPU using the ASTRA-toolbox [Van+16], which provides a collection of high-performance building blocks for tomography algorithm development. Here, for performance reasons, the forward projection is not the exact adjoint of the backward projection and vice versa. Our implementation of this method is available on Github [Laga].

3.4 Results and discussion

3.4.1 Common data deficiencies

Sparse view and noisy data

Figure 3.4a and **Figure 3.4b** show the MAE and SSIM for reconstructions with a varying number of projection angles and varying noise levels, respectively. We observe that the MR filters are close to the optimal manually selected filter (see **Appendix 3.6.2** for details about manually tuned filters) for all considered cases with respect to the quantitative measures. Moreover, we observe that the manually selected filters are only optimal for a certain range of cases. This is illustrated in **Figure 3.5** where reconstructions of noisy data are shown with several filters.

Varying cone angles

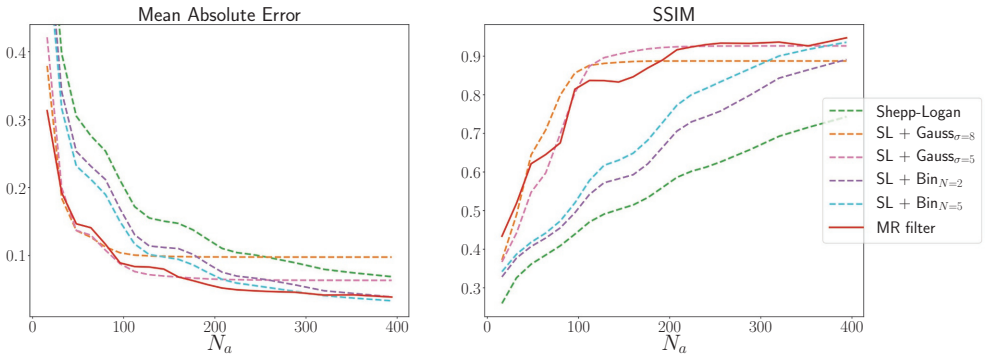
So far we have considered a relatively small cone angle of 5.7 degrees. In this section we show the effect of varying the cone angle. **Figure 3.6** shows the MAE and SSIM for a range of cone angles.

We observe that all the reconstruction methods react similarly to the change in cone angle with unchanged relative performance. **Figure 3.7** illustrates the effect of a large cone angle with strong artifacts at the top and bottom of the reconstruction. The effect of these artifacts are also reflected in the quantitative measures.

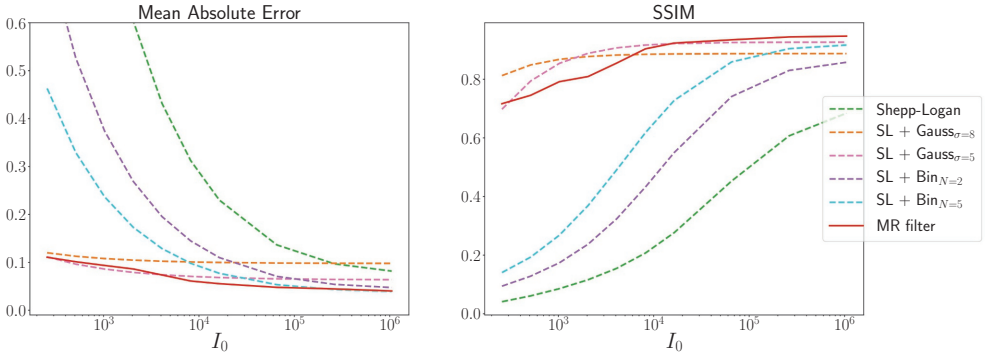
3.4.2 Task based problems

In this section we test the performance of MR filters for specific tasks, namely segmentation and porosity computation. **Figure 3.8** shows the percentage of misqualified voxels for an Otsu global thresholding segmentation [Ots79] for several filters and varying noise levels.

We observe that the less smoothing filters lead to a lower percentage of misqualified voxels. Looking at **Figure 3.9** we observe that the MR filter still leads to a less noisy reconstruction, but the smaller pores are lost in the segmentation.



(a) Varying number of projections angles.



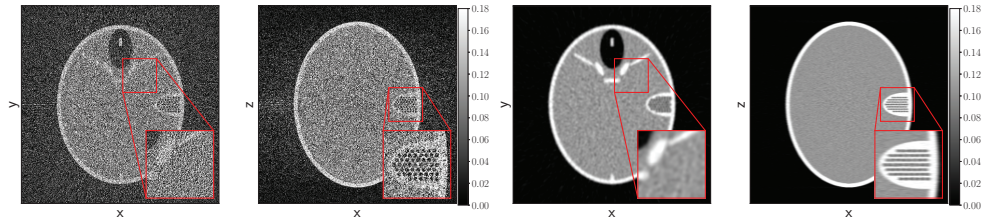
(b) Varying emitted photon counts I_0.

Figure 3.4: Comparing quantitative measures of FDK reconstructions with varying filters. The data is generated from the FORBILD head phantom.

Looking at **Figure 3.9** we observe that reconstructions with the MR filter contain less noise than the reconstructions with SL + Bin$_{N=5}$ filter, which is in line with the experiments in **Section 3.4.1**. Additionally, we see that, although the SL + Bin$_{N=2}$ has a lower percentage of misqualified voxels in this case, it still contains noise in the object and false positives in the background.

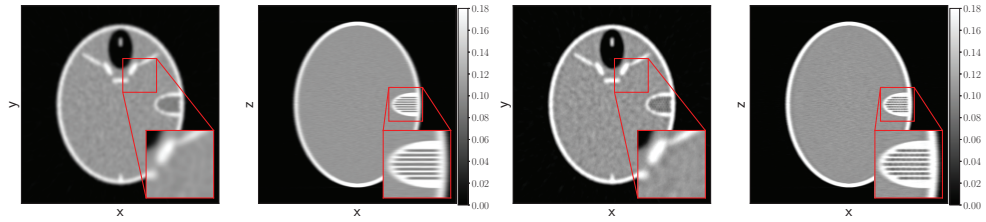
From these segmentations we can also count the pores inside the phantom. A pore is defined as an open space in the segmented volume and in **Figure 3.10** we show the pore size distributions for several noise levels. First of all, we note that the segmentations with the SL + Bin$_{N=2}$ filter for $I_0 = 256$ contain too much noise to compute a sensible pore distribution. Second, we observe that the MR filter underestimates the number of pores, whereas the filters with binomial smoothing tend to overestimate the number of pores.

Considering these observations we can conclude that, although MR filters do



(a) FDK reconstruction with Shepp-Logan filter.

(b) FDK reconstruction with a SL + Gauss$_{\sigma=5}$ filter.



(c) FDK reconstruction with a SL + Gauss$_{\sigma=8}$ filter.

(d) FDK reconstruction with MR filter.

Figure 3.5: Reconstructions from the FORBILD phantom; the dataset has 360 equidistant projection angles and emitted photon count $I_0 = 256$.

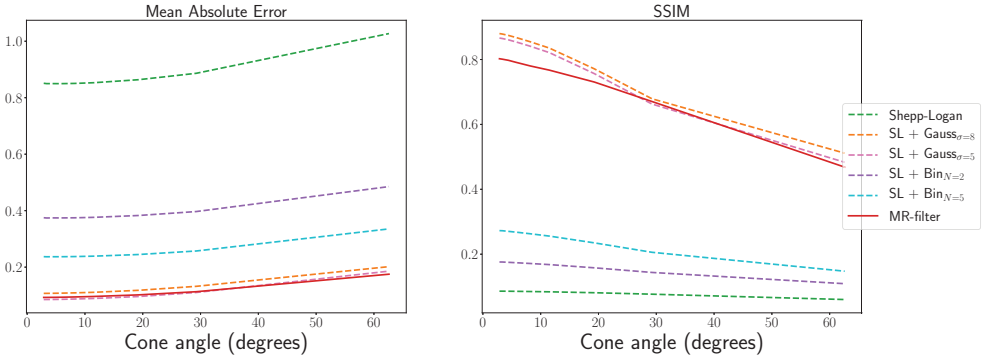


Figure 3.6: Reconstruction quality measures computed from the FORBILD phantoms with 360 equidistant projection angles, emitted photon count $I_0 = 1024$, and varying cone angles.

not lead to the best segmentations or pore size distributions, they still perform robustly in all the considered cases.

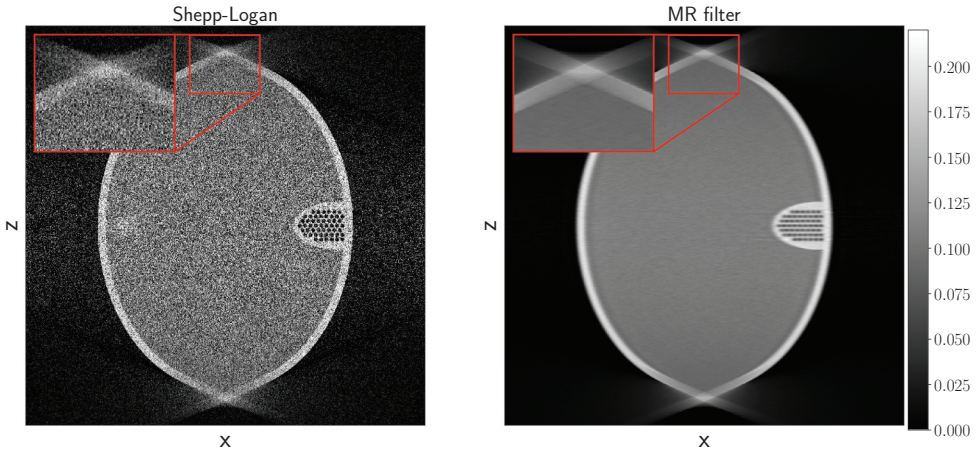


Figure 3.7: Reconstructions using Shepp-Logan filter (Left) and MR filter (Right) computed on the FORBILD phantom with 360 equidistant projection angles, cone angle of 62.6 degrees and an emitted photon count $I_0 = 1024$.

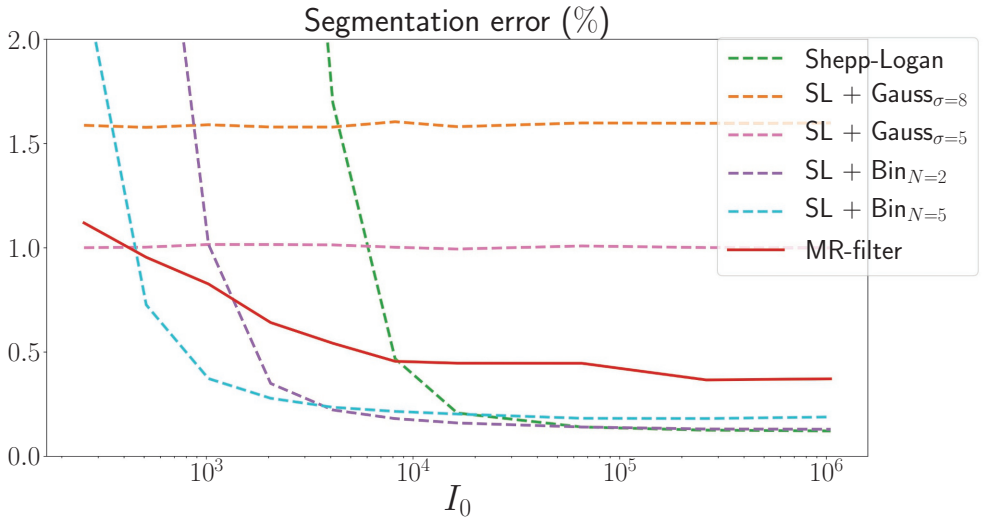


Figure 3.8: Segmentation errors for various filters. The projection data is simulated with varying emitted photon counts I_0 from a foam phantom.

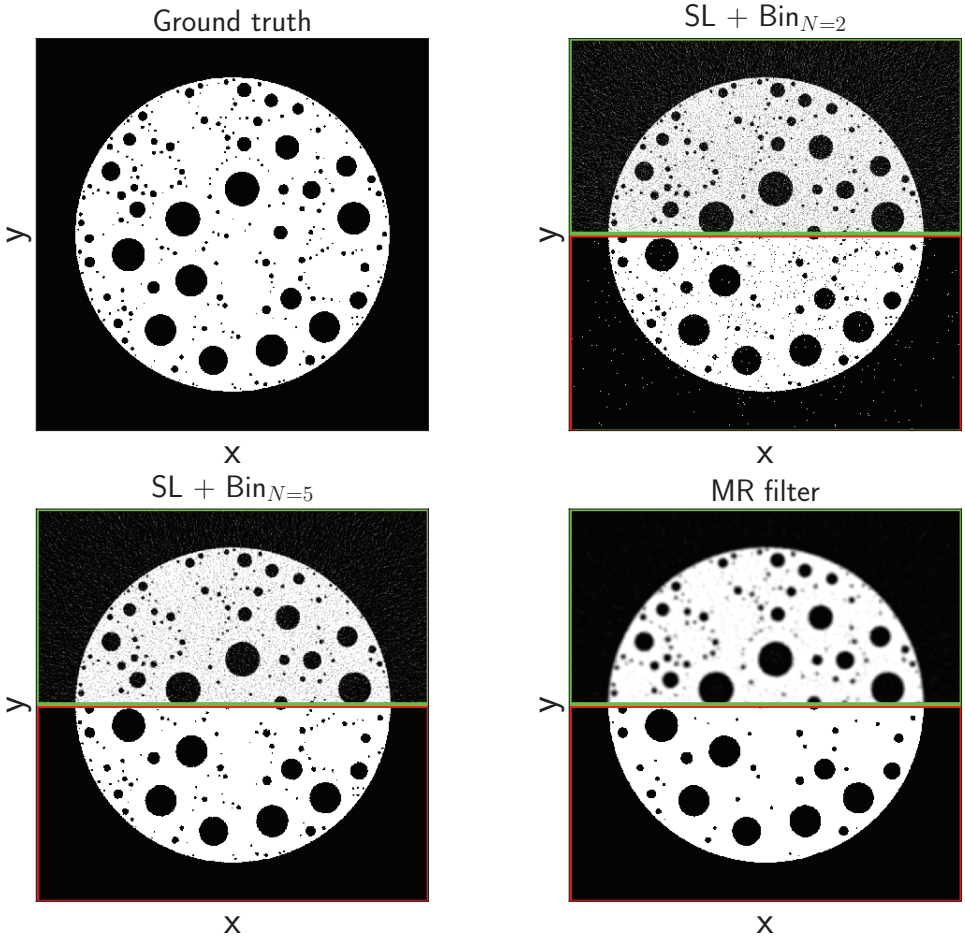


Figure 3.9: Reconstructions (top half, green border) and segmentations (bottom half, red border) of projection data from a foam phantom with emitted photon count $I_0 = 2048$ and 360 projection angles.

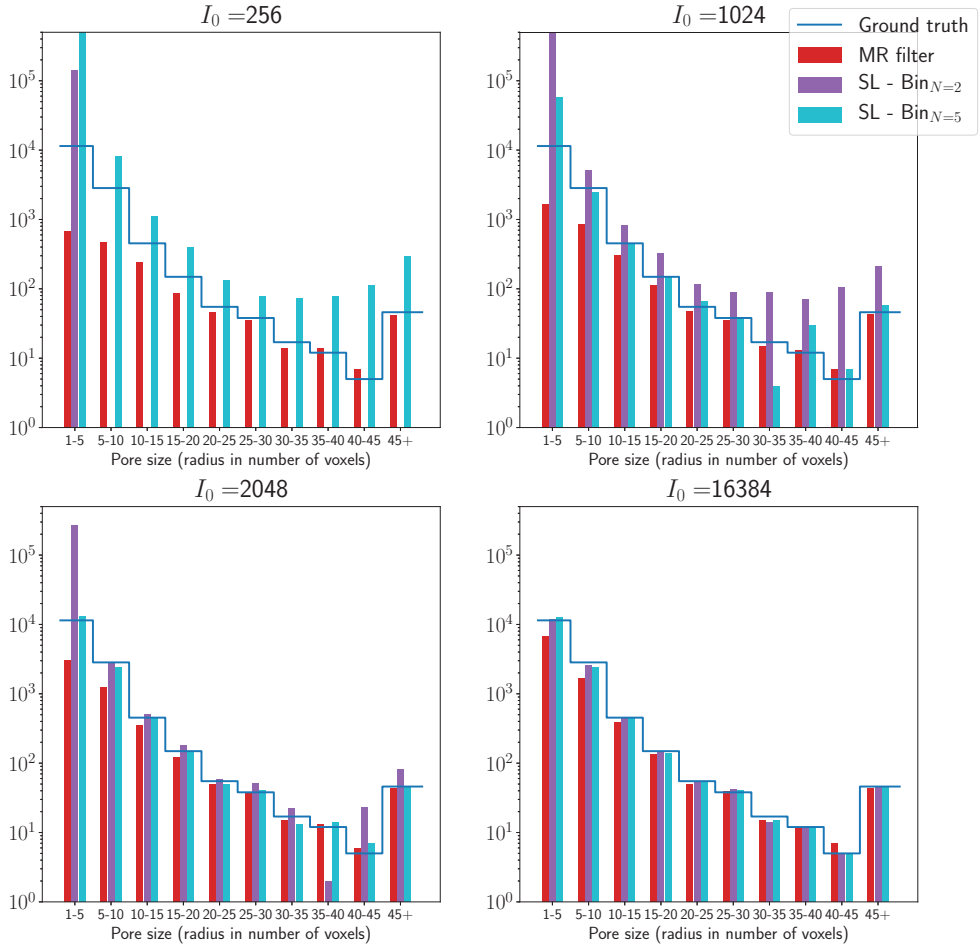


Figure 3.10: Pore size distributions for three different filters and the ground truth distribution. The size of a pore is defined as the radius of the pore, which is expressed in a number of voxels. Furthermore, the pores are grouped into bins with similar radii (5 voxel intervals). The projection data is simulated with varying emitted photon counts from the foam phantom.

3.4.3 Filter analysis

In this section we analyse the MR filters. In **Figure 3.11** we show the Fourier representation of several types of filters, namely: analytic, manually tuned, and MR filters. We show MR filters from four different cases. Comparing the filters we observe that the shape of the MR filters is different from the manually selected filters, *i.e.*, they have a relatively lower amplitude and a longer tail. From the MTFs, which are shown in **Figure 3.12**, we also observe a slight difference between the three types of filters. The MTFs of the analytic and manually selected filters all have a similar gradual drop in frequencies, whereas the MTFs of the MR filters have a steeper drop and a longer tail compared to the SL + Gauss filters, which were the filters that led to the most comparable results for the (NOI, 1) and (NOI, 2) scenarios.

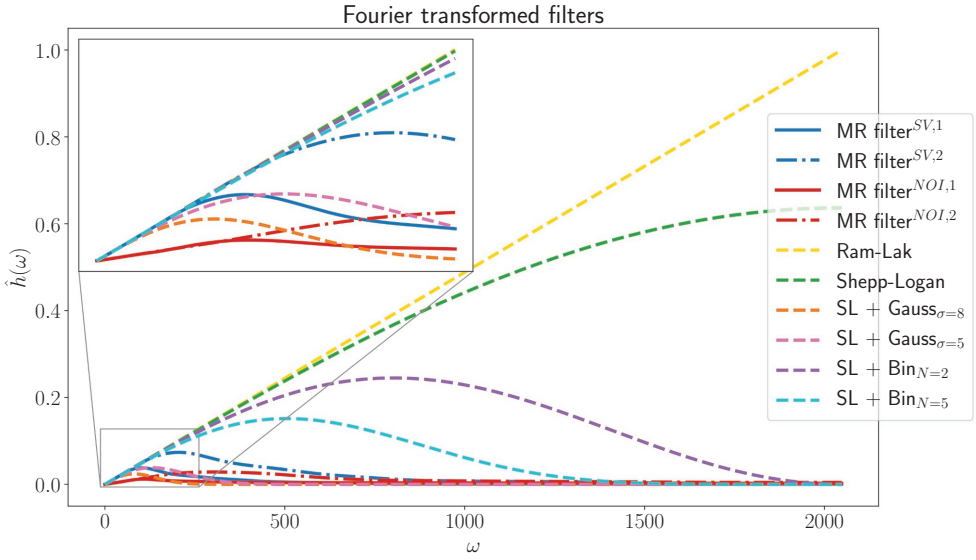


Figure 3.11: Fourier representations of various filters. Here the MR filters denoted with (SV, 1) and (SV, 2), are computed on the FORBILD head phantom with 96 equidistant and 192 projection angles, respectively. The MR filters denoted with (NOI, 1) and (NOI, 2) are computed on the FORBILD phantom with 360 equidistant projection angles and emitted photon count $I_0 = 256$ and $I_0 = 16384$, respectively.

Lastly, we consider the image bias introduced by the MR filters. The MR filters use the FDK algorithm to reconstruct the data. Therefore, their reconstructions will suffer similar limitations as the reconstructions with standard filters. In **Figure 3.13** we show the cross section along the x-axis of averaged reconstructions computed

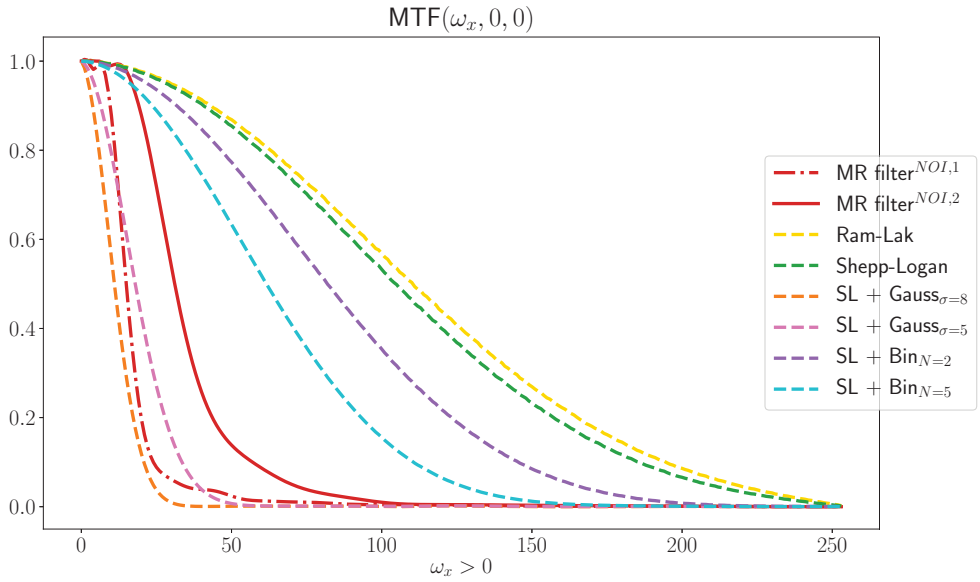


Figure 3.12: Modulation Transfer Functions of several filters in the positive ω_x direction. The MR filters are computed on datasets with 360 equidistant projection angles of the FORBILD phantom and emitted photon counts $I_0 = 256$ (NOI, 1) and $I_0 = 16384$ (NOI, 2).

over 500 random noise instantiations and their standard deviations of 3 phantoms: a cylinder with constant density, a cylinder with a ramp in its density and the FORBILD phantom. We clearly see the bias-variance trade-off between the more smoothing filters and the Ram-Lak filter. Additionally, we see that the MR filters adapt to the data and even try to fit the edges, at the cost of overshooting around these edges.

3.4.4 Experimental data

In **Table 3.1** the quantitative measures for the experimental data with respect to the gold standard reconstruction (recall **Section 3.4.4** and **Figure 3.3**). First we observe that the MR filters have a lower MAE than the other filters. In **Figure 3.14** we see that the intensity of the MR filter reconstruction is closer to the gold standard reconstruction, which explains the difference in MAE. This difference is less prominent in the SSIM, because the SSIM is designed to be less influenced by scaling differences. Lastly, we observe that the reconstruction with the MR filter for the low-dose dataset still contains noise, which explains the relatively low SSIM.

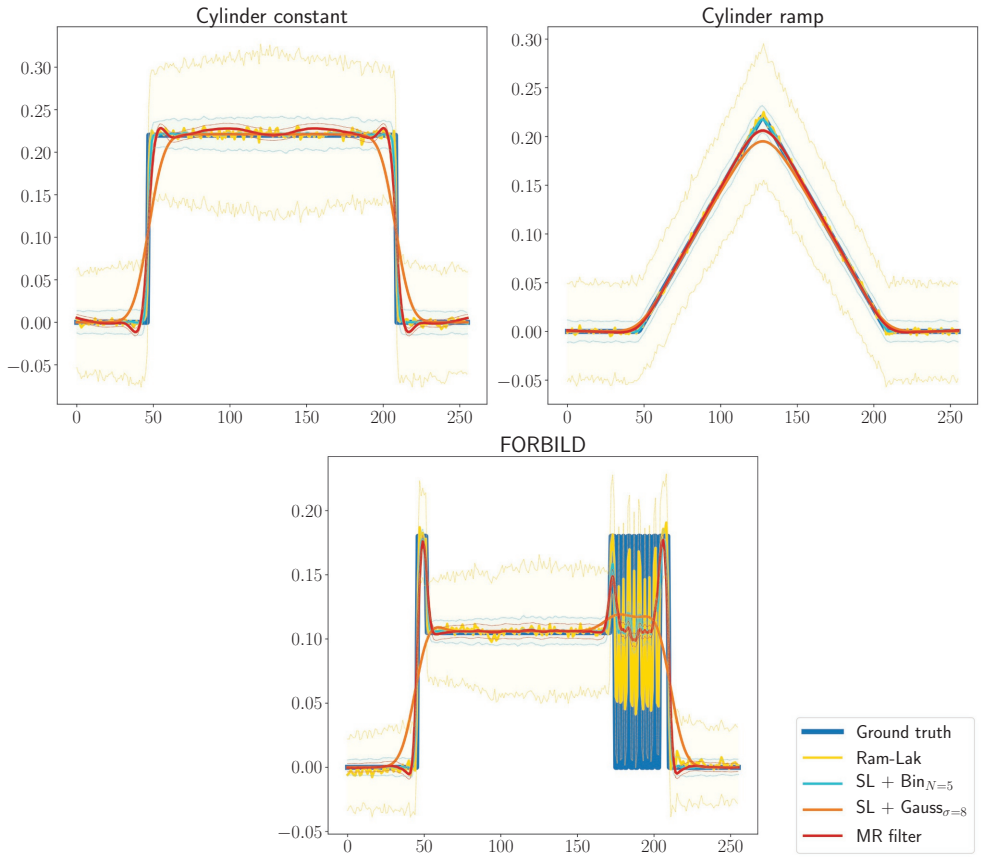
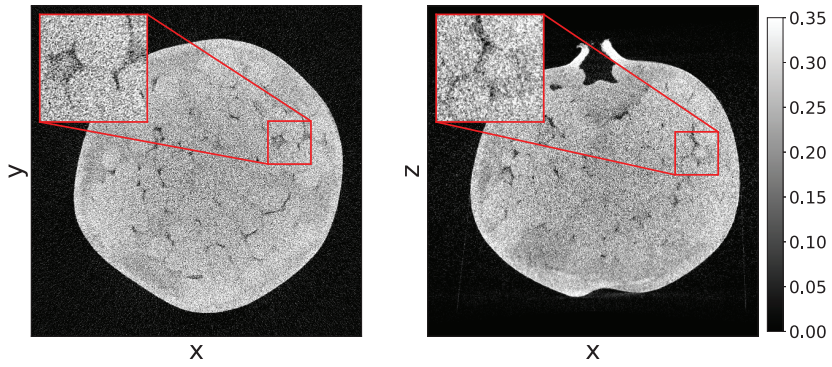
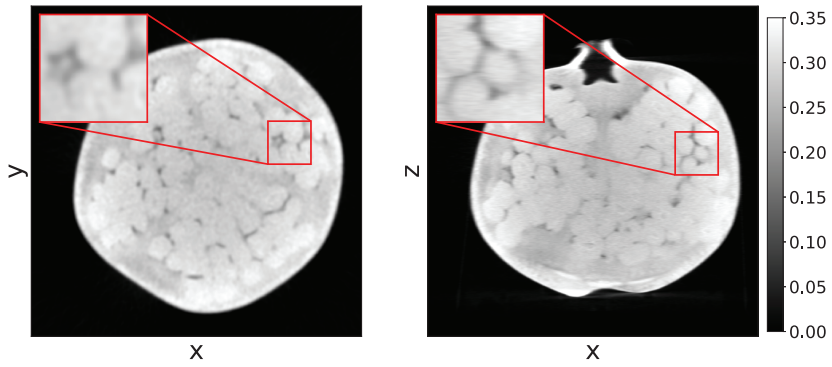


Figure 3.13: Average reconstructions and standard deviation of the x-axis for a (Left) constant cylinder phantom (Middle) ramp cylinder phantom (Right) FORBILD phantom. The average and standard deviation are computed over 500 reconstructions with 256^3 voxels of input data with different initialisations of emitted photon count $I_0 = 256$ and 360 projection angles.

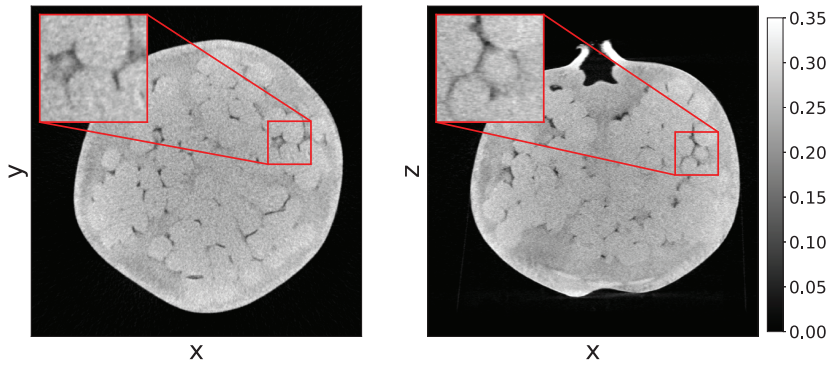
This indicates that the estimation for the regularization parameter is relatively low, which is most likely due to noise in the low resolution high quality reconstruction used to determine this parameter.



(a) FDK reconstruction with Shepp-Logan filter.



(b) FDK reconstruction with SL - Gauss$_{\sigma=8}$ filter.



(c) FDK reconstruction with MR filter.

Figure 3.14: Reconstructions of the low-dose Pomegranate dataset with 500 equidistant projection angles.

Table 3.1: Reconstruction quality measures from the high- and low-dose Pomegranate1 dataset with 64 and 500 equidistant projection angles (Left & Right), respectively.

High dose, 64 projection angles		
Method	MAE	SSIM
Shepp-Logan	0.2290	0.1789
SL + $\text{Bin}_{N=2}$	0.2839	0.1948
SL + $\text{Bin}_{N=5}$	0.2620	0.2350
SL + $\text{Gauss}_{\sigma=5}$	0.2112	0.5365
SL + $\text{Gauss}_{\sigma=8}$	0.2037	0.6304
MR filter	0.0711	0.6500

Low dose, 500 projection angles		
Method	MAE	SSIM
Shepp-Logan	0.2426	0.1799
SL + $\text{Bin}_{N=2}$	0.2794	0.2147
SL + $\text{Bin}_{N=5}$	0.2513	0.2745
SL + $\text{Gauss}_{\sigma=5}$	0.2144	0.6983
SL + $\text{Gauss}_{\sigma=8}$	0.2152	0.7294
MR filter	0.0636	0.5934

3.5 Conclusions and outlook

We have proposed a computationally efficient and automated method to compute a FDK-filter for a given imaging scenario (scanned object, number of angles, dose) that is optimal with respect to an objectively defined quality criterion. For cone-beam CT scanners used in research environments, where many different objects are scanned and parameters are often varied, our method can be used to automatically determine a filter that yields accurate results across a range of scanning conditions and tasks to be performed.

The experimental results demonstrate that for a variety of objects, scan settings (number of angles and noise levels), and tasks (porosity quantification, threshold-based segmentation), the MR filters computed by our approach are not task or problem specific and yield accurate results in terms of several different metrics that are comparable to manually selected filters.

Although the computational cost of computing an MR-filter is substantially lower than running an iterative reconstruction algorithm, it is much higher than the computational cost of FDK with a fixed filter. When carrying out batches of scans for similar objects, one can reuse an MR-filter computed for a particular object to further bring down the computational cost.

3.6 Appendices

3.6.1 Filter approximation

We want to design an expansion operator E to reduce the dimension needed to describe a filter. **Figure 3.15** shows five examples of analytic FDK filters as

presented in [Nat01; Buz08]. From the figure we see that the filters are symmetric and have most information on a fine grid around the origin.

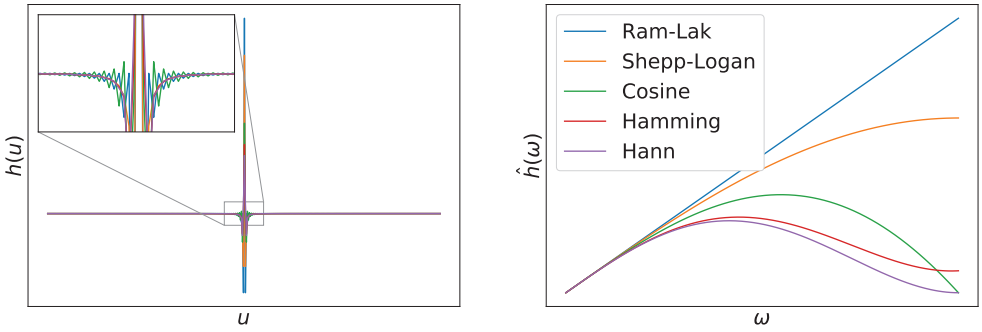


Figure 3.15: Standard filters in the spatial (left) and Fourier (right) domain.

To emulate this behavior, we introduce a re-sampling scheme as used in [BK06; PB13; PB14]. We do this by defining the discretised filter space $\mathbb{D}_u$ to be a uniform grid with $2N_u$ points on the interval $[-D_u, D_u]$ and grid points u_j. Now we re-sample these points into bins β_i with length:

$$d_i = \begin{cases} \frac{\Delta_u}{2}, & \text{for } i = 0, \\ \Delta_u, & \text{for } 0 < i < b, \\ 2^{b-i}\Delta_u, & \text{for } i \geq b, \end{cases} \tag{3.10}$$

with $\Delta_u = \frac{D_u}{N_u}$, the length of a single detector pixel. Here the indices i denote the place of the bin with respect to the central bin, *i.e.*, $\beta_0 = [0, d_0]$ and $\beta_1 = (d_0, d_0 + d_1]$, etc. This binning strategy results in a grid with uniform spacing around the origin and, depending on the binning parameter b, an exponentially coarsening grid for the outer regions. Instead of piece-wise constant basis functions as were used in [BK06; PB13; PB14], we will use piece-wise linear basis functions. This is because we observed that filters computed with these basis functions behave regularly in Fourier space. We define these functions as such:

$$\phi^{\text{lin}}_{i,u} = \begin{cases} \frac{s_i - u_j}{d_{i-1}}, & \text{for } \left|u_j\right| \in \beta_{i-1}, \\ \frac{u_j - s_i}{d_i}, & \text{for } \left|u_j\right| \in \beta_i, \end{cases} \tag{3.11}$$

where s_i is the boundary value of the i^{th} bin, such that $\beta_i = (s_i, s_{i+1}]$.

Spatial representations of filters created with piece-wise constant and linear basis functions are shown in **Figure 3.16**. we use the binning parameter $b = 2$, which strongly limits the computational effort of computing a filter. In experiments

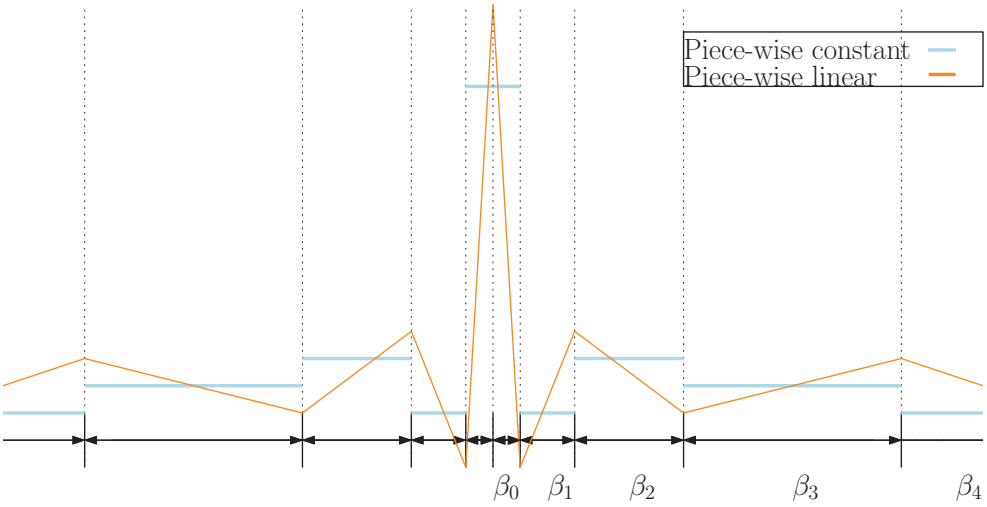


Figure 3.16: Examples of exponentially binned filters with piece-wise constant and linear basis functions. Binning parameter $b = 2$.

we observed that selecting a binning parameter greater than 2 does not improve the quality of the reconstructions substantially.

3.6.2 Filters selected for comparison

Manually selected filters for the FDK algorithm are often analytic filters combined with low-pass filters. These filters are typically manually adapted to the scan conditions. We selected several of such filters as reference methods for our results. We consider two types of low-pass filters.

Binomional filters are defined as:

$$\text{Bin}_N = \underbrace{([1\ 1] \otimes [1\ 1] \otimes \cdots \otimes [1\ 1])}_{N \text{ times}} / 2^{N+1}, \qquad \text{with } N \in \mathbb{N}_{>0}, \tag{3.12}$$

and N the order of the binomial filter.

Gaussian filters are defined as:

$$\text{Gauss}_{\sigma,j} = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{(j-N_u/2)^2}{2\sigma^2}}, \qquad \text{with } \sigma > 0, \tag{3.13}$$

with $\text{Gauss}_{\sigma,j}$ the j^{th} element of $\text{Gauss}_\sigma \in \mathbb{R}^{N_u}$ and σ the standard deviation of the Gaussian in pixels.

We combine the filters by convolving the low-pass and analytic filters in the spatial domain and cutting them off. These filters are specified by first referencing

the analytic filter, then the low-pass filter and lastly the value of the parameter related to this low-pass filter.

For comparing the results of our proposed MR filter with a set of low pass filters combined with analytic filters (as defined above), we carried out a search over the parameter space of these filters (specifically $N \in \{1, ..., 8\}$ and $\sigma \in \{1, ..., 10\}$) and selected four filter settings such that for each of the experiments reported in the chapter, at least one of the selected filters has quality metrics close to the optimum across the parameter space (with respect to MAE, SSIM, and task specific metrics). The selected filters are the SL + $\text{Gauss}_{\sigma=5,8}$ filters and the SL + $\text{Bin}_{N=2,5}$ filters.

3.6.3 Methods used in experiments

3.6.4 Pore size distribution

We derive the pore size distribution from the cumulative pore size distribution of a segmentation. The cumulative pore distribution is computed through inverting the segmentation, which leads to pores being particles, and step-wise erosion of these resulting particles. By controlling the number of voxels that we erode, we know how many pores there are with a certain radius (this radius is expressed in the number of voxels).

3.6.5 Modulation transfer function

The *modulation transfer function* (MTF) of a reconstruction method is defined as the magnitude of the Fourier transform of its *point spread function* (PSF). However, due to aliasing, computing the MTF directly from the PSF is unstable. Therefore, instead of measuring the PSF directly, we measure the *edge spread function* (ESF) and compute its derivative, which coincides with the PSF. Following the ASTM standard [AST13], we consider a homogeneous cylinder, with its axis on the z-axis, as the measured object. Since the FDK algorithm and the object are radially symmetric, every ray from the center of the cylinder to the edge of the reconstruction domain is a realization of the ESF. Defining ϕ to be the angle that this ray makes with the x-axis we can compute the average ESF over all ϕ, which we denote by $\mathbb{E}_\phi$ (we limit ourselves to the z=0 plane). Next we compute the gradient of this average ESF and use it to compute the MTF acting in the z=0 plane.

In mathematical terms, the MTF in the z=0 plane, or w.l.o.g. the x-direction, due to the radial symmetry, related to a filter $\mathbf{h}$ in the FDK algorithm is defined as

follows:

$$\mathrm{MTF}_{\mathbf{h}}(\omega_x) = \left|\mathscr{F}_{1D}\left\{\mathrm{PSF}_{\mathbf{h}}\right\}(\omega_x, 0, 0)\right|, \tag{3.14}$$

$$= \left|\mathscr{F}_{1D}\left\{\tfrac{d}{dx}\mathrm{ESF}_{\mathbf{h}}\right\}(\omega_x, 0, 0)\right|, \tag{3.15}$$

$$\approx \left|\mathscr{F}_{1D}\left\{\tfrac{d}{dx}\left(\mathbb{E}_{\phi}\left[F(\mathbf{y}_{\mathscr{C}}, \mathbf{h})_{z=0}\right]\right)\right\}(\omega_x, 0, 0)\right| \tag{3.16}$$

where $\mathbf{y}_{\mathscr{C}}$ is the cone-beam projection data of the aforementioned cylinder.

Chapter 4

Neural Network Feldkamp-Davis-Kress algorithm

4.1 Introduction

Circular cone-beam (CCB) Computed Tomography (CT) has become an integral part of non-destructive imaging in a broad spectrum of applications, such as industrial quality control [GUV11], materials sciences [Die+14; Bul+16] and medical imaging [For+02; GKT17]. Limitations on the scanning process caused by the need to scan a large number of objects in a short amount of time lead to measurements with a low number of projection angles or high noise levels. Additionally, CT reconstruction has become a *big data* problem due to the development of readily available high-resolution CT-scanners [TESb; TESa; Can]. This stresses the need for computationally efficient reconstruction methods that are applicable to a broad spectrum of high-resolution problems and produce accurate results from data with a high noise levels, low number of projection angles or large cone angles.

In practice, if computational efficiency is a constraint and especially for high-resolution problems, *direct methods* (*e.g.*, the filtered backprojection (FBP) algorithm [Nat01], the Feldkamp-Davis-Kress (FDK) algorithm [FDK84] and the Katsevich algorithm [Kat03]) are still the common choice of reconstruction method

This chapter is based on:

A computationally efficient reconstruction algorithm for circular cone-beam computed tomography using shallow neural networks. *MJ Lagerwerf, DM Pelt, WJ Palenstijn, KJ Batenburg*. Journal of Imaging (Submitted for publication).

[PSV09]. While *iterative methods* have been shown to be more accurate for noisy and limited data problems [ROF92; BKP10; SP08; Jia+10; Niu+14; EF03], they have a significantly higher computational cost. Consequently there have been efforts to improve the accuracy of direct methods by computing data-specific or scanner-specific filters [Zen12; Nie+12; BP12; PB14; Lag+20]. Although these strategies do improve the reconstruction accuracy, they also add significant computational effort or are specific to one modality, *e.g.*, tomosynthesis [Kun+07].

An emerging approach for improving direct methods is to use *machine learning* to remove artifacts from the reconstructions. The idea is to use high-quality reconstructions to train a neural network that removes artifacts from low-quality reconstructions using a supervised learning approach. This *post-processing* approach has shown promising results for computed tomography using deep neural networks (DNNs) [Jin+17; PBS18; Kid+18]. Deep neural network structures contain a large number of layers, leading to millions of trainable parameters and therefore require a large amount of training data [PS18]. This is problematic in CT imaging, since there is often a limited amount of training data available, *e.g.*, due to scanning time, dose, and business-related concerns. Moreover, for the available data there are often no reference datasets or annotations available [Wan+18]. The large amount of training data and large number of parameters also lead to long training times. While for standard 2D networks the training time ranges between a couple of hours and a couple of days (see **Section 4.5.1**), for 3D networks the training time becomes prohibitively long [Çiç+16] (*i.e.*, weeks). Therefore, to apply post-processing to 3D problems the reconstruction volume can be considered as a stack of 2D problems [RFB15; PBS18] for which one 2D network is trained and then applied in a *slice-by-slice* fashion to the 3D volume. Although this strategy reduces the training time and the training data constraints, applying a 2D network to all slices can still be computationally intensive due to the number of slices in the 3D volume. A more in-depth discussion on current developments related to machine learning methods in CT imaging is given in **Section 4.2**.

In this chapter we propose the Neural Network FDK (NN-FDK) reconstruction algorithm. It is a direct reconstruction method that is designed to produce accurate results from noisy data, data with a low number of projection angles, or a large cone angle, but still maintains a similar computational efficiency and scalability as the standard FDK algorithm. Moreover, the algorithm has a fast training procedure, and requires a limited amount of training data.

The NN-FDK algorithm is an adaptation of the standard FDK algorithm using a shallow multilayer perceptron network [Bis06] with one fully connected hidden layer, a low number of trainable parameters and low memory constraints. We will show it is possible to interpret the weights of the first layer of the perceptron

network as a set of learned filters for the FDK algorithm. We can then use the FDK algorithm to evaluate the network efficiently for all voxels simultaneously to arrive at an accurate reconstruction for the CCB CT problem.

The NN-FDK algorithm is an extension of the method proposed in [PB13] for the Filtered Backprojection (FBP) algorithm [Nat01]. The derivation of the approach outlined in [PB13] relies on the shift-invariance property of the FBP algorithm. We will show that, although the FDK algorithm does not have this shift-invariance property, we can derive a similar method for the FDK algorithm. Moreover, the proposed strategy can be extended to any linear filtered backprojection type reconstruction method.

Using both simulated and experimental data, we compare the proposed method with the standard FDK algorithm, SIRT [VV90] with a nonnegativity constraint (SIRT^+), which is a commonly used iterative algorithm for CT problems, and two 2D deep neural networks (U-net [RFB15] and MSD [PBS18]) trained to remove reconstruction artifacts from slices of standard FDK reconstruction. We show that the NN-FDK algorithm is faster to evaluate than all but the standard FDK algorithm and orders of magnitude faster to train than the considered DNNs, with only a slight reduction in reconstruction accuracy compared to the DNNs.

The chapter is structured as follows. In **Section 4.3** we give definitions and introduce our method. In **Section 4.4** we introduce the data and the parameters used for the experiments. The experiments and their results are shown and discussed in **Section 4.5**. The chapter is summarized and concluded in **Section 4.6**.

4.2 Related work

Using *machine learning* methods is an emerging approach in CT imaging [Wan+18]. *Deep learning* methods have shown promising results for many applications within the development of CT reconstruction methods [KMY17]. For the sake of exposition, we split these machine learning approaches into two categories: (i) Improving standard reconstruction methods by replacing components of the reconstruction method with networks specifically trained for the application; and (ii) improving the image quality of reconstructions computed with existing reconstruction methods by training neural networks to perform *post-processing* in order to remove artifacts or reduce noise.

Examples of the first strategy (improving standard reconstruction methods) applied to iterative methods are the learned primal-dual reconstruction algorithm [AÖ17; AÖ18], variational networks [Kob+17; Ham+18], plug and play priors [VBW13; REM17; RS18], and learned regularizers [LÖS18; Muk+20]. These methods achieve promising results in reconstruction accuracy and generalizability.

However, their high computational cost limits the applicability if high throughput is required. Examples for this strategy applied to direct methods are the NN-FBP method [PB13], and also the NN-FDK method introduced in this chapter. These methods are designed to improve the image quality of direct methods for data with limitations (*e.g.*, data with noise or a low number of projection angles) while maintaining their computational efficiency.

Examples of the second strategy (learned post-processing) have demonstrated substantial improvements in reconstruction quality for CT imaging [RFB15; KMY17; PS18; Jin+17]. This is aided by the fact that the post-processing problem can be viewed as a classic imaging problem — *e.g.*, denoising, segmentation, inpainting, classification — for which many effective machine learning methods have already been developed [SLD17; PCC18; Zha+17]. Although the general trend is towards deeper networks to make such networks more expressive [YHC18], this can lead to problems with scalability for large 3D image datasets.

The rise in popularity of machine learning in CT is driven by the increased computational possibilities and although these advances are sufficient to handle most 2D problems, scaling towards 3D problems can be problematic, due to memory constraints. This is illustrated in **Figure 4.5** in **Section 4.5.1**, where we plotted the memory constraints for applying a 2D and 3D U-net and MSD network in terms of gigabytes (GiB) of memory as a function of the size of the image. This shows that in theory one could apply a 2D MSD network to images of 7500×7500 pixels (with a 24GiB GPU), but in 3D this limit lies around $400 \times 400 \times 400$ voxels. Considering that CT problems range between $256 \times 256 \times 256$ (small image size) up to $4096 \times 4096 \times 4096$ images, this gives an indication that scalability can become an issue, especially for 3D problems.

When applying machine learning techniques for improving the reconstruction quality in CT, a balance must be struck between image quality, running time, and memory requirements. Here we propose a method that achieves relatively high accuracy, while also being computationally efficient and scalable.

4.3 Method

The NN-FDK algorithm is a reconstruction algorithm with a machine learning component, meaning that a number of parameters of the reconstruction algorithm are optimized through *supervised learning* [AB09]. Similar to the network presented in [PB13], the *NN-FDK network* is a two layer neural network with a hidden layer and an output layer. We design the network such that it reconstructs one single voxel, but handles all voxels in a similar manner. This means that we only have to train one network for a full reconstruction. We consider the NN-FDK algorithm to

have three parts: The *NN-FDK network*, the *NN-FDK reconstruction algorithm* and the *training process*.

We introduce the reconstruction problem, FDK algorithm, a filter approximation method and the definition of a perceptron in **Section 4.3.1**. In **Section 4.3.2** we give the *NN-FDK reconstruction algorithm* and derive from this algorithm the *NN-FDK network*. The input of the network that is needed in the *training process* is a pre-processed version of the input of the reconstruction algorithm. In **Section 4.3.3**, we discuss how to compute this pre-processing step for all voxels simultaneously and we introduce the optimization problem and related notation for the training process. Lastly, we summarize and discuss the characteristics of the method in **Section 4.3.4**.

4.3.1 Preliminaries

Reconstruction problem

In this chapter we focus exclusively on the circular cone-beam (CCB) geometry, where the object rotates with respect to a point source and a planar detector, acquiring 2D cone-beam projections. The reconstruction problem for the CCB geometry can be modeled by a system of linear equations

$$W\mathbf{x} = \mathbf{y}, \tag{4.1}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the vector describing the reconstruction (*i.e.*, every element coincides with a voxel value), $\mathbf{y} \in \mathbb{R}^m$ is the vector describing the measured projection data, and $W \in \mathbb{R}^{m\times n}$ is a discretized version of the *cone-beam transform* or *forward projection*. For the sake of simplicity we assume that the volume consists of $n = N\times N\times N$ voxels and the detector consists of $N\times N$ pixels. We denote the number of angles with N_a, so we have $m = N_a \times N \times N$.

FDK algorithm & filter approximation

The FDK algorithm, as presented in [FDK84], is a filtered backprojection-type algorithm that solves the CCB reconstruction problem (4.1) approximately. First, for each projection angle, it applies a *reweighting* step, $r : \mathbb{R}^{N_a\times N\times N} \to \mathbb{R}^{N_a\times N\times N}$, that adapts the cone-beam data such that it approximately behaves as fan-beam data. Second, it applies a *filtering* step, that convolves the data with a one-dimensional *filter* $\mathbf{h}$ in a line-by-line fashion, $(-*-)_{1\mathrm{D}} : \mathbb{R}^{2N} \times \mathbb{R}^{N_a\times N\times N} \to \mathbb{R}^{N_a\times N\times N}$. Last, it applies a *backprojection* step. This step transforms the filtered projection data to the image domain. Using the notation of (4.1), the FDK algorithm is given

by

$$\mathrm{FDK}(\mathbf{y}, \mathbf{h}) = W^T(\mathbf{h} * r(\mathbf{y}))_{1\mathrm{D}}, \tag{4.2}$$

with W^T the transpose of W. The operator W^T is also known as the *backprojection operator*.

In [PB13; PB14; Lag+20] exponential binning is used to approximate filters, leading to $N_e \approx \log N$ coefficients to describe a filter. This approximation can be seen as a matrix $E \in \mathbb{R}^{2N \times N_e}$ applied to a coefficient vector $\mathbf{h}_e \in \mathbb{R}^{N_e}$:

$$\mathbf{h} \approx E\mathbf{h}_e. \tag{4.3}$$

The implementation details of this filter approximation can be found in [Lag+20].

Perceptron

In a similar manner as in [Bis06] we define a *perceptron* or *node* $\mathsf{P} : \mathbb{R}^l \to \mathbb{R}$ as a non-linear activation function $\sigma : \mathbb{R} \to \mathbb{R}$ applied to a weighted sum of the input $\eta \in \mathbb{R}^l$ with the weights $\xi \in \mathbb{R}^l$ and a bias $b \in \mathbb{R}$:

$$\mathsf{P}_{\xi,b}(\eta) = \sigma(\eta \cdot \xi - b) \tag{4.4}$$

In this chapter we will only consider the sigmoid function as activation function, *i.e.*, $\sigma(t) = 1/(1 + e^{-t})$.

A multilayer perceptron is a network structure containing two types of layers with perceptrons, where each perceptron operates on the outputs of the previous layer. These layers are, in order, any number of *hidden layers*, and the *output layer*. Note that the number of hidden layers and number of hidden nodes N_h in these layers can be chosen freely.

4.3.2 Reconstruction algorithm & Network design

We formulate the NN-FDK reconstruction algorithm in a similar fashion as the NN-FBP method in [PB13]. See **Algorithm 3** for a schematic representation. The NN-FDK reconstruction algorithm consists of N_h individual FDK algorithms executed on the input data y, each using its own (exponentially binned) filter $\mathbf{h}_e^k \in \mathbb{R}^{N_e}$. It combines these N_h volumes into a single reconstruction, using point-wise application of the activation function σ and an output perceptron with parameters $b_o, b_k \in \mathbb{R}$, and $\xi \in \mathbb{R}^{N_h}$.

We use $\theta = (\xi, b_o, \mathbf{h}_e^k, b_k)$ as short-hand for the full set of parameters of the NN-FDK reconstruction algorithm. The full algorithm is then given by the following equation.

$$\text{NN-FDK}_\theta(\mathbf{y}) = \sigma\Big(\sum_{k=1}^{N_h} \xi_k \sigma\left(\text{FDK}(\mathbf{y}, E\mathbf{h}_e^k) - b_k\right) - b_o\Big) \tag{4.5}$$

The FDK algorithm is a bilinear map in the input projection data and the used filter. Therefore, for fixed input projection data $\mathbf{y}$ and an expanded exponentially binned filter $E\mathbf{h}_e$, the FDK algorithm can be written as a linear map $F_\mathbf{y}$ applied to $E\mathbf{h}_e$. The product $F_\mathbf{y}E$ can be considered as a matrix of size $N^3 \times N_e$, and the v-th voxel of the output of the FDK algorithm is given by the inner product of $\mathbf{h}_e$ with $(F_\mathbf{y}E)_{v:}$, the v-th row of the matrix $F_\mathbf{y}E$. Using the definition of a perceptron (4.4) we can show the following:

$$(\text{NN-FDK}_\theta(\mathbf{y}))_v = \sigma\Big(\sum_{k=1}^{N_h} \xi_k \sigma\left((F_\mathbf{y}E\mathbf{h}_e^k)_v - b_k\right) - b_o\Big), \tag{4.6}$$

$$= \sigma\Big(\sum_{k=1}^{N_h} \xi_k \sigma\left((F_\mathbf{y}E)_{v:}\mathbf{h}_e^k - b_k\right) - b_o\Big), \tag{4.7}$$

$$= \mathsf{P}_{\xi,b_o}\left(\left[\mathsf{P}_{\mathbf{h}_e^k,b_k}((F_\mathbf{y}E)_{v:})\right]_k\right). \tag{4.8}$$

Therefore, we define the two-layer perceptron network $\mathsf{N}_\theta : \mathbb{R}^{N_e} \to \mathbb{R}$:

$$\mathsf{N}_\theta(\mathbf{q}) = \mathsf{P}_{\xi,b_0}\left(\left[\mathsf{P}_{\mathbf{h}_e^k,b_k}(\mathbf{q})\right]_k\right). \tag{4.9}$$

This is our NN-FDK network, and as we derived above, it has the following relationship with the NN-FDK reconstruction algorithm:

$$\mathsf{N}_\theta((F_\mathbf{y}E)_{v:}) = (\text{NN-FDK}_\theta(\mathbf{y}))_v. \tag{4.10}$$

This relationship shows that we can *evaluate* the NN-FDK reconstruction algorithm efficiently on full input projection data at once, but also *train* the NN-FDK network efficiently with each *individual* voxel $(\mathbf{x}_{\text{HQ}})_v$ in a high quality reconstruction yielding a training pair with input $(F_\mathbf{y}E)_{v:}$ and target $(\mathbf{x}_{\text{HQ}})_v$. A schematic representation of the network is given in **Figure 4.1**.

Note that we arrive at the same network structure as found in [PB13] for FBP, using only the properties that the FDK algorithm is a bilinear map in the data and the filter, and that all operations can be applied point-wise. Using this reasoning we can derive a similar network structure for any FBP-type method satisfying these conditions.

Even though we use the same network structure as [PB13], the way we compute inputs to the network is different. In [PB13], the input to the NN-FBP network is

explicitly calculated by shifting and adding projection data for each reconstruction pixel. The FDK algorithm has additional weighting factors and lacks the shift-invariance property, which makes the approach presented in [PB13] not directly applicable. In the next section, we detail an alternative method to compute the input. The same approach could be applied to the NN-FBP method, similarly simplifying the network input computations.

Algorithm 3 Neural Network FDK reconstruction algorithm

1: Given a set of parameters, $\theta := \left(\xi, b_o, \mathbf{h}_e^k, b_k\right)$.
2: Compute H_k for all nodes k of the hidden layer:
3: **for** $k = \{1, 2, .., N_h\}$ **do**
4: $\quad \mathrm{H}_k(\mathbf{y}) = \sigma\left(\mathrm{FDK}(\mathbf{y}, E\mathbf{h}_e^k) - b_k\right)$
5: Compute the output of the output layer:
$\mathrm{NN\text{-}FDK}_\theta(\mathbf{y}) = \sigma\left(\sum_{k=1}^{N_h} \xi_k H_k(\mathbf{y}) - b_o\right)$

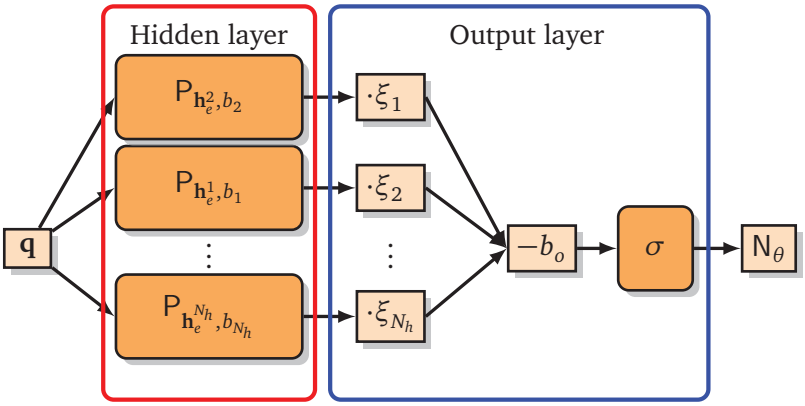


Figure 4.1: Schematic representation of the NN-FDK network, $\mathrm{N}_\theta : \mathbb{R}^{N_e} \to \mathbb{R}$, with N_h hidden nodes. Note that if we take $q = (F_{\mathbf{y}}E)_v$: we get $q \cdot \mathbf{h}_e^k = (\mathrm{FDK}(\mathbf{y}, E\mathbf{h}_e^k))_v$ in the perceptrons of the hidden layer and the output of the network is equal to the v-th voxel of the NN-FDK reconstruction algorithm.

4.3.3 Training process

Training and validation data

We will train our network using supervised learning, where we assume that we have N_{TD} and N_{VD} datasets available for training and validation, respectively.

These datasets consist of low quality tomographic input data and a high quality reconstruction from which we randomly draw a total of N_{T} training pairs and N_{V} validation pairs. Note that we ensure that every drawn pair is unique and that an equal number of pairs is taken from each dataset. Moreover, to avoid selecting too many training pairs from the background we only take training pairs from a region of interest (ROI) around the scanned object. This ROI is defined from the high quality reconstruction as the voxels in the reconstructed object plus a buffer of roughly $0.2N$ voxels around it.

Recall from the previous section that given low quality tomographic data $\mathbf{y}$ and a high quality reconstruction $\mathbf{x}_{\mathrm{HQ}}$ the matrix $F_{\mathbf{y}}E$ contains each input vector $Z = \left(F_{\mathbf{y}}E\right)_{v:} \in \mathbb{R}^{N_e}$ corresponding to the target voxel $O = (\mathbf{x}_{\mathrm{HQ}})_v$. However, due to memory constraints $F_{\mathbf{y}}E$ cannot be computed directly as a matrix product. Therefore, we observe that each column of $F_{\mathbf{y}}E$ is an FDK reconstruction with a specific filter:

$$(F_{\mathbf{y}}E)_{:j} = F_{\mathbf{y}}E\mathbf{e}_j = \mathrm{FDK}(\mathbf{y}, E\mathbf{e}_j), \tag{4.11}$$

with $\mathbf{e}_j \in \mathbb{R}^{N_e}$ the unit vector with all entries equal to zero except for the j-th element.

Learning problem

The parameters of the NN-FDK network are learned by finding the set of parameters $\theta^\star$ that minimize the *loss function* $\mathscr{L}$ on the training set. We minimize the ℓ^2-distance between the network output and the target voxel for all training pairs in T:

$$\theta^\star = \operatorname*{argmin}_\theta \mathscr{L}(\theta, T) = \operatorname*{argmin}_\theta \tfrac{1}{2} \sum_{j=1}^{N_{\mathrm{T}}} \left(O_j - \mathsf{N}_\theta(Z_j)\right)^2. \tag{4.12}$$

To minimize the loss function we use a quasi-Newton optimization scheme, the *Levenberg-Marquardt algorithm* (LMA) as proposed in [Lev44; Mar63]. This is a combination of gradient descent and the Gauss-Newton algorithm, improving the stability of Gauss-Newton while retaining its fast convergence and it is specifically designed to minimize a non-linear least squares problem such as (4.12). Note that the small number of parameters of the proposed network allows us to use such a method. Lastly, to avoid overfitting we check whether every update of the parameters also reduces the loss function on the validation set. We discuss the specifics of this algorithm in **Appendix 4.7.2**.

Method comparison: Goals

	Reconstruction		Training	
Method	Time	Accuracy	Data	Time
NN-FDK	++	?	++	+++
DNN	±	+++	±	- - -
FDK	+++	- -		
SIRT$^+$	- -	+		

Table 4.1: Comparison of reconstruction methods with respect to the goals formulated in **Section 4.1**. We consider a DNN to be 2D deep convolutional neural network (U-net & MSD-net) applied in slice-by-slice fashion to a standard FDK reconstruction. Reconstruction accuracy is defined as the accuracy of a method when reconstructing low quality data, *e.g.*, data with high noise or a low number of projection angles.

4.3.4 Method characteristics & comparison

To conclude the method section we compare the characteristics of the NN-FDK algorithm to those of several other methods. These methods are two 2D post-processing DNNs (U-net [RFB15] and MSD-net [PS18]) applied in a slice-by-slice fashion, the SIRT$^+$ algorithm [VV90] and the FDK algorithm. We focus our discussion on the goals formulated in **Section 4.1** and show a summary of this comparison in **Table 4.1**. The reconstruction accuracy will be discussed in **Section 4.5**.

Computational efficiency

We approximate the reconstruction time by counting how many times it has to evaluate its most expensive computations. For simplicity we assume that a backprojection takes approximately the same time as a forward projection, T_{BP}.

- **FDK:** The FDK algorithm consist of one reweighting, filtering and backprojection step, *i.e.*, :

$$T_{\text{FDK}} \approx T_{\text{BP}}. \tag{4.13}$$

- **NN-FDK:** The NN-FDK algorithm performs one FDK reconstruction per hidden node N_h. Therefore the reconstruction time becomes:

$$T_{\text{NN-FDK}} \approx N_h T_{\text{BP}}. \tag{4.14}$$

- **SIRT$^+$:** The SIRT$^+$ method evaluates a forward and backprojection for each iteration. For N_{iter} iterations, the reconstruction time becomes:

$$T_{\text{SIRT}^+} \approx 2N_{\text{iter}} T_{\text{BP}}. \tag{4.15}$$

- **DNN:** To evaluate a DNN an FDK reconstruction is performed and a 2D network is applied per slice of the FDK reconstruction.

$$T_{\text{DNN}} \approx T_{\text{BP}} + N T_{\text{DNN}}, \tag{4.16}$$

  with T_{DNN} the time it takes to apply a 2D DNN.

On a modern GPU and with $N = 1024$ and $N_a = 360$, we found in our experiments that $T_{\text{BP}} \approx 10$ s and $T_{\text{DNN}} \approx 0.5$ s.

Comparing the reconstruction times, we see that NN-FDK is similar to FDK when the number of nodes N_h is small, which is the case since we will take N_h=4 (see **Section 4.4.3**). For DNNs the computational load of applying a 2D network leads to relatively high reconstruction times compared to the FDK algorithm. Lastly, we note that the number of iterations N_{iter} often lies between the 20 and 200, making SIRT$^+$ several times slower than the (NN-)FDK algorithm.

Number of trainable parameters

The number of trainable parameters is closely related to the amount of training data required to train a network [PS18]. From the definition of the NN-FDK network (4.5) we can compute the number of trainable parameters $|\theta|$:

$$|\theta| = (N_e + 2)N_h + 1, \tag{4.17}$$

with $N \gg N_h, N_e > 0$. Taking $N_h = 4$ and $N = 1024$ gives $|\theta| = 61$, which is several orders of magnitude lower than the typical numbers of parameters in a DNN (several tens of thousands to millions).

Training time

In the training step a solution to the minimization problem (4.12) is computed. For the NN-FDK algorithm this problem has N_{T} samples and $|\theta|$ unknowns. In a similar fashion we can formulate a least squares problem for training a DNN. Even assuming that we only take the same number of training samples to train the DNNs, this least squares problem is already orders of magnitude larger than that for NN-FDK due to the difference in the number of trainable parameters. Moreover, the LMA (the algorithm used to train NN-FDK) approaches quadratic convergence,

which means it will need fewer iterations to converge than a first order scheme such as ADAM [KB14], which is often used for training DNNs. Considering these two observations we expect the training time of the NN-FDK algorithm to be lower than the training time of the DNNs.

4.4 Experimental setup

We carried out a range of experiments to assess the performance of the NN-FDK algorithm with respect to the goals formulated in **Section 4.1** compared to several alternative methods. In this section we introduce the setup of these experiments. We describe the simulated data in **Section 4.4.1** and the experimental data in **Section 4.4.2**. In **Section 4.4.3** we discuss the specific network structure for the NN-FDK algorithm and the training parameters used. Finally, we give the quantitative measures we use to compare the reconstruction in **Section 4.4.4**.

4.4.1 Simulated data

We consider two types of phantom families for the simulated data experiments: the *Fourshape phantom family* and the *Random Defrise phantom family*. Examples are shown in **Figure 4.2** and **Figure 4.3**, respectively. The Fourshape phantom family contains three random occurrences of each of four types of objects: an ellipse, a rectangle, a Gaussian blob and a Siemens star. For evaluation and visualization of the reconstructions we fixed one realization that clearly shows at least one of all the four objects and we will refer to this phantom as the *Fourshape test phantom*. The Random Defrise phantom family is a slight adaptation of the phantom introduced in [KND98], which is a common phantom for assessing the influence of imaging artifacts due to the cone angle. Here we vary the intensities, orientations and sizes of the disks making sure they do not overlap. Again, we define a test phantom for evaluation and visualization, which is in this case the standard *Defrise phantom* without alternating intensities (right in **Figure 4.3**). To simulate realistic settings, we scale the phantoms to fit inside a 10 cm cube, and use an attenuation coefficient of $\mu = 0.22\ \text{cm}^{-1}$, approximating that of various common plastics at 40 keV [HS95]. These phantoms are defined through geometric parameters, and can therefore be generated for any desired N. For our experiments we will take $N = 1024$. Details about how we generate the data are given in **Appendix 4.7.1**.

To compute a high quality reconstruction $\mathbf{x}_{\text{HQ}}$ that can be used as target for training (recall **Section 4.3.3**) we consider a simulated dataset with $N_a = 1500$ projection angles, low noise ($I_0 = 2^{20}$ emitted photon count) and cone angle of 0.6 degrees and reconstruct this problem with the standard FDK algorithm using a

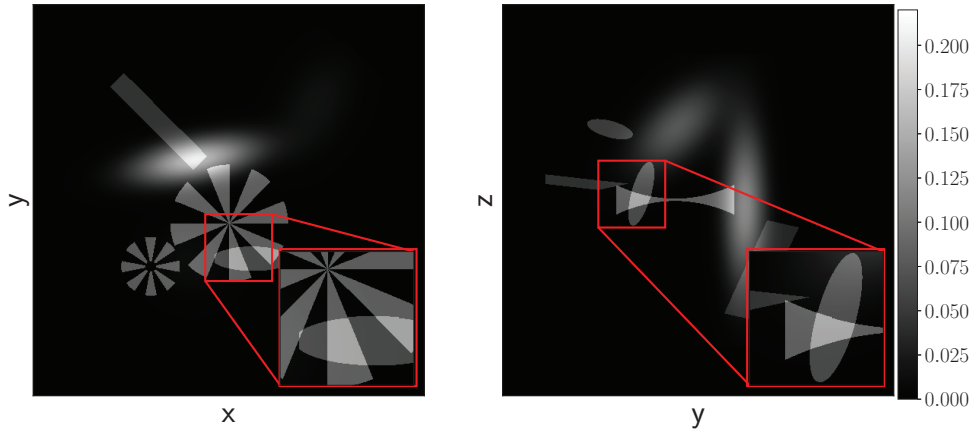


Figure 4.2: Slices, (Left) $z = 0$, (Right) $x = 0$, of the Fourshape test phantom. This phantom is designed such that at least one of all objects can clearly be observed in the slices.

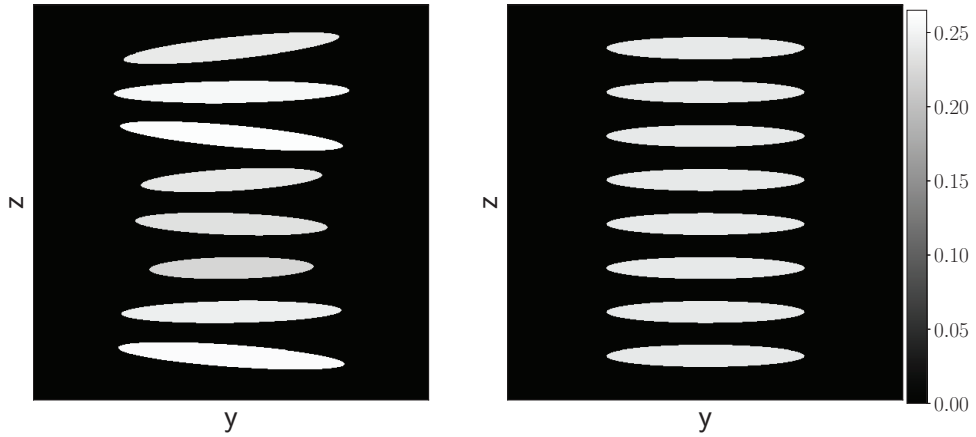


Figure 4.3: The $x = 0$ slice for a Random Defrise phantom (Left) and the standard Defrise phantom without alternating intensities from [KND98] (Right).

Hann filter [Nat01].

4.4.2 Experimental data

For experimental data we consider a set of CT scans that were recorded using the custom-built and highly flexible FleX-ray CT scanner, developed by XRE NV and located at CWI [Cob+20]. This scanner has a flat panel detector with 972 × 768

pixels and a physical size of 145.34×114.82 mm. This set of 42 scans was set up to create high noise reconstruction problems and low noise reconstruction problems with a low number of projection angles.

We acquired high-dose (low noise) and low-dose (high noise) scans of 21 walnuts. The datasets contain 500 equidistantly spaced projections over a full circle. The distance from the center of rotation to the detector was set to 376 mm and the distance from the source to the center of rotation was set to 463 mm. The scans were performed with a tube voltage of 70 kV. The high-dose scan was collected with a tube power of 45 W and an exposure time of 500 ms per projection. The low-dose scan was collected with a tube power of 20 W and an exposure time of 100 ms per projection. To create a low noise reconstruction problem with a low number of projection angles we considered the high-dose scan but only took every 16-th projection angle. As high quality reference reconstructions we used SIRT^+ reconstructions with 300 iterations (SIRT^+_{300}) of the high-dose scans with all available projection angles ($N_a = 500$). We will refer to these reconstructions as the *gold standard* reconstruction and we show such a reconstruction in **Figure 4.4**. These datasets are available at Zenodo [LCB20].

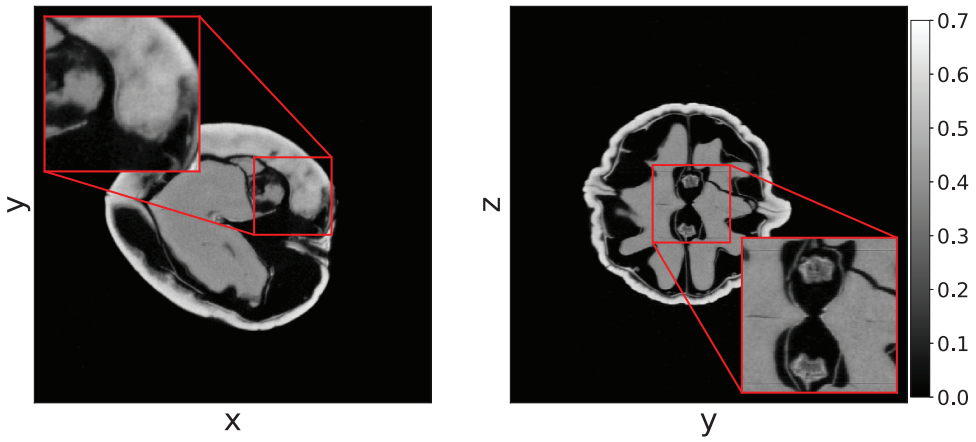


Figure 4.4: The $z = 0$ (Left) and $y = 0$ (Right) slice of the gold standard reconstruction of the high-dose dataset of the 21^{st} walnut with full number of projection angles. The projection data is acquired using the FleX-ray scanner located at the CWI [LCB20].

4.4.3 Parameter settings NN-FDK

Network structure

In our initial experiments we found that taking more FDK-perceptrons improved the accuracy of the networks, at the cost of increasing the training and reconstruction time. We found that $N_h = 4$ FDK-perceptrons led to a good balance between accuracy and reconstruction time, which is similar to the findings in [PB13].

Training data

We found that, similar to the findings in [PB13], taking $N_\mathrm{T} = 10^6$ voxels for training and $N_\mathrm{V} = 10^6$ for validation is sufficient for training an NN-FDK network.

The network structures and training procedure used for the U-nets and MSD networks are discussed in **Appendix 4.7.1**.

4.4.4 Quantitative measures

To quantify the accuracy of the reconstructions we consider two measures, the test set error (TSE) and the structural similarity index (SSIM). These measures compare the reconstructed image $\mathbf{x}_r$ to a high quality reconstruction $\mathbf{x}_\mathrm{HQ}$ on the ROI (as discussed in **Section 4.3.3**).

The TSE is the average loss[1] of the test set, where the test set is all the voxels defined in the ROI of $\mathbf{x}_\mathrm{HQ}$:

$$\mathrm{TSE}(\mathbf{x}_r, \mathbf{x}_\mathrm{HQ}) = \frac{1}{N_\mathrm{ROI}} \mathscr{L}(\mathscr{I}_\mathrm{ROI}(\mathbf{x}_\mathrm{HQ}), \theta), \tag{4.18}$$

$$= \frac{1}{2N_\mathrm{ROI}} \left\| \mathscr{I}_\mathrm{ROI}(\mathbf{x}_\mathrm{HQ} - \mathbf{x}_r) \right\|_2^2. \tag{4.19}$$

with $\mathscr{I}_\mathrm{ROI} : \mathbb{R}^{N^3} \to \mathbb{R}^{N^3}$ the masking function for the ROI and N_ROI the number of voxels in the ROI.

The SSIM [Wan+04] is implemented based on the scikit-image 0.13.1 [Wal+14] package, where all the constants are set to default and the filter is uniform with a width of 19 pixels.

[1] Recall (4.12) in **Section 4.3.3**

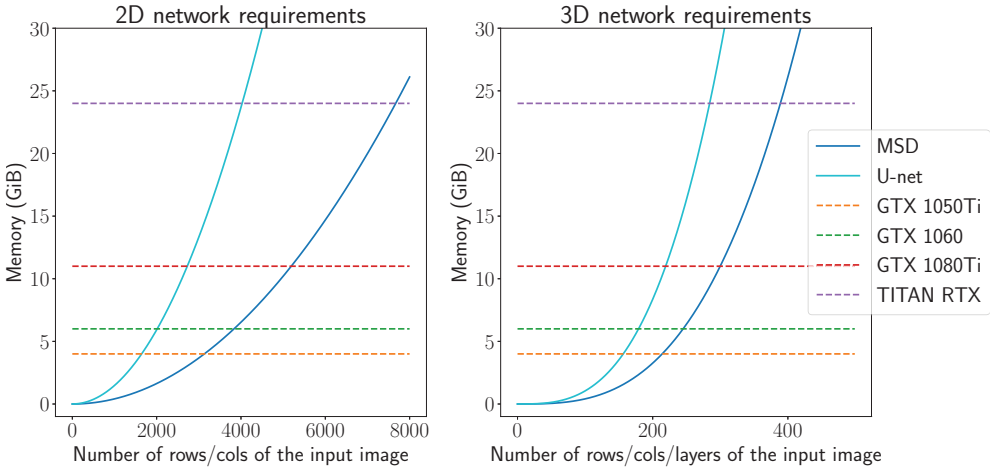


Figure 4.5: The required memory to store all intermediate images for applying a 2D and 3D U-net and MSD network as a function of the input image size.

4.5 Results and discussion

4.5.1 Scalability

Memory scaling

The required memory to store all intermediate images for a forward pass of a 2D or a 3D U-net and MSD network as a function of the input image size is shown in **Figure 4.5**. Considering that CT imaging problems typically range from $256 \times 256 \times 256$ up to $4096 \times 4096 \times 4096$ we conclude from these figures that full 3D networks do not fit into GPU memory for higher resolutions and that even for 2D U-nets not all resolutions fit on the GPU. As a forward pass of the NN-FDK algorithm requires only one additional reconstruction volume[2] compared to the FDK algorithm, the memory requirements of the NN-FDK algorithm are roughly 2 times the memory required by the FDK algorithm.

Training time

In **Figure 4.6** we compare the training processes by plotting the progress of the network training (measured by the TSE) as a function of the number of voxels that the network has seen during training. We see that the NN-FDK has seen $1.1 \cdot 10^8$

[2]Technically a forward pass of the NN-FDK algorithm can be done for every voxel separately, however, for the sake of comparison we assume a forward pass is for a full reconstruction volume.

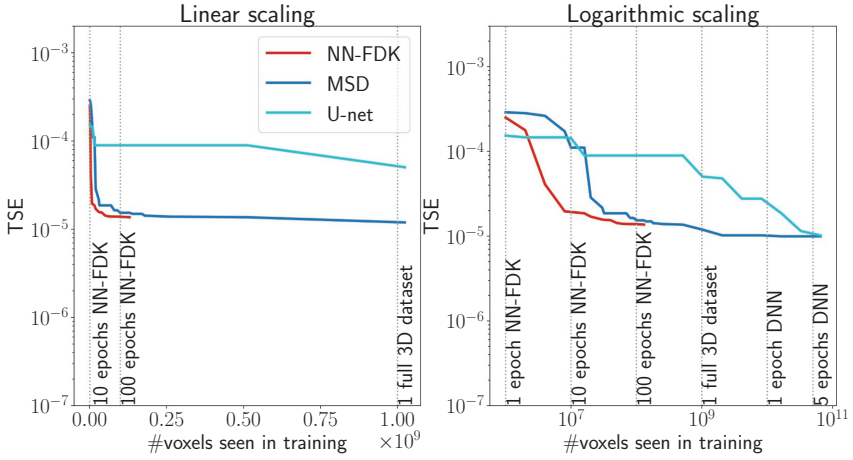


Figure 4.6: The TSE as a function of the number of voxels the training process has seen. We report the lowest TSE up till that point. The networks are trained on randomly generated Fourshape phantoms with size $N = 1024$, $N_a = 32$ projection angles and no noise. (Left) Linear scaling in the number of voxels ranging from 1 epoch for the NN-FDK (10^6 voxels), to 1 full 3D dataset (10^9 voxels). (Right) Logarithmic scaling in the number of voxels. Ranging from 1 epoch for the NN-FDK network (10^6 voxels) to 5 epochs for a DNN ($5 \cdot 10^{10}$ voxels).

voxels when it converges to TSE$= 1.4 \cdot 10^{-5}$, whereas, MSD and U-net have seen $5.1 \cdot 10^8$ voxels and $3.2 \cdot 10^9$ voxels, respectively, at the point they first achieve a similar TSE. Important to note is that both U-net and MSD are not yet converged when they match the TSE of NN-FDK, and in general the DNNs achieve lower TSEs than NN-FDK.

In **Table 4.2** we show various timings and properties with respect to the training process. These timings are recorded using one Nvidia GeForce GTX 1080Ti with 11GiB memory. We define a converged training process as 100 epochs without improvement on the validation set error and the number of epochs to converge as the epoch with the lowest validation set error during a converged training process. From these results we see that the size of the training problem influences the time per epoch as an NN-FDK epoch is sub-second and the time per epoch for DNNs is in the range of hours.

In practice, we observed that after 2 days of training for the DNNs, any additional training only achieved marginal improvements. Therefore, in the following experiments we train all DNNs for 2 days with one Nvidia GeForce GTX 1080Ti GPU, unless mentioned otherwise.

Training process

	NN-FDK$_4$	MSD	U-net
Voxels seen in one epoch	$1 \cdot 10^6$	$1.1 \cdot 10^{10}$	$1.1 \cdot 10^{10}$
Time per epoch	0.1336 (s)	0.95 (h)	2.36 (h)
Time to converge	28 (s)	± 10 (d)	± 14 (d)
Epochs to converge	110	128	42
Epochs in 2 days	-	45	18

Table 4.2: Timings and properties of the considered training processes. We define a converged training process as 100 epochs without improvement on the validation set error. The epochs to converge is therefore the epochs computed of such a process minus 100. The training was performed using one Nvidia GeForce GTX 1080Ti GPU (11 GiB).

Reconstruction time

We measured the average reconstruction times and corresponding standard deviation over 120 reconstructions with resolution $N^3 = 1024^3$ and $N_a = 360$ projection angles. These reconstructions are computed using one Nvidia GeForce GTX 1080Ti with 11 GiB memory. The results are shown in **Table 4.3**. We define the reconstruction time as the time it takes to compute the full 3D volume. This means for U-net and MSD, an FDK reconstruction needs to be computed and the network needs to be applied $N = 1024$ times to a 2D slice. Although every application can be done within a second (U-net $\approx 0.3s$, MSD $\approx 0.7s$) this leads to long reconstruction times.

Reconstruction times

FDK	SIRT$^+_{200}$	NN-FDK$_4$	U-net	MSD
28 ± 8	3225 ± 916	76 ± 8	382 ± 69	809 ± 86

Table 4.3: Average and standard deviation of the reconstruction times (in seconds) computed over 120 reconstruction problems with $N = 1024$ and $N_a = 360$ projection angles. These reconstructions are computed using one Nvidia GeForce GTX 1080Ti GPU (11 GiB).

4.5.2 Reconstruction accuracy for simulated data

For evaluating the reconstruction accuracy using simulated data, we consider 16 cases: 6 different noise levels, 5 different numbers of projection angles and 5

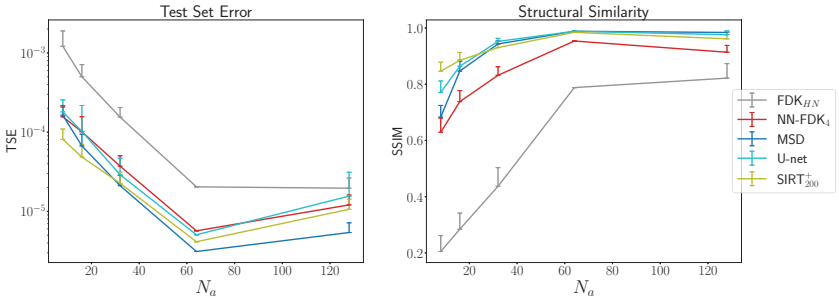
different cone angles. For each case an NN-FDK, MSD and U-net network was trained. For the training process of NN-FDK we used $N_{\mathrm{T}} = 10^6$ training voxels and $N_{\mathrm{V}} = 10^6$ validation voxels from $N_{\mathrm{TD}} = 10$ and $N_{\mathrm{VD}} = 5$ datasets, respectively. For U-net and MSD we took the same datasets for training and validation (10 for training and 5 for validation), and used all voxels in these datasets for the training process. The NN-FDK networks were trained till convergence and the DNNs were trained for 48 hours. Note that in a few cases we had to retrain the DNNs because of inconsistent results (*i.e.*, cases with more information achieving a lower reconstruction accuracy), possibly because they got stuck in local minima of the loss function.

In **Figure 4.7** we show the average and standard deviation of the TSE and the SSIM for the considered cases. We observe that U-net and MSD achieve the most accurate results and that NN-FDK and SIRT$^+$ closely follow. The FDK algorithm is lowest in all categories. Between NN-FDK and SIRT$^+$ we see that NN-FDK performs best for the noisy reconstruction problems and SIRT$^+$ achieves better results for the reconstruction problems without noise. We visualize the noise for the lowest and highest I_0 in **Figure 4.8** by showing a line profile through the center of the $z = 0$ slice. Here we see that for the noisiest problems the amplitude of the noise can be as high as the maximum value of the phantom. In **Figure 4.9** we show 2D slices of reconstructions of the test phantoms for the three types of reconstruction problems. In all cases we still observe reconstruction artifacts, but comparing these to the baseline FDK reconstructions, the majority is removed or suppressed.

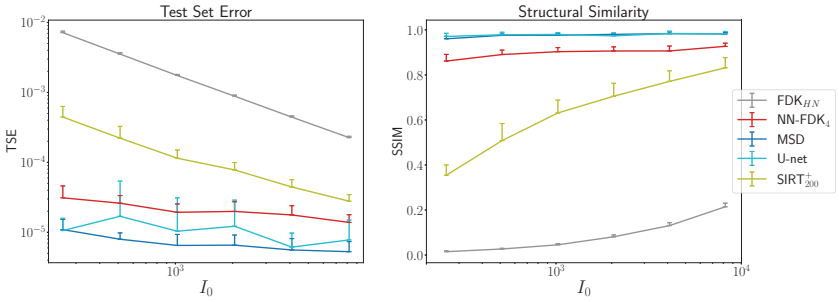
4.5.3 Reconstruction accuracy for experimental data

In this section we use the datasets discussed in **Section 4.4.2** to assess the reconstruction accuracy on experimental data. In a similar fashion as for the simulated data, we trained a network for the low-dose reconstruction problem and a network for the high-dose reconstruction problem with $N_a = 32$ projection angles with the notable exception that U-net and MSD were trained till convergence. The results are presented in **Table 4.4**.

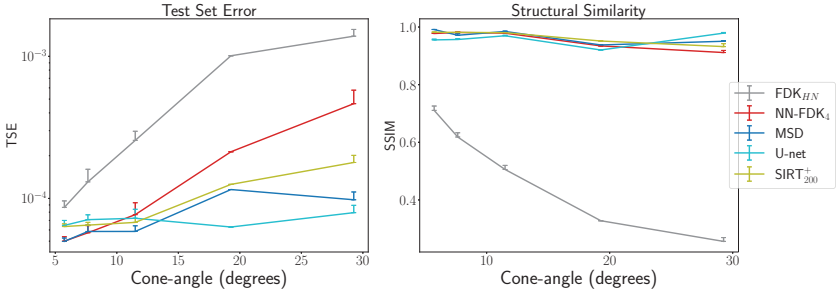
Comparing the results to the simulated data experiments we see that SIRT$^+$ performs worse on the experimental data, even with the additional regularization of early stopping. This is most likely due to the high-dose datasets still containing noise, whereas this is completely absent in the simulated data experiments. These differences are illustrated in **Figure 4.10** where 2D slices of the reconstructions for the high-dose reconstruction problem with $N_a = 32$ projection angles are shown.



(a) The average and standard deviation of the TSE and SSIM as a function of number of projection angles N_a computed over 20 randomly generated phantoms Fourshape family.



(b) The average and standard deviation of the TSE and SSIM as a function of the emitted photon count I_0 computed over 20 randomly generated phantoms of the Fourshape family.



(c) The average and standard deviation of the average TSE and SSIM as a function of the cone angle computed over 20 randomly generated phantoms of the Defrise family.

Figure 4.7: The average and standard deviation of the TSE and SSIM. These results are discussed in **Section 4.5.2**. For each number of projection angles, noise level, cone angle and training scenario one specific network is trained and used to evaluate the 20 reconstruction problems.

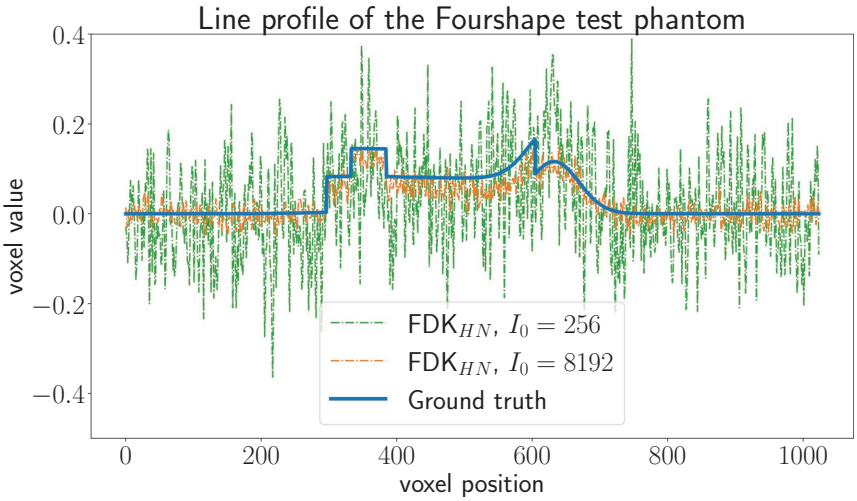


Figure 4.8: Line profile through the center of the $z = 0$ slice of the Fourshape test phantom. We show the ground truth profile, the profile of the FDK reconstruction with lowest emitted photon count $I_0 = 256$, and the profile of the FDK reconstruction with the highest emitted photon count $I_0 = 8196$.

	Experimental data			
	High-dose, low number of projection angles		Low-dose	
Method	TSE	SSIM	TSE	SSIM
FDK_{HN}	5.54±3.43e-03	0.224±0.076	1.40±0.05e-03	0.334±0.104
$SIRT^{+}_{200/20}$	9.94±0.15e-04	0.603±0.087	1.92±0.08e-03	0.584±0.083
NN-FDK_4	8.03±1.39e-04	0.946±0.010	1.14±0.23e-04	0.965±0.012
U-net	**4.10±1.06e-04**	**0.964±0.009**	1.02±0.45e-04	**0.980±0.006**
MSD	4.23±0.97e-04	**0.964±0.009**	**7.82±2.86e-05**	0.980±0.007

Table 4.4: Average and standard deviation of the quantitative measures computed over 6 walnut datasets. The high-dose low projection angle reconstruction problem has $N_a = 32$ projection angles, the low-dose reconstruction problem has $N_a = 500$ projection angles. The best results per experiment are highlighted.

4.5.4 Segmentation experiment for experimental data

To assess the performance of the different reconstruction approaches in a segmentation task, we focus here on the segmentation of the shell and kernel of walnuts, based on our experimental CT data. The review [Ber+20] provides an overview of segmentation problems in walnut imaging, and their relevance. For segmenting

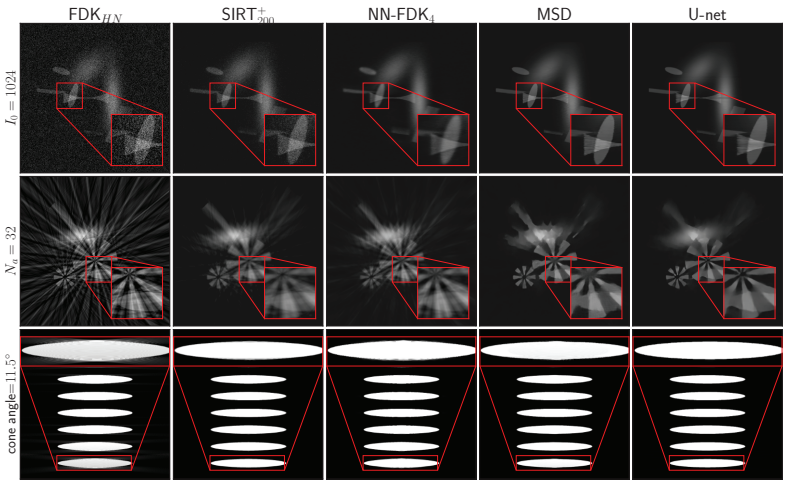


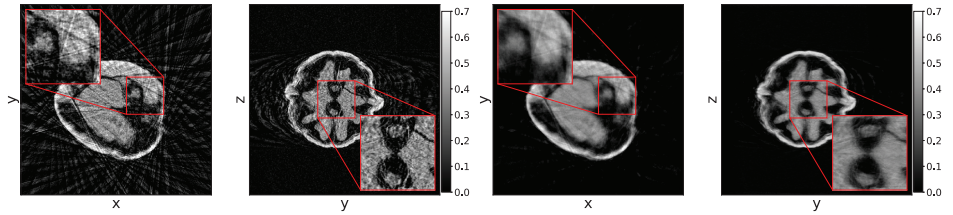
Figure 4.9: Two-dimensional slices of the reconstructions for the considered reconstruction methods. (Top) Slice $x = 0$ of the Fourshape test phantom reconstruction problem with $N_a = 360$ projection angles and $I_0 = 1024$ emitted photon count. (Middle) Slice $z = 0$ of the Fourshape test phantom reconstruction problem with $N_a = 32$ projection angles. (Bottom) Slice $x = 0$ of the Defrise reconstruction problem with $N_a = 360$ projection angles and a cone angle of 11.5 degrees.

the 3D volume after the reconstruction, we used a deterministic segmentation algorithm that combines thresholding, the watershed algorithm and prior knowledge of the scanned objects. Details of this method are discussed in **Appendix 4.7.1**.

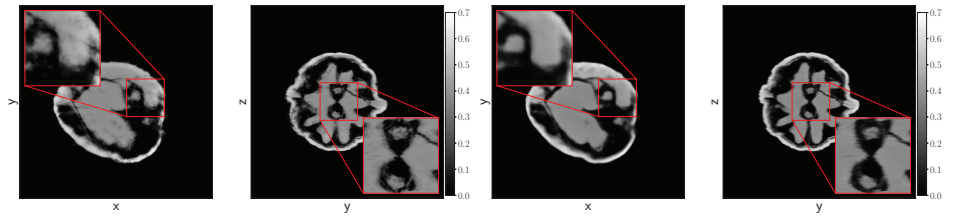
For determining the accuracy of the segmentation of an object — *i.e.*, shell, empty space and kernel of the walnut — we consider three metrics: volume error, mislabeled voxels and the Dice coefficient [Dic45]. We define a segmentation S as a reconstruction volume with value 1 if the voxel is in the object (shell, kernel or empty space) and 0 if outside the object. Furthermore we define the norm of a segmentation as the sum: $|S| = \sum_i^{N^3} (S)_i$. Using this notation we can compute the measures in the following manner:

$$V_{\mathrm{err}} = \frac{|S_{\mathrm{rec}}| - |S_{\mathrm{GS}}|}{|S_{\mathrm{GS}}|}, \qquad \mathrm{ML}_{\mathrm{err}} = \frac{|S_{\mathrm{rec}} - S_{\mathrm{GS}}|}{|S_{\mathrm{GS}}|}, \qquad \mathrm{DC} = \frac{2|S_{\mathrm{rec}} \cap S_{\mathrm{GS}}|}{|S_{\mathrm{rec}}| + |S_{\mathrm{GS}}|}, \tag{4.20}$$

with GS denoting the gold standard reconstruction.



(a) FDK_{HN} (b) SIRT^{+}_{200} reconstruction.



(c) NN-FDK_4 reconstruction. (d) MSD reconstruction.

Figure 4.10: Slices $z = 0$ and $x = 0$ of several reconstruction methods of the high-dose dataset of the 21^{st} walnut with 32 projection angles.

In **Table 4.5** we show the results for computing these metrics on the 6 walnuts not considered in the training process. We observe that MSD performs best in segmenting the shell and U-net performs best at segmenting the empty space and kernel and NN-FDK is close to both DNNs and in some cases even better than MSD for segmenting the empty space and kernel. Comparing NN-FDK to standard FDK we observe a significant improvement.

4.5.5 Data requirements

To test the influence of the amount of training data on the reconstruction quality we performed an experiment with three different training scenarios:

- **Scenario 1**. One dataset available. Here we take the training and validation data from the same dataset.
- **Scenario 2**. Two datasets available. Here we take the training and validation data from the separate datasets.
- **Scenario 3**. Fifteen datasets available. Again the training and validation data are picked from separate datasets, but now the training and validation pairs come from several datasets, specifically 10 training datasets ($N_{\text{TD}} = 10$) and 5 validation datasets ($N_{\text{VD}} = 5$). This is the scenario used in the previous experiments.

Segmentation errors			
Method	Shell	Empty space	Kernel
Volume errors			
FDK_{HN}	0.127 ± 0.078	0.146 ± 0.091	0.128 ± 0.092
$SIRT^{+}_{200}$	0.082 ± 0.047	0.104 ± 0.078	0.050 ± 0.074
$NN\text{-}FDK_4$	0.068 ± 0.035	0.045 ± 0.035	0.029 ± 0.032
U-net	0.055 ± 0.019	**0.029 ± 0.017**	**0.012 ± 0.016**
MSD	**0.028 ± 0.010**	0.059 ± 0.075	0.035 ± 0.050
Mislabeled voxels			
FDK_{HN}	0.168 ± 0.087	0.190 ± 0.98	0.144 ± 0.081
$SIRT^{+}_{200}$	0.133 ± 0.026	0.182 ± 0.118	0.101 ± 0.048
$NN\text{-}FDK_4$	0.103 ± 0.026	0.087 ± 0.023	0.072 ± 0.018
U-net	0.092 ± 0.028	**0.073 ± 0.024**	**0.059 ± 0.019**
MSD	**0.086 ± 0.038**	0.116 ± 0.094	0.061 ± 0.039
Dice coefficient			
FDK_{HN}	0.922 ± 0.036	0.895 ± 0.061	0.934 ± 0.033
$SIRT^{+}_{200}$	0.934 ± 0.016	0.908 ± 0.061	0.947 ± 0.028
$NN\text{-}FDK_4$	0.951 ± 0.012	0.955 ± 0.013	0.964 ± 0.008
U-net	0.955 ± 0.013	**0.963 ± 0.012**	**0.971 ± 0.010**
MSD	**0.957 ± 0.018**	0.939 ± 0.055	0.971 ± 0.018

Table 4.5: The average and standard deviation of the three metrics computed over the 6 low-dose walnut datasets with $N_a = 500$ projection angles. The metrics are computed using (4.20). The best results are highlighted.

We fix the number of voxels used for training and validation at $N_T = 10^6$ and $N_V = 10^6$ for all scenarios. For comparison we trained a U-net and a MSD network with the same training scenarios, with the exception that all voxels from the datasets are used. For training scenario 1 the slices are divided into a training and a validation set. More specifically, every fourth slice is used for validation.

We performed this experiment for two simulated data problems, a high noise level (emitted photon count $I_0 = 256$) and a large cone angle (29.3 degrees), and the two experimental data problems. For the sake of brevity we show only the results for the high noise simulated data reconstruction problem (**Table 4.6**) and the high noise experimental data reconstruction problem (**Table 4.7**). The results for the other reconstruction problems are given in **Appendix 4.7.3**. Comparing quantitative measures between the different scenarios we see that the reconstruction accuracy improves as more data is used for the simulated data experiment, but remains about the same for the experimental data experiment. This can be

explained by the variation in the objects used in the reconstruction problems. Recall that the Fourshape phantom family has a large variety in its phantoms, *i.e.*, three instances of four randomly generated objects, and the variety within the walnut datasets is small, *i.e.*, similar shapes, sizes and structures. This indicates that if objects are similar, one training dataset may already be sufficient to train networks that achieve a high reconstruction accuracy.

Note that although the training scenarios for NN-FDK and the DNNs use the same number of datasets, the number of voxels considered for training the NN-FDK network is constant over all three scenarios and is several orders of magnitude lower than the number of voxels considered for training the DNNs. This opens up future possibilities for reducing the training data requirements to only need a high quality reconstruction of a certain region of interest.

Simulated data, high noise

TSE			
Method	1 dataset	2 datasets	15 datasets
NN-FDK$_4$	4.97±4.68e-05	4.19±3.60e-05	2.51±1.14e-05
U-net	1.06±1.36e-05	2.45±2.87e-05	8.06±3.63e-06
MSD	1.12±0.41e-05	1.12±0.40e-05	**7.94±3.16e-06**
SSIM			
NN-FDK$_4$	0.831±0.065	0.844±0.065	0.884±0.030
U-net	0.884±0.075	0.932±0.050	**0.979±0.009**
MSD	0.961±0.013	0.962±0.013	0.974±0.008

Table 4.6: Average and standard deviation of the quantitative measures computed over 20 Fourshape phantoms for varying training scenarios. The reconstruction problems have an emitted photon count of $I_0 = 256$ and $N_a = 360$ projection angles. The best results are highlighted.

4.6 Summary and conclusion

We have proposed the Neural Network FDK (NN-FDK) algorithm, a reconstruction algorithm for the circular cone-beam (CCB) Computed Tomography (CT) geometry with a machine learning component. The machine learning component of the algorithm is designed to learn a set of FDK filters and to combine the FDK reconstructions done with these filters. This leads to a computationally efficient reconstruction algorithm, since one only needs to compute and combine the FDK reconstructions for this learned set of filters. Due to parametrization of

Experimental data, low-dose

TSE			
Method	1 dataset	2 datasets	15 datasets
NN-FDK$_4$	1.16±0.25e-04	1.23±0.25e-04	1.14±0.23e-04
U-net	1.27±0.38e-04	1.23±0.35e-04	1.02±0.45e-04
MSD	1.28±0.41e-04	1.16±0.35e-04	**7.82±2.86e-05**
SSIM			
NN-FDK$_4$	0.973±0.009	0.968±0.011	0.965±0.012
U-net	0.979±0.008	0.978±0.008	**0.980±0.006**
MSD	0.979±0.008	0.979±0.008	0.980±0.007

Table 4.7: Average and standard deviation of the quantitative measures computed over 6 walnuts for varying training scenarios. The datasets are low-dose and have $N_a = 500$ projection angles. The best results are highlighted.

the learned filters, the NN-FDK network has a low number of trainable parameters (<100) and can be trained efficiently with the Levenberg-Marquardt algorithm with approximate quadratic convergence rate.

We compared the NN-FDK algorithm to SIRT with a nonnegativity constraint (SIRT$^+$), the standard FDK algorithm and two deep neural networks (DNNs), namely a 2D U-net and a 2D MSD network applied in a slice-by-slice fashion to a 3D volume. We have shown that the NN-FDK algorithm has the lowest reconstruction time after the standard FDK algorithm. We have also shown that the NN-FDK algorithm achieves a reconstruction accuracy that is similar to that of SIRT$^+$ for simulated data and a higher accuracy than that of SIRT$^+$ for experimental data. The DNNs achieved the highest reconstruction accuracy, but training those networks took between 2 days (1 training and validation dataset) and 2 weeks (15 training and validation datasets), whereas all the NN-FDK networks were trained within 1 minute.

To conclude, the NN-FDK algorithm is a computationally efficient reconstruction algorithm that can reconstruct CCB CT reconstruction problems with high noise, low projection angles or large cone angles accurately. The training process is efficient and requires a low amount of training data, making it suitable for application to a broad spectrum of large scale (up to $4096 \times 4096 \times 4096$) reconstruction problems. Specifically, the NN-FDK algorithm can be used improve image quality in high throughput CT scanning settings, where FDK is currently used to keep pace with the acquisition speed using readily available computational resources.

4.7 Appendices

4.7.1 Implementation

Data generation

For our simulated data experiments we take $N = 1024$, which means that reconstructions and reference images are defined on a 1024^3 equidistant voxel grid, and the projection data on a 1024^2 equidistant detector grid per projection angle. However, to avoid using the same operator for reconstructions as for the data generation we generate the input data at a higher resolution. More specifically, we generate a phantom at $N = 1536$, forward project this phantom to the data space with size $N_a \times 1536^2$ and apply a bilinear interpolation per projection angle to arrive at a 1024^2 detector grid, resulting in input data with the desired resolution $N_a \times 1024^2$. We set the source radius to 10 times the physical size of the phantom, resulting in a cone angle of 5.7 degrees. To generate noise we compute a noise free photon count I from clean projection data $\mathbf{y}_c$ and use that to generate a Poisson distributed photon count from which we compute $\mathbf{y}$:

$$I = I_0 e^{-\mathbf{y}_c}, \qquad I_{\text{noise}} \sim \text{Pois}(I), \qquad \mathbf{y} = -\log\left(\frac{I_{\text{noise}}}{I_0}\right), \tag{4.21}$$

with I_0 the emitted photon count. Higher I_0 implies a higher dose and therefore less noise in the data.

Deep neural networks

Application strategy We train 2D DNNs to remove artifacts from 2D slices of an FDK reconstruction. We train one network that handles all slices in the reconstructions.

Training DNNs We train the DNNs with ADAM [KB14] and stop training after 48 hours of training on a Nvidia GeForce GTX 1080Ti GPU, the network with the lowest validation set error during this training process will be used for the reconstructions.

U-net and MSD network structures For U-net we will take four up and down layers as presented in [RFB15]. For the MSD networks we take 100 layers with one input and one output layer and the dilations as suggested in [PS18].

Code-base

We implemented the NN-FDK framework using Python 3.6.5 and Numpy 1.14.5 [WCV11]. For the parameter learning we used the Levenberg-Marquardt algorithm implementation from [PB13]. The reconstruction algorithm is implemented using ODL [AKÖ17], the ASTRA-toolbox [Van+16], PyFFTW [FJ05] and the exponential binning framework for filters from [Lag+20]. For performance reasons the simulated phantoms are generated through C++ using Cython [Beh+11].

For the evaluation of U-nets we took the PyTorch [Pas+19] implementation used in [Hen+19]. The MSD-nets are implemented using the package published with [PBS18].

All the code related to this chapter can be found on Github [Lagb].

Segmentation algorithm

This algorithm consists of several steps:

1. Apply a Gaussian filter to the reconstruction.
2. Compute a histogram of the filtered reconstruction and determine the peaks relating to the background, kernel and shell.
3. Determine the shell and kernel segmentations using a threshold based on the found peaks.
4. Apply the watershed algorithm on the shell segmentation. This gives the total volume inside the walnut.
5. Remove the kernel from the total volume inside the walnut to attain the empty space segmentation.

Further details about this implementation can be found on our Github [Lagb].

4.7.2 Levenberg-Marquardt algorithm

Given the learning problem (4.12), the update rule for the Levenberg-Marquardt algorithm (LMA) ([Lev44; Mar63]) is given by:

$$\theta^{i+1} = \theta^i + \mathbf{t}^i, \tag{4.22}$$

with $\mathbf{t}^i$ the update vector. This is computed by solving the following equation for $\mathbf{t}^i$

$$\left(J_i^T J_i + \lambda_i I\right)\mathbf{t}^i = -\frac{\partial \mathscr{L}}{\partial \theta}(\theta^i, T) = -J_i^T \sum_{j=1}^{N_\mathrm{T}} \left(O_j - \mathsf{N}_\theta(Z_j)\right) \tag{4.23}$$

where $\lambda_i > 0$ is the step parameter and J_i the $m \times n$ Jacobian matrix of $\mathsf{N}_{\theta^i}(\mathbf{Z})$ with respect to θ^i, with $\mathbf{Z}$ the vector containing all inputs from the training set T. We can solve (4.23) using a Cholesky decomposition.[3]

To ensure convergence, only updates that improve the training error are accepted, *i.e.*, if the following is true:

$$\mathscr{L}(\theta^i, T) > \mathscr{L}(\theta^i + \mathbf{t}^i, T), \tag{4.24}$$

If this is not the case we change the step parameter λ_i to $a\lambda_i$ with $a > 1$ and compute a new update vector $\mathbf{t}^i$. When an update is accepted we change the step parameter to $\lambda_{i+1} = \lambda_i / a$.

We use two stopping criteria for the LMA. Firstly, we stop if we cannot find a suitable θ^{i+1}, using several indicators for this:

- The norm of the gradient $\frac{\partial \mathscr{L}}{\partial \theta}(\theta^i)$ is too small
- The step size λ_i is too big
- After N_{up} rejected updates.

The second stopping criterion checks whether the parameters θ^i improve the validation set error. More specifically, we terminate the LMA when the validation set error has not improved for N_{val} iterations.

In **Algorithm 4** the LMA is summarized. The random initialisation is done with the Nguyen-Widrow initialization method [NW90]. For our experiments we take $N_{\text{up}} = 100$, $\lambda_0 = 10^5$, $a = 10$ and $N_{\text{val}} = 100$.

Algorithm 4 Levenberg-Marquardt algorithm

1: Compute random initialization θ^0 using [NW90]
2: **repeat**
3: Compute $\mathbf{t}^i$ until we accept an update θ^{i+1}.
4: **until** N_{up} updates were rejected **or**
$\mathscr{L}(\theta^i, V)$ did not improve N_{val} times **or**
$\left\| \frac{\partial \mathscr{L}}{\partial \theta}(\theta^{i+1}) \right\|$ is too small **or** λ_{i+1} is too big.
5: Set $\theta^\star$ equal to the θ^i with the lowest validation error.

[3] $J_i^T J_i$ is positive semi-definite and $\lambda_i > 0$, therefore the left hand side of (4.23) is positive definite.

4.7.3 Results data requirement experiment

Simulated data, large cone angle

TSE			
Method	1 dataset	2 datasets	15 datasets
NN-FDK$_4$	6.47±1.19e-04	4.70±1.16e-04	4.82±1.13e-04
U-net	1.04±0.27e-04	1.02±0.17e-04	8.23±0.85e-05
MSD	2.44±1.43e-04	1.53±0.17e-04	**6.52±0.43e-05**
SSIM			
NN-FDK$_4$	0.825±0.018	0.904±0.011	0.910±0.007
U-net	**0.974±0.015**	0.971±0.021	0.973±0.010
MSD	0.954±0.006	0.937±0.004	0.966±0.002

Table 4.8: Average and standard deviation of the quantitative measures computed over 20 different Defrise phantoms for varying training scenarios. The reconstruction problems have a cone angle of 29.2 degrees and $N_a = 360$ projection angles. The best results are highlighted.

Experimental data, high-dose, 32 projection angles

TSE			
Method	1 dataset	2 datasets	15 datasets
NN-FDK$_4$	8.14±1.45e-04	8.68±1.43e-04	8.03±1.39e-04
U-net	7.56±1.52e-04	6.85±1.56e-04	**4.10±1.06e-04**
MSD	7.82±0.41e-04	6.51±0.35e-04	4.23±0.97e-04
SSIM			
NN-FDK$_4$	0.950±0.010	0.948±0.010	0.946±0.011
U-net	0.955±0.011	0.930±0.023	**0.964±0.009**
MSD	0.955±0.010	0.947±0.014	**0.964±0.009**

Table 4.9: Average and standard deviation of the quantitative measures computed over the 6 datasets for varying training scenarios. These are the high-dose datasets from [LCB20] with $N_a = 32$ projection angles. The best results are highlighted.

Chapter 5

Noise2Filter: self-supervised learning and real-time reconstruction algorithm

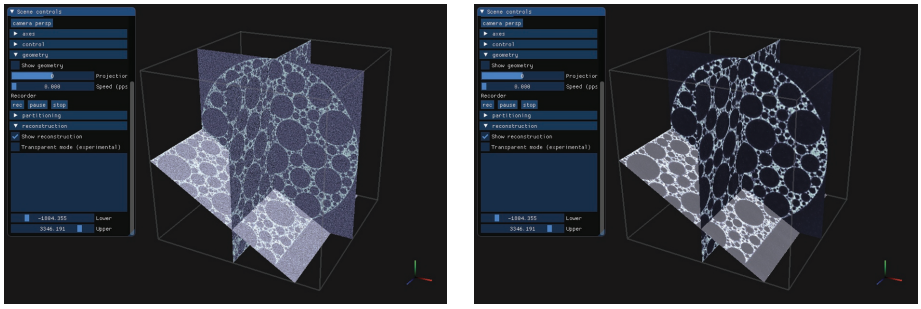
5.1 Introduction

Computed tomography is a non-destructive imaging technique with applications in biology [San+14], energy research [Xu+20], materials science [Gar+18], and many other fields [De +18]. In a tomographic scan, a rotating object is positioned between a source emitting penetrating radiation and a detector that captures the projections of the object. Tomographic reconstruction algorithms compute a 3D image of the interior of the object from its projections. Besides extensive use in medical and laboratory settings, tomography is routinely used at synchrotron facilities, where advances in the last decade have enabled time-resolved imaging of the interior structure of a rapidly changing object [San+14; Xu+20; Gar+18]. So far, reconstruction algorithms are typically operated offline, enabling visualization of the object only after a scan has completed.

Recent advances in tomographic reconstruction enable real-time interrogation of the reconstructed volume during the scanning process using a quasi-3D recon-

This chapter is based on:

Noise2Filter: fast, self-supervised learning and real-time reconstruction for 3D Computed Tomography. *MJ Lagerwerf, AA Hendriksen, JW Buurlage, KJ Batenburg.* Machine Learning: Science and Technology, Early access, 2020.



(a) FBP reconstruction (b) Noise2Filter reconstruction

Figure 5.1: Real-time reconstructions using FBP and Noise2Filter of a high-noise acquisition using the RECAST3D software package. The highlighted slice is currently being moved.

struction protocol [Buu+18; Buu+19]. In this framework, arbitrarily oriented slices are selected for reconstruction and can be interactively rotated and translated, after which they are reconstructed and visualized virtually instantaneously. This creates the illusion of having access to the full reconstructed 3D volume, but at a fraction of the computational cost. The quasi-3D reconstruction protocol has been implemented in the RECAST3D software package. The information gained from this quasi-3D visualization can be used to directly steer the tomographic experiment, for instance, by adjusting an external parameter — such as temperature — in response to changes in the interior of the object. In addition, the object can be re-positioned, or other acquisition parameters can be adjusted to facilitate the best possible reconstruction [Van+20].

Real-time 3D reconstruction is computationally demanding and data sizes are substantial — data acquisition rates of 7.7GB per second are not uncommon [Buu+19]. To attain real-time visualization, the quasi-3D reconstruction protocol is essentially limited to filtered backprojection type methods, since it exploits the locality of backprojection to obtain fast reconstructions. Filtered backprojection (FBP) methods are sensitive to measurement noise, leading to errors in the reconstructed slices [Buz08]. Therefore, application of these methods in the quasi-3D reconstruction protocol is not well-suited to high-noise acquisitions [PBS18; Xu+20], as illustrated in **Figure 5.1.a**.

In this chapter, we combine a learning-based filtered reconstruction method with a self-supervised training strategy to obtain Noise2Filter, a denoising FBP-type reconstruction algorithm that can be applied in a quasi-3D reconstruction protocol. This algorithm is designed to be both fast to train and fast to evaluate. Moreover, no additional training data is required other than the measured projection data.

For dynamic scans, our method enables a possible use case where a static scan is performed — with the exact same acquisition rates as the dynamic scan — permitting the Noise2Filter method to be trained immediately. After training for tens of seconds, real-time visualization of the dynamic experiment can ensue, as illustrated in **Figure 5.1.b**. In addition, we note that Noise2Filter can be used as a stand-alone reconstruction method.

The first component of our method is the Neural Network filtered backprojection (NN-FBP) method [PB13]. This method learns a set of filters, along with additional weights, and then forms the reconstructed image as a non-linear function of the individual FBP reconstructions, resulting in higher image quality than standard FBP. However, its application requires the availability of ground truth or high-quality reconstructed images.

This limitation can be overcome using the second component of our method, Noise2Inverse [HPB20], which is a recent machine learning method designed to train denoising convolutional neural networks (CNNs) in inverse problems in imaging. To train a denoising CNN, the method splits the measured projection data to obtain multiple statistically independent reconstructed slices, which are presented to the network during training, without requiring additional high-quality data.

Our main contribution is that we show how to combine the NN-FBP method with the Noise2Inverse training strategy. In addition, we demonstrate that NN-FBP training can be substantially accelerated as compared to previous methods [PB13]. We evaluate our method on both simulated and experimental datasets, comparing to both conventional filter-based methods and supervised NN-FBP. Finally, we demonstrate that the method can be used in a quasi-3D reconstruction protocol, and exhibit its potential use for dynamic control of tomographic experiments.

The chapter is structured as follows. In the next section, we introduce the tomographic reconstruction problem and the filtered backprojection algorithm. In addition, we introduce quasi-3D reconstruction, NN-FBP, and Noise2Inverse. These methods are combined in Section 5.3, where we describe the Noise2Filter method. In Sections 5.4 and 5.5, we describe experiments to analyze the reconstruction accuracy of Noise2Filter on real and simulated CT datasets. Moreover, we study the hyper-parameters of the proposed method. We discuss these results in Section 5.6.

5.2 Preliminaries

5.2.1 Reconstruction problem

In parallel-beam tomography, an unknown object rotates with respect to a planar detector and a parallel source beam. Projections are acquired at a finite number N_a of rotation angles, yielding 2D images defined on an $N \times N$ pixel grid. The reconstruction problem can be modeled by a system of linear equations

$$W\mathbf{x} = \mathbf{y}, \tag{5.1}$$

where the vector $\mathbf{x} \in \mathbb{R}^n$ denotes the unknown object, $\mathbf{y} \in \mathbb{R}^m$ describes the measured projection data, and $W = (w_{ij})$ is an $m \times n$ matrix where w_{ij} denotes the contribution of object voxel j to detector pixel i. For the sake of simplicity we assume that the volume consists of $n = N \times N \times N$ voxels, and the projection dataset contains $m = N_\mathrm{a} \times N \times N$ pixels.

5.2.2 Filtered backprojection methods

We consider the filtered backprojection (FBP) method for parallel beam tomography [Nat01]. The FBP algorithm is a two step algorithm. First, the data $\mathbf{y} \in \mathbb{R}^m$ is convolved over the width of the detector with a one-dimensional filter $\mathbf{h} \in \mathbb{R}^{N_\mathrm{f}}$. Next, the *backprojection* $W^T : \mathbb{R}^m \to \mathbb{R}^n$ is applied to compute a reconstruction $\mathbf{x}_{\mathrm{FBP}} \in \mathbb{R}^n$. Expressing the FBP algorithm in terms of $\mathbf{h}$, $\mathbf{y}$ and W yields

$$\mathrm{FBP}(\mathbf{y}, \mathbf{h}) = W^T(\mathbf{y} * \mathbf{h}) = \mathbf{x}_{\mathrm{FBP}}. \tag{5.2}$$

Observation 1 (FBP is two-step). *The FBP algorithm consists of a* filtering step *and a* backprojection step*, and both can be computed separately. That is, the filtering can be performed in advance, and the backprojection can occur on demand. This technique will be used throughout the chapter.*

We observe that the FBP algorithm can be described by a linear operator when fixing either $\mathbf{y}$ or $\mathbf{h}$. This will be exploited in the discussion of learned filter methods in Section 5.2.4.

5.2.3 Quasi-3D reconstruction

A property shared by filtered-backprojection type algorithms is that they are *local*, in the sense that each voxel of the reconstructed volume can be computed directly from the filtered data by backprojecting onto only that voxel [Buu+18]. Therefore, if one is interested in a subset of the reconstructed volume, much of

the computational cost of a full 3D reconstruction can be avoided. Specifically, if the reconstructed subset is a rectangular box or a slice, efficient backprojection algorithms such as those implemented in the ASTRA toolbox [Van+16] can be used. This reduces the computational cost of the backprojection step by an order of N.

Observation 2 (Locality). *The backprojection operator is* local. *Computing the backprojection for a single voxel or a subset of voxels is therefore substantially faster than computing the backprojection for all voxels.*

This methodology has been implemented in the RECAST3D software package [Buu+18], which exposes a limited number of arbitrarily oriented 2D slices. These slices are interactive and can be manipulated by the technician of the tomographic experiment. This technique for real-time visualization has been successfully applied in practice to acquisitions in micro-CT systems [Cob+20], synchrotron tomography [Buu+19], and electron tomography [Van+20].

5.2.4 NN-FBP reconstruction algorithm

The NN-FBP algorithm learns a set of suitable filters and a set of weights, and then forms a non-linear model that combines the individual FBP reconstructions. The algorithm may be considered as a multi-layer perceptron [HTF09] that operates pointwise on a collection of suitable reconstructions. A schematic representation of the NN-FBP algorithm is given in **Figure 5.2**, a mathematical description is given below.

To obtain these reconstructions, we first make some general observations: a filter $\mathbf{h}$ can be seen as a vector in $\mathbb{R}^{N_\mathrm{f}}$, and the FBP method is linear in the filter when fixing the measured projection data $\mathbf{y}$. Therefore, an FBP reconstruction can be expressed as a linear combination in the basis of the filter. Let $\mathbf{e}_1, \ldots, \mathbf{e}_{N_\mathrm{f}}$ be any basis for the space of filters $\mathbb{R}^{N_\mathrm{f}}$, such as the standard basis. Define the reconstruction of $\mathbf{y}$ filtered by a basis element $\mathbf{e}_i$ as

$$\mathbf{x}_{\mathbf{e}_i} := W^T (\mathbf{y} * \mathbf{e}_i). \tag{5.3}$$

Then we can write the FBP reconstruction as a linear combination of these reconstructions

$$\mathbf{x}_{\mathrm{FBP}}(\mathbf{y}, \mathbf{h}) = \sum_{i=1}^{N_\mathrm{f}} h_i \mathbf{x}_{\mathbf{e}_i} = \sum_{i=1}^{N_\mathrm{f}} W^T (\mathbf{y} * h_i \mathbf{e}_i) = W^T (\mathbf{y} * \mathbf{h}), \tag{5.4}$$

where h_i denotes the coordinate of the ith basis element $\mathbf{e}_i$.

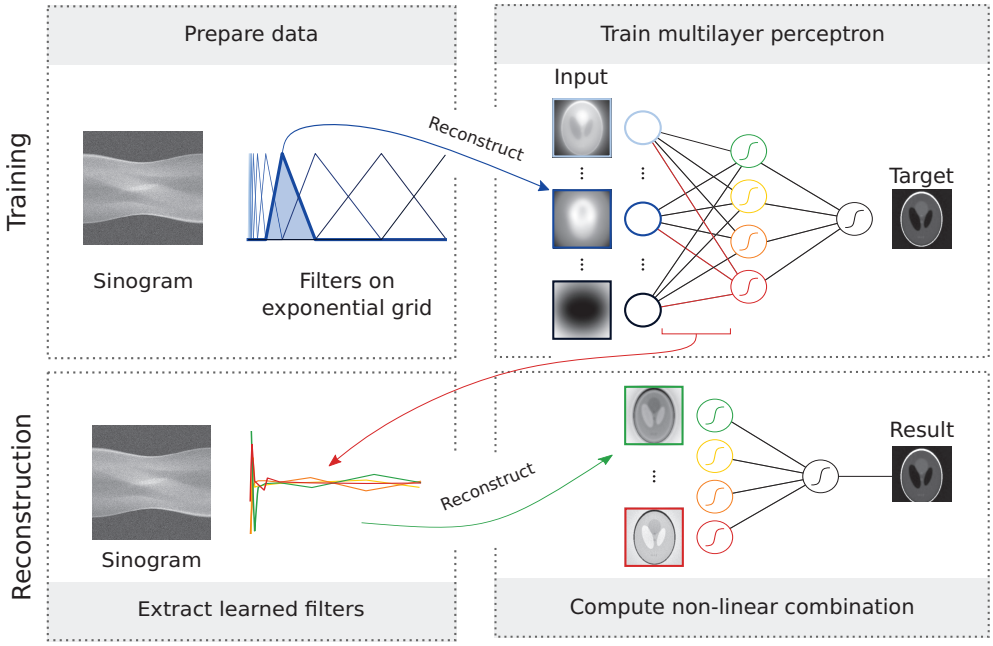


Figure 5.2: An illustration of the NN-FBP method as applied to a noisy 2D Shepp-Logan sinogram. Before training, the data is reconstructed with filters $\mathbf{e}_1, \ldots, \mathbf{e}_{N_e}$, defined on an exponential grid. These reconstructions $\mathbf{x}_{\mathbf{e}_1}, \ldots, \mathbf{x}_{\mathbf{e}_{N_e}}$ are used as input for training a multilayer perceptron, as described in Equation (5.5). The training target is a high-quality reconstruction. For reconstruction, learned filters $\mathbf{h}^1, \ldots, \mathbf{h}^{N_\mathrm{h}}$ are extracted from the network (as indicated by the red arrow). Reconstructions are computed using the learned filters, and a non-linear combination is computed, as described in Equation (5.6).

Given a set of N_h filters $\mathbf{h}^1, \ldots, \mathbf{h}^{N_\mathrm{h}}$, we can define a multi-layer perceptron (MLP) with one hidden layer as a function of the reconstructions $\mathbf{x}_{\mathbf{e}_1}, \ldots, \mathbf{x}_{\mathbf{e}_{N_\mathrm{f}}}$

$$\mathrm{MLP}_\theta(\mathbf{x}_{\mathbf{e}_1}, \ldots, \mathbf{x}_{\mathbf{e}_{N_\mathrm{f}}}) = \sigma\left(\sum_{k=1}^{N_\mathrm{h}} a_k \sigma\left(\sum_{i=1}^{N_\mathrm{f}} h_i^k \mathbf{x}_{\mathbf{e}_i} - \mathbf{b}_k\right) - \mathbf{b}_0\right), \tag{5.5}$$

where σ is a non-linear activation function, such as the sigmoid. The multi-layer perceptron has free parameters $\theta = (\mathbf{a}, \mathbf{b}, \mathbf{h}^1, \ldots, \mathbf{h}^{N_\mathrm{h}})$. Plugging Equation (5.4) into Equation (5.5), we obtain the NN-FBP reconstruction algorithm

$$\text{NN-FBP}_\theta(\mathbf{y}) = \sigma\left(\sum_{k=1}^{N_\mathrm{h}} a_k \sigma\left(\mathrm{FBP}(\mathbf{y}, \mathbf{h}^k) - \mathbf{b}_k\right) - b_0\right), \tag{5.6}$$

which is amenable to fast, parallel computation because it is a non-linear combination of FBP reconstructions.

Observation 3 (pointwise). *Note that the multi-layer perceptron operates* pointwise *on the voxels of the reconstructed volumes. Therefore, a* single *voxel can be computed without having to reconstruct other voxels. This observation connects to the observation of locality on Page 111, and will return several times in this chapter.*

Supervised training [HTF09] is used to determine the free parameters of the MLP defined in Equation (5.5). The goal is to approximate a suitable *target* reconstruction $\mathbf{x}_{\text{Target}}$ by minimizing

$$\left\| \text{MLP}_\theta(\mathbf{x}_{\mathbf{e}_1}, \ldots, \mathbf{x}_{\mathbf{e}_{N_\text{f}}}) - \mathbf{x}_{\text{Target}} \right\|_2^2, \tag{5.7}$$

i.e., the mean square error with respect to the target reconstruction.

The size of the training problem in Equation (5.7) is related to the number of reconstructed volumes $\mathbf{x}_{\mathbf{e}_1}, \ldots, \mathbf{x}_{\mathbf{e}_{N_\text{f}}}$ and the size of these reconstructions, which suggests two techniques that may be used to accelerate training. First, to reduce the number of reconstructions, the filter is expressed on an exponentially binned grid, which grows logarithmically in the width of the filter. Since the filter width is proportional to the number of pixels in each detector row, we have $N_e = O(\log N_\text{f}) = O(\log N)$. This technique yields suitable filter approximations, as observed in [PB13; Lag+20]. Second, training may be accelerated by sampling *a subset of voxels* on which to minimize Equation (5.7), rather than the full volume. Subsampling is possible because NN-FBP operates pointwise, as noted in Observation 3.

To summarize, we can split the NN-FBP algorithm into three parts, namely: (1) *data preparation*, where the input training data $\mathbf{x}_{\mathbf{e}_1}, \ldots, \mathbf{x}_{\mathbf{e}_{N_e}}$ is computed, (2) *network training*, where the weights $\theta^\star$ for the network are determined using a supervised learning approach, and (3) the *reconstruction algorithm*, which is summarized in **Algorithm 5**.

We use the same network architecture as proposed in [PB13]. The hyperparameters used in this chapter are discussed in **Section 5.4.2**.

5.2.5 Noise2Inverse training

Noise2inverse is a technique to train a convolutional neural network (CNN) to denoise reconstructed images in a self-supervised manner [HPB20]. This means that no additional training data is required beyond the acquired noisy measurements. The key idea is change the training strategy by splitting the projection dataset

Algorithm 5 NN-FBP reconstruction algorithm

1: Given projections $\mathbf{y}$ and a set of parameters $\theta^\star := \left(\mathbf{a}, \mathbf{b}, \mathbf{h}^1, \ldots, \mathbf{h}^{N_\text{h}}\right)$.
2: Compute the FBP reconstruction using the learned filters:
3: **for** $k = \{1, 2, .., N_h\}$ **do**
4: $\quad \mathbf{x}_{\mathbf{h}^k} = \text{FBP}(\mathbf{y}, \mathbf{h}^k)$
5: Compute a non-linear combination of these reconstructions:
$\text{NN-FBP}_{\theta^\star}(\mathbf{y}) = \sigma\left(\sum_{k=1}^{N_\text{h}} a_k \sigma\left(\mathbf{x}_{\mathbf{h}^k} - \mathbf{b}_k\right) - b_0\right)$

into subsets, computing sub-reconstructions with these subsets and train a neural network mapping one sub-reconstruction to another.

First, the projection data is split into N_s sub-datasets such that projection images from successive angles are placed in different sub-datasets $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{N_\text{s}}$. The network is trained to predict the reconstruction from one subset using the reconstruction of the other subsets. Training therefore aims to find the parameter $\theta^\star$ that minimizes

$$\sum_{j=1}^{N_\text{s}} \left\| \text{CNN}_\theta(\text{FBP}(\mathbf{y}_j)) - \text{FBP}(\mathbf{y}_{l\neq j}) \right\|_2^2, \tag{5.8}$$

where $\text{FBP}(\mathbf{y}_j)$ denotes the reconstruction from one subset of the data, and $\text{FBP}(\mathbf{y}_{l\neq j})$ denotes the FBP reconstruction of the remaining subsets. We observe that the FBP reconstruction of a projection dataset is the mean of the FBP reconstruction of each projection image individually, which enables us to obtain

$$\text{FBP}(\mathbf{y}_{l\neq j}) = \frac{1}{N_\text{s} - 1} \sum_{l\neq j} \text{FBP}(\mathbf{y}_l). \tag{5.9}$$

Now the *original training data* can be denoised by applying the trained network to each subreconstruction individually and averaging to obtain

$$\mathbf{x}_\text{N2I} = \frac{1}{N_\text{s}} \sum_{i=1}^{N_\text{s}} \text{CNN}_{\theta^\star}(\text{FBP}(\mathbf{y}_i)). \tag{5.10}$$

In the previous discussion, we have assumed that the target images are reconstructed from more subsets than the input images. As in [HPB20], we call this the *1:X* strategy. A reverse *X:1* training strategy is also possible. Here, the target is a single subreconstruction and the input is reconstructed from the remaining sub-datasets.

Note that convolutional neural networks take into account the surrounding structure of a voxel, typically a 2D slice, and thus do not operate pointwise. Therefore, these networks are an example where Observation 3 does not apply.

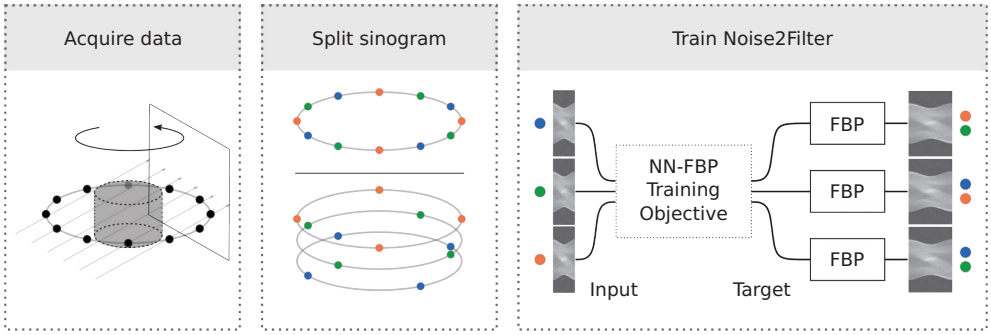


Figure 5.3: An illustration of the training of the Noise2Filter method. Data is acquired using a 3D parallel beam geometry. For each detector row, the sinogram is split into three sub-datasets such that acquisitions from successive projection angles belong to different sub-datasets. Each sub-dataset is used as input for NN-FBP training; the remaining sub-datasets are used in the target FBP reconstruction. This illustration depicts the *1:X* strategy. In the *X:1* strategy, the input is computed from the majority of the data, and the target from the minority, rather than vice versa.

5.3 Noise2Filter method

Our proposed method combines the three ideas introduced in the previous section. The NN-FBP method is trained on a single projection dataset using the Noise2Inverse training strategy. This enables fast reconstruction of arbitrarily oriented slices using the NN-FBP reconstruction algorithm in a quasi-3D reconstruction protocol.

Training The training procedure for the Noise2Filter method is similar to the NN-FBP procedure described in [PB13], with two notable exceptions. First, instead of minimizing the supervised training objective in Equation (5.7), Noise2Filter minimizes a self-supervised training objective similar to Equation (5.8). Second, training voxels are sampled from a subset of the reconstructed volume, rather than the full volume.

As in Noise2Inverse, the projection data $\mathbf{y}$ is split into N_s subdatasets with FBP reconstructions $\mathbf{x}_{\mathrm{FBP},j}, j = 1,\ldots,N_\mathrm{s}$. For each subdataset $\mathbf{y}_j$, we denote with $\mathbf{x}_{j,\mathbf{e}_i}$ a reconstruction filtered with basis element $\mathbf{e}_i$.

Training aims to minimize the difference between the MLP output of a subset of projection data and the FBP reconstruction of the remaining data. For the *1:X* training strategy, the MLP operates on a single subset of the data and the target is reconstructed from the remaining subsets. For the *X:1* training strategy, on the

other hand, the target is reconstructed from a single subdataset, and the MLP operates on the remaining subsets. The self-supervised training objective thus becomes:

$$\sum_{j=1}^{N_\mathrm{s}} \left\| \mathrm{MLP}_\theta(\mathbf{x}_{j,\mathbf{e}_1}, \ldots, \mathbf{x}_{j,\mathbf{e}_{N_e}}) - \mathbf{x}_{\mathrm{FBP},l\neq j} \right\|_2^2, \quad \textbf{(X:1 strategy)} \tag{5.11}$$

$$\sum_{j=1}^{N_\mathrm{s}} \left\| \mathrm{MLP}_\theta(\mathbf{x}_{l\neq j,\mathbf{e}_1}, \ldots, \mathbf{x}_{l\neq j,\mathbf{e}_{N_e}}) - \mathbf{x}_{\mathrm{FBP},j} \right\|_2^2, \quad \textbf{(1:X strategy)} \tag{5.12}$$

with

$$\mathbf{x}_{\mathrm{FBP},l\neq j} = \frac{1}{N_\mathrm{s}-1} \sum_{l\neq j} \mathbf{x}_{\mathrm{FBP},l}, \quad \mathbf{x}_{l\neq j,\mathbf{e}_i} = \frac{1}{N_\mathrm{s}-1} \sum_{l\neq j} \mathbf{x}_{l,\mathbf{e}_i}. \tag{5.13}$$

A schematic summary of the *1:X* training strategy is given in **Figure 5.3**.

The second difference is related to the voxels that are considered for the training. Like NN-FBP, we minimize the training objective on a random sample of N_T voxels. We have $N_\mathrm{T} \ll N^3$, and increasing the sample size in response to increasing object size has been observed to yield diminishing returns. Unlike NN-FBP, training voxels are sampled only from the reconstructions of the axial, frontal, and longitudinal *ortho-slices*, rather than the full volume. This choice substantially reduces the computational effort of the data preparation step, as shown below.

Data preparation We discuss the *1:X* strategy; similar statements hold true for the *X:1* strategy.

The data preparation step is the most computationally expensive part of the method. In this step, an input reconstruction $\mathbf{x}_{l\neq j,\mathbf{e}_i}$ is computed for each subdataset $\mathbf{y}_j$ and each basis element $\mathbf{e}_i$. In addition, a target reconstruction $\mathbf{x}_{\mathrm{FBP},j}$ is computed for each subdataset, resulting in a total of $N_\mathrm{s}(N_e+1)$ reconstructions. These reconstructions are computed on the ortho-slices instead of the full volume. Due to locality — see Observation 2 — the computational cost of the data preparation is therefore reduced by an order of N.

Note that the computational cost of the FBP algorithm scales linearly in the number of projection angles, therefore the computational cost of this step is equal to $3(N_e+1)$ FBP reconstructions of a 2D slice. Splitting the projection data thus has no adverse effect on the performance.

Reconstruction The reconstruction algorithm is almost identical to the NN-FBP reconstruction algorithm described in **Algorithm 5**. Whereas the aim of NN-FBP

is to reconstruct the full volume, we aim only to reconstruct slices on demand. Therefore, reconstruction can be substantially accelerated.

We make use of Observation 1 that the FBP algorithm can be split in a filtering and backprojection step. First, the acquired projection data is filtered with the learned filters and cached. Then, a single slice can be reconstructed using **Algorithm 5**, which can occur in real-time due to the locality of the backprojection (Observation 2) and the pointwise nature of the multi-layer perceptron (Observation 3). Therefore, the reconstruction can be integrated in the quasi-3D reconstruction protocol, computing reconstructions of arbitrarily oriented slices in real time.

We note that the reconstruction step deviates slightly from the Noise2Inverse reconstruction described in Equation (5.10). Rather than averaging separate reconstructions of each subset of the projection data, Noise2Filter computes a reconstruction using the learned filters directly from all data. In the context of self-supervised learning, this technique has been observed to yield improved results [BR19].

Noise2Filter summary

The Noise2Filter method consists of three steps. A summary of these steps, and specifically the computations performed, is given below:

1. **Data preparation** Compute the input and target training pairs from the measured projection data $\mathbf{y}$. Specifically, split the measured projection data into N_s equal sub-datasets and compute the following for the ortho-slices:

$$\mathrm{FBP}(\mathbf{y}_i, \mathbf{h}) \text{ for } i = 1, \dots, N_\mathrm{s} \tag{5.14}$$

$$\mathrm{FBP}(\mathbf{y}_i, \mathbf{e}_j) \text{ for } i = 1, \dots, N_\mathrm{s}, j = 1, \dots, N_e. \tag{5.15}$$

   The computational effort of this step is equal to $3(N_e+1)$ FBP reconstructions of a 2D slice.

2. **Training** Obtain a random sample of N_T voxels on the ortho-slices for inclusion in the training set. Compute the optimal parameters $\theta^\star$ that minimizes the training objective with respect to the sampled voxels. Note that the training time depends on the size of the training set, which may be fixed independent of the object size.

3. **Reconstruction** Using the computed parameters $\theta^\star$, compute an NN-FBP reconstruction for the desired 2D slices. Recall from Equation (5.6) that the computational cost of an NN-FBP reconstruction is equivalent to N_h FBP reconstructions.

The network architecture used for the Noise2Filter method is the same as the architecture used for the NN-FBP method and the considered hyperparameters are discussed in **Section 5.4.2**.

5.4 Experimental setup

In this section we discuss the setup of the experiments. Specifically, we describe the data used in the experiments, the implementation of NN-FBP and Noise2Filter, and the measures used to quantify these comparisons.

5.4.1 Simulated data

A phantom was generated by removing 100,000 randomly-placed non-overlapping balls from a foam cylinder. The `foam_ct_phantom` package [PBS18] was used to generate analytic projection images with 2× supersampling, were each pixel's value is averaged over four equally-spaced rays through the pixel. The result contains 1024 equally-spaced projection images with 512×768 pixels.

In each experiment, the simulated projection images were corrupted with Poisson noise of various levels of intensity, by altering the incident photon count I_0 per pixel. Specifically, we compute the mean measured photon I_{mean} count for an incident photon count I_0 from the analytic projection images $\mathbf{y}_{\text{analytic}}$:

$$I_{\text{mean}} = I_0 e^{-y_{\text{analytic}}}. \tag{5.16}$$

Given the mean measured photon count, we draw from a Poisson distribution the measured photon count I with respect to I_0 and compute the corresponding noisy projection data $\mathbf{y}$:

$$I \sim \text{Pois}(I_{\text{mean}}) \qquad\qquad \mathbf{y} = -\log\left(\frac{I}{I_0}\right). \tag{5.17}$$

The average absorption of the sample was 10%. Reconstructions without Poisson noise and with Poisson noise ($I_0 = 1000$) are shown in Figure 5.5.

5.4.2 NN-FBP and Noise2Filter

Noise2Filter and NN-FBP benefit from a shared implementation. Therefore, most almost all implementation details are the same. As in the original NN-FBP implementation [PB13], the number of learned filters is set to $N_{\text{h}} = 4$, the non-linear activation function is the sigmoid, the exponential binning parameter is set to 2, but the filters are piece-wise linear — rather than piece-wise constant — as proposed

in [Lag+20]. Moreover, changes have been made to the shared implementation in order to accelerate data preparation, training, and reconstruction.

In the data preparation step, reconstructions are computed of the ortho-slices rather than the full volume. These reconstructions are performed using the RECAST3D software package [Buu+18].

Some changes have been made to the training procedure. As in the original implementation, the training objective is minimized using the Levenberg-Marquadt algorithm (LMA), which requires that the data samples are split into a training set and a validation set. Compared to the original implementation, however, the number of training samples is reduced from 10^6 to $5 \cdot 10^4$, and training is stopped after the validation set error has not improved for 10 epochs (originally 100 epochs were used). The effect of this reduction is discussed in **Section 5.5.2**. In addition, the original CPU implementation of the training process is accelerated by performing computations on the graphics processing unit (GPU) using PyTorch [Pas+17]. Final reconstructions are computed using the RECAST3D software package [Buu+18].

NN-FBP The free parameters for the NN-FBP method are trained and tested on *separate* tomographic datasets. The training dataset consists of paired noisy and noiseless reconstructions. Supervised training minimizes the training objective in Equation (5.7).

Noise2Filter The Noise2Filter parameters are optimized using self-supervised training on the noisy test dataset, rather than on a separate training dataset. No noiseless reconstructions are necessary for training. Depending on the training strategy (*X:1* or *1:X*), training minimizes either Equation (5.11) or (5.12).

5.4.3 Quantitative measures

Reconstruction accuracy is quantified using the the *Peak Signal-to-Noise Ratio* (PSNR) and the *Structural Similarity* (SSIM) index [Wan+04] metrics. Both metrics were computed with respect to the noiseless reconstructed images and using a data range that was determined by the minimum and maximum intensity of the noiseless reconstructed images. If not otherwise mentioned, the reported metrics are the average of the metric as computed on the three ortho-slices.

5.5 Experiments & Results

We performed several experiments to evaluate the Noise2Filter method. We provide a short summary below.

Reconstruction accuracy We compare Noise2Filter to supervised NN-FBP training and several standard FBP improvement strategies in terms of reconstruc-

tion accuracy.

Hyperparameter analysis Implementation choices in the design of the Noise2Filter method are analyzed, including the number of training samples, training strategy (*X:1* or *1:X*), and number of splits.

Timing An analysis of data preparation, training, and reconstruction speed is given.

Experimental data The method is applied to experimental data, including a showcase that illustrates the potential for use in dynamic control.

5.5.1 Reconstruction accuracy comparison

In this section, we assess the reconstruction accuracy of the Noise2Filter method. We compare to other filter-based reconstruction techniques in terms of reconstruction accuracy. Specifically, we compare to a baseline FBP reconstruction (with a Ram-Lak filter) and FBP with standard noise reduction techniques — Gaussian filtering (FBP_G) and frequency scaling (FBP_{sc}). These two methods are discussed in more detail in **Appendix 5.7.1**. In addition, we compare to the NN-FBP, which is trained on a separate training dataset with ground truth images.

The comparison is performed on the simulated foam dataset with varying levels of Poisson noise. The incident photon count I_0 was varied between 1000 and 32,000 in powers of two.

For each of the methods, parameter selection was performed as follows. For Noise2Filter, training was performed on the noisy test set. For NN-FBP, training was performed on a separate training dataset. For both methods, training was repeated 20 times to obtain statistics for the PSNR and SSIM. For Gaussian filtering and frequency scaling, the parameters maximizing the SSIM on the test set were determined using a linear grid search.

The Noise2Filter method with the *1:X* training strategy and 3 splits is used. We find that this yields consistent results at various noise levels.

The quantitative measures for the ortho-slices are shown in **Figure 5.4**. For all noise levels, the Noise2Filter metrics are higher than FBP with frequency scaling or Gaussian filtering. The NN-FBP method attains the best metrics, although the difference with Noise2Filter decreases as the noise level decreases. The difference in reconstruction accuracy is illustrated in **Figure 5.5**, where the ground truth phantom, reconstructions, and residuals for all considered methods are shown for the incident photon count $I_0 = 1000$. Notice that NN-FBP and Noise2Filter remove the noise in the voids, unlike the FBP methods.

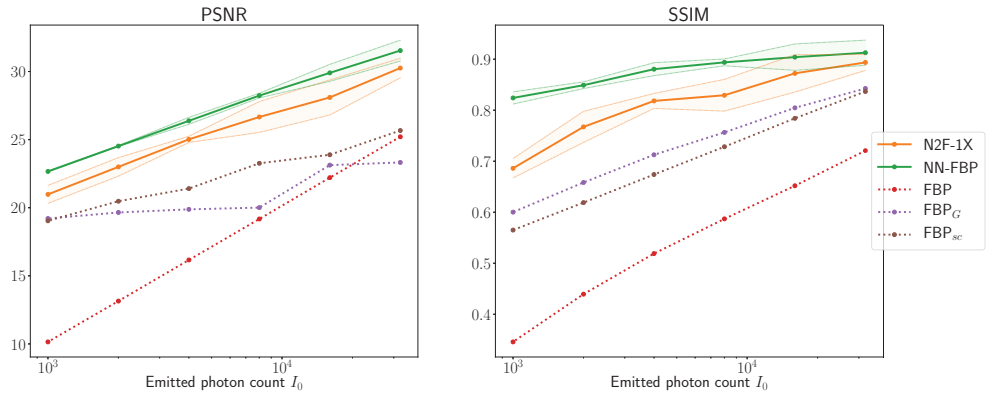


Figure 5.4: Reconstruction accuracy comparison of Noise2Filter (N2F-1X), NN-FBP, and FBP with Gaussian filtering, frequency scaling, and default filter. For varying noise levels, the average (line) and standard deviation (shaded region) over 20 trials of the PSNR and SSIM are reported.

5.5.2 Hyper parameter analysis

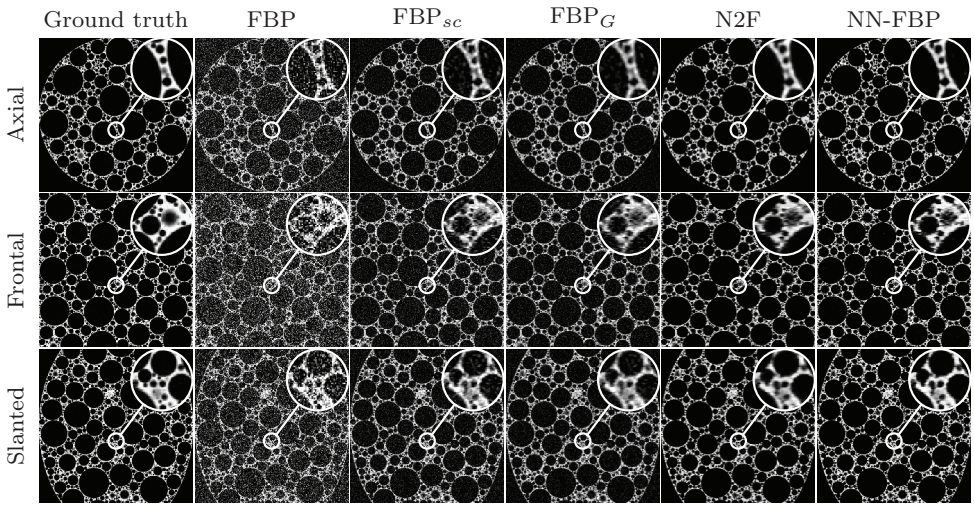
We consider three hyper parameters for the N2F method: the number of samples considered for training, the training strategy *X:1* or *1:X* and the number of splits N_s for the measured projection data.

First, we analyzed the reconstruction accuracy as a function of N_T, the number of training samples used in the training process. Here, the number of validation samples is fixed to 10% of the number of training samples. Noise was applied to the projection dataset equivalent to $I_0 = 1000$. The results for this experiment are shown in **Figure 5.6**. We observe that increasing the number of voxels yields virtually no increase in PSNR or SSIM beyond $N_T = 5 \cdot 10^4$ voxels.

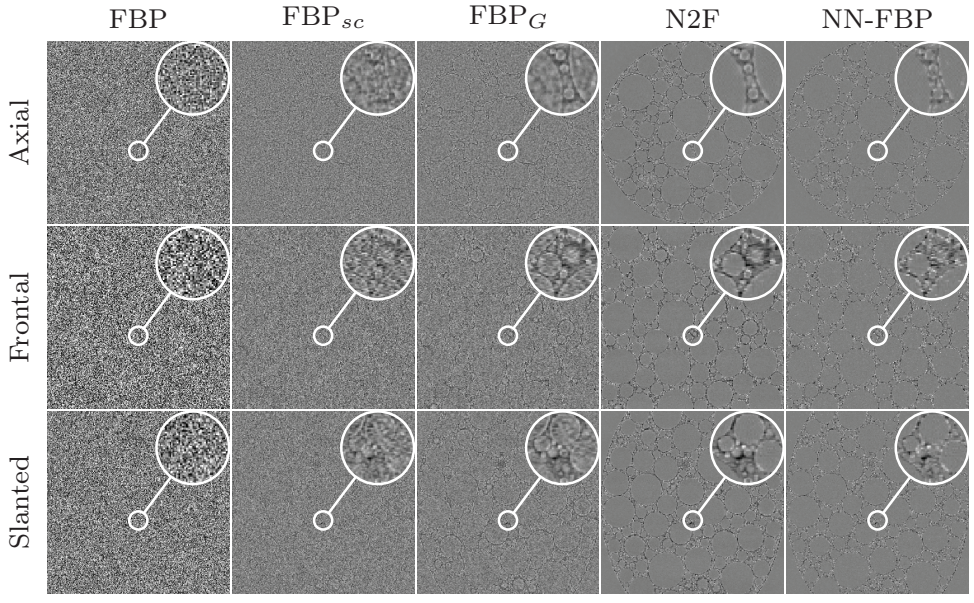
Second, we compare the training strategies and the number of splits on the simulated foam dataset for two noise levels, $I_0 = 1000$ and $I_0 = 8000$. For various values of the number of splits, 20 networks were trained and used to reconstruct the projection data. The average and standard deviation of the PSNR and SSIM are shown in **Figure 5.7**. For both noise levels we observe that the *1:X* strategy with 3 splits obtains the best SSIM and close to the best PSNR.

5.5.3 Timing comparison

We give timings for the data preparation, training, and reconstruction step of the Noise2Filter method. The computations were performed on a server with 375 GB of RAM and made use of a single Nvidia GeForce RTX 2080 Ti GPU (Nvidia, Santa



(a) Reconstructions



(b) Residuals

Figure 5.5: Reconstructions and residuals of the FBP algorithm, FBP with frequency scaling (FBP$_{sc}$, $sc = 0.4$), FBP with Gaussian filtering (FBP$_G$, $\sigma = 1.5$), Noise2Filter (N2F), and NN-FBP on a simulated foam phantom with photon count $I_0 = 1000$. Results are shown on an axial, frontal, and 45° slanted slice. The insets are zoomed by a factor of four.

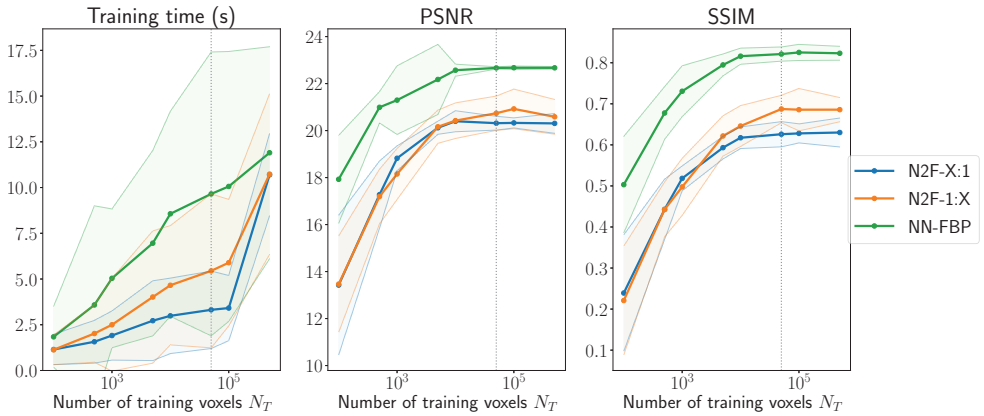


Figure 5.6: Training time and reconstruction accuracy for varying amounts of training voxels N_T. The mean (line) and standard deviation (shaded region) over 50 trials are reported. For both NN-FBP and Noise2Filter, increasing N_T yields diminishing returns in terms of PSNR and SSIM beyond $N_T = 5 \cdot 10^4$, as indicated by the dashed line.

Data size				Duration (seconds)		
# voxels	# pixels	# angles	N_e	DP	FBP	N2F
128^3	128×192	256	10	0.34	0.003	0.009
256^3	256×384	512	11	1.34	0.006	0.024
512^3	512×768	1024	12	6.08	0.030	0.114
1024^3	1024×1536	2048	13	44.00	—	—

Table 5.1: Benchmark results for the data preparation (DP) and reconstruction steps. FBP and Noise2Filter (N2F) reconstructions are performed on a single slice from filtered projection data. Due to memory constraints, some reconstructions were not performed, as indicated by a —.

Clara, CA, USA).

We computed the mean and standard deviation of the training time and number of epochs over 50 trials, resulting in a training time of $5.45 \pm 4.21s$ and a number of epochs of 58.21 ± 34.73.

In **Table 5.1** we report the reconstruction times of one 2D slice using the RECAST3D framework for standard FBP and the Noise2Filter method. We see that Noise2Filter is roughly 4 times slower than standard FBP, which is expected considering that we use $N_h = 4$ learned filters.

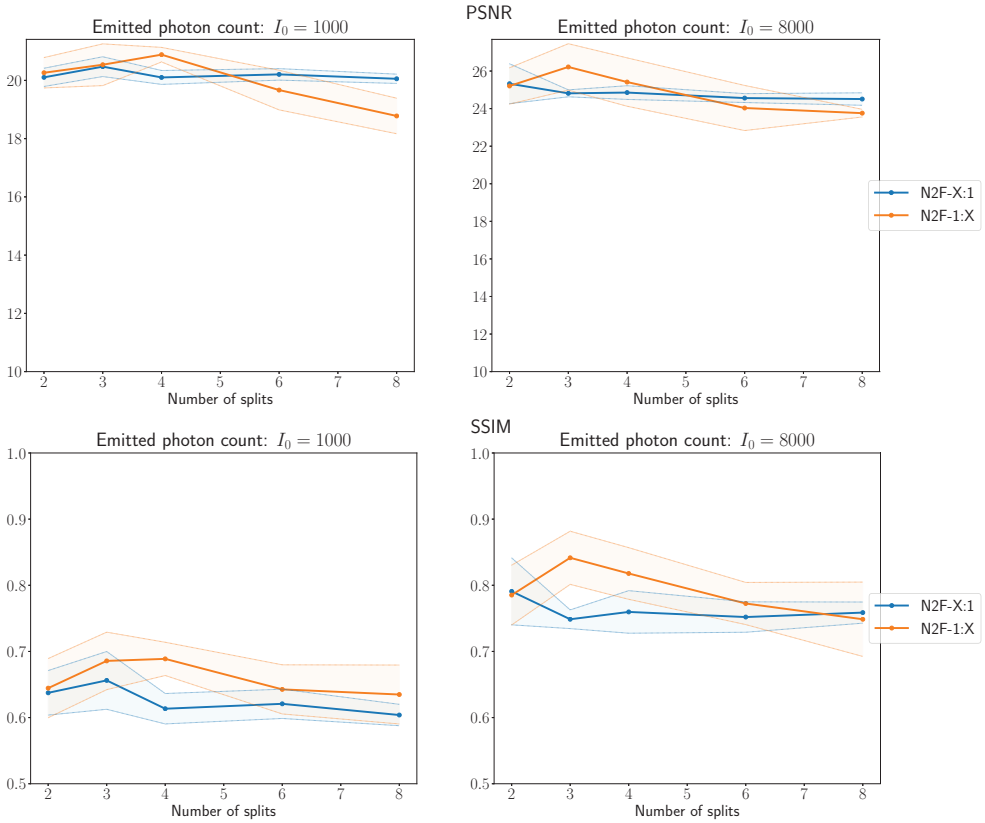


Figure 5.7: A comparison of Noise2Filter reconstruction accuracy for varying number of splits N_s and training strategies X:1 and 1:X. Mean (line) and standard deviation (shaded region) over 20 trials of the PSNR and SSIM are plotted for noise levels $I_0 = 1000$, and $I_0 = 8000$.

5.5.4 TomoBank dynamic dataset

We consider two experiments with an experimental dynamic tomographic dataset, consisting of 60 scans at consecutive time steps. First, we train Noise2Filter on the data from the first time step and use the trained reconstruction method to compute reconstructions for later time steps. This experiment aims to reveal the ability of Noise2Filter to generalize over dynamics in time. Second, we consider determining the correct center of rotation using Noise2Filter.

The experimental data is taken from the public TomoBank repository [De +18] and was acquired at the TOMCAT beamline at the Swiss Light Source (Paul Scherrer Institut, Switzerland). In this experiment, sub-second X-ray tomographic

microscopy was used to investigate liquid water dynamics in a fuel cell during operation. The experiment took less than 6 seconds, during which 60 scans were acquired. A scan consists of 301 projections taken by a detector with 1100×1440 detector pixels. Without loss of generality we have set the pixel size to 1, which means the linear attenuation coefficient — *i.e.*, the intensity of the reconstructions — is expressed in attenuation per pixel. Note that there is no reference reconstruction available for these experiments. Therefore, the analysis of these experiments is purely qualitative.

First, we train a Noise2Filter network at the first time step $T = 0$ and use this network to evaluate all further time steps. **Figure 5.8** shows the results for this strategy for $T = 0, 19, 39, 59$ and the FBP reconstructions at these time steps. There is no visible deterioration of the reconstruction accuracy over time, indicating that the trained network generalizes over the whole experiment.

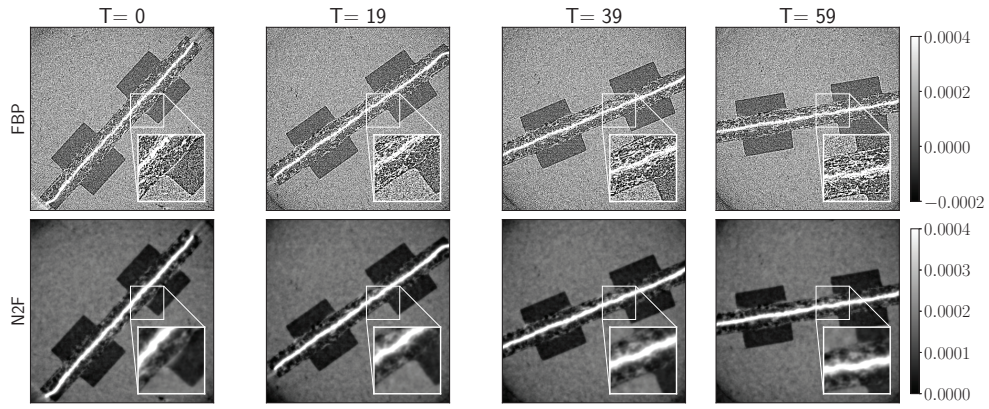


Figure 5.8: Reconstruction of the fuel cell at various time steps using FBP and Noise2Filter (N2F). The Noise2Filter method was trained on the first time step and also used to reconstruct later time steps. The insets are zoomed by a factor two.

Second, we consider determining the correct center of rotation. In the presence of noise, determining the correct center of rotation for a dataset can be difficult and is often performed after acquiring the measured projection data. Using the tools developed in [Van+20], the center of rotation can be adapted interactively in real-time. In **Figure 5.9** we show Noise2Filter and FBP reconstructions with shifted centers of rotation at the first time step. We note that no retraining was performed for Noise2Filter: the network parameters were determined once using a shift of 0 pixels. In the FBP reconstructions, the center of rotation artifacts (half moons) are difficult to discern. In the Noise2Filter reconstruction, however, these artifacts are both clearly visible, and visibly disappear at a shift of 19 pixels, which

coincides with the reported center of rotation in [De +18].

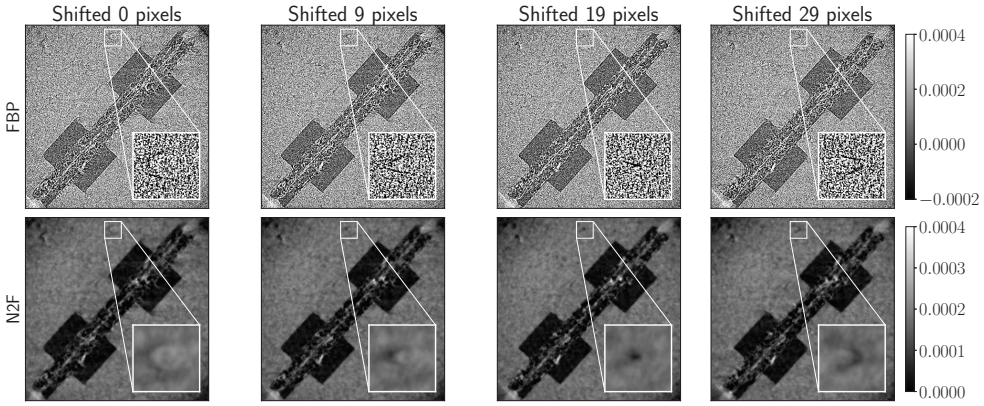


Figure 5.9: Reconstructions of a fuel cell at various centers of rotation using FBP and Noise2Filter (N2F). In the inset, a center of rotation artifact is highlighted, which disappears at a shift of 19 pixels. The distance between the detector pixels, or pixel pitch, for this dataset is 2.75 μm. The insets are zoomed by a factor four.

5.6 Conclusion and outlook

We have introduced Noise2Filter, a machine learning method for denoising filter-based reconstruction that does not require any additional training data beyond the acquired measurements. We show that this self-supervised method improves reconstruction accuracy compared to standard filter-based methods, and has limited loss of accuracy compared to its supervised counterpart (NN-FBP). The method exhibits sub-minute training times and reconstruction times in the order of hundred milliseconds, which demonstrates the potential for use in quasi-3D reconstruction for real-time visualization of tomographic experiments. In addition, we demonstrate that visual calibration of the center of rotation is possible, which illustrates the potential of our method for use in the dynamic control of tomographic experiments where noise is a challenge.

This method enables operators of dynamic experiments to directly adjust for external parameters — such as temperature — in response to changes in the measured object, even with high acquisition noise. Moreover, it can be used in high-throughput real-time quality control applications, where a fast scanning protocol leads to data with high acquisition noise.

5.7 Appendices

5.7.1 Standard FBP improvement strategies

In addition to standard FBP and NN-FBP, the Noise2Filter method is compared to two commonly used strategies to improve the reconstruction accuracy of the FBP algorithm for noisy data [Rus17].

Gaussian filtering

In this strategy the standard filter $\mathbf{h}$ in the FBP algorithm is convolved with a Gaussian filter $G_\sigma \in \mathbb{R}^{N_f}$ to smooth the noise in the reconstructions, with σ the standard deviation of the Gaussian. The elements j of the filter G_σ are defined as follows:

$$(G_\sigma)_j = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(j-N_f/2)^2}{2\sigma^2}}, \tag{5.18}$$

resulting in the smoothed reconstruction $\text{FBP}_G(\mathbf{y}, \mathbf{h}, \sigma) = W^T(\mathbf{y} * (\mathbf{h} * G_\sigma))$.

Frequency scaling

This strategy removes the higher frequencies from the FBP reconstruction. This is done by setting the frequencies above a threshold f_{sc} in Fourier domain of the filter $\mathbf{h}$ equal to zero and using this filter in the standard FBP algorithm, obtaining $\text{FBP}_{sc}(\mathbf{y}, \mathbf{h}_{sc}) = W^T(\mathbf{y} * \mathbf{h}_{sc})$.

For these strategies we optimized the choice of variable by computing reconstructions with a range of variables and taking the reconstruction with the highest SSIM.

Chapter 6

Conclusion

One of the main goals of the research presented in this thesis is to bridge the gap between theoretical research and practical use of CT reconstruction methods. Throughout this research the focus is on using mathematical insights from various fields to tackle practical problems encountered when using CT reconstruction methods: making state-of-the-art reconstruction methods more accessible to users without the need to fully understand the underlying mathematical theory, and improving already accessible reconstruction methods such that these methods are more widely applicable.

In **Chapter 2** we presented a framework in which one can efficiently explore different parameter choices for a reconstruction method. The framework was developed for a class of reconstruction methods: variational methods. These reconstruction methods are very effective in computing accurate reconstructions if the correct regularization parameter is chosen. Picking the optimal regularization parameter is not a straightforward process and often requires experience and understanding of the reconstruction method. The proposed method requires only a rough guess of the range of the regularization parameter from which approximate reconstructions can be computed using pixel-wise interpolation. The approximations can be computed efficiently and the choice of optimal regularization parameter is reduced to picking the optimal reconstruction from these approximations.

Filtered-backprojection methods are the among the easiest-to-use reconstruction methods due to their easy to choose parameters and the efficiency with which a reconstruction can be computed. However, the challenge for these methods is that they require measured projection data with a large number of projection images and low noise levels to produce accurate results. **Chapters 3**, **4** and **5** aim to improve FBP-type methods using various strategies, while maintaining the ease

of use.

The reconstruction accuracy of FBP-type methods, such as the FDK algorithm, can be improved by adapting the filter used in the algorithm. The process of determining the optimal filter is often a trial-and-error process. Therefore, we formulate in **Chapter 3** an optimization problem from which we can automatically compute the optimal filter for the measured projection data or similar projection data.

Due to the efficiency of FBP-type methods they are an excellent candidate for real-time tomography, *i.e.*, reconstructing the measured projection data as it is acquired. However, data acquired in a real-time scanning protocol often has a low number of projection images and high noise levels. In **Chapter 4** we have shown that the Neural Network Filtered-backprojection (NN-FBP) algorithm — an algorithm shown to be fast and accurate for parallel beam — can be extended to general FBP-type methods and specifically to the FDK algorithm.

The challenge with the NN-FBP and NN-FDK algorithm is that they contain a machine learning component trained using supervised learning. This limits the applicability of these methods to cases where high quality reference data is available. In **Chapter 5** we have shown that this problem can be circumvented by using the Noise2Inverse training to train the NN-FBP network, which only uses noisy measured projection data to train the network. Moreover, we have shown that this training process is very fast — *i.e.*, sub-minute — and that the NN-FBP algorithm can be applied in the RECAST3D real time quasi 3D reconstruction framework. The resulting method is dubbed the Noise2Filter (N2F) method and can be used to reconstruct arbitrarily oriented 2D slices of a 3D reconstruction volume in real-time. And although the N2F method combines several state-of-the-art concepts, the number of reconstruction parameters that have to be set is limited. Moreover, the choice of the reconstruction parameters is straightforward.

The effects of the regularization in NN-FBP and NN-FDK can mainly be observed in the x, y-plane, whereas the effects in the z-direction are less pronounced. Therefore, I believe that adapting these methods to 2D filters could lead to even better results. This could be achieved by using the proposed multilayer perceptron framework and moving to full 2D filters or slab filters, *i.e.*, a stack of several 1D filters. Alternatively, the multilayer perceptron framework could be replaced by a CNN framework. Ideally when developing such methods, the locality, pointwise, and two-step properties — as described in **Chapter 5** — are maintained such that the method could be applied in the RECAST3D framework.

To automate a reconstruction method one should know what the reconstruction parameters are that lead to an optimal reconstruction. However, as we have seen in this thesis, there are many different metrics and conditions for what an

"optimal" reconstruction is. Even with a ground truth or high quality reference reconstruction it is not agreed upon which metric indicates the best reconstructions. Therefore, instead of computing a reconstruction and then using that reconstruction to answer an application specific question, one could combine the two steps in one method and directly answer the question. This could be achieved by combining a classifier network architecture with reconstruction method and considering the reconstruction parameters as trainable parameters. This way the classifier network automatically determines which reconstructions are best to use to answer the posed question.

We have considered several machine learning methods throughout this thesis and there are many more being applied and developed. Although the training procedure for these methods rely roughly on the same mechanism, almost every network architecture has a different best practice to train the networks. Understanding why these differences work best and standardizing training procedures will greatly improve the ease of use of a machine learning method.

To conclude, with the rise in popularity of machine learning methods, the commercial availability of CT scanners, and the development of high-resolution detectors the field of CT imaging is still generating many interesting research questions from both an application and a theoretical point of view.

Bibliography

[Aar+15] W. van Aarle et al. 'The ASTRA Toolbox: a platform for advanced algorithm development in electron tomography'. In: *Ultramicroscopy* 157 (2015), pp. 35–47.

[AB09] M. Anthony and P. L. Bartlett. *Neural network learning: Theoretical foundations*. Cambridge University Press, 2009.

[AK84] A. H. Andersen and A. C. Kak. 'Simultaneous algebraic reconstruction technique (SART): a superior implementation of the ART algorithm'. In: *Ultrasonic imaging* 6.1 (1984), pp. 81–94.

[AKÖ17] J. Adler, H. Kohr and O. Öktem. *ODL 0.6.0*. Apr. 2017.

[AÖ17] J. Adler and O. Öktem. 'Solving ill-posed inverse problems using iterative deep neural networks'. In: *Inverse Problems* 33.12 (2017), p. 124007.

[AÖ18] J. Adler and O. Öktem. 'Learned primal-dual reconstruction'. In: *IEEE transactions on medical imaging* 37.6 (2018), pp. 1322–1332.

[AST13] ASTM. *Standard Test Method for Measurement of Computed Tomography (CT) System Performance*. 2013.

[Beh+11] S. Behnel et al. 'Cython: The Best of Both Worlds'. In: *Computing in Science Engineering* 13.2 (2011), pp. 31–39.

[Ber+20] A. Bernard et al. '3D characterization of walnut morphological traits using X-ray computed tomography'. In: *preprint* (2020).

[Bis06] C. M. Bishop. *Pattern recognition and machine learning*. Springer Science+ Business Media, 2006.

[BK06] K. J. Batenburg and W. A. Kosters. 'A neural network approach to real-time discrete tomography'. In: *IWCIA*. Springer. 2006, pp. 389–403.

[BKP10] K. Bredies, K. Kunisch and T. Pock. 'Total generalized variation'. In: *SIAM Journal on Imaging Sciences* 3.3 (2010), pp. 492–526.

[BM12] G. Blanchard and P. Mathé. 'Discrepancy principle for statistical inverse problems with application to conjugate gradient iteration'. In: *Inverse Problems* 28.11 (2012), p. 115011.

[BP12] K. J. Batenburg and L. Plantagie. 'Fast approximation of algebraic reconstruction methods for tomography'. In: *IEEE Transactions on Image Processing* 21.8 (2012), pp. 3648–3658.

[BR19] J. Batson and L. Royer. 'Noise2Self: Blind Denoising by Self-Supervision'. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by K. Chaudhuri and R. Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, USA: PMLR, 2019, pp. 524–533.

[Bri+18] B. Bringmann et al. 'The homotopy method revisited: Computing solution paths of ℓ_1-regularized problems'. In: *Mathematics of Computation* 87.313 (2018), pp. 2343–2364.

[BT09] A. Beck and M. Teboulle. 'A fast iterative shrinkage-thresholding algorithm for linear inverse problems'. In: *SIAM Journal on Imaging Sciences* 2.1 (2009), pp. 183–202.

[Bul+16] T. Bultreys et al. 'Fast laboratory-based micro-computed tomography for pore-scale research: illustrative experiments and perspectives on the future'. In: *Advances in Water Resources* 95 (2016), pp. 341–351.

[Bur+13] M. Burger et al. 'An adaptive inverse scale space method for compressed sensing'. In: *Mathematics of Computation* 82.281 (2013), pp. 269–299.

[Bur+16] M. Burger et al. 'Spectral decompositions using one-homogeneous functionals'. In: *SIAM Journal on Imaging Sciences* 9.3 (2016), pp. 1374–1408.

[Buu+18] J.-W. Buurlage et al. 'Real-time quasi-3D tomographic reconstruction'. In: *Measurement Science and Technology* 29.6 (2018), p. 064005.

[Buu+19] J.-W. Buurlage et al. 'Real-time reconstruction and visualisation towards dynamic feedback control during time-resolved tomography experiments at TOMCAT'. In: *Scientific Reports* 9.1 (2019), pp. 1–11.

[Buz08] T. M. Buzug. *Computed tomography: from photon statistics to modern cone-beam CT*. Springer Science & Business Media, 2008.

[Can] Canon Medical Systems USA, Inc. *Aquilon™ Precision, ULTRA High Resolution CT*. `https://us.medical.canon/products/computed-tomography/aquilion-precision/`. [Accessed: 24-Nov-2020].

[Çiç+16] Ö. Çiçek et al. '3D U-Net: learning dense volumetric segmentation from sparse annotation'. In: *International conference on medical image computing and computer-assisted intervention*. Springer. 2016, pp. 424–432.

[CLB18] S. B. Coban, M. J. Lagerwerf and K. J. Batenburg. *High-resolution cone-beam scan of two pomegranates with two dosage levels*. Oct. 2018.

[Cob+20] S. B. Coban et al. 'Explorative Imaging and Its Implementation at the FleX-ray Laboratory'. In: *Journal of Imaging* 6.4 (2020), p. 18.

[Com] G. E. Company. *Discovery™ CT750 HD, The leading edge of CT clarity*. http://www3.gehealthcare.com/~/media/documents/us-global/products/computed-tomography/brochures/discovery-ct-750-hd/gehealthcare-brochure_discover-ct750-hd.pdf. [Accessed: 24-Nov-2020].

[CP11] A. Chambolle and T. Pock. 'A first-order primal-dual algorithm for convex problems with applications to imaging'. In: *Journal of Mathematical Imaging and Vision* 40.1 (2011), pp. 120–145.

[De +18] F. De Carlo et al. 'TomoBank: a Tomographic Data Repository for Computational X-Ray Science'. In: *Measurement Science and Technology* 29.3 (2018), p. 034004.

[Dic45] L. R. Dice. 'Measures of the amount of ecologic association between species'. In: *Ecology* 26.3 (1945), pp. 297–302.

[Die+14] M. Dierick et al. 'Recent micro-CT scanner developments at UGCT'. In: *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms* 324 (2014), pp. 35–40.

[EF03] I. A. Elbakri and J. A. Fessler. 'Efficient and accurate likelihood for iterative image reconstruction in X-ray computed tomography'. In: *Medical Imaging 2003: Image Processing*. Vol. 5032. International Society for Optics and Photonics. 2003, pp. 1839–1850.

[FDK84] L. Feldkamp, L. Davis and J. Kress. 'Practical cone-beam algorithm'. In: *JOSA A* 1.6 (1984), pp. 612–619.

[Fes10] J. A. Fessler. 'Model-based image reconstruction for MRI'. In: *IEEE Signal Processing Magazine* 27.4 (2010), pp. 81–89.

[FJ05] M. Frigo and S. G. Johnson. 'The design and implementation of FFTW3'. In: *Proceedings of the IEEE* 93.2 (2005), pp. 216–231.

[For+02] E. Ford et al. 'Cone-beam CT with megavoltage beams and an amorphous silicon electronic portal imaging device: Potential for verification of radiotherapy of lung cancer'. In: *Medical physics* 29.12 (2002), pp. 2913–2924.

[Gar+18] F. García-Moreno et al. 'Time-resolved in situ tomography for the analysis of evolving metal-foam granulates'. In: *Journal of Synchrotron Radiation* 25.5 (2018), pp. 1505–1508.

[GBH70] R. Gordon, R. Bender and G. T. Herman. 'Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and X-ray photography'. In: *Journal of Theoretical Biology* 29.3 (1970), pp. 471–481.

[GHO99] G. H. Golub, P. C. Hansen and D. P. O'Leary. 'Tikhonov regularization and total least squares'. In: *SIAM Journal on Matrix Analysis and Applications* 21.1 (1999), pp. 185–194.

[GKT17] J. C. Galicia, J. Kawilarang and P. Z. Tawil. 'Clinical Endodontic Applications of Cone Beam-Computed Tomography in Modern Dental Practice'. In: *Open Journal of Stomatology* 7.07 (2017), p. 314.

[GMD06] D. J. Godfrey, H. McAdams and J. T. Dobbins. 'Optimization of the matrix inversion tomosynthesis (MITS) impulse response and modulation transfer function characteristics for chest imaging'. In: *Medical physics* 33.3 (2006), pp. 655–667.

[Goc16] M. Gockenbach. *Linear Inverse Problems and Tikhonov Regularization*. 32. The Mathematical Association of America, 2016.

[GUV11] F. Giudiceandrea, E. Ursella and E. Vicario. 'A high speed CT scanner for the sawmill industry'. In: *Proceedings of the 17th international non destructive testing and evaluation of wood symposium*. University of West Hungary Sopron, Hungary. 2011, pp. 14–16.

[Ham+18] K. Hammernik et al. 'Learning a variational network for reconstruction of accelerated MRI data'. In: *Magnetic Resonance in Medicine* 79.6 (2018), pp. 3055–3071.

[Han92] P. C. Hansen. 'Analysis of discrete ill-posed problems by means of the L-curve'. In: *SIAM review* 34.4 (1992), pp. 561–580.

[Han99] P. C. Hansen. 'The L-curve and its use in the numerical treatment of inverse problems'. In: (1999).

[Hen+19] A. A. Hendriksen et al. 'On-the-Fly Machine Learning for Improving Image Resolution in Tomography'. In: *Applied Sciences* 9.12 (2019), p. 2445.

[Her09] G. T. Herman. *Fundamentals of computerized tomography: image reconstruction from projections*. Springer Science & Business Media, 2009.

[HO93] P. C. Hansen and D. P. O'Leary. 'The use of the L-curve in the regularization of discrete ill-posed problems'. In: *SIAM Journal on Scientific Computing* 14.6 (1993), pp. 1487–1503.

[HPB20] A. A. Hendriksen, D. M. Pelt and K. J. Batenburg. 'Noise2Inverse: Self-supervised deep convolutional denoising for tomography'. In: *IEEE Transactions on Computational Imaging* (2020).

[HS95] J. H. Hubbell and S. M. Seltzer. *Tables of X-ray mass attenuation coefficients and mass energy-absorption coefficients 1 keV to 20 MeV for elements Z= 1 to 92 and 48 additional substances of dosimetric interest*. Tech. rep. National Inst. of Standards and Technology-PL, Gaithersburg, MD (United States). Ionizing Radiation Div., 1995.

[Hsi+09] J. Hsieh et al. 'Computed tomography: principles, design, artifacts, and recent advances'. In: SPIE Bellingham, WA. 2009.

[HTF09] T. Hastie, R. Tibshirani and J. Friedman. 'The Elements of Statistical Learning'. In: *Springer Series in Statistics* (2009).

[Jia+10] X. Jia et al. 'GPU-based fast cone beam CT reconstruction from undersampled and noisy projection data via total variation'. In: *Medical Physics* 37.4 (2010), pp. 1757–1760.

[Jin+17] K. H. Jin et al. 'Deep convolutional neural network for inverse problems in imaging'. In: *IEEE Transactions on Image Processing* 26.9 (2017), pp. 4509–4522.

[JOP+01] E. Jones, T. Oliphant, P. Peterson et al. *SciPy: Open source scientific tools for Python*. [Online: Accessed 24-Nov-2020]. 2001.

[Kac37] S. Kaczmarz. 'Genäherte Auflösung von Systemen linearer Gleichungen'. In: *Bull. Int. Acad. Pol. Sei. Lett. A* (1937), pp. 335–37.

[Kat03] A. Katsevich. 'A general scheme for constructing inversion algorithms for cone beam CT'. In: *International Journal of Mathematics and Mathematical Sciences* 2003.21 (2003), pp. 1305–1321.

[KB14] D. P. Kingma and J. Ba. 'Adam: A method for stochastic optimization'. In: *arXiv preprint arXiv:1412.6980* (2014).

[Kid+18] S. Kida et al. 'Cone beam computed tomography image quality improvement using a deep convolutional neural network'. In: *Cureus* 10.4 (2018).

[KMY17] E. Kang, J. Min and J. C. Ye. 'A Deep Convolutional Neural Network Using Directional Wavelets for Low-Dose X-Ray CT Reconstruction'. In: *Medical Physics* 44.10 (2017), e360–e375.

[KND98] H. Kudo, F. Noo and M. Defrise. 'Cone-beam filtered-backprojection algorithm for truncated helical data'. In: *Physics in Medicine & Biology* 43.10 (1998), p. 2885.

[Kob+17] E. Kobler et al. 'Variational networks: connecting variational methods and deep learning'. In: *German conference on pattern recognition*. Springer. 2017, pp. 281–293.

[KS01] A. C. Kak and M. Slaney. *Principles of computerized tomographic imaging*. SIAM, 2001.

[Kun+07] H. Kunze et al. 'Filter determination for tomosynthesis aided by iterative reconstruction techniques'. In: *9th International Meeting on Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine*. 2007, pp. 309–312.

[Laga] M. J. Lagerwerf. *Data-dependent filters for FDK algorithm*. `https://github.com/MJLagerwerf/ddf_fdk`. [Accessed: 24-Nov-2020].

[Lagb] M. J. Lagerwerf. *Neural Network FDK algorithm*. `https://github.com/MJLagerwerf/nn_fdk`. [Accessed: 24-Nov-2020].

[Lagc] M. J. Lagerwerf. *Pixel-wise interpolation for regularization parameter space exploration*. `https://github.com/MJLagerwerf/reg_param`. [Accessed: 24-Nov-2020].

[Lag+20] M. J. Lagerwerf et al. 'Automated FDK-filter selection for Cone-beam CT in research environments'. In: *IEEE Transactions on Computational Imaging* Early acces (2020).

[LB] G. Lauritsch and H. Bruder. *FORBILD Head Phantom*. `http://www.imp.uni-erlangen.de/phantoms/head/head.html`. [Accessed: 24-Nov-2020].

[LCB20] M. J. Lagerwerf, S. B. Coban and K. J. Batenburg. *High-resolution cone-beam scan of twenty-one walnuts with two dosage levels*. Zenodo, Apr. 2020.

[Lev44] K. Levenberg. 'A method for the solution of certain non-linear problems in least squares'. In: *Quarterly of applied mathematics* 2.2 (1944), pp. 164–168.

[LÖS18] S. Lunz, O. Öktem and C.-B. Schönlieb. 'Adversarial regularizers in inverse problems'. In: *Advances in Neural Information Processing Systems*. 2018, pp. 8507–8516.

[Mar63] D. W. Marquardt. 'An algorithm for least-squares estimation of nonlinear parameters'. In: *Journal of the society for Industrial and Applied Mathematics* 11.2 (1963), pp. 431–441.

[Mid+01] P. A. Midgley et al. 'Z-Contrast tomography: a technique in three-dimensional nanostructural analysis based on Rutherford scatteringElectronic supplementary information (ESI) available: 3D animations of a Pd–Ru bimetallic catalyst generated from a tomographic reconstruction of HAADF STEM images.' In: *Chemical Communications* 10 (2001), pp. 907–908.

[Muk+20] S. Mukherjee et al. 'Learned convex regularizers for inverse problems'. In: *arXiv preprint arXiv:2008.02839* (2020).

[Nat01] F. Natterer. *The mathematics of computerized tomography*. SIAM, 2001.

[Nie+12] T. Nielsen et al. 'Filter calculation for X-ray tomosynthesis reconstruction'. In: *Physics in medicine and biology* 57.12 (2012), p. 3915.

[Niu+14] S. Niu et al. 'Sparse-view X-ray CT reconstruction via total generalized variation regularization'. In: *Physics in Medicine & Biology* 59.12 (2014), p. 2997.

[NW90] D. Nguyen and B. Widrow. 'The truck backer-upper: An example of self-learning in neural networks'. In: *Advanced neural computers*. Elsevier, 1990, pp. 11–19.

[OSu85] J. D. O'Sullivan. 'A fast sinc function gridding algorithm for Fourier inversion in computer tomography'. In: *IEEE transactions on Medical Imaging* 4.4 (1985), pp. 200–207.

[Ots79] N. Otsu. 'A threshold selection method from gray-level histograms'. In: *IEEE Transactions on Systems, Man, and Cybernetics* 9.1 (1979), pp. 62–66.

[Pas+17] A. Paszke et al. 'Automatic differentiation in PyTorch'. In: *NIPS-W*. 2017.

[Pas+19] A. Paszke et al. 'PyTorch: An Imperative Style, High-Performance Deep Learning Library'. In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035.

[PB13] D. M. Pelt and K. J. Batenburg. 'Fast tomographic reconstruction from limited data using artificial neural networks'. In: *IEEE Transactions on Image Processing* 22.12 (2013), pp. 5238–5251.

[PB14] D. M. Pelt and K. J. Batenburg. 'Improving filtered backprojection reconstruction by data-dependent filtering'. In: *IEEE Transactions on Image Processing* 23.11 (2014), pp. 4750–4762.

[PBS18] D. M. Pelt, K. J. Batenburg and J. Sethian. 'Improving tomographic reconstruction from limited data using mixed-scale dense convolutional neural networks'. In: *Journal of Imaging* 4.11 (2018), p. 128.

[PCC18] C. S. Perone, E. Calabrese and J. Cohen-Adad. 'Spinal Cord Gray Matter Segmentation Using Deep Dilated Convolutions'. In: *Scientific Reports* 8.1 (2018).

[Pla] Planmeca. *Planmeca AINO™ filter*. https://www.planmeca.com/imaging/3d-imaging/planmeca-aino/. [Accessed: 24-Nov-2020].

[Pre+11] A. J. Pretorius et al. 'Visualization of parameter space for image analysis'. In: *IEEE Transactions on Visualization and Computer Graphics* 17.12 (2011), pp. 2402–2411.

[PS18] D. M. Pelt and J. A. Sethian. 'A mixed-scale dense convolutional neural network for image analysis'. In: *Proceedings of the National Academy of Sciences* 115.2 (2018), pp. 254–259.

[PSV09] X. Pan, E. Y. Sidky and M. Vannier. 'Why do commercial CT scanners still employ traditional, filtered back-projection for image reconstruction?' In: *Inverse Problems* 25.12 (2009), p. 123009.

[REM17] Y. Romano, M. Elad and P. Milanfar. 'The Little Engine That Could: Regularization By Denoising (RED)'. In: *SIAM Journal on Imaging Sciences* 10.4 (2017), pp. 1804–1844.

[RFB15] O. Ronneberger, P. Fischer and T. Brox. 'U-net: Convolutional networks for biomedical image segmentation'. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[RL71] G. Ramachandran and A. Lakshminarayanan. 'Three-dimensional reconstruction from radiographs and electron micrographs: application of convolutions instead of Fourier transforms'. In: *Proceedings of the National Academy of Sciences* 68.9 (1971), pp. 2236–2240.

[ROF92] L. I. Rudin, S. Osher and E. Fatemi. 'Nonlinear total variation based noise removal algorithms'. In: *Physica D: Nonlinear Phenomena* 60.1-4 (1992), pp. 259–268.

[RS18] E. T. Reehorst and P. Schniter. 'Regularization By Denoising: Clarifications and New Interpretations'. In: *CoRR* (2018). arXiv: 1806.02296 [cs.CV].

[Rus17] P. Russo. *Handbook of X-ray imaging: physics and technology*. CRC Press, 2017.

[San+14] T. dos Santos Rolo et al. 'In Vivo X-Ray Cine-Tomography for Tracking Morphological Dynamics'. In: *Proceedings of the National Academy of Sciences* 111.11 (2014), pp. 3921–3926.

[Sch+09] O. Scherzer et al. *Variational methods in imaging*. Springer, 2009.

[Sed+14] M. Sedlmair et al. 'Visual parameter space analysis: A conceptual framework'. In: *IEEE Transactions on Visualization and Computer Graphics* 20.12 (2014), pp. 2161–2170.

[SJP12] E. Y. Sidky, J. H. Jørgensen and X. Pan. 'Convex optimization problem prototyping for image reconstruction in computed tomography with the Chambolle–Pock algorithm'. In: *Physics in Medicine & Biology* 57.10 (2012), p. 3065.

[SLD17] E. Shelhamer, J. Long and T. Darrell. 'Fully Convolutional Networks for Semantic Segmentation'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.4 (2017), pp. 640–651.

[SLX+16] J. Sun, H. Li, Z. Xu et al. 'Deep ADMM-Net for compressive sensing MRI'. In: *Advances in neural information processing systems*. 2016, pp. 10–18.

[SP08] E. Y. Sidky and X. Pan. 'Image reconstruction in circular cone-beam computed tomography by constrained, total-variation minimization'. In: *Physics in Medicine and Biology* 53.17 (2008), p. 4777.

[TESa] TESCAN. *TESCAN DynaTOM, High temporal resolution 4D X-ray imaging*. https://www.tescan.com/product/micro-ct-for-materials-science-tescan-dynatom/. [Accessed: 24-Nov-2020].

[TESb] TESCAN. *TESCAN UniTOM XL, Modular and versatile high resolution 3D X-ray imaging*. https://www.tescan.com/product/micro-ct-for-materials-science-tescan-unitom-xl/. [Accessed: 24-Nov-2020].

[Tuy83] H. K. Tuy. 'An inversion formula for cone-beam reconstruction'. In: *SIAM Journal on Applied Mathematics* 43.3 (1983), pp. 546–552.

[Vai82] G. M. Vainikko. 'The discrepancy principle for a class of regularization methods'. In: *USSR computational mathematics and mathematical physics* 22.3 (1982), pp. 1–19.

[Van+16] W. Van Aarle et al. 'Fast and flexible X-ray tomography using the ASTRA toolbox'. In: *Optics Express* 24.22 (2016), pp. 25129–25147.

[Van+20] H. Vanrompay et al. 'Real-Time Reconstruction of Arbitrary Slices for Quantitative and In Situ 3D Characterization of Nanoparticles'. In: *Particle & Particle Systems Characterization* (2020), p. 2000073. eprint: https://www.onlinelibrary.wiley.com/doi/pdf/10.1002/ppsc.202000073.

[VBW13] S. V. Venkatakrishnan, C. A. Bouman and B. Wohlberg. 'Plug-And-Play Priors for Model Based Reconstruction'. In: *2013 IEEE Global Conference on Signal and Information Processing* (2013).

[VV90] A. Van der Sluis and H. A. van der Vorst. 'SIRT-and CG-type methods for the iterative solution of sparse linear least-squares problems'. In: *Linear Algebra and its Applications* 130 (1990), pp. 257–303.

[Wal+14] S. van der Walt et al. 'Scikit-Image: image processing in Python'. In: *PeerJ* 2 (June 2014), e453.

[Wan+04] Z. Wang et al. 'Image quality assessment: from error visibility to structural similarity'. In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612.

[Wan+18] G. Wang et al. 'Image reconstruction is a new frontier of machine learning'. In: *IEEE Transactions on Medical Imaging* 37.6 (2018), pp. 1289–1296.

[Wat94] D. W. Watt. 'Column-relaxed algebraic reconstruction algorithm for tomography with noisy data'. In: *Applied optics* 33.20 (1994), pp. 4420–4427.

[WCV11] S. v. d. Walt, S. C. Colbert and G. Varoquaux. 'The NumPy array: a structure for efficient numerical computation'. In: *Computing in Science & Engineering* 13.2 (2011), pp. 22–30.

[WKL19] D. Wu, K. Kim and Q. Li. 'Computationally efficient deep neural network for computed tomography image reconstruction'. In: *Medical Physics* 46.11 (2019), pp. 4763–4776.

[Xu+20] H. Xu et al. 'Optimal Image Denoising for In Situ X-ray Tomographic Microscopy of Liquid Water in Gas Diffusion Layers of Polymer Electrolyte Fuel Cells'. In: *Journal of The Electrochemical Society* 167.10 (2020), p. 104505.

[YHC18] J. C. Ye, Y. Han and E. Cha. 'Deep convolutional framelets: A general deep learning framework for inverse problems'. In: *SIAM Journal on Imaging Sciences* 11.2 (2018), pp. 991–1048.

[Zen12] G. L. Zeng. ‘A filtered backprojection algorithm with characteristics of the iterative Landweber algorithm’. In: *Medical physics* 39.2 (2012), pp. 603–607.

[Zha+17] K. Zhang et al. ‘Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising’. In: *IEEE Transactions on Image Processing* 26.7 (2017), pp. 3142–3155.

List of publications

Publications that are part of this dissertation:

- An Efficient Interpolation Approach for Exploring the Parameter Space of Regularized Tomography Algorithms. *MJ Lagerwerf, WJ Palenstijn, F Bleichrodt, KJ Batenburg.* Fundamenta Informaticae (Volume: 172), number 2, pp. 143–167, 2020.
- Automated FDK-Filter Selection for Cone-Beam CT in Research Environments. *MJ Lagerwerf, WJ Palenstijn, H Kohr, KJ Batenburg.* IEEE Transactions on Computational Imaging (Volume: 6), pp. 739–748, 2020.
- A computationally efficient reconstruction algorithm for circular cone-beam computed tomography using shallow neural networks. *MJ Lagerwerf, DM Pelt, WJ Palenstijn, KJ Batenburg.* Journal of Imaging, Early access, 2020.
- Noise2Filter: fast, self-supervised learning and real-time reconstruction for 3D Computed Tomography. *MJ Lagerwerf, AA Hendriksen, JW Buurlage, KJ Batenburg.* Machine Learning: Science and Technology (Volume 2), number 1, 2020.

Publications that are not part of this dissertation:

- A framework for directional and higher-order reconstruction in photoacoustic tomography. *YE Boink, MJ Lagerwerf, W Steenbergen, SA van Gils, S Srirang, C Brune.* Physics in Medicine & Biology (Volume: 63), number 4, pp. 045018, 2018.
- Directional sinogram inpainting for limited angle tomography. *R Tovey, M Benning, C Brune, MJ Lagerwerf, SM Collins, RK Leary, PA Midgley, CB Schönlieb.* Inverse problems (Volume: 35), number 2, pp. 024004, 2019.

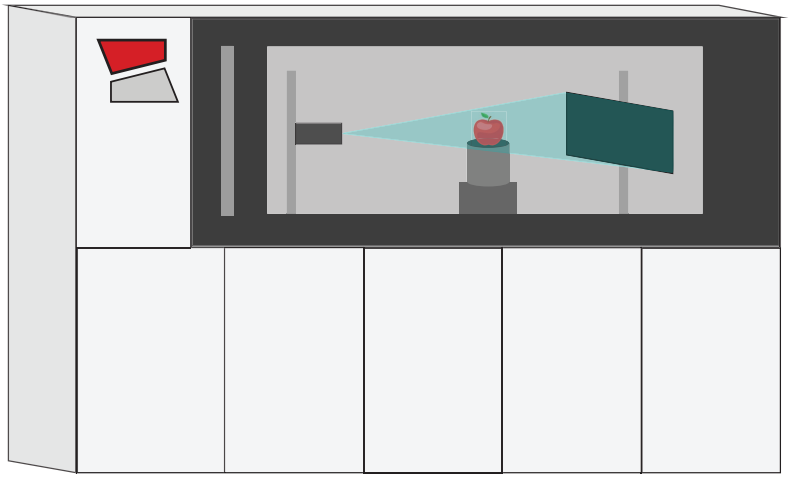
Samenvatting in het Nederlands

This chapter contains a layman summary of the research presented in this thesis, and is written in Dutch.

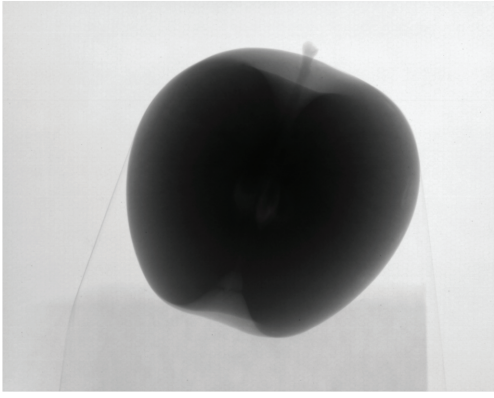
Voor een groot aantal toepassingen is het interessant om de binnenkant van een object in beeld te brengen, zonder het object te vernietigen. Voorbeelden van toepassingen zijn legio, en liggen op zeer diverse vlakken: het productieproces van computerchips controleren, het inspecteren van een brug op barsten in de ondersteunende balken, of het onderzoeken van de organen van een patiënt.

Tomografische reconstructie is de wiskundige rekenmethode die vaak gebruikt wordt voor dit soort toepassingen. In tomografie wordt een golf die door materialen heen kan bewegen gebruikt om metingen van het object te maken (zie **figuur 1**). Deze metingen worden gedaan uit verschillende posities en worden gecombineerd tot één grote dataset. Deze dataset wordt vervolgens gebruikt om berekeningen te doen die een beeld opleveren van het binnenste van het gemeten object. Deze beelden worden ook wel *reconstructies* genoemd. Een voorbeeld van een reconstructie is gegeven in **figuur 1c**. Voorbeelden van golven die gebruikt worden in de tomografie om reconstructies mee te maken zijn: geluidsgolven; gebruikt voor ultrasound en fotoakoestische metingen; X-rays gebruikt voor CT scanners, magnetische velden; gebruikt voor MRI scanners, en elektronen; gebruikt voor elektronemicroscopen.

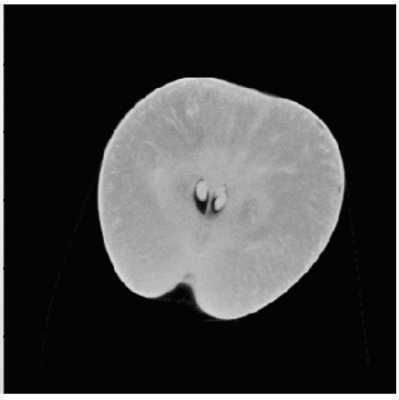
In dit proefschrift focussen we voornamelijk op *computed tomography*, ofwel CT. Deze vorm van tomografie wordt veel in medische toepassingen gebruikt, maar ook in het ontwikkelen van nieuwe materialen, en zelfs in het onderzoeken en digitaliseren van museumobjecten. In het algemeen worden eerst de CT-metingen gedaan, om daarna pas reconstructie te berekenen. Maar als deze reconstructie berekend zou kunnen worden terwijl de CT-metingen gedaan worden zijn er ineens een heleboel nieuwe interessante toepassingen mogelijk. Dit betekent dat het mogelijk is om het object direct te observeren tijdens de scan en op deze observaties te reageren. Neem bijvoorbeeld het stresstesten van een bouwmateriaal. Hier wordt druk op een materiaal geleverd tot het breekt. Doordat de scheuren



(a) Illustratie van een CT scan.

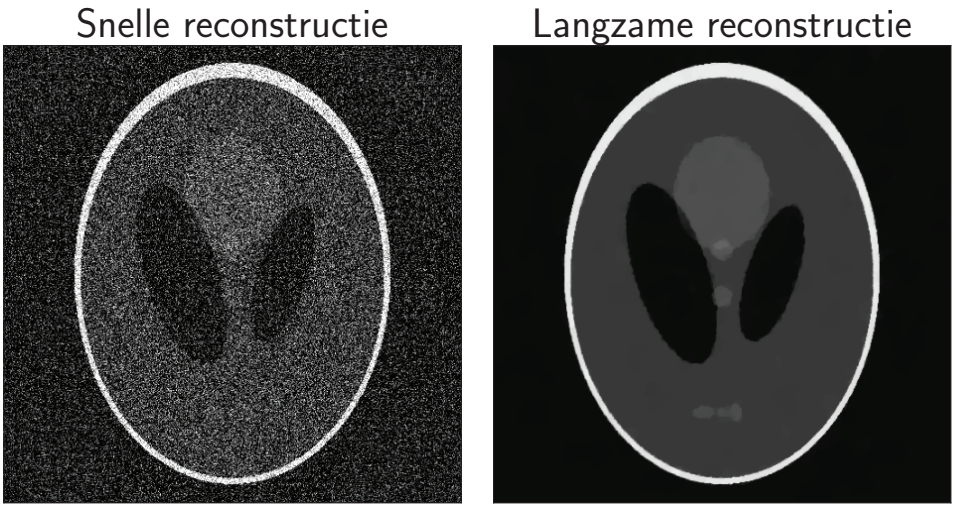


(b) Voorbeeld van een CT-meting



(c) Voorbeeld van een reconstructie.

Figuur 1: Tijdens een CT scan worden X-ray golven door een object gestuurd, in dit geval een appel. Achter het object staat een detector die meet hoeveel golven door het object zijn gekomen. Als er minder golven zijn gemeten, dan was 'meer object' aanwezig en wordt de meting donkerder. Een voorbeeld van zo'n meting zien we in figuur (b). Vervolgens draaien we het object en doen we nog een meting. Als het object een vol rondje heeft gedraaid dan is er voldoende informatie om een reconstructie te maken. Figuur (c) laat een 2D doorsnede zien van een 3D-reconstructie. Deze reconstructie laat een doorsnede zien van een appel zonder dat de appel is doorgesneden.



(a) Rekentijd: 0,09 seconden. (b) Rekentijd: 47,5 seconden.

Figuur 2: Twee verschillende reconstructiemethodes toegepast op computer gesimuleerde CT-metingen met veel ruis.

nu gezien worden op het moment dat ze ontstaan kun je ophouden met druk geven voordat het materiaal volledig is gebroken en kun je onderzoeken wat het effect is van andere veranderingen op deze scheuren, zoals bijvoorbeeld temperatuurverschillen. Een ander voorbeeld is het scannen van een patiënt. Als de dokter direct een reconstructie heeft van de ingewanden kan er ingezoomd worden als er niet voldoende detail is. Bovendien kan de dokter direct conclusies trekken uit de observaties en hoeft de patiënt niet onnodig lang te wachten op de uitslag van het onderzoek.

Om live reconstructies te kunnen berekenen tijdens de CT scan moeten twee processen real-time gedaan worden: de scan en de berekeningen. De standaard manier van CT scannen is te langzaam om live beelden te kunnen berekenen. Daarom moet het scanprocess versneld worden. Dit kan op twee manieren: 1) door minder metingen te doen of 2) door sneller een meting te doen. Ten eerste, minder metingen betekent minder informatie voor de berekeningen, en ten tweede, sneller meten betekent meer ruis in de metingen en dus minder precieze metingen. Dit levert een probleem op, want reconstructiemethodes die snel genoeg zijn om real-time uit te rekenen leveren slechte beelden op als de CT-metingen ruis of weinig informatie bevatten. En reconstructiemethodes die wel om kunnen gaan met dit soort CT-metingen kosten veel tijd om toe te passen. Een voorbeeld hiervan is gegeven in **figuur 2**. Hier laten we reconstructies zien

van computer gesimuleerde CT-metingen met veel ruis. Voor 2D-CT-reconstructies kunnen we snellere computers gebruiken en real-time reconstructies maken, maar voor 3D-CT-reconstructies is dit niet meer mogelijk. Dit betekent dat voor live 3D-CT-reconstructies een nieuwe reconstructiemethode moet worden bedacht die snelle berekeningen kan doen en alsnog accurate reconstructies kan maken van CT-metingen die veel ruis bevatten en weinig informatie. Dit hebben we onderzocht in **hoofdstuk 4** en **5**.

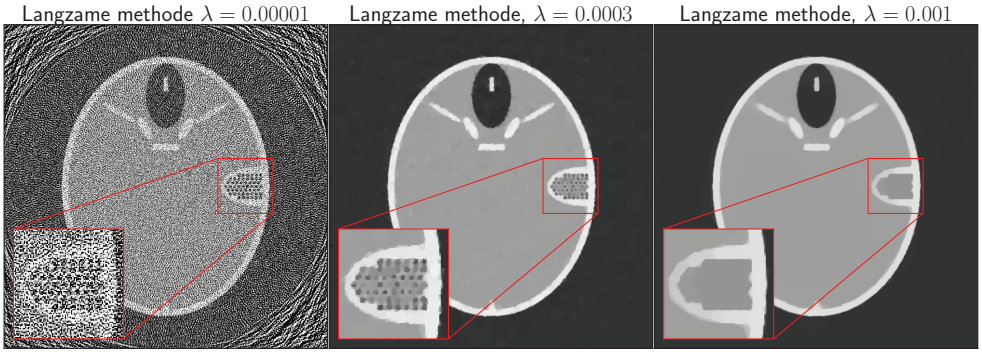
Het toepassen van reconstructiemethodes is niet altijd even makkelijk. In **hoofdstuk 2** en **3** hebben we nieuwe methodes ontwikkeld om bestaande reconstructiemethodes beter toepasbaar te maken.

Het automatisch correct toepassen van reconstructiemethodes

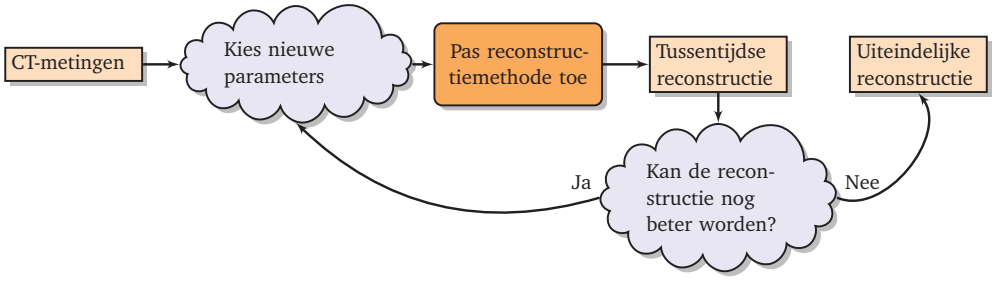
Een reconstructiemethode is vergelijkbaar met een kookrecept. Er zijn simpele recepten, waar zonder veel moeite iets lekkers uitkomt. Maar er zijn ook heel moeilijke recepten waar echt iets geweldigs uitkomt. De uitdaging met deze moeilijke recepten is dat als er iets misgaat de resultaten ook makkelijk minder goed kunnen worden. Zo ook met reconstructiemethodes.

Reconstructiemethodes hebben een set aan parameters[1], en de keuze van deze parameters beïnvloedt de kwaliteit van de reconstructie, zoals we kunnen zien in **figuur 3**. In dit figuur zien we drie verschillende reconstructies met de langzame reconstructiemethode van precies dezelfde CT metingen. Hier is het enige verschil tussen de drie reconstructies de parameterkeuze λ. Hoe je deze reconstructieparameters moet kiezen is niet altijd even duidelijk voordat je een reconstructie ziet met de gekozen parameter. Daarom komt het uitrekenen van een reconstructie in de praktijk vaak neer op een proces van 'trial-and-error', zoals beschreven in **figuur 4**. Daarom hebben we in **hoofdstuk 2** en **3** wiskundige methodes ontwikkeld om de keuze voor parameters te optimaliseren. In **hoofdstuk 2** ontwikkelen we een methode voor langzame reconstructiemethodes, zoals de methode die we eerder gezien hebben in **figuur 2**. Het toepassen van deze reconstructiemethodes duurt lang en elke keer dat je een suboptimale reconstructieparameter kiest moet je de reconstructiemethode nog een keer toepassen. Om de ontwikkelde methode toe te kunnen passen moet je een paar keer een reconstructie uitrekenen. Maar als dit gedaan is kun je vervolgens voor (bijna) alle parameterkeuzes direct een voorbeeldreconstructie genereren en is de parameterkeuze zo simpel als gewoon de beste reconstructie kiezen uit een verzameling van reconstructies.

[1] Parameters zijn waardes die van tevoren gekozen moeten worden om een methode te laten werken. Voorbeelden hiervan in de echte wereld zijn: de versnelling en zadelhoogte van een fiets, of de stand en temperatuur van een oven.



Figuur 3: Voorbeelden van reconstructies met de langzame reconstructiemethode van **figuur 2**. Het enige verschil tussen deze drie reconstructies is de parameterkeuze λ. Dit laat zien dat de parameterkeuze erg belangrijk is en ook voor problemen kan zorgen als deze niet goed genoeg is.



Figuur 4: Schematische weergave van het toepassen van een reconstructie methode. Als de mogelijke keuzes voor de set van reconstructieparameters klein is en het effect van de parameterkeuze duidelijk is, dan is de reconstructie methode makkelijk te gebruiken.

In **hoofdstuk 3** formuleren we een wiskundig model om automatisch de beste parameterkeuze te vinden voor de eerder besproken snelle reconstructiemethodes. Vergeleken met de langzame reconstructiemethodes kan je veel meer parameterkeuzes testen. Maar als je dat met de hand moet doen, dan kun je deze methode niet meer gebruiken in een geautomatiseerde omgeving. De parameters die we hier optimaliseren zijn vergelijkbaar met de parameters die in de machine learning-methode van **hoofdstuk 4** worden geleerd, maar in dit geval leren we alleen de parameters voor één snelle reconstructie methode en hebben we helemaal geen voorbeeld CT-metingen nodig. Daarmee is deze methode toepasbaar voor een

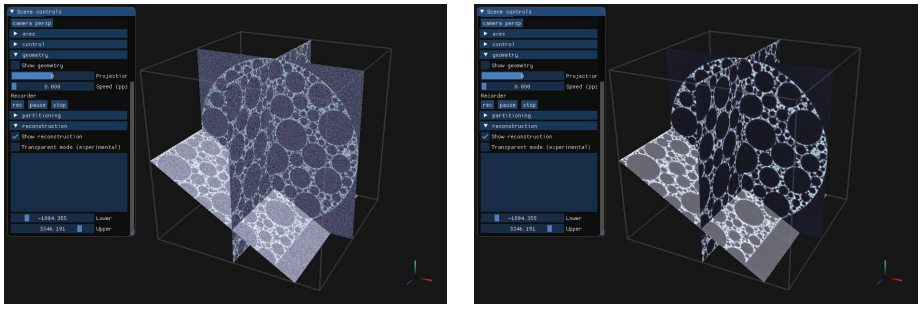
breed scala van CT-metingen.

Snelle en accurate reconstructiemethodes

In **hoofdstuk 4** ontwikkelen we een accurate en snelle reconstructiemethode voor snelle CT scans, dus voor CT scans die CT-metingen met weinig informatie en veel ruis opleveren. Het idee achter deze methode is dat we een snelle reconstructiemethode ontwikkelen door een bestaande snelle methode te combineren met een *machine learning*-methode. De machine learning-methode is in dit geval een wiskundig model dat vier snelle reconstructiemethodes combineert. In dit model is een deel van de parameters nog niet gekozen. Deze parameters worden vervolgens gekozen (of geleerd) door CT-metingen en bijbehorende accurate reconstructies te geven en de computer te laten bedenken welke parameters bij deze combinatie het beste passen. Als deze parameters geleerd zijn kunnen we het model toepassen op nieuwe CT-metingen die lijken op de metingen die we gebruikt hebben tijdens het leerproces. Ondanks dat we meer berekeningen aan het doen zijn, is deze methode nog steeds snel, omdat alle componenten een snel te evaluaren reconstructiemethode zijn. Doordat we de methode toepassen op CT-metingen die vergelijkbaar zijn met de voorbeelden die we hebben gebruikt tijdens het leerproces zijn de reconstructies niet alleen snel, maar ook accuraat.

Een uitdaging voor het gebruiken van deze machine learning-methode is het feit dat er CT-metingen beschikbaar moeten zijn voor het leerproces die vergelijkbaar zijn met de CT-metingen waar we het op toe willen passen en waar een accurate reconstructie voor is. Dit is moeilijk omdat voor sommige toepassingen het heel moeilijk is om een accurate reconstructie te krijgen, gezien we niet anders hebben dan snelle CT metingen met onvoldoende informatie.

Daarom combineren we in **hoofdstuk 5** de reconstructiemethode uit **hoofdstuk 4** met trainingsstrategie waarbij geen voorbeelden nodig zijn. Deze trainingsstrategie is gebaseerd op de observatie dat de ruis in één CT meting niet beïnvloed wordt door de ruis in een andere CT meting. Dit is vergelijkbaar met het feit dat twee dobbelsteenworpen elkaar niet beïnvloeden. Deze observatie kunnen we gebruiken om trainingsvoorbeelden te creëren vanuit de CT-metingen waar we de methode op willen toepassen. Dit betekent dat deze methode het meest effectief is voor CT-metingen met veel ruis. Bovendien laten we zien dat deze methode ook gebruikt kan worden om quasi-3D real-time reconstructies te maken. Met quasi-3D bedoelen we dat het geen volledige 3D-reconstructie is, maar 3 willekeurig te kiezen doorsneden van een 3D-reconstructie. We laten een voorbeeld van quasi-3D-reconstructies zien in **figuur 5**. Hier is links de originele methode voor deze strategie en rechts de methode ontwikkeld in **hoofdstuk 5**.



(a) Standaard snelle reconstructie (b) Nieuwe snelle reconstructie

Figuur 5: Real-time reconstructies gemaakt met de quasi-3D reconstructiestrategie. Het gescande object is een cilinder met bubbels erin. Hier rekenen we drie doorsneden van de 3D-reconstructie uit in plaats van de volledige 3D-reconstructie. Doordat er op elk moment kan worden besloten om een andere doorsnede weer te geven is het volledige 3D-volume real-time beschikbaar.

Conclusie en vervolg

Het onderzoek beschreven in deze thesis laat zien dat je CT reconstructiemethodes bruikbaarder kunt maken, en geschikt kunt maken voor een bredere doelgroep, door slim gebruik te maken van wiskundige modellen. Bovendien hebben we laten zien dat je bestaande reconstructiemethodes zodanig kan verbeteren dat je ze kan gebruiken om real-time CT reconstructies te maken. De volgende stap in het onderzoek zal zijn dat er gewerkt gaat worden om deze methodes toe te passen in de eerder beschreven voorbeelden. Dit zal, zoals het altijd gaat met nieuwe methodes toepassen in de praktijk, nieuwe uitdagingen, problemen en onderzoeksvragen opleveren.

Curriculum Vitae

Marinus Jan Lagerwerf was born in 1990 in Wageningen, The Netherlands, and completed his secondary education (VWO) in 2009 at the R.S.G. Pantarijn. In 2013, he received his undergraduate degree at the University of Twente in applied Mathematics. He obtained his MSc degree (cum laude) in Applied Mathematics from the University of Twente in 2015, with a focus on mathematical imaging. During his master's he visited the Cambridge Image Analysis group at Cambridge University as a research assistant. His master's thesis entitled "Higher order variational methods for photoacoustic tomography", was written under the supervision of Christoph Brune and Srirang Manohar. He started his PhD research at Leiden University under supervision of Joost Batenburg in 2016. The research was carried out at Centrum Wiskunde & Informatica (CWI) Amsterdam. He made research visits to University of Twente in Enschede, EMAT in Antwerpen and KTH in Stockholm. He attended international conferences, workshops, and summer schools in Bordeaux, Antwerpen, Obergurgl, Stockholm, Grenoble and Bologna.

Automatic and Efficient Tomographic Reconstruction Algorithms