Discrete Tomography

by

Jara Charlotte Aarts

to obtain the degree of Bachelor of Science

at the Delft University of Technology,

to be defended publicly on Tuesday June 18, 2024 at 11:00 AM.

Student number:	5362792	
Project duration:	April 19, 2024 – June 18, 2024	
Thesis committee:	dr. H.N. Kekkonen,	TU Delft, supervisor
	dr. C. Kraaikamp,	TU Delft

TU Delft

Layman's Abstract

Medical imaging is a technique to make an image of the interior of a body thereby revealing internal structures which would normally be hidden behind skin and bones. The goal here is to visualizing the function of organs and tissues to be able to diagnose diseases and detect deficiencies in a noninvasive way. A CT scan is one of the most used medical imaging tools and is a abbreviation of Computed Tomography. A CT scanner machine however does not directly provide an image. The machine makes measurements of the body and from these measurements an image needs to be produced which reconstructs the interior of the body. These reconstructions involve quite some mathematics and thus there exist multiple mathematical models to tackle this reconstruction problem. When using such a mathematical method one has to careful because the measurements of the body made by the CT scanner machine may contain noise, so to say additional information, and this noise can cause the image to be blurry. Various mathematical tools exist which improve the quality and accuracy of the reconstructed CT image. Using those tools, doctors are more able to make a correct diagnosis. This thesis explains how the reconstruction of a CT image comes to be and describes multiple of such mathematical tools. The thesis mainly focuses on one method in specific called DART by analyzing the reconstructions made with this method and comparing it to reconstructions made using other techniques.

Abstract

This thesis aims to introduce the reader to the mathematical principles underlying X-ray computed tomography. It begins with the derivation of both filtered and unfiltered back-projection reconstruction techniques. Following this, the thesis delves into two algorithms based on algebraic reconstruction techniques (ART): the Simultaneous Algebraic Reconstruction Technique (SART) and the Discrete Algebraic Reconstruction Technique (DART). This includes the derivation and implementation of both methods and the analysis and comparison of the DART algorithm to the aforementioned methods, all using simulated data. These ART methods approach the reconstruction problem by iteratively optimizing a large system of linear equations. In addition, DART leverages prior knowledge on grey values to steer the final reconstruction towards one that only contains these pre-determined grey values. Based on the experiments using simulated data it is shown that DART effectively deals with both errors in the approximation of the grey values as well as with noisy projection data. Furthermore, it is found that DART computes relatively accurate reconstructions using only a small number of projections or projections from a small angular range.

Contents

1

Introduction

In 1895, Wilhelm Röntgen discovered X-rays [11]. This groundbreaking discovery has forever changed the way physicians go about their work. This tool for medical imaging provides a look into the hidden structures of the physical world. Since 1895 the medical imaging field has experienced major developments, the emergence of X-ray computed tomography (CT) soon followed after the establishment of computers around the 1970's. This development brought medical imaging techniques to a new level. Where X-ray images are *planar* (two-dimensional), CT images are *cross-sectional* (three-dimensional). Although X-ray (computed) tomography is not entirely risk-free [6], it presented the first noninvasive medical imaging method. Previous to this discovery, patients had to be cut open for us to be able to see inside of the body which is invasive and harmful to the human body. We are now able to diagnose patients and do research into how the human body is composed without physically invading the body. Mathematical methods are used in reconstructing these medical images from X-ray projections. Allan Cormack was the first person to realise the importance of mathematics in the reconstruction of medical images from projections. He recognised that the reconstruction problem involved determining a function from its line integrals [13]. In 1972 Godfrey Hounsfield used the methods developed by Allan Cormack to compute the very first computed tomography (CT), together they received the Nobel Prize in Medicine in 1979 [11]. To this day this method still lays the foundation of medical imaging. In the method, Allan Cormack uses the *Radon transform* to model how X-rays propagate through an object and are detected by a sensor. Initially, the effect of the X-rays within the object was determined by using a method called *Back-Projection*, which attempts to reverse the projection process. Following this discovery, numerous other reconstruction techniques were developed, aiming to improve the reconstruction quality. Drawing from various branches of mathematics, making X-ray imaging methods interdisciplinary. For instance, the problem can be approached with a Bayesian inverse model. Using random variables, the reconstruction problem becomes a question of statistical inference. Alternatively, considering the discretized version of the Radon transform the problem can be formulated with matrix notation. This formulation allows for retrieving the solution using inversion, which may propose an unstable problem for which multiple regularization techniques exist. Additionally, the matrix notation of the discrete version can be seen as a system of linear equations, resulting in a more algebraic approach. All methods have their advantages, the Bayesian method allows for quantification of uncertainty in its solution but is more complex to implement than the other mentioned methods. This thesis will study a version of the algebraic approach, this method has the advantage of enabling the incorporation of prior knowledge.

The aim of this thesis is firstly to introduce the underlying principles of X-ray computed tomography. This includes the more physical aspects, like the ionization of atoms, as well as the Radon transform. The second objective is to reconstruct the internal structure of objects from measurement data using the Discrete Algebraic Reconstruction Technique (DART). DART incorporates a sub-algorithm for which several options are possible. This thesis, however, employs the simultaneous algebraic reconstruction technique (SART). Besides the DART and SART algorithm reconstruction using Filtered Back Projection is introduced to be able to compare the performance of the methods. The discussed methods are shown through reconstructions of

simulated data in MATLAB. The Shepp-Logan phantom, a widely-used test image that models a human head for which the resolution can be determined, is used as simulated data. Figure 1.1 shows two Shepp-Logan phantoms, one of smaller resolution (on the left) in which the image is made up of 50×50 pixels and one of larger resolution (on the right) where the image is made up of 200×200 pixels. An explanation of how these test images can be used as simulated data and how the reconstructions can be tested both visually and using multiple performance metrics is also included in this thesis.

This thesis adheres to the following structure. Chapter 2 provides an introduction to the physical properties related to X-rays. This introduction enables the derivation of the fundamental attenuation law. Leading to the Radon transform from which we are able to introduce Filtered Bac- Projection as the continuous case of X-ray computed tomography. Chapter 3 includes the more applicable case of X-ray computed tomography, discrete tomography described as an inverse problem. Discrete measurements are taken and pixels with constant values form the basis of the reconstruction. Furthermore, the chapter includes explanations on important concepts of inverse problems known as well-posedness and inverse crime. In Chapter 4 Algebraic Reconstruction Techniques are introduced and an explanation of SART is provided from which some reconstructions are shown. Thereafter, Chapter 5 contains a detailed explanation into the DART algorithm together with several reconstructions. Lastly, the main takeaways and discussion points can be found in Chapter 6.

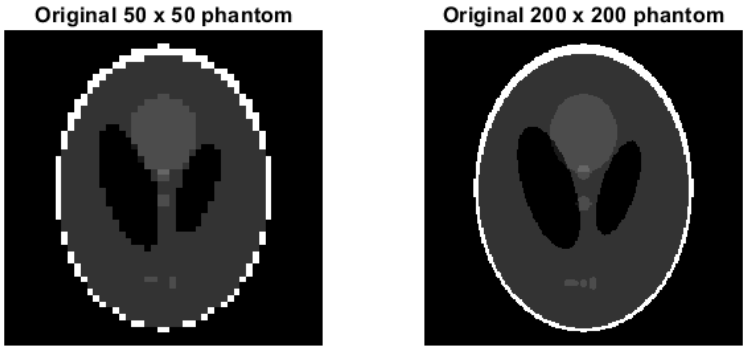


Figure 1.1: The original Shepp-Logan phantom of resolution 50×50 on the left and resolution 200×200 on the right.

2

X-rays

When studying X-ray computed tomography, it is essential to understand the basics of the underlying physical properties. This chapter provides an introduction to the physics behind the X-ray; Sections 2.1 and 2.2 will dive into the physical aspects of tomography. Most importantly, explaining ionization through the atomic structure. Eventually, we are able to get more into the mathematics of it, in Section 2.3 numerous attenuation laws will be derived and in Section 2.4 the Radon transform is introduced and related to the exponential attenuation law. Furthermore Section 2.4 contains an explanation on filtered and unfiltered back-projection. This chapter is based on a book by Price and Links [11] and a book by Mueller and Siltanen [9].

2.1. Ionization

A *tomogram* is an image of a plane, or 'slice', of the body. Therefore X-ray computed tomography generates an image of a slice within the body, representing the conveyance of an X-ray through the human body. Computing a tomogram relies on two separate components. Firstly a source, the 'X-ray tube', produces a beam of ionizing radiation which is then transmitted through the patient. Whilst passing through the patient the the intensity of the radiation is reduced, this phenomenon is called *attenuation*. The second component, the radiation detectors, thereafter collect the new intensity of the beams on the other side of the patient, this is depicted in Figure 2.1. Here the X-ray source is positioned at $x = 0$ and the X-ray detector is positioned at $x = x_d$. The amount in which the intensity is attenuated is determined by the attenuation ability, which is a physical characteristic of materials. The individual processes happening inside the source and detector are outside the scope of this paper, for those interested, further details can be found in [14]. What happens in between will however be explained briefly.

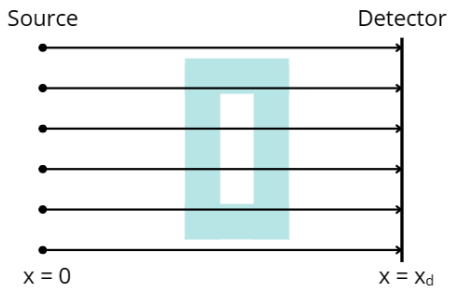


Figure 2.1: Visual representation of X-rays travelling from the X-ray source $x = 0$ through an object to the X-ray detector $x = x_d$.

Electromagnetic radiation

Where the amount of attenuation is determined by the characteristics of the material the radiation beam passes through, the existence of attenuation is caused by the beam's ionization ability. X-rays are electromagnetic radiation (EM), a type of radiation capable of ionization. That is, radiation capable of ejecting electrons from atoms creating a free electron and ion. A free electron is thus an electron that is not bound to an atom and an ion is an atom that has either lost or gained electrons. Other types of EM are UV rays, visible light and radio waves. Generally, electromagnetic radiation with a large enough energy level is considered ionizing, resulting in X-rays being ionizing but visible light not. Electromagnetic radiation can act as a particle and a wave since the phenomenon has no mass and no charge. When treated as a particle, EM radiation can be seen as "packets" of energy which are called *photons*. On the contrary, when considered to be a wave, the radiation can be characterised by its wavelength.

Atomic structure

The ejection of electrons depends on the atomic structure. An atom consists of a centre called the *nucleus* which is orbited by *electrons*, as can be seen in Figure 2.2. Since a stable atom is always electrically neutral and the nucleus has a positive charge, there must be as many electrons as there is a positive charge to balance the charge. These orbiting electrons are arranged in shells, where each shell can contain a growing amount of electrons as we move further from the nucleus. It is common for the electrons to be positioned in the shell as close to the nucleus as possible, since there the energy level is as low as possible. This guideline is called the *ground state*, it is nature's preferred arrangement of electrons over the shells. Another important characteristic of electrons is that it is energetically favourable to be bound to an atom rather than to be a free electron. Implying that the energy of a complete atom is smaller than the energy of the electron plus the energy of the atom without that electron. This difference in energies is called the *electron binding energy* and plays an important role in radiation. We denote the shell in which an electron is located with the orbit number. Note that electron binding energy decreases when the orbit number increases. Thus if an electron is positioned in a shell further away from the nucleus the electron binding energy is lower than when the electron is close to the nucleus. The electron binding energy plays an important role. When radiation passes through a certain material it also passes through the atoms within. More specifically, it passes energy to orbiting electrons in that atom. If this energy is greater or equal to the electron binding energy this electron gets ejected from the atom, this process is called *ionization*. The ejected electron is called the photo-electron. On the contrary, when the energy of the passing radiation is lower, the electron will move to a higher shell which will leave a 'hole' in the lower shell.

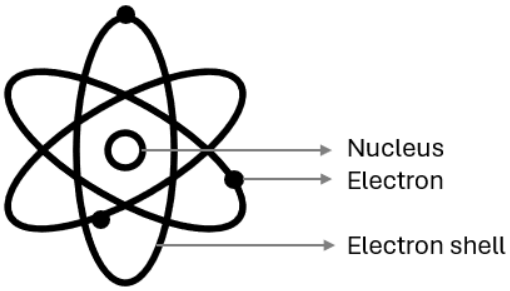


Figure 2.2: Visualization of atomic structure.

2.2. Effects

The electron binding energy, being the reason for an electron to be either ejected or moved, can cause two different effects, the *photoelectric effect* and the *Compton scatter*. As these effects both occur when an X-ray photon interacts with an atom the difference is rooted in the level of absorption.

Photoelectric effect

Firstly, in the photoelectric effect, the photon interacts with the atom, where the photon will be absorbed completely, causing the atom to eject an electron. This happens as the photon that is absorbed carries energy, as explained the energy level of a stable atom is always neutral and thus the energy level has to be restored. This is done by transferring the energy of the photon to an electron within the atom. The transferred energy helps to overcome the binding energy which holds the electron in the atom. Thus the electron is ejected from the atom, taking a part of the energy with it. When this interaction happens in one of the inner shells, it leaves the atom with a 'hole' which must be filled by an electron from an outer orbit, now the outer orbit experiences a loss in energy which in turn creates an EM photon called the *characteristic radiation*. This process is shown in Figure 2.3. It is called characteristic radiation since the energy of the photon equals the difference in electron binding energies of the shells in consideration. This energy level differs per atom and thus is a *characteristic* of particular atoms.

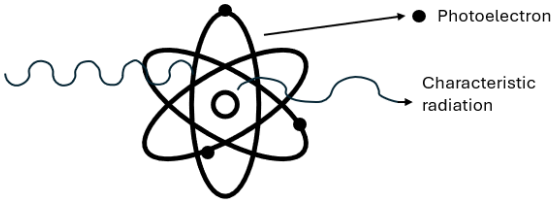


Figure 2.3: Visualization of the photoelectric effect.

Compton scatter

Secondly, in Compton scattering the photon interacts with an outer shell electron and is only partly absorbed creating a loss of energy and changing its direction. This direction change limits the resolution of X-ray images. The outer shell electron, now called the *Compton electron*, is in this case also ejected from the atom. Compton scattering is depicted in Figure 2.4.

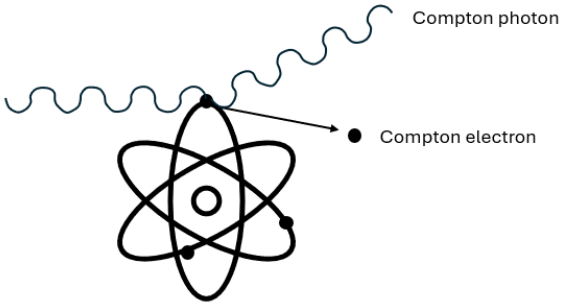


Figure 2.4: Visualization of Compton scatter.

2.3. Attenuation

As explained the Photoelectric effect and Compton scattering cause a loss in strength of the EM radiation beam, and as stated we call this *attenuation*. The difference in strength depends on the characteristics of the atom which the radiation passes through. To be able to model this, we define several characteristics of the X-ray beam. Firstly, we can measure the strength of an X-ray beam which is commonly referred to as *the photon fluence*.

Definition 2.1. *The* photon fluence Φ *is the number of photons N in a radiation beam spread over the area A*

that the beam covers and is therefore given by

$$\Phi = \frac{N}{A}.$$

A possibility is to take time into account as well since measurements take place over fixed time interval Δt, this gives

$$\Phi = \frac{N}{A\Delta t}.$$

Secondly, we define the *intensity I* of a beam by the photon fluence together with the energy these photons carry:

Definition 2.2. *The intensity of a radiation beam is given by*

$$I = E\Phi$$

where $E = h\nu$ with h being Plank's constant and ν being the frequency of the beam.

In reality, photon beams are poly-energetic, meaning that the photons may have differing energy levels. The poly-energetic case however is more complicated thus the mono-energetic, in which all N photons are of the same energy, will be considered.

Starting off with placing a homogeneous object, being of one material, between the photon source and the detector. We can determine the total change in the number of photons due to attenuation in an object using the following theorem

Theorem 2.1. *Let N_0 be the initial number of photons and let N_d describe the number of photons measured by the detector. Furthermore, let Δx be the distance an X-ray travels through the object. Assume that the attenuation coefficient, μ, is constant. Then the* fundamental photon attenuation law *is given by*

$$N_d = N_0 e^{-\mu\Delta x} \tag{2.1}$$

Proof. Let N be the total number of photons present in the beam and take N' to be the number of photons measured by the detector. Note that $N' \leq N$ since some photons will be absorbed due to the photoelectric effect and some photons may be deflected away from the detector due to Compton scattering. We expect that the amount of photons that are lost due to attenuation, $\Delta N = N - N'$, is proportional to N and Δx thus giving

$$\Delta N = -\mu N \Delta x.$$

Letting the object become thin and considering N to be a continuous quantity leads to

$$\frac{dN}{N} = -\mu dx$$

as the distance travelled by the X-ray through the object is also becoming smaller when the object becomes thinner. We are interested in the total change in the number of photons between the source, which is equal to the initial number of photons N_0, and the number of photons at the detector, which is assumed to be N_d, due to attenuation while travelling through an object for a distance of Δx. We find this by integration:

$$\begin{aligned}
\int_{N_0}^{N_d} \frac{dN}{N} &= -\int_0^{\Delta x} \mu dx \\
\frac{N_d}{N_0} &= e^{-\mu\Delta x} \\
N_d &= N_0 e^{-\mu\Delta x},
\end{aligned}$$

which concludes our proof. □

Similarly, let I_0 be the initial intensity, this can be determined through calibration. The calibration can be done by sending an X-ray through empty space and detecting the non-attenuated intensity. In the mono-energetic case equation (2.1) can be written in terms of intensity as the energy level is irrelevant.

$$I = I_0 e^{-\mu \Delta x} \tag{2.2}$$

From now on we will only consider the intensity as Equations (2.1) and (2.2) are similar when considering the mono-energetic case. Moving on to the non-homogeneous case, where the object consists of multiple materials. In that case, the attenuation coefficient depends on the position x within the object as well, therefore the following theorem applies.

Theorem 2.2. *Let $I(x)$ be the intensity of the beam at position x and let I_0 be the initial intensity. Take Δx to be the distance an X-ray travelled through the object to get to position x. Assume that the attenuation coefficient $\mu(x)$ depends on position x within the object. Then the* exponential attenuation law *is given by*

$$I(x) = I_0 exp\Big\{-\int_0^{\Delta x} \mu(x)dx\Big\}. \tag{2.3}$$

Proof. Following the proof of Theorem 2.1 we must solve

$$\frac{dI(x)}{I} = -\mu(x)dx.$$

We are interested in the total change in intensity between the source, which is equal to $I(0) = I_0$, and then the intensity at position x, which is assumed to be $I(x)$, due to attenuation while travelling through an object for a distance of Δx. We find this by integration:

$$\begin{aligned}
\int_{I_0}^{I(x)} \frac{dI(x)}{I} &= -\int_0^{\Delta x} \mu(x)dx \\
\frac{I(x)}{I_0} &= e^{-\int_0^{\Delta x} \mu(x)dx} \\
I(x) &= I_0 e^{-\int_0^{\Delta x} \mu(x)dx},
\end{aligned}$$

which concludes our proof. □

Equation (2.3) models the line integrals which can be interpreted to be the individual radiation beams shown in Figure 2.1. Let $I(x_d) = I_d$ to be the intensity at the position of the detector for which we write x_d. Let Δx_d denote the distance the X-ray travelled through the object to get to the detector. Note that since I_d and I_0 are known we now can determine the linear attenuation coefficient by the log difference in intensities:

$$-\int_0^{\Delta x_d} \mu(x)dx = log\Big(\frac{I_d}{I_0}\Big) = log(I_d) - log(I_0). \tag{2.4}$$

This equation forms the basis of X-ray computed tomography.

The above model is a simplification of the reality. Besides a photon beam being poly-energetic multiple other issues are being neglected. For instance, the used photon count is modelled to be deterministic whilst it is better modelled as a random variable. To take this into account without making it too complicated the measurement is often modelled as

$$-\int_0^{\Delta x_d} \mu(x)dx + \epsilon = log(I_d) - log(I_0).$$

with $\epsilon \sim N(0, \sigma^2)$ a Gaussian random variable. The noise amplitude σ^2 can be estimated by measuring the same object repeatedly and calculating the sample variance.

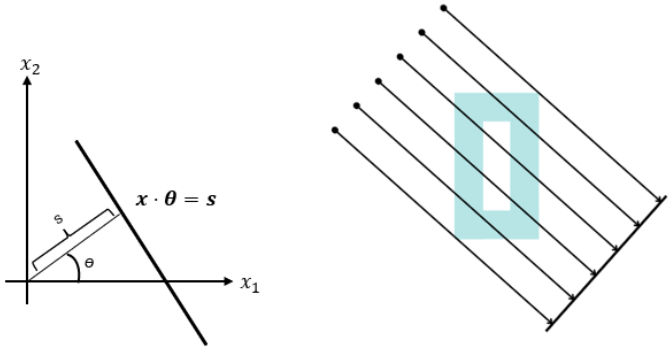


Figure 2.5: Illustration of line $L_{s,\theta}$ on the left and six parallel X-ray beams with fixed θ and differing s on the right.

2.4. Continuous tomography

Recall that we want to model the events happening during X-ray computed tomography. The exponential attenuation law provides a step in the right direction. However, to be able to reconstruct a traditional X-ray image considering one X-ray separately is not enough. A collection of X-rays is needed. Additionally, when we consider X-ray computed tomography we need collections of X-rays from different angles around the object. For this we consider lines

$$L_{s,\theta} = \{x \in \mathbb{R}^2 | x \cdot \Theta = s\}, s \in \mathbb{R}$$

which are illustrated in Figure 2.5. Here Θ is the unit vector with angle $\theta \in \mathbb{R}$ (in radians) with respect to the x_1-axis, denoted

$$\Theta = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}.$$

Note that line $L_{s,\theta}$ is perpendicular to $[s\cos\theta, s\sin\theta]^T$ trough the point $(s\cos\theta, s\sin\theta)$, thus we can write it as

$$L_{s,\theta} = \{x = (x_1(t), x_2(t)) = (s\cos\theta + t\sin\theta, s\sin\theta - t\cos\theta) \in \mathbb{R}^2 | t \in \mathbb{R}\}. \tag{2.5}$$

Modelling the difference in intensity along the line $L_{s,\theta}$ can be done using the *Radon transform*. In a two-dimensional space, the Radon transform is an integral transform mapping a function to line integrals [16].

Definition 2.3. *The Radon Transform of function $u(s,\theta) : \mathbb{R}^2 \to \mathbb{R}$ is defined as*

$$\mathscr{R}u(s,\theta) = \int_{x\cdot\theta=s} u(x)dx$$

with $s \in \mathbb{R}$ and $\theta \in [0, 2\pi]$.

Note that the Radon transform of the attenuation coefficient gives the same expression as the derivation we made by examining the physical properties of the intensity in Equation (2.4) when we take $\mathbf{x} = (x_1(t), x_2(t))$ to be the same as in Equation (2.5) and by noting that taking $\theta = \frac{\pi}{2}$ models an equation parallel to the x-axis:

$$\begin{aligned}
\mathscr{R}\mu(s,\theta) &= \int_{x\cdot\theta=s} \mu(x)dx \\
&= \int_{-\infty}^{\infty} \mu(x_1(t), x_2(t))dt \\
&= \int_{-\infty}^{\infty} \mu(scos\frac{\pi}{2} + tsin\frac{\pi}{2}, ssinn\frac{\pi}{2} - tcos\frac{\pi}{2})dt \\
&= \int_{-\infty}^{\infty} \mu(t,s)dt \\
&= \int_{0}^{\Delta x_d} \mu(x,y)dx \\
&= log\Big(\frac{I_d}{I_0}\Big).
\end{aligned}$$

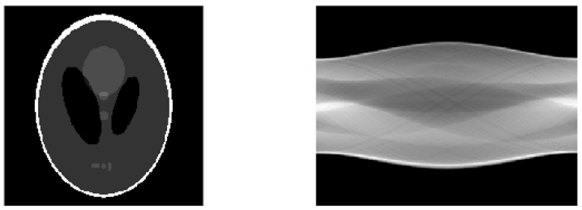


Figure 2.6: On the left the Shepp-Logan phantom at resolution 200×200 and on the right the measured data (sinogram) from 180 angles ranging from 0 to π.

Note that the radon transform models the CT measurements in which $\mu(x_1, x_2)$, being the attenuation coefficient, corresponds to the underlying unknown we wish to reconstruct. A pictorial representation of the radon transforms $\mathscr{R}\mu(s,\theta)$, where s and θ serve as rectilinear coordinates (forming a straight line), of $\mu(x_1, x_2)$ is a *sinogram*. This represents the data which is needed to reconstruct $\mu(x_1, x_2)$. Figure 3.1 shows an object, the Shepp-Logan phantom, and its sinogram, which is generated by making projections (180 to be exact) of the phantom with an angle range of 0 to π. We start with a projection at angle $\theta = 0$, this is a projection comprising vertical integrals of the object (to understand this the reader is referred to Figure 2.5). Note that in Figure 2.6 the phantom is the most narrow in the horizontal direction, therefore there will be the least data retrieved from vertical integrals and thus the sinogram, starting at the left is also the most narrow. As we move along our angle range, our sinogram becomes wider as our projections widen. Arriving at the angle of $\theta = \frac{\pi}{2}$, notice that the phantom is at its widest and thus giving us the widest projection which is evident in the middle of the sinogram. Approaching an angle of $\theta = \pi$ we return to a projection comprising almost vertical lines and there is no need to proceed as these projections are redundant.

2.4.1. Back-projection

As explained in Definition 2.3 , the radon transform models the CT measurements. We, however, want to reconstruct an image of attenuation coefficient μ given these measurements. During the reconstruction process the data from the radon transform, which is represented by the sinogram, is taken and processed so that it shows the cross-section which it came from. For this reconstruction process, we use the method of *back-projection* which will now be introduced.

Consider a projection for angle $\theta = \theta_0$ with radon transform $\mathscr{R}\mu(s,\theta_0)$. Generally, there are an infinite number of functions $\mu(x_1, x_2)$ that could result in this projection and thus it is not possible to retrieve $\mu(x_1, x_2)$ from a single projection. Speaking to the readers intuition, if $\mathscr{R}\mu(s,\theta_0)$ is large for $s = s_0$, then $\mu(x_1, x_2)$ must be large somewhere (or everywhere) along line L_{s_0,θ_0}. Creating an image with that property can be done by letting every point on that line equal $\mathscr{R}\mu(s_0,\theta_0)$. Now repeating this for all values of s will result in the *back-projection image* $b_{\theta_0} = \mathscr{R}\mu(s,\theta_0)$ which is shown in Figure 2.7 for $\theta_0 = \frac{pi}{2}$. When considering multiple angles the separate reconstructions will be placed on top of each other. The reconstruction with two angles, see Figure 2.8, already shows more of the shape of the original object but also contains a lot more colours, this phenomenon we will address as artefacts. Adding up the back-projection images of all the measured angles ranging from 0 to π by integral, gives us the *back-projection*, formally defined by

Definition 2.4. *Back-projection $\mathscr{B}$ of function $f = f(s,\theta)$ is*

$$\mathscr{B}f(x) = \int_0^\pi f(x_1\cos\theta + x_2\sin\theta, \Theta)\,d\theta$$

Applying this back-projection to the radon transform of the attenuation coefficient $\mu(x_1, x_2)$ we get

$$\mathscr{B}\mathscr{R}\mu(x_1, x_2) = \int_0^\pi \mathscr{R}\mu(x_1\cos\theta + x_2\sin\theta, \Theta)\,d\theta = \int_0^\pi \mathscr{R}\mu(s, \Theta)\,d\theta.$$

When applying the back-projection method to (noise-free) measurements retrieved from the Shepp-Logan phantom it can be noted that the reconstruction, using 180 angles as shown in Figure 2.10, is still very blurry. In fact, no matter how much angles will be used this method will always give a blurry reconstruction. This is

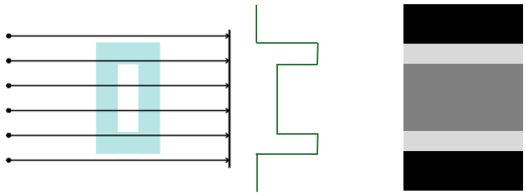


Figure 2.7: Reconstruction using back-projection for one angle.

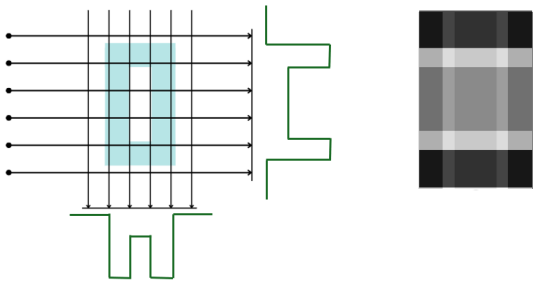


Figure 2.8: Reconstruction using back-projection for two different angles.

caused by letting every point on the projection line equal the measurement for one angle. Thus if two points lie on a line from one measurement these points will be given the same value during reconstruction whilst one point may lie inside of the object and the other outside of the object.

2.4.2. Filtered Back-projection

As the reconstructions using back-projection are still quite blurry we will introduce a second method, *filtered back-projection*. Filtered back-projection uses Fourier transforms to suppress certain frequencies and amplify others during the back-projection giving a less blurry result. In order to define the filtered back-projection method we need to concepts, that of the Fourier transform and the central slice theorem which will now both be defined.

Definition 2.5. *Let $u(x) : \mathbb{R}^n \to \mathbb{R}$ be an integrable function. The* Fourier transform *of u, denoted $\mathscr{F}u$, is defined as*

$$\mathscr{F}_n u(\xi) = \hat{u}(\xi) = \int_{\mathbb{R}^n} u(x) e^{-i2\pi x\cdot\xi} dx$$

If u and $\hat{u} \in L^1(\mathbb{R})$, then the inverse Fourier transform is defined by

$$\mathscr{F}_n^{-1} u(x) = \int_{\mathbb{R}^n} u(\xi) e^{i2\pi x\cdot\xi} d\xi.$$

Additionally, we need to define a relation between the one dimensional Fourier transform of the radon transform and the two dimensional Fourier transform of the object.

Theorem 2.3. *(Central slice theorem). Let u be integrable on $\mathbb{R}^2$. For all $s \in \mathbb{R}$ and $\theta \in [0, 2\pi]$ we find that*

$$\mathscr{F}_2 u(r\cos\theta, r\sin\theta) = \mathscr{F}_1 \mathscr{R} u(r, \Theta),$$

where $\mathscr{F}_1$ is the Fourier transform with respect to the first parameter.

Proof. Note that

$$\mathscr{F}_2 u(r\cos\theta, r\sin\theta) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} u(x_1, x_2) e^{-i2\pi(x_1,x_2)\cdot(r\cos\theta, r\sin\theta)} dx_1 dx_2$$

Using the following change of variables

$$x = (x_1, x_2) = (s\cos\theta - t\sin\theta, s\sin\theta + t\cos\theta).$$

We get the following determinant of the Jacobian

$$det\begin{bmatrix} \frac{\partial x_1}{\partial t} & \frac{\partial x_1}{\partial s} \\ \frac{\partial x_2}{\partial t} & \frac{\partial x_2}{\partial s} \end{bmatrix} = det\begin{bmatrix} -\sin\theta & \cos\theta \\ \cos\theta & \sin\theta \end{bmatrix} = -1,$$

thus $dx_1 dx_2 = dtds$. Note that $x_1 r\cos\theta + x_2 r\sin\theta = rs$ and thereby write

$$\begin{aligned}\mathscr{F}_2 u(r\cos\theta, r\sin\theta) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} u(s\cos\theta - t\sin\theta, s\sin\theta + t\cos\theta)e^{-i2\pi rs}dtds \\ &= \int_{-\infty}^{\infty}\left(\int_{-\infty}^{\infty} u(s\cos\theta - t\sin\theta, s\sin\theta + t\cos\theta)dt\right)e^{-i2\pi rs}ds \\ &= \int_{-\infty}^{\infty}\mathscr{R}u(s,\Theta)e^{-i2\pi rs}ds \\ &= \mathscr{F}_1\mathscr{R}u(r,\Theta),\end{aligned}$$

Where Fourier transform $\mathscr{F}_1$ is taken with respect to the first parameter. □

We are now able to propose filtered back-projection with which we can recover $\mu(x_1, x_2)$ from it's radon transform

Theorem 2.4. *Let u and û be integrable functions. Then*

$$u(x) = \int_0^{\pi}\int_{-\infty}^{\infty} e^{i2\pi r x\cdot\Theta}\mathscr{F}_1\mathscr{R}u(r,\Theta)|r|drd\Theta.$$

Proof. Note that

$$u(x) = \mathscr{F}_2^{-1}\mathscr{F}_2 u(x) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\mathscr{F}_2 u(\xi_1,\xi_2)e^{i2\pi(x_1,x_2)\cdot(\xi_1,\xi_2)}d\xi_1 d\xi_2.$$

Do change of variables $(\xi_1, \xi_2) = (r\cos\theta, r\sin\theta)$ to be able to use the central slice theorem. Here we have $r \in \mathbb{R}$ and $\theta \in [0, \pi]$. We get the following determinant of the Jacobian

$$det\begin{bmatrix}\frac{\partial\xi_1}{\partial r} & \frac{\partial\xi_1}{\partial\theta} \\ \frac{\partial\xi_2}{\partial r} & \frac{\partial\xi_2}{\partial\theta}\end{bmatrix} = det\begin{bmatrix}\cos\theta & -r\sin\theta \\ \sin\theta & r\cos\theta\end{bmatrix} = r,$$

thus giving $d\xi_1 d\xi_2 = |r|drd\theta$. Now using the central limit theorem and the above change of variables we can write

$$\begin{aligned}u(x) &= \int_0^{\pi}\int_{-\infty}^{\infty}\mathscr{F}_2 u(r\cos\theta, r\sin\theta)e^{i2\pi(x_1,x_2)\cdot(r\cos\theta, r\sin\theta)}|r|drd\theta \\ &= \int_0^{\pi}\int_{-\infty}^{\infty}\mathscr{F}_1\mathscr{R}u(r,\Theta)e^{i2\pi r(x_1,x_2)\cdot(\cos\theta,\sin\theta)}|r|drd\theta.\end{aligned}$$

□

Recall that the radon transform of μ is given by the measurement and thus we can use above theorem to recover the attenuation coefficient from noiseless full-angle X-ray measurements. Note in the above theorem first the Fourier transform is calculated of the radon transform where after we calculate the inverse Fourier transform of this multiplied with filter $|r|$, this gives the ability to filter out low-frequency components and amplifying high-frequency components. Then we conclude the filtered back-projection process by taking the back-projection of the filtered Fourier transform. Figure 2.10 also shows a reconstruction using filtered back-projection of noise-free measurements. The noise free measurements, depicted in a noise free sinogram, can be seen in the left of Figure 2.9. Using this sinogram gives a perfect reconstruction. However, when 5% noise is added to the measurements, we see that the sinogram, shown on the right of Figure 2.9, is pretty noise and the reconstruction is also noisy.

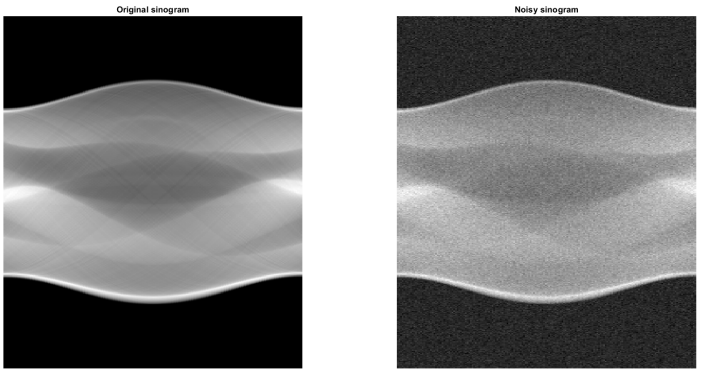


Figure 2.9: On the left the ideal measurement, so a noise free sinogram made from the original Shepp-Logan phantom as shown in Figure 2.10 using 180 projections ranging from 0 to π. On the right the same sinogram but with 5% noise.

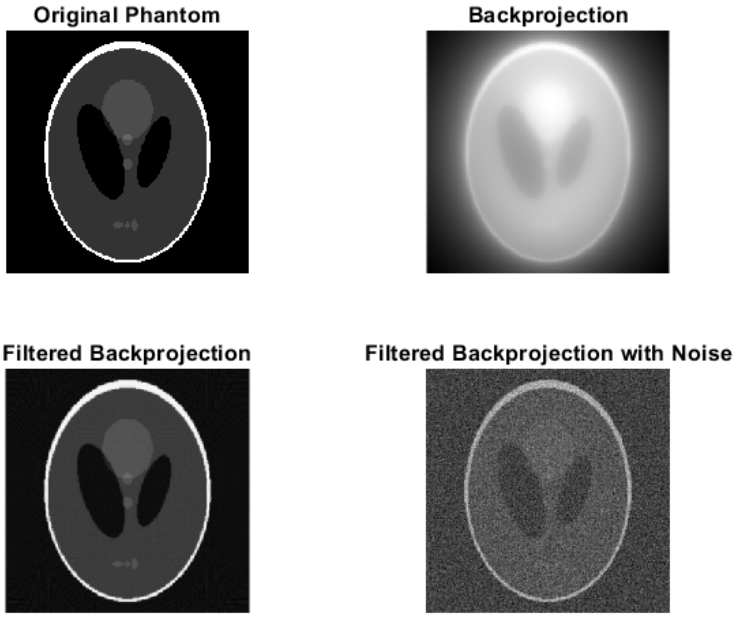


Figure 2.10: Original Shepp-Logan phantom compared to the unfiltered back-projection and filtered back-projection without and with 5% noise

3

Discrete tomography

Filtered back-projection models a solution for a continuous problem, implying that we have continuous data. In reality, the data is generated via a finite collection of lines as shown in Figure 2.5. In this chapter, X-ray computed tomography will be derived as a discrete inverse problem.

3.1. Inverse Problems

In a *direct problem* the cause is known and we want to know the effect. An *inverse problem* is the opposite of the direct problem, we know what the effect is but what was the cause? In X-ray computed tomography the direct problem is to determine a projection image, which was defined in Section 2.4 as a sinogram, when we know the internal structure of the body precisely. The corresponding inverse problem is to determine how the internal structure of the body looks like when all we are given are projection images. Recall that these projection images are the measurements made by the detector when X-ray beams have been sent through the patient and the attenuated intensity leaves the body. Figure 3.1 shows a Shepp-Logan phantom together with its noiseless sinogram determined by making 180 projections of the phantom with angle range $[0, \pi]$.

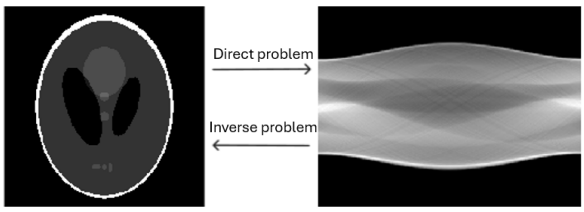


Figure 3.1: The Shepp-Logan phantom at resolution 200×200 and the measured data (sinogram) from 180 angles.

Having defined the concepts of a direct and an inverse problem, we need to delve into the criteria that determine whether a proposed problem, either direct or inverse, is tractable. More specifically, we distinguish between *well-posed* and *ill-posed* problems to define the conditions under which a solution to the given problem can be reliably obtained and utilized. The direct problem is well-posed; generally numerically stable, can be solved reliably and has a unique solution. A problem is well-posed if it complies with the following conditions as defined by Jacques Hadamard in 1923

- **H1 Existence**: there should be at least one solution.
- **H2 Uniqueness**: there should be at most one solution.
- **H3 Stability**: the solution must depend continuously on data.

A problem is ill-posed if it breaks at least one of the above conditions. In contrary to the direct problem, the inverse problem is generally ill-posed, thus more difficult to solve than the direct problem and very sensitive to noise and errors. Thus we need to ensure that we model the inverse problem such that it is well-posed. For this, we first define a basis model for the discrete inverse problem and then we will look at how this formulation can break well-posedness.

3.2. Discretization

We model the direct problem as follows. The measurements, which are made by the X-ray detector, are stacked in a vector such that we get $m \in \mathbb{R}^k$. Here k depends on the measurement device. In the case of X-ray computed tomography, k is determined by how many measurements are made by the X-ray detector. Then, the unknown in the case of X-ray computed tomography is μ, the attenuation coefficient, which is also stacked in a vector. The unknown μ is originally modelled as a piece-wise continuous function but to be able to model this as a discrete problem we need to introduce a finite-dimensional approximation of $\mu \in \mathbb{R}^D$. Naturally, μ does not have one ideal discretization, we have to choose the discretization method and level d. In Figure 3.2 multiple discretization levels are shown of the Shepp-Logan phantom, here the value for D expresses in how many pixels the projection image is divided in total. The discretization level d gives the number of pixels in which the image is divided in terms of the height and width of the image, therefore $D = d^2$. The bigger d, the more details the reconstruction will show but also the more computing power is needed to make that reconstruction. We want to estimate μ to be able to recreate the internal structure of the patient. For this, we need the *forward operator* A. The forward operator maps the unknown object μ (cause) to the measurement m (effect) which is perturbed by noise n. A is modelled as a matrix $A \in \mathbb{R}^{k\times D}$ such that it agrees with the formatting of m and μ. With that, we are able to describe the discrete inverse problem in the following manner

$$m = A\mu + n. \tag{3.1}$$

The direct problem of determining m, is well posed if A is well-defined, single-valued and continuous [9]. The inverse problem, determining μ, is then ill-posed when A^{-1} does not exist or is not continuous. Recall that $m \in \mathbb{R}^k$, $\mu \in \mathbb{R}^D$ and $A \in \mathbb{R}^{k\times D}$, therefore we are able to write Equation (3.1) in the following manner

$$\begin{bmatrix} m_1 \\ \vdots \\ m_k \end{bmatrix} = \begin{bmatrix} a_{11} & \dots & a_{1D} \\ \vdots & \ddots & \vdots \\ a_{k1} & \dots & a_{kD} \end{bmatrix} \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_D \end{bmatrix} + \begin{bmatrix} n_1 \\ \vdots \\ n_k \end{bmatrix}. \tag{3.2}$$

The inverse problem, recover $\mu \in \mathbb{R}^D$ from the measurement $m = A\mu + n$, can fail to fulfil the well-posedness conditions as follows

- **H1** Assume that $D < k$. Then we have more measurements than unknowns, which makes the system *overdetermined*. Which means that when noise is present there is no solution to solve (3.1). For X-ray computed tomography this would result in pixels that are left 'empty' where $A^{-1}m$ is not defined. How these empty pixels look depends on the initialization of the problem as we will see in Chapter 4, most of the time it means that the pixels are simply black.

- **H2** Assume that $D > k$. Then there are more unknowns than measurements and the system is *underdetermined*. Now there are several solutions possible. For X-ray computed tomography this means that the measurements can belong to several attenuation coefficients, thus we do not know which material was measured. For X-ray computed tomography, this implies that the values of the reconstruction can be off, as there are multiple possibilities. This causes the reconstruction to be off.

- **H3** Assume that $D = k$ and there exists A^{-1}. The problem may be extremely sensitive to errors in the measurement, producing a reconstruction dominated by noise. This is the case when the *condition number* $\kappa = \frac{\lambda_1}{\lambda_k}$ is very large. Here λ_1 and $\lambda_k > 0$ are the biggest and smallest singular values of A. What these singular values are and why a big condition number results in an unstable solution will be explained in Section 3.3.

In Section 3.3 we will also see that the last condition will affect the X-ray computed tomography reconstruction the most. With Equation (3.2), we have modelled X-ray computed tomography in a discrete manner.

Figure 3.2: Shepp-Logan phantoms with discretizations 25x25, 50x50, 100x100 and 200x200.

However, we have not established yet the precise meaning of the entries of the separate components m, μ and A. Thus, we will now adapt the tomography model as established in Sections 2.3 and 2.4 to fit (3.2). Firstly, we will model our measurements in a discrete manner. Let $\theta \in [0,\pi]$ be sampled with equidistant steps

$$\theta_j = \theta_1 + (\frac{j-1}{J-1})\pi, \qquad 1 \leq j \leq J$$

with θ_1 being a reference angle and J being the total number of steps taken, so the amount of different measurement angles. Let s be sampled with equidistant steps over interval $[-S,S]$ where $S > 0$ gives

$$s_n = -S + 2(\frac{n-1}{N-1})S, \qquad 1 \leq n \leq N$$

with N is the total number of steps. Thus for J angles, we measure from N different distances to the origin, as in Figure 2.5, and this gives us $k = JN$ measurements in total. Therefore we can write our measurements taken during X-ray computed tomography using Definition 2.3 as follows

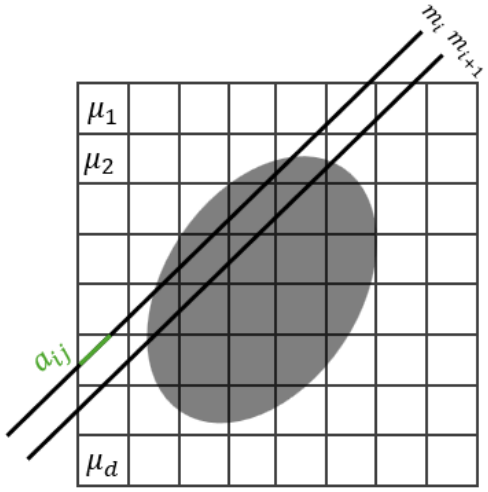
$$m = \begin{bmatrix} \int_{L_1} \mu(x_1(t), x_2(t))\,dt \\ \vdots \\ \int_{L_k} \mu(x_1(t), x_2(t))\,dt \end{bmatrix}.$$

Secondly, we model the unknown by dividing the projection image into D pixels, where we can set the attenuation coefficient constant in each pixel. In the continuous case, we have established that the log difference of the intensity equals the summation over the attenuation coefficients times an infinitely small distance which resulted in the integral. For the discrete case we will now do the same, we however leave the distance to be determined by the pixels instead of letting them become very small. This leaves us with a summation over all distance intervals of the attenuation times the distance we call this the *ray-sum*. An estimation of one noiseless measurement m_i, the value measured by the detector along a radiation beam (thus line integral L_i), relates to the entries of one row in matrix A, which denotes the distance that beam travels through a pixel. Multiply that row by the attenuation coefficient values and we get

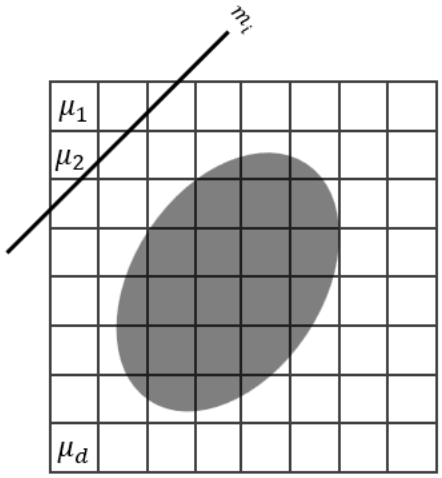
$$m_i = \int_{L_i} \mu(x)\,dx \approx \sum_{j=1}^{D} a_{ij}\mu_j \tag{3.3}$$

with a_{ij} being the distance beam L_i travels through pixel j, see Figure 3.3a.

Note that a_{ij} are the entries of the forward operator A with $i = 1,...,k$, $j = 1,...,d$. For many pixels $a_{ij} = 0$, this makes matrix A a sparse matrix. When a beam travels through an object it only passes through a certain area. The attenuation ability of the beam has an impact on the pixels which it travels through and the intensity of the pixels that are not crossed by this beam are left unchanged. To be more precise, considering Figure 3.3b, beam i only crosses 5 pixels, therefore measurement m_i is only affected by the attenuation coefficients of those 5 pixels. The entries a_{ij} of A corresponding to those 5 pixels have a non-zero value whilst all the other a_{ij} are equal to zero making sure that in Equation (3.3) the attenuation coefficients of all the other pixels do not contribute to measurement m_i. With Equation (3.3) and taking noise into account again we are back to the expression in Equation (3.2).



(a) Square grid imposed over the unknown image with assigned cell values μ_i and parallel measurements m_i and m_{i+1}. Highlighted value a_{ij} denotes the distance beam m_i travels through pixel j.



(b) Visualization of beam i travelling through only 5 pixels, ensuring matrix A to be a sparse matrix.

3.3. Reconstruction methods

Now that we have modelled X-ray computed tomography in a discrete manner we can start with solving the inverse problem, reconstructing the measured object. This can be done in several ways, first of all, we will look at the naive reconstruction method which is simple and intuitive but has big problems.

Assume that matrix A is a square and invertible matrix. When looking at our model $m = A\mu + n$, we can simply rewrite it such that we solve for the unknown

$$\begin{aligned} m &= A\mu + n \\ A^{-1}m &= \mu + A^{-1}n \\ \mu &= A^{-1}m - A^{-1}n. \end{aligned} \tag{3.4}$$

Considering the above equation, we see that the inverse of A is also applied to the noise, thus even if the noise $\|n\|$ is small it could still be that $\|A^{-1}n\|$ is large. This results in an unstable solution, breaking Hamard condition H3 and making the reconstruction inaccurate.
We will first test this method with ideal data. Let $n = 0$, data which does not contain any noise. In Figure 3.4 left you can see the naive reconstruction gives a perfect reconstruction. This perfect reconstruction is sadly too good to be true as this result is obtained with *inverse crime*. Inverse crime occurs with simulated data when the exact same model is used to generate the measurements and to invert it. In reality, there can the discrepancies and uncertainties in the model. Also, the detector, when making measurements, can introduce noise. This noise can be caused by slight variations in sensitivity within the detector or it can be caused by environmental effects like temperature dependence or mechanical vibrations. These factors can all impact the quality of the reconstruction. When working with simulated data we can choose to perturb the data such that it models these variations and includes noise present in real-life data. By doing this we avoid inverse crime and create reconstructions which better reflect the performance of a method in real-life applications. To avoid inverse crime we will generate the data (sinogram) using a 200×200 phantom, then interpolate the data to the 55×55 resolution used in the reconstruction. Additionally, we introduce some noise with the measurements, this will be be explained more thoroughly in Section 4.2.1.

However, when matrix A is not square, when the rows or columns of A are not linearly dependent or when A has eigenvalue zero we can not invert it. Instead, we can search for an approximation of the unknown μ_{approx} such that the distance between $A\mu_{approx}$ and m is as small as possible. This more general solution is the *least squares naive solution*

$$\underset{\mu}{\operatorname{argmin}} \|A\mu - m\|_2^2$$

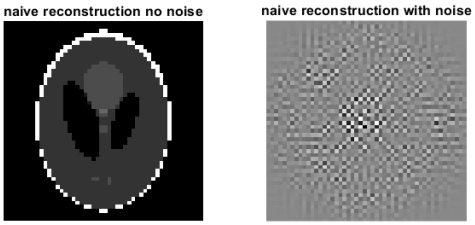


Figure 3.4: Naive reconstruction applied to noise-free data and the naive reconstruction applied to data with 0.1% noise.

which is solved by

$$\mu_{recon} = (A^T A)^{-1} A^T m$$

with A^T being the transpose of A. Note that when we have more measurements than unknowns, the problem is overdetermined and we break Hamard condition H2. Therefore when noise is present this method does not have a solution. Furthermore, this method has some other problems as minimizing over the distance potentially yields in multiple solutions, as multiple attenuation coefficients μ could result in the same distance between $A\mu_{approx}$ and m, which breaks well-posedness condition H2. Also, when there is noise present we will create the same situation as in Equation (3.4) where the multiplication of A^{-1} with n could result in an unstable solution, breaking Hamard condition H3

The above-stated problems can be solved in multiple manners. Firstly, the *Truncated singular value decomposition (TSVD)* is a method in which instead of A^{-1} a truncated version of the pseudo inverse of matrix A is used to ensure stability. This pseudo inverse is obtained by factorizing the matrix A into multiple matrices using *singular value decomposition*. One of the matrices from this singular value decomposition is a diagonal matrix consisting of fractions within the denominator the square roots of the eigenvalues of $A^T A$, we call these the *singular values*. Since the singular values are in the denominator, very small singular values will result in a reconstruction dominated by the noise term $A^{-1}n$. By truncation the smallest singular values from this diagonal and setting them to zero we ensure stability. Another method to avoid instability is using *Tikhonov regularisation* or *Total variation regularisation* which are both methods that add a certain penalty term to the least squares problem to address the instability. Further information into these three methods can be read in a book by Mueller and Siltanen [9].

4

Algebraic Reconstruction Techniques

In Chapter 3 we have defined the discrete model for X-ray computed tomography. This discrete approach is more simplistic than the continuous approach with transform-based methods as in Section 2.4. On the other hand, it can therefore also lack in speed and accuracy. The continuous transform based solutions are however not always possible, for example in situations where it is not possible to measure a large number of projections. One method that does work in the case of a limited amount of projections is the algebraic reconstruction technique. Other methods, like total variation regularisation as shortly discussed in Section 3.3 also perform quite well in the case of limited projection data, the advantage here however is that the matrix inversion operation becomes an iterative operation therefore it is more easily implemented when the matrix is not square. In this chapter, a specific version of the algebraic reconstruction technique, called SART, is discussed. SART attempts to improve the accuracy of algebraic techniques. First, in Section 4.1, we lay the basis of algebraic reconstruction techniques. Thereafter, in Section 4.2 this theory is used to define the improved version of this basis technique. The contents of this chapter are based on the book by Slaney and Kak [15] and the article by Andersen and Kak [1].

4.1. Projection representation

As in Section 3.2 we divide the unknown, the image which we want to reconstruct, into pixels. More specific, we cover the image with a square grid and we assume that the unknown, the attenuation coefficient μ in the case of X-ray computed tomography, is constant within each pixel cell. Thus we denote the constant value of pixel, or cell, j with μ_j. Recall that D is the total number of pixels, meaning that $\mu \in \mathbb{R}^D$. We again look at the representation in Figure 3.3a, however now keep X-ray computed tomography in mind. The fat blue line in Figure 4.1 represents a radiation beam. Where a_{ij} in Figure 3.3a represented the distance beam i travels through pixel j we now look at w_{ij} which represents the contribution of pixel j to ray integral i. This factor w_{ij} is equal to the area of pixel j covered by ray i. Like the a_{ij}'s most w_{ij}'s are zero. The interpretation of w_{ij}, being to consider radiation as a fat beam, may come closer to reality as radiation beam is not exactly a 'line' as the interpretation of a_{ij} suggests. However, a detector is only able to detect a discrete amount of values, spread over a few millimeters for each detector value and a beam has a width of only a few militarise [12]. Therefore we choose to use the notation and interpretation of a_{ij} to keep a more simplistic model, and thus we consider Figure 3.3a with the following equation again

$$m_i = \sum_{j=1}^{D} a_{ij}\mu_j, \qquad i = 1, ..., k. \tag{4.1}$$

Equation (4.1) provides us with k different equations as $i = 1, ..., k$, with D unknowns in each equation giving us D degrees of freedom. To be able to better explain this we will look at an example in which we consider a

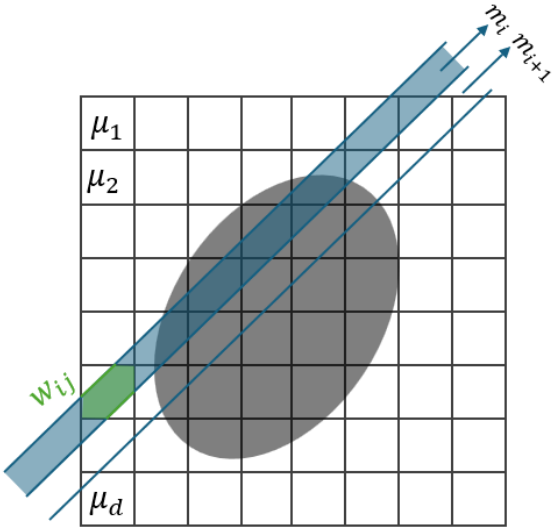


Figure 4.1: Square grid imposed over the unknown image with assigned cell values μ_i and parallel measurements m_i and m_{i+1}. Highlighted value w_{ij} denotes the contribution of pixel j to ray integral i.

two-dimensional case using equations

$$m_1 = \vec{a}_1\vec{\mu} = a_{11}\mu_1 + a_{12}\mu_2 \tag{4.3}$$
$$m_2 = \vec{a}_2\vec{\mu} = a_{21}\mu_1 + a_{22}\mu_2, \tag{4.4}$$

this case is visualized in Figure 4.2. We consider $D = 2$ pixels, and thus we have 2 unknowns representing the pixel values denoted μ_1 and μ_2. Additionally, we have two measurements m_1, m_2 and therefore values $a_{11}, a_{12}, a_{21}, a_{22}$ define the forward operator. Note that for instance a_{12} denotes the contribution that beam 1 has to cell 2 as visualized in Figure 4.2. Now these two equations, with two unknowns, result in two degrees of freedom and therefore we can visualize the equations from (4.2) as in Figure 4.4 with a μ_1-axis and a μ_2-axis. When we consider the more general case, where we have D unknowns we get an image covered by a grid of D cells (pixels) and thus the image will be defined by it's definite pixel values $(\mu_1, ..., \mu_D)$. The definite pixel values can be considered to represent a single point in a D-dimensional space. Just like (μ_1, μ_2) with fixed values denotes one point in the 2-dimensional space. Only now we can not visualize the problem anymore as there are D axes, and thus each equation $m_i = \sum_{j=1}^{D} a_{ij}\mu_j$ represents a hyperplane. Figure 4.3 shows for example two equations in a 3-dimensional space, note that since there are 3 degrees of freedom the equations represent a plane in stead of a line. When system of Equations (4.1) has a unique solution, the intersection of all the hyperplanes is a single point providing that unique solution. Note that Figure 4.3 does not have a unique solution as there are only two planes in the 3-dimensional space. The problem is underdetermined and thus the intersection of the equations results in a line, so many solutions, instead of one point. In Figure 4.4 there is however a unique solution and it is located at the intersection, denoted S. The process of finding such a unique solution will described using the 2-dimensional case.

The solution to Equations (4.2) can be found by making an initial guess $\vec{\mu}^{(0)}$, most often this initial guess is taken to be the zero vector. The initial guess is projected onto the first line in our system of equations m_1, see Figure 4.4(1). This results in one point on the first line $\vec{\mu}^{(1)}$ where after this point is again projected onto the second line, resulting in a new guess $\vec{\mu}^{(2)}$ shown in Figure 4.4(3). The solution on the second line is back projected on the first line and so on. If a unique solution exists, the iterations will always converge to that point. This procedure is show in Figure 4.4, where the unique solution is the intersection of the two lines denoted S. Note that initial guess in the D-dimensional case is denoted as $\vec{\mu}^{(0)} = (\mu_1^{(0)}, ..., \mu_D^{(0)})$. This method of solving a system of equations is known as the *Kaczmarz method*.

Theorem 4.1. *Take $\vec{a}_i$ to be the i^{th} row vector of forward operator A, let m_i be the measurement of the i^{th} beam.*

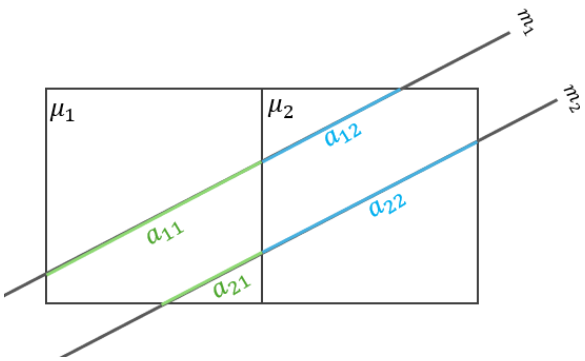


Figure 4.2: Visualization of 2-dimensional case.

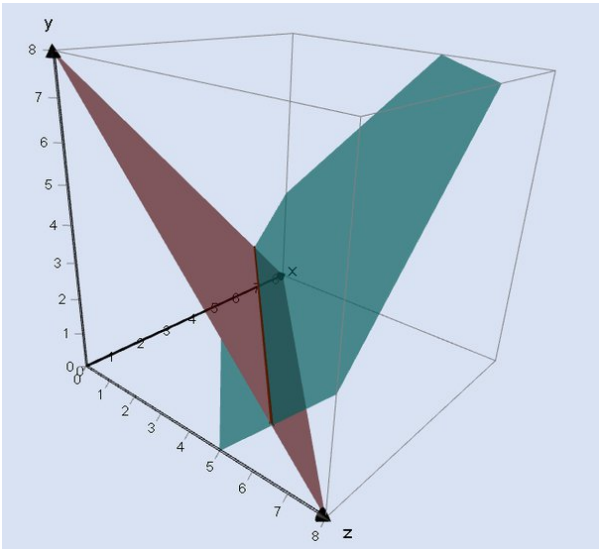


Figure 4.3: Enter Caption, `https://www.quora.com/What-is-the-intersection-of-two-planes-called`

Take $\vec{\mu}^{(j)}$ to be the projection of the (j-1)th guess on the ith measurement line, then this is calculated by

$$\vec{\mu}^{(j)} = \vec{\mu}^{(j-1)} - \frac{(\vec{\mu}^{(j-1)} \cdot \vec{a}_i - m_i)}{\vec{a}_i \cdot \vec{a}_i} \vec{a}_i. \tag{4.5}$$

Proof. For clarification we use the 2-dimensional case such that we can easily link it to the visualization in Figure 4.4. The proof of the D-dimensional case is however the exact same except for the indices.

Let $\vec{\mu}^{(0)}$ be the initial guess. When considering Figure 4.5, we want to project $\vec{\mu}^{(0)}$ onto m_1 to get a better guess $\vec{\mu}^{(1)}$. Also, note that our new guess is our current guess minus the projection on line $m_1 = \vec{a}_1\vec{\mu}$, thus we have

$$\vec{\mu}^{(j)} = \vec{\mu}^{(j-1)} - projection.$$

The projection, depicted as the line from the initial guess $\vec{\mu}^{(0)}$, denoted C for simplicity, to D in Figure 4.5, will therefore be derived by only considering the first equation in (4.2), $m_1 = \vec{a}_1\vec{\mu}$. We set initial guess $\vec{\mu}^{(0)}$. To find an expression for line segment CD we need to determine the direction and the length of this line segment. The direction of CD can be determined using vector $\vec{a}_1$ as $\vec{a}_1$ is perpendicular to the equation $m_1 = \vec{a}_1\vec{\mu}$. We normalize $\vec{a}_1$ to determine the direction of CD,

$$\frac{\vec{a}_1}{\sqrt{\vec{a}_1 \cdot \vec{a}_1}}. \tag{4.6}$$

Now note that the length of line segment CD is equal to

$$|CD| = |B| - |A|$$

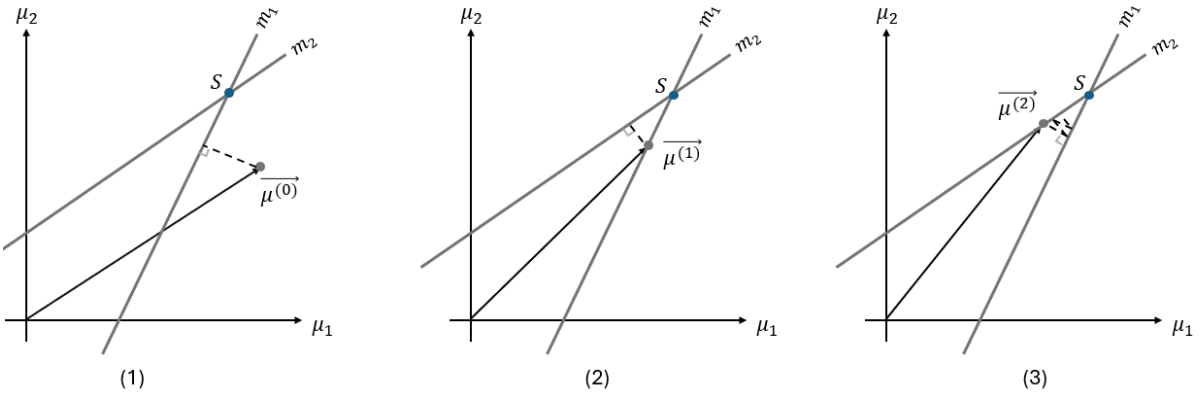


Figure 4.4: Visualization for iteratively solving algebraic equations, starting with initial guess $\vec{\mu}^{(0)}$ projected on the line corresponding to the first equation. The resulting point is projected onto the line corresponding to the second equation. With only two equations present, the projections will be continued back and forth between the two lines as illustrated. Intersection S denotes the unique solution to the problem as proposed in Equations (4.2).

where $|A|$ is the perpendicular distance from the origin to the line $m_1 = \vec{a}_1\vec{\mu}$ which is given by

$$|A| = \frac{\vec{a}_1}{\sqrt{\vec{a}_1 \cdot \vec{a}_1}}\vec{\mu} = \frac{m_1}{\sqrt{\vec{a}_1 \cdot \vec{a}_1}}$$

and note that the perpendicular distance from the origin to $\vec{\mu}^{(0)}$ denotes $|B|$ with

$$|B| = \frac{\vec{a}_1}{\sqrt{\vec{a}_1 \cdot \vec{a}_1}}\vec{\mu}^{(0)}.$$

Combining this we get

$$\begin{aligned} |CD| &= |B| - |A| \\ &= \frac{\vec{a}_1}{\sqrt{\vec{a}_1 \cdot \vec{a}_1}} \cdot \vec{\mu}^{(0)} - \frac{m_1}{\sqrt{\vec{a}_1 \cdot \vec{a}_1}} \\ &= \frac{\vec{a}_1 \cdot \vec{\mu}^{(0)} - m_1}{\sqrt{\vec{a}_1 \cdot \vec{a}_1}}. \end{aligned} \tag{4.7}$$

Now the only things that is left is combining Equation (4.6) and (4.7)

$$\begin{aligned} CD &= \frac{\vec{a}_1}{\sqrt{\vec{a}_1 \cdot \vec{a}_1}} \frac{\vec{a}_1 \cdot \vec{\mu}^{(0)} - m_1}{\sqrt{\vec{a}_1 \cdot \vec{a}_1}} \\ &= \frac{\vec{a}_1 \cdot \vec{\mu}^{(0)} - m_1}{\vec{a}_1 \cdot \vec{a}_1}\vec{a}_1. \end{aligned}$$

Therefore the orthogonal projection of $\vec{\mu}^{(0)}$ onto m_1 is given by

$$\vec{\mu}^{(1)} = \vec{\mu}^{(0)} - \frac{\vec{a}_1 \cdot \vec{\mu}^{(1)} - m_1}{\vec{a}_1 \cdot \vec{a}_1}\vec{a}_1$$

which can be generalised to

$$\vec{\mu}^{(j)} = \vec{\mu}^{(j-1)} - \frac{\vec{a}_i \cdot \vec{\mu}^{(j)} - m_i}{\vec{a}_i \cdot \vec{a}_i}\vec{a}_i.$$

This concludes our proof. □

Using the Kaczmarz method in the algebraic reconstruction technique follows the steps as described in the example using Figure 4.4 and can be described with the following pseudo-code

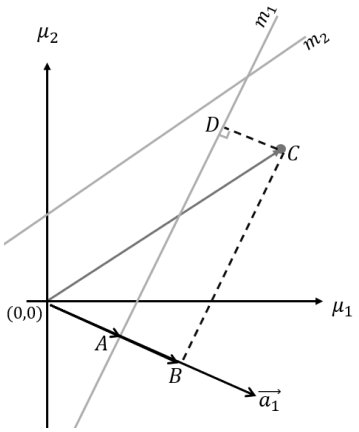


Figure 4.5: Visualization derivation Equation (4.5) using line $\vec{a}_1 \cdot \vec{\mu} = m_1$.

Algorithm 1 The SART algorithm

Set initial guess $\vec{\mu}^{(0)}$
for m = 1 to selected maximum iteration **do**
  for i = 1 to D **do**
    $j = (m-1)k + i$
    $\vec{\mu}^{(j)} = \vec{\mu}^{(j-1)} - \frac{\vec{a}_i \cdot \vec{\mu}^{(j)} - m_i}{\vec{a}_i \cdot \vec{a}_i} \vec{a}_i$
  end
end

Note that in one iteration every measurement line will be considered once, therefore the current reconstruction denoted j depends on in which iteration we are and which measurement we are considering. The amount of iterations for which this algorithm is run needs to be predetermined. Tanabe [17] has proven that if there exists a unique solution $\vec{\mu}_s$ to the system of equations in (4.1), then

$$\lim_{n\to\infty} \vec{\mu}^{(nk)} = \vec{\mu}_s. \tag{4.8}$$

This convergence of the current reconstruction to the exact solution takes a lot of iterations. However, this convergence to the exact solution is not necessarily needed as a not fully converged reconstruction can still be close enough to the wanted solution [18]. To ensure a fast convergence to a reconstruction of acceptable quality one can consider to make sure that the hyperplanes are perpendicular to each other. In Figure 4.6 note that when the hyperplanes have a very small angle between them it takes much more iterations to reach the solution than when there is a large angle between the hyperplanes. In theory perpendicular hyperplanes would be best for convergence, orthogonal hyperplanes could be ensured using the Gramm-Schmidt procedure. This however is computationally not feasible. Another method would be to ensure pairwise orthogonal hyperplanes. Easier however is to choose the order of considered angles wisely. The hyperplanes represent ray integrals and therefore it is likely that adjacent rays will be almost parallel. Thus by choosing hyperplanes which represent a more separated range of ray integrals the convergence is also enhanced.

We will now consider the Hadamard conditions for well-posedness regarding the algebraic reconstruction technique. When the system is overdetermined and there is noise present in the measurement there is no solution as there is not one intersection of all hyperplanes as can be seen in Figure 4.7. Note that the guesses $\vec{\mu}^{(j)}$ can not converge to one solution and therefore will oscillate around the intersections instead. When the system is underdetermined there are multiple solutions possible, like in Figure 4.3. In that case Tanabe [17] proved that the iterative process converges to a solution, denoted $\vec{\mu}_s$. This $\vec{\mu}_s$ is the point on the line which is closest to the initial guess which minimizes $|\vec{\mu}^{(0)} - \vec{\mu}_s|$. Thus in the case of the initial guess being close to line m_i, $\vec{\mu}_s$ will be the point on line m_i such that $\vec{\mu}_s$ is minimized. Note that in the 2-dimensional example at the beginning of this section that would mean that the projection of our initial guess, so $\vec{\mu}^{(1)}$ would result in the best reconstruction. However, it could be that our initial guess is closest to another line than m_1, say $m_i (i \neq 1)$

, and then the best reconstruction would not be given by $\vec{\mu}^{(1)}$. Thus when we let the amount of iterations go to infinity as in Equation (4.8) we would get solution $\vec{\mu}_s$ minimizing $|\vec{\mu}^{(0)} - \vec{\mu}_s|$ however as we can not compute infinitely many iterations the given solution may not be the best one.

This iterative method, besides being computationally efficient, is attractive since it is possible to incorporate prior knowledge about the object. μ can for instance be set to zero outside a certain range of values, as we know that if we consider grey values they should range from 0 to 255. A problem which arises is the

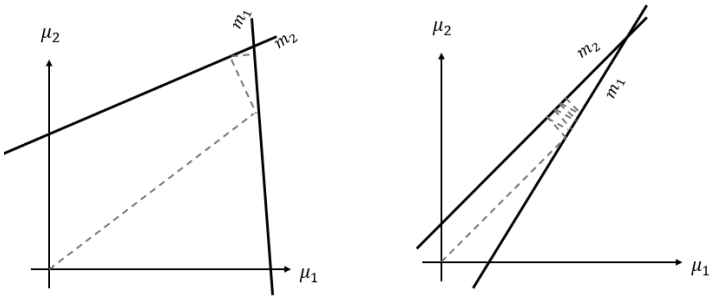


Figure 4.6: Illustration slow versus fast convergence depending on the angle between projections.

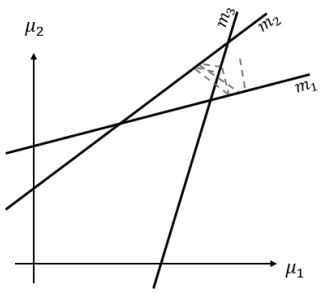


Figure 4.7: Illustration overdetermined system resulting in non-unique solution.

amount of storage needed for all the different weights a_{ij}. Therefore it can be chosen to replace the a_{ij}'s by 1's and 0's, depending on if the center of the j^{th} cell is within the i^{th} ray. This approximation of the weights introduces an error which causes *salt and pepper noise*. The amount of noise is worsened due to the iterative nature of the method. An initial guess is made and with each iteration the goal is to adjust this guess such that it minimizes the distance from the current guess to the measured projections m_i. The measured projections however contain noise, and each iteration will try to fit to this noise. Over multiple iterations the small errors in the initial data m_i and a_{ij}'s can accumulate and become more problematic. This effect can be reduced by introducing *relaxation*. There the cell value is updated by the preceding guess minus only a fraction of the projection instead on the whole projection

$$\vec{\mu}^{(j)} = \vec{\mu}^{(j-1)} - \lambda \frac{\vec{a}_i \cdot \vec{\mu}^{(j)} - m_i}{\vec{a}_i \cdot \vec{a}_i} \vec{a}_i.$$

4.2. Simultaneous Algebraic Reconstruction Technique

In the above explained method one projection at a time is processed. To even further reduce the noise discussed in the above section all projections can be processed simultaneously. This approach is a variation on the algebraic approach explained in Section 4.1 called *Simultaneous Algebraic Reconstruction Technique* or *SART* in short. The simultaneous approach is incorporated by replacing Equation (4.5) with the following equation

$$\vec{\mu}^{(j)} = \vec{\mu}^{(j-1)} - \lambda \sum_{i=1}^{k} \frac{(\vec{\mu}^{(j-1)} \cdot \vec{a}_i - m_i)}{\vec{a}_i \cdot \vec{a}_i} \vec{a}_i. \tag{4.9}$$

Besides this, SART also differs from the above described basic algebraic reconstruction technique as it uses a longitudinal *Hamming window*. Applying a Hamming window means that we apply additional weights during the reconstruction process as follows

$$\vec{\mu}^{(j)} = \vec{\mu}^{(j-1)} - \sum \frac{(\vec{\mu}^{(j-1)} \cdot \vec{a}_i - m_i \cdot c(i))}{\vec{a}_i \cdot \vec{a}_i} \vec{a}_i. \tag{4.10}$$

This additional weight $c(i)$ emphasizes the correction terms near the middle of the ray relative to the correction terms applied at its ends. Weights $c(i)$ are defined by the following equation

$$c(i) = 0.54 - 0.46 \cos\left(2\pi \frac{i}{k}\right).$$

The Hamming window is used because we are dealing with circular reconstruction regions, we take caution in weighing the correction terms when a ray enters the circle and when it exits the circle again as is showed in Figure 4.8. Lastly, SART uses bilinear interpolation. Bilinear interpolation is a method of interpolating within a 2-dimensional grid. First linear interpolation in performed in one direction and then in the other direction. The linear interpolation is performed over directly neighbouring pixels (thus over a 4-connected neighbourhood as will be explained in Section 5.2.2). This ensures that the values of neighbouring pixels are considered in calculating the pixel value as well. Therefore creating smooth transitions between different different coloured areas. This also enhances the capability to handle noise as the noisy pixel values can be smoothed out by the interpolation. For more detail into how the hamming window weights are calculated or how the bilinear interpolation is applied during the algorithm the reader is referred to the book by A. Kak [15].

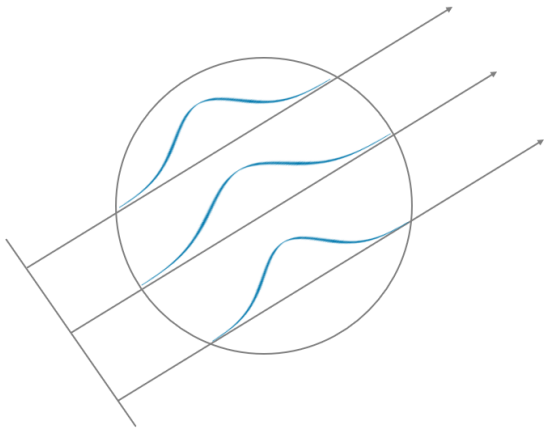


Figure 4.8: Hamming window, depicted as blue lines, for a set of three straight parallel rays over a round object

4.2.1. Implementation of SART

We will look at multiple reconstructions of the Shepp-Logan phantom using the SART method. This implementation is made in MATLAB using the ASTRA Toolbox [2], the code can be found in Appendix A.

We first consider the most idealistic case where we have access to sufficient data, a SART reconstruction in which we have parallel projector beams which range from 0 to π radians in 180 different angles with no noise present. When we create a sinogram with this amount of measurement the ASTRA Toolbox depends the amount of beams used on the amount of detector values, thus the amount of unknowns. Therefore, when we consider a Shepp-logan phantom of resolution 50×50, 50 beams will be used to create the measurements. The measurements are made with inverse crime. The projections are considered in order $[0:5:175, 1:5:176, 2:5:177, 3:5:178, 4:5:179]$, this makes sure we start with the angle at 0 radians and take steps of 5 until we get to the angle at 175 radians and then go back to the start again until we have considered all angles. Thus creating the following sequence of angles $[0, 5, ..., 175, 1, 5, ..., 176, ..., 179]$. By doing this

we make sure that we do not consider adjacent rays as they will almost be parallel and that delays convergence as is explained in Section 4.1. The reconstruction of this idealist case is shown in Figure 4.9 and 1800 iterations are used here.

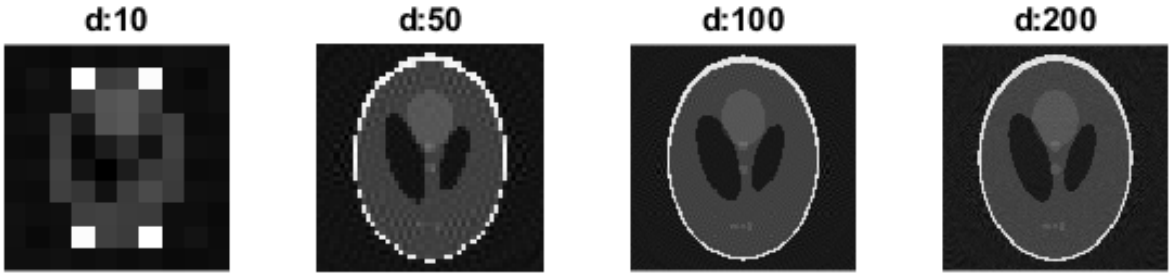


Figure 4.9: SART reconstruction of the Shepp-Logan phantom without noise for discretization levels 10×10, 50×50, 100×100 and 200×200. 180 different angles are considered for projection rays considered in order $[0:5:175, 1:5:176, 2:5:177, 3:5:178, 4:5:179]$

Next this simplistic reconstruction will be compared to reconstructions with differing parameters only looking at the case where $d = 100$. Multiple performance metrics are used, the *Structural Similarity Index* (SSIM) measures the similarity between two images in which it primarily focuses on differences in structural information like luminance and contrast [8]. In contrast to this, the *Relative Error* (RE) is a measure for the difference between the true grey value of the reconstruction and the original image. Lastly the *Pixel Error* (PE) quantifies the total number of pixels from the reconstruction that differs fro the original image. Note in Figure 4.10 image (a) is the same as the $d : 100$ case in Figure 4.9, the settings for this reconstruction will be used as basis and for the other reconstructions parameters will be changed or added to show the effect. Since these images are created with a discretization of $d = 100$ there are $10,000$ pixels in total per image. Firstly, in reconstruction (b), we added a Gaussian noise level of 0.1% to the measurement which is present in all other reconstructions as well. Now note in reconstruction (c) that a white haze is present. Here a linear projection order is chosen, thus the angle sequence $[0, 1, ..., 179]$ is used in the reconstruction. As explained before this works less good than the projection order used in reconstruction (b) since adjacent angles are almost parallel resulting in the reconstruction guess oscillating around the actual value in stead of converging towards it. When looking at the performance metrics they all agree with this theory, the relative error and pixel error are higher than the previous reconstructions and the similarity index is lower. However, the difference is not alarmingly big, this can be explained as we used many iterations thus ensuring convergence to a reconstruction of good quality. Furthermore, in reconstruction (d) only 45 projection angles are used in stead of 180, causing a lot of background noise and the projection lines to be clearly visible. In reconstruction (e) prior knowledge is used by putting the constraint on the values to be non-negative. As the value 0 corresponds to black all negative values are clipped to zero and thus set to black. This reduces the amount of background noise, note that the pixel error is more than halved in comparison to reconstruction (b) for which all the other settings are the same. Lastly when using less iterations, in this case 50, a lot more background noise is present but also the phantom itself is less clear The three small dots in the lower portion of the phantom are almost invisible and the boundaries are less clear. Overall looking at the performance measures reconstruction (e) is the best and reconstruction (f) is the worst, which visually it indeed is.

Note that in the reconstructions in Figure 4.10 inverse crime is present. Next we will reconstruct the image again using differing parameter settings but now we will avoid inverse crime. This will be done by first generating the Shepp-Logan phantom using a higher resolution, this means using a larger value for the dimension d of the phantom grid. A high resolution phantom ensures that all the features of the phantom are respresented in detail. Thereafter, using this high resolution phantom we will generate the measurements, thus the sinogram. Now comes the difference, to simulate a more realistic scenario and to avoid invers crime we interpolate the high resolution sinogram to a lower resolution. We reduce the resolution of the sinogram data which reflects the loss of detail that occurs in real world data retrieval. Now with this interpolated sino-

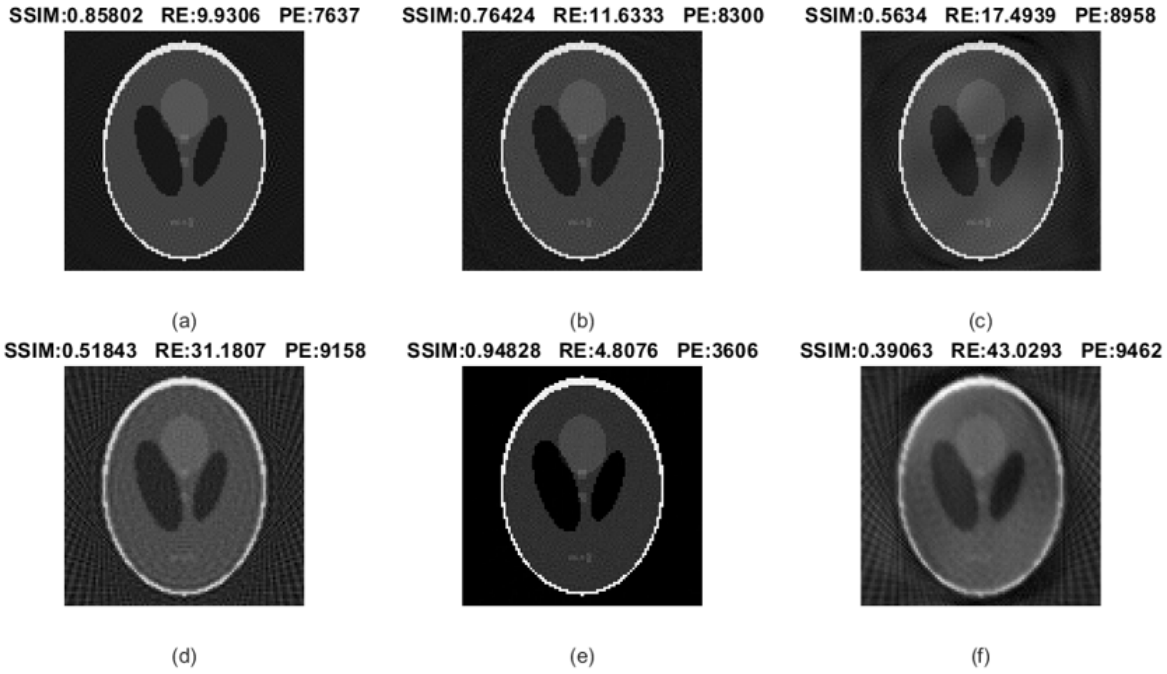


Figure 4.10: Reconstructions with inverse crime (a) simple reconstruction, (b) reconstruction with a Gaussian noise level of 0.1%, (c) reconstruction with projection order $[0, 1, ..., 179]$ and 0.1% of Gaussian noise, (d) reconstruction with 45 projection angles and 0.1% of Gaussian noise, (e) reconstruction with non-negativity constraint to the pixel values and 0.1% of Gaussian noise, (f) reconstruction with 50 iterations.

gram we perform the reconstruction. Additionally, adding randomized noise, so Gaussian distributed noise for instance, also helps to avoid inverse crime. Both steps ensure that the reconstruction process is not able to unrealistically benefit from knowing under which the data has been generated creating a more realistic reconstruction.

In Figure 4.11 the reconstructions are shown, the same parameter settings are used for each reconstruction as in Figure 4.10. In reconstruction (c) a white haze is still present and reconstructions (d) and (f) still show some lines, however all reconstructions look more alike now. This is caused as when avoiding inverse crime different discretizations are used for the data generation and the reconstruction, ensuring that the reconstruction process reflects complexities and inaccuracies of real-world scenario's better. Thus the reconstructions are more blurry and noisy but more realistic. Even though the reconstructions look more alike, the performance measures still show that reconstructions (c), (d) and (f) are not favourable.

Note that some reconstructions from Figure 4.11 are better than it's corresponding reconstruction from Figure 4.10. This improvement can be due to the fact that when using different discretization levels and methods for reconstruction than with which we generated the data the resulting discrepancies force the algorithm to handle interpolation errors and inaccuracies. This results in a reconstruction that is better able to handle noise. This improvement is due to the model containing inverse crime being able to perfectly fit the data. The same methods and levels for for instance discretization are used for both data generation as reconstruction, thus the model knows exactly how to use the measurements. However when we create measurements avoiding inverse crime the algorithm can not perfectly fit the data and this may result in a more generalized and smoothed result. The specific reconstructions that have a lower relative error in Figure 4.11 compared to Figure 4.10 are (d) where we used a limited amount of angles to create the measurements and (f) where we used less iterations. These specific cases can also be explained. In case (d) with a limited amount of angles, so limited data, the reconstruction from Figure 4.10 shows artifacts in the background and more noise. Because the data is incomplete the model with inverse crime might overfit on this limited data, trying to reconstruct exactly what was measured and only that. Whilst the reconstruction without inverse crime is less prone to overfitting the data as it simply can not match the data perfectly. This leads to a more stable result in which noise can be better dealt with. Then for the reconstructions (f) where fewer iterations were used, when committing inverse crime the model may not able to converge properly. However when avoiding inverse crime,

the algorithm is focussed on the main features of the data potentially leading to better performance with limited iterations.

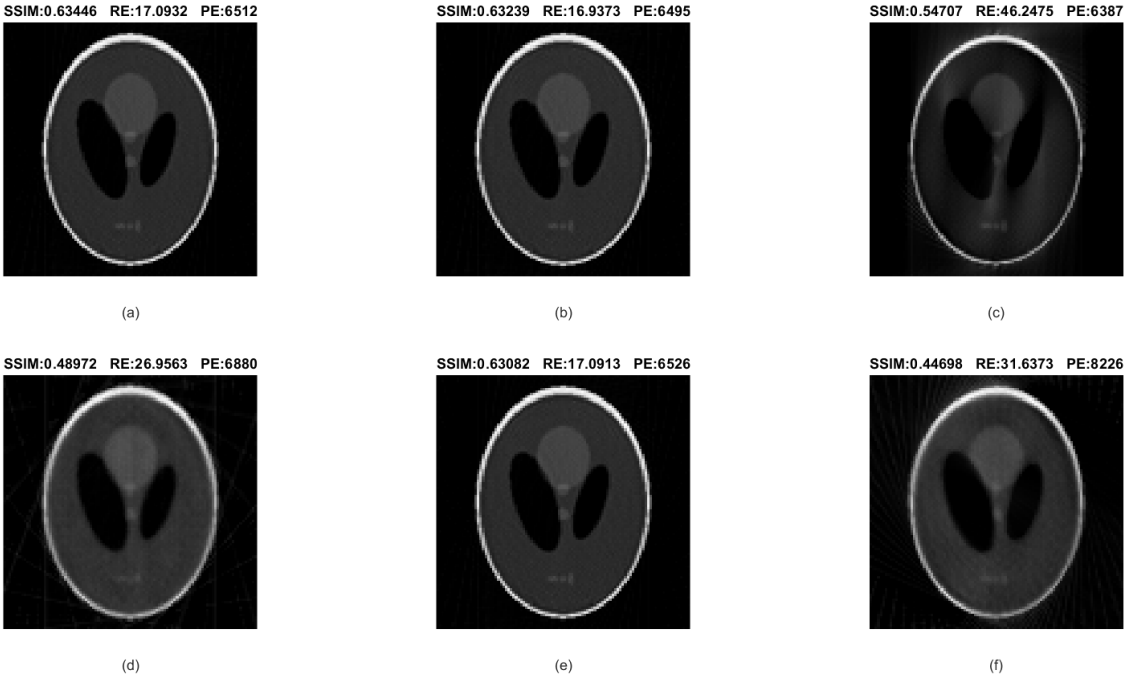


Figure 4.11: Reconstructions without inverse crime (a) simple reconstruction, (b) reconstruction with a Gaussian noise level of 0.1%, (c) reconstruction with projection order $[0,1,...,179]$ and 0.1% of Gaussian noise, (d) reconstruction with 45 projection angles and 0.1% of Gaussian noise, (e) reconstruction with non-negativity constraint to the pixel values and 0.1% of Gaussian noise, (f) reconstruction with 50 iterations.

5

DART

SART is a good starting point, but we will try to incorporate more prior knowledge with the goal of increasing accuracy and the ability to deal with noisy projection data while decreasing computational effort even more. We do this using DART, *Discrete Algebraic Reconstruction Technique*. This reconstruction algorithm alternates iteratively between update steps in which a sub-algorithm is used to make a reconstruction and discretization steps in which we incorporate prior knowledge on the grey levels. As a sub-algorithm, we will use SART as described in Section 4.2 as a sub-algorithm.

The prior knowledge that will be exploited in DART is based on the assumption that the unknown object consists of a small number of different materials, all corresponding to a characteristic grey value. This grey value is approximately constant and thus we assume it to be. Based in this assumption we have that bone is of one constant density. Realistically, this is not the case, however to see whether a bone is broken we do not need to see all the different densities. DART consists of several steps, in this chapter, an overview of these steps will be given firstly in Section 5.1, whereas in Section 5.2 these steps will be further explained individually. Lastly, in Section 5.3 some results will be presented. The contents of this chapter are based on an article by Batenburg and Sijbers [3] and the implementation, for which the code can be found in Appendix B, is based on code by D. Oeronymakis and M. Ieronmaki [7].

5.1. Algorithm overview

As explained above, DART is an iterative process structured in various steps, which will now be shortly introduced with the help of Figure 5.1. In Section 5.2 each step will be defined in more detail. Note that in Figure 5.1 a discretization of $d = 200$ is used so that there are $D = 40,000$ pixels in total, and the projection data was not perturbed by noise. Further parameter settings and the approach of generating Figure 5.1 can be found in Appendix B.3.

We start with an initial reconstruction computed using SART as can be seen in Figure 5.1(b). This SART reconstruction will be segmented such that it only contains certain grey levels as is shown in Figure 5.1(c). The amount of grey levels and their exact values are considered prior knowledge, as we approximately know what tissues are present in for instance the brain. This segmented reconstruction will be divided into two sets, a set with free pixels and a set with fixed pixels. These fixed pixels are pixels that are either within the interior of the object far away enough from a boundary or they are background pixels not too close to a boundary. The fixed pixels are set to the correct grey value as we are sure of what value they need to be, with this we are again incorporating the prior knowledge we have about the object. The remaining pixels, closer to a boundary, are called boundary pixels and are depicted in Figure 5.1(d). Fixing all non boundary pixels to a set value does however limit the possibility of creating another boundary if there is for example a hole in the object. To be able to deal with holes a small subset of non boundary pixels are randomly selected from the fixed pixels during each DART iteration, as shown in Figure 5.1(e). This updating of non-boundary pixels also helps with noise and errors, this will be explained further in Section 5.2.2. The boundary pixels are combined

with the additional randomly selected fixed pixels as is depicted in Figure 5.1(f) and are defined to be the free pixels. For these free pixels, we compute several SART iterations again. After the SART reconstruction, the fixed pixels are combined with the newly computed values of the free pixels. During the SART reconstruction each free pixel can vary independently possibly resulting in large variations, therefore as the last step of the DART procedure smoothing is applied to the newly computed pixel values. We use Gaussian smoothing with a standard deviation of 1 pixel. This completes one iteration of the DART algorithm. The result of this single iteration is then used as input for the next iteration. DART terminates when a certain stopping criterion is met. This stopping criterion can for instance be a fixed amount of iterations or can depend on a performance measure with which we want to reach a certain performance level. Note that in Figure 5.1 with each reconstruction step within one iteration the pixel error (PE) decreases, implying an improved result. This process is shown in Figure 5.2.

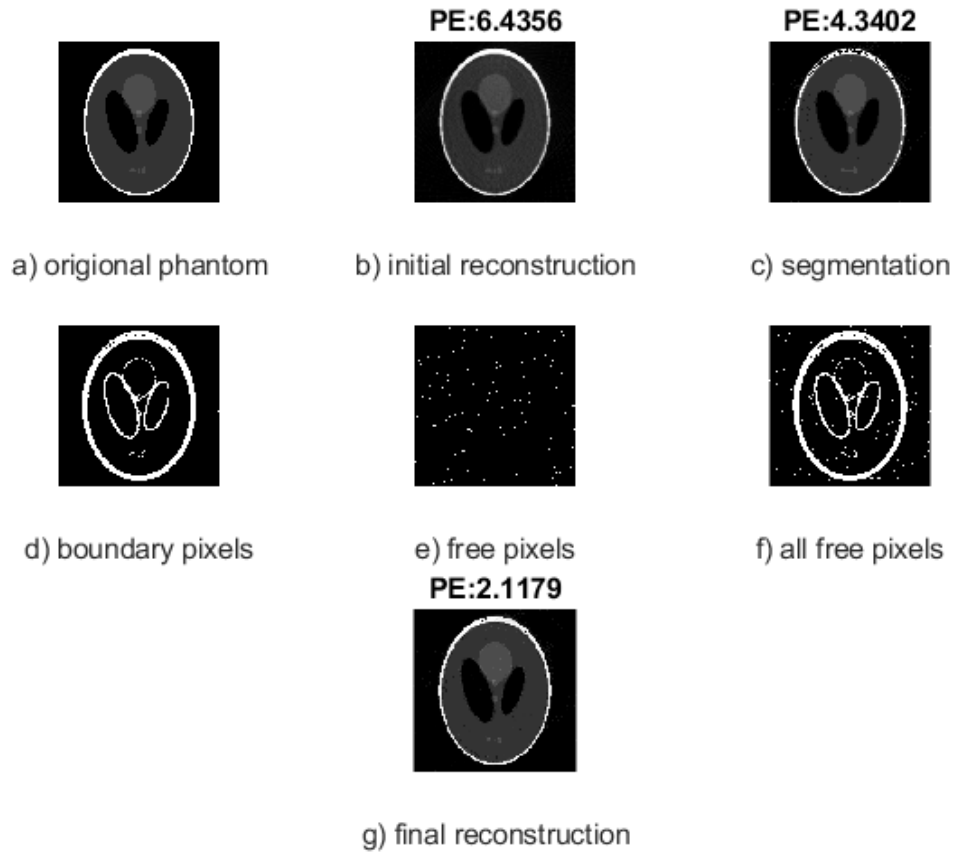


Figure 5.1: Steps of the DART algorithm applied to the Shepp-Logan phantom: (a) original phantom, (b) initial SART reconstruction, (c) grey value segmentation, (d) boundary pixels, (e) additional free pixels with probability p, (f) all free pixels, (g) final reconstruction after 10 iterations.

5.2. Algorithm definition

As in Section 3.2, we have J angels for which we measure from N different distances to the origin, giving us $k = JN$ different measurements in total. Let $\vec{m} \in \mathbb{R}^k$ again be a vector containing all (m_i) measurements and let $\vec{\mu} = (\mu_i) \in \mathbb{R}^D$ represent the unknown object. Finally, let $A = (a_{ij}) \in \mathbb{R}^{k \times D}$ be the projection matrix that maps the unknown to the measurement

$$A\vec{\mu} + \vec{n} = \vec{m}.$$

Note that $\vec{n}$ here denotes the noise. Recall that $\vec{\mu}$ is the attenuation coefficient per pixel, thus the entries of $\vec{\mu}$ determine the grey value in the corresponding pixel. If the attenuation coefficient is low it will result in a

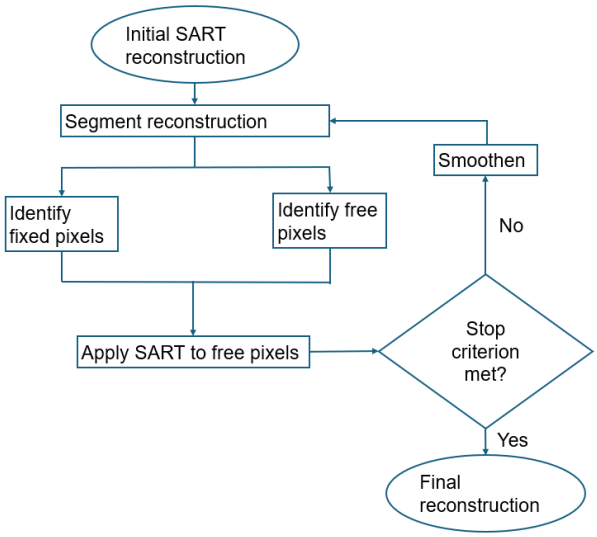


Figure 5.2: Flowchart of the DART algorithm.

darker grey value and a higher coefficient will result in a lighter grey value. Furthermore, entries (a_{ij}) determine the weight of the contribution of pixel i to measurement j, which is the intersection length between the pixel and the projection line. Furthermore we are given grey levels $R = \{\rho_1, ..., \rho_l\}$.

As we are given measured data $\vec{m}$, know $A \in \mathbb{R}_{\geq 0}^{k \times D}$ and can estimate the size of $\vec{n}$ we are left with the inverse problem of finding $\vec{\mu}$ such that $A\vec{\mu} + \vec{n} = \vec{m}$ just as before. The estimation of $\vec{n}$ can be done by measuring empty space, so without an object present.

5.2.1. Segmentation

In each iteration of the algorithm, the current reconstructed image gets segmented such that a new image is obtained which has only grey values from the set $R = \{\rho_1, \rho_2, ..., \rho_l\}$. Typically, grey values range from 0 to 255, with 0 meaning the pixel is black and 255 meaning the pixel is white. R is considered to be prior knowledge. It could for instance be that we want to reconstruct a binary image, thus we assume that we have two grey levels (black and white). From this assumption we determine that only grey values $\{0, 255\}$ are present and thus by segmenting the reconstruction we make sure that no other grey values are present besides black and white. When we consider the Shepp-Logan phantom, the following grey values are present $R = \{0, 25.5, 51, 76.5, 102, 255\}$.

To segment the image, we first need to determine a thresholding scheme according to which we will assign a grey value from set R to each pixel. Define thresholds $\tau_1, ..., \tau_{l-1}$ by $\tau_i = \frac{\rho_i + \rho_{i+1}}{2}$. In this way, the thresholds lie in between two grey values. Next define thresholding function $r : \mathbb{R} \to \mathbb{R}$ as

$$r(v) = \begin{cases} \rho_1 & \text{if} \quad v < \tau_1 \\ \rho_2 & \text{if} \quad \tau_1 \leq v < \tau_2 \\ \vdots & \\ \rho_l & \text{if} \quad \tau_{l-1} \leq v \end{cases}$$

where v is the pixel intensity, so the grey value, of the considered pixel.

5.2.2. Pixel selection

The pixel selection can start when the image is segmented into only a few grey values. For this we first define what it means to be neighbouring pixels, we use an *8-connected neighbourhood*, also called an indirect neighbourhood of Moore neighbourhood.

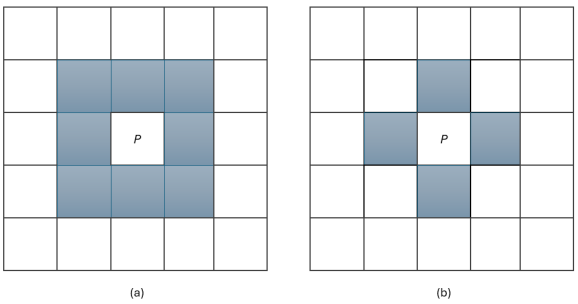


Figure 5.3: Visualization of different types of connectedness (a) an 8-connected neighbourhood (b) an 4-connected neighbourhood.

Definition 5.1. *Pixel Q is an* 8-connected neighbour *of pixel P if they share either an edge or a vertex. Thus pixel (x, y) has neighbours* $(x-1, y, 1)$, $(x-1, y)$, $(x-1, y+1)$, $(x, y-1)$, $(x, y+1)$, $(x+1, y-1)$, $(x+1, y)$, $(x+1, y+1)$.

In Figure 5.3 (a) such an 8-connected neighbourhood is shown and in Figure 5.3 (b) an 4-connected neighbourhood is shown also known as the direct neighbour. An 8-connected neighbourhood is used since this also captures diagonal connections which leads to a smoother and more accurate reconstruction. Furthermore, the inclusion of diagonal neighbours can also help average out noise which again creates a more accurate reconstruction. One would choose to incorporate an 4-connected neighbourhood to reduce computational load.

Next, we define the concept of *boundary pixels*.

Definition 5.2. *Pixel Q is a boundary pixel if at least one neighbour of Q has a different grey value.*

To reduce the impact of noise we introduce a threshold here, changing the formal definition slightly to enhance performance. We consider a neighbour to have a different grey value when the grey value of the two pixels, ranging from 0 to 255, differ more than 10.

Now the set of *free pixels* is the set of boundary pixels with additionally a random sample of non-boundary pixels. These additional free pixels allow for the creation of new boundaries in the reconstruction and help reduce the impact of noise in the measurements. A pixel may be fixed to the wrong grey value due to noise, thus adding some of the fixed pixels into the free pixel subset allows for correction of this mistake. In Section 5.3.6 we explore what amount of free pixels delivers the best reconstruction. We define the amount of fixed pixels that are added into the subset of free pixels as follows

Definition 5.3. *The* fix probability, *denoted $0 \leq p \leq 1$, is the independent probability with which a non-boundary pixel is fixed on its current value. Therefore $1-p$ denotes the independent probability with which a non-boundary pixel is included in the subset of free pixels.*

5.2.3. SART with free pixels

When the free pixels are determined, the SART algorithm will be applied to those free pixels. Consider the previously determined system of equations as in (3.2) without noise

$$\begin{bmatrix} | & & | \\ \vec{a}_1 & \dots & \vec{a}_D \\ | & & | \end{bmatrix} \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_D \end{bmatrix} = \vec{m}. \tag{5.1}$$

Now when we consider variable μ_i to be a fixed pixel, thus setting it at a fixed grey value v_i, we transform system (5.1) to the following

$$\begin{bmatrix} | & & | & | & & | \\ \vec{a}_1 & \dots & \vec{a_{i-1}} & \vec{a_{i+1}} & \dots & \vec{a_D} \\ | & & | & | & & | \end{bmatrix} \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_{i-1} \\ \mu_{i+1} \\ \vdots \\ \mu_D \end{bmatrix} = \vec{m} - v_i \vec{a}_i. \tag{5.2}$$

This new system has the same amount of equations as system (5.1), the number of variables, however, decreases by the number of fixed pixels. Although SART is used as reconstruction method, the reduction in the number of variables makes DART computationally efficient. Even if some pixels are fixed at a wrong value in one or more iterations K. Batenburg and J. Sijbers showed that the algorithm still converges towards the original object [3].

5.2.4. Smoothing

Reducing variables can cause fluctuations in the grey values of pixels that are not fixed. By solely adjusting the values of the free pixels, errors that occur due to fixing a pixel to the wrong grey value are unaccounted for. To reduce the influence of errors we apply a *Gaussian smoothing filter* with a standard deviation of 1 pixel to the free pixels. This regularization step is applied at the end of each iteration, after applying SART, except for the last iteration.

A Gaussian smoothing filter with a standard deviation of 1 pixel is a 3×3 matrix, called the *kernel*. This kernel contains values that are determined by a Gaussian function serving as weights. The weight of the central kernel entry typically being higher than that of the surrounding entries. The matrix, or kernel, moves over the image during which for each pixel in the image the kernel is centered on that pixel. The pixels that the kernel covers are multiplied with the corresponding weights in the Gaussian kernel. The resulting values are summed together to provide a weighted average which will replace the original pixel value. This results in a smoothed image where the noise is reduced but the overall structure is preserved.

5.2.5. Termination

DART is a *heuristic algorithm*, meaning that it does not guarantee an optimal reconstruction under predetermined conditions. Therefore, in this thesis, as termination criterion a fixed number of iterations is performed. Another termination criterion could be set using a threshold for a certain performance metric, for example the relative error. The relative error could be calculated between the current reconstruction and one before that to see if there is much change between iterations. Such that, when the relative error reaches below a certain level, the process is terminated.

5.3. Results

The performance of the DART algorithm is evaluated by comparing the reconstructions of DART to those of aforementioned reconstruction methods while differing the parameter settings. In this Section we will analyse the results of certain experimental parameter settings, using this analysis we will be able to draw our conclusions in Chapter 6. For the evaluations the Shepp-Logan phantom is used, which was introduced in Chapter 2. Additionally a binary image is used to evaluate the performance when the only present grey values are black and white. Figure 5.4(a) shows the original image used, which can be downloaded from the ASTRA Toolbox documentation [10]. It has a resolution of 512×512, as this is an image size common in practical CT applications [4]. Figure 5.4(b) shows the generated measurement, the sinogram, using 180 projections with an angle range of 0 to π. This is an ideal measurement, meaning that it was not perturbed by noise. When reconstructing an image from measurements each time the same size image (512×512) is used. The measurements are however sampled using different parameter settings, the number of projections and

the angle range are for instance varied during each specific experiment. Additionally noise may be added to perturb the measurements, it was chosen to not perturb every measurement with noise but also look at noiseless reconstructions to be able to analyse the effects of differing certain parameters to create better understanding. The used parameter settings will be mentioned for each experiment and can be found in Appendix B. To create the reconstructions the ASTRA Toolbox [2] was used and the implementation was based on that of D. Ieronymakis and M. Ieronymaki [5].

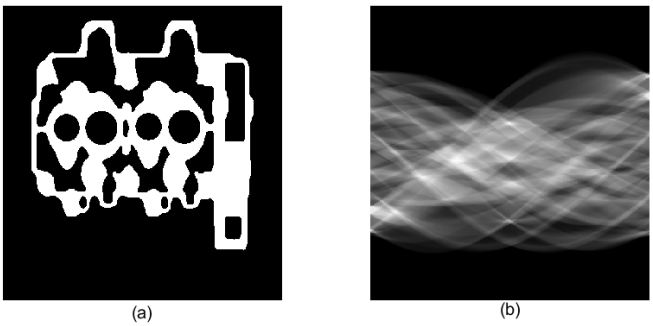


Figure 5.4: Binary phantom and the belonging sinogram (a) binary phantom (b) sinogram of the binary image.

5.3.1. Tuning parameters

In Figure 2.10 it was shown that, if enough projections and a big enough angular range are used to make the measurements, filtered back projection is able to obtain an almost perfect reconstruction for noiseless measurements. Since DART is a heuristic algorithm, as explained in Section 5.2.5, there are no pre-determined conditions for which DART is able to create a perfecr reconstruction. The quality of the reconstruction using DART strongly depends on the amount of iterations that are used for DART and the amount of iterations that are used in the sub algorithm, which in our case is SART. Additionally, the fix probability affects the quality of the reconstruction. Before starting with experiments in which the performance of DART will be evaluated, we will tune these parameters. By doing this we will make sure DART's performance can be evaluated fairly. The parameters will be tuned using 90 projections equally divided over an angle range of 0 to π. Table 5.1 and Figure 5.5 show the results obtained during the tuning experiment.

To start of 200 iterations were chosen for DART, 3 iterations were chosen for SART and the fixed probability was set to 0.85 since these were the parameters used in the reference paper [3]. As can be seen in Figure 5.5(a) these settings do not obtain a preferred reconstruction. In addition, the computation time using these parameter settings was the longest of all parameter setting tries. While working with SART in Chapter 4 the computation time was significantly lower than for this first reconstruction. Thus the presumption was made that the extensive computation time was due to DART, therefore in the second try the amount of DART iterations was lowered to 100 and to compromise, the amount of SART iterations was increased to 10 while keeping the fix probability the same. This resulted in a slightly better reconstruction as the relative error has decreased and Figure 5.5(b) already shows more of the round characteristic of the Shepp-Logan phantom. To try to improve even further, the amount of DART iterations was even more decrease to 50 and with this 100 SART iterations were used, again keeping the fix probability the same. This resulted in a reconstruction in which all properties of the original Shepp-Logan phantom are visible. Trying to improve it even further with 10 DART iterations and 1000 SART iterations while still keeping p the same again resulted in a lower relative error. The amount of SART iterations however seemed extensive, thus a reconstruction with 10 DART iterations and only 200 SART iterations were tried. The 200 SART iterations were chosen here since that amount of iterations proved to be sufficient when working with SART in Chapter 4. It can be concluded that indeed a 1000 iterations for SART is extensive as 200 SART iterations resulted in an improved relative error compared to the 1000 SART iterations and significantly reduced the computational efforts due to reduction in iterations it was chosen to proceed with these parameter settings. Lastly the fix probability was increased to 0.9 and this

resulted in a higher relative error. Computation time decreased using a fix probability of 0.9, therefore we will move forward with the following parameter settings

- Dart iterations: 10
- SART iterations: 200
- fix probability p: 0.9

unless specified otherwise. When trying to decrease the fix probability, the computation time became an obstacle. Thus, as we want to be able to compare as much reconstructions as possible, I chose to break the program while reconstructing with a lower fix probability. The parameter tuning experiment was only run for the Shepp-Logan phantom as we are most interested in the performance for a phantom containing multiple grey levels. However, since our implementation approach was similar to that of [7], based on their tuning experiment it was concluded that the experiment was sufficient to conclude these parameter settings to be appropriate.

An interesting observation can be made about Figure 5.5(a) and (b). These reconstructions are quite far away from the original phantom and almost look like a reconstruction made with limited angle data, which we will encounter in Section 5.3.3. This is mainly due to the small number of SART iterations as only 3 or 10 iterations are not sufficient to refine the reconstruction before being segmented in the next DART step. While te SART reconstruction has not converged yet to an acceptable reconstruction, DART will apply the different steps, which possibly makes the reconstruction even worse. By enforcing the SART reconstruction to be segmented into a few grey levels the reconstruction might get further away form the original even more. The segmentation can introduce artifacts, like we see when we use limited angle data and therefore the reconstructions have the same properties. Additionally, this many DART iterations worsen the reconstruction as well since the errors made in the early iterations accumulate leading to a final reconstruction that still contains these errors and artifacts. Therefore an appropriate DART-SART iteration balance is of importance. There should be enough SART iterations such that the initial reconstruction is of sufficient quality and there must not be too much DART iterations such that the errors get accumulated too much. However, there should be enough of both to be able to create a reconstruction of acceptable quality.

Shepp-Logan phantom				
	DART iter	SART iter	p	Relative Error
a	200	3	0.85	78.9
b	100	10	0.85	71.0
c	50	100	0.85	24.1
d	10	1000	0.85	22.3
e	10	200	0.85	21.9
f	10	200	0.9	23.8

Table 5.1: Parameter tuning results, corresponding to the images in Figure 5.5, for different values of the amount of DART and SART iterations and fix probability on the Shepp-Logan phantom, based on the relative error.

5.3.2. Varying number of projections

Firstly, we will analyze the performance of the different reconstruction methods with respect to the considered number of projections. It is intuitive that for more projections all methods will perform better, therefore only cases of a small number of projections are evaluated. A proof that reconstructions of all methods described in this thesis improve as the number of projections increases can be read in [3].

Reconstructions were evaluated for 20 and 10 projections divided over an angle range from 0 to π for DART, SART and Filtered Back Projection (FBP). Here we evaluated both the Shepp-Logan phantom and the phantom introduced by Figure 5.4. Lastly note that noiseless measurements were used. In Figure 5.6 all above mentioned reconstructions are shown. As expected, since more measurements, thus more data and more knowledge, result in a better reconstruction. The errors increase and the similarity index decreases

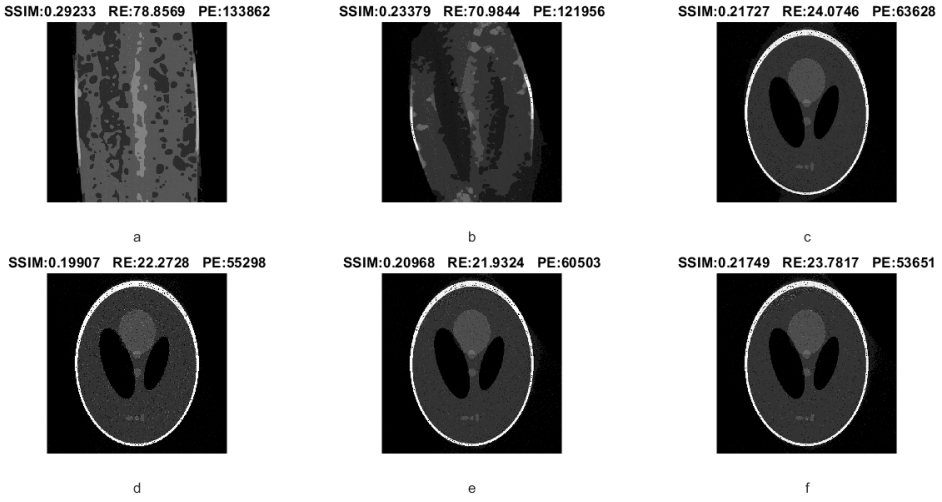


Figure 5.5: Comparison of the Shepp-Logan phantom during tuning parameter settings as specified in Table 5.1

when the number of projections decreases from 20 to 10 projections for all methods. There seems to be a minimum number of projections required to obtain an acceptable reconstruction. Although this minimum differs per phantom and per reconstruction method, it seems to lie closer to 20 projections than to 10 for each reconstruction phantom and method pair in Figure 5.6. For more exact and substantiated answers, additional experiments should be conducted in which for instance the number of projections could range from 15 to 25. For each parameter setting DART achieves the best result when looking at the performance metrics. When comparing the reconstructions visually it can be concluded that FBP shows the least performance regarding limited projections. FBP shows artefacts in the background as well as in the phantom itself. These artefacts are aliasing effects. *Aliasing* is the effect of incorrectly reconstructing the original measurement because the sampling frequency is too low. The sampling frequency is how often the X-ray detector captures data points, which is also influenced by how many projection angles we choose to measure from. If this is too low when the *Nyquist sampling criterion* is not met.

Definition 5.4. *Take f_{max} to be the highest frequency detected in the measurement and let f_s be the frequency with which the samples, or measurements, are taken. Then the* Nyquist rate *is defined as*

$$f_s > 2f_{max}.$$

When this Nyquist rate is not met high-frequency components will be misinterpreted as low-frequency components, resulting in streaking artifacts in the reconstructions.

Comparing SART and DART there we may not be able to appoint one winner. While DART is more detailed, it also shows more noise. In comparison, SART looks more smoothed. When solely comparing reconstructions (c) and (d) to (g) and (h) respectively, DART can be appointed as the winner as the reconstruction should only consist of black and white pixels. Therefore the noise can be distinguished easily. The DART reconstructions in comparison to the SART reconstructions also show a better ability to reconstruct small regions. Figure 5.6(c) for example shows all different regions of the phantom whereas Figure 5.6(g) does not show the smallest black oval regions in the bottom of the phantom. However, comparing the Shepp-Logan phantoms in (a) and (b) to (e) and (f) it becomes more difficult as now multiple grey levels are present and noise can not be distinguished as easily. The application of the method decides which method, DART or SART, is better to use. In medical applications, one could imagine that a wrongly coloured segment, like in the lower right portion of the phantom in Figure 5.6(b), could result in a wrong conclusion. The difference in performance for these two phantoms can be explained by how DART incorporates prior knowledge on grey levels. DART uses the fact that we know what certain grey levels should be present in the reconstruction, it however does not incorporate any information on what grey levels should be used where. When only using black and white we know all other grey levels should be changed to either black or white. However, when looking at a phantom with more grey levels it becomes more difficult to say that a certain pixel is assigned

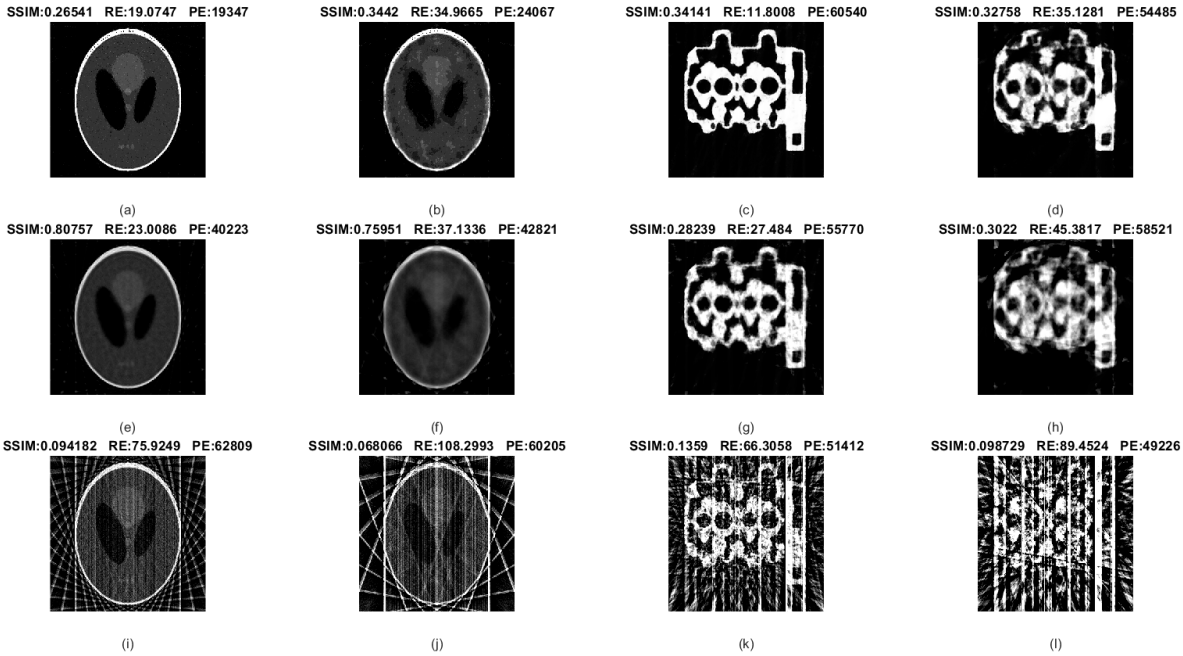


Figure 5.6: Comparison of DART (row 1), SART (row 2) and FBP (row 3) for Shepp-Logan phantom (columns 1 and 2) and phantom introduced in 5.4 (columns 3 and 4) using 20 (column 1 and 3) and 10 projections (column 2 and 4). (a) DART 20 projections (b) DART 10 projections (c) DART 20 projections (d) DART 10 projections (e) SART 20 projections (f) SART 10 projections (g) SART 20 projections (h) SART 10 projections (f) FBP 20 projections (j) FBP 10 projections (k) FBP 20 projections (l) FBP 10 projections.

to a wrong grey value if it is a value of our prior knowledge list. Therefore we may end up with the right grey levels, but in the wrong places.

5.3.3. Varying number of angles

Figure 5.7: Person in a CT scanner, https://my.clevelandclinic.org/health/diagnostics/4808-ct-computed-tomography-scan

In a medical application of tomography reconstruction techniques we are able to collect data from a full angular range of 180°, so from 0 to π. Meaning that we are able to look at the object in question, in that case the patient, from every angle as a ct scanner device wraps around the whole patient as is shown in Figure 5.7. However, in other, more industrial, applications the angular range to sample data from is often more limited. Industrial applications of tomography reconstruction techniques could for instance be material analysis, which materials are present in a certain construction and what are their properties. Another industrial application is quality control in which the quality of manufactured parts is analysed for instance in the automotive or aerospace industry. As this is only a small portion of all the applications of discrete tomography, it shows the versatile applications of reconstruction techniques besides the medical one. Therefore, although it may not be of importance to X-ray applications, it is still of importance to also test the performance of DART considering a limited angle range.

When using a limited amount of angles there will always be a part of the object that is not measured and therefore will always be improperly reconstructed as the data is simply non existent. Thus, to keep the reconstructions somewhat meaningful it was chosen to conduct this limited angle range experiment first using half of the full angular range corresponding to $\frac{\pi}{2}$ or approximately 90° and the secondly one third of the full angular range is used corresponding to $\frac{\pi}{3}$ or approximately 60°. In this the projections are sampled at 1° intervals, thus the number of projections increases linearly with the angular range. In Figure 5.8 reconstructions are shown again using DART, SART and FBP for both phantoms using angular ranges of $\frac{\pi}{2}$ and $\frac{\pi}{3}$. Note that again FBP, in the bottom row, shows the least performance. The FBP reconstructions shows artifacts, in the form of vertical and horizontal streaks. When comparing the DART and SART reconstructions (Figure 5.8(c),(d) and (g),(h) respectively) of the phantom from Figure 5.4 we can again conclude that DART has the better performance. Especially the DART reconstruction in Figure 5.8(c), examined visually, shows a reconstruction quite close to the original phantom. Then considering the DART and SART reconstructions of the Shepp-logan phantom in Figure 5.8(a), (b) and (e), (f). The same effects as in Section 5.3.2 can be noted as SART again produces more smoothed reconstructions and DART shows a more pixel dominated image. The same conclusion can therefore be repeated, depending on the application one can prefer one of the two methods.

5.3.4. Noisy measurements

The previous reconstructions were all obtained from noiseless measurements. In reality, measurements often contain a certain amount of noise. Therefore DART's performance with respect to noisy projection data needs to be evaluated. The original measurements were polluted with Poisson-distributed noise, with noise levels ranging from 0% noise to 17% noise. Poisson distributed noise is chosen since our measurements are based on the detection of photons, being discrete and independent particles. Because of the independence and each photon having a constant probability the Poisson distribution applies to this situation.

As expected all reconstruction methods showed decreasing reconstruction quality when increasing the noise level both visually as performance metric wise. In the experiment conducted by [7] it was concluded that the errors increased linearly with increasing Poisson noise perturbing the measurements for all reconstruction methods evaluated in this thesis. Not enough noise levels were tested to conclude this within this thesis due to computational limitations. In Figure 5.9 that all reconstructions perform relatively well with the highest noise level of 17%. Again, reconstructions using SART (row 2 Figure 5.9) show the smoothest results, reconstructions using FBP (row 3 Figure 5.9) show the most background artifacts and reconstructions using DART (row 1 Figure 5.9) show the most pixel dominated results. Although FBP shows more artifacts in the background, the reconstructed phantom itself shows all details present in the orignial Shepp-logan phantom. For the noise level of 17% the performance metrics agree that FBP shows the lowest performance. However, analyzing visually, I do not fully agree with the performance metrics. The reconstruction using FBP

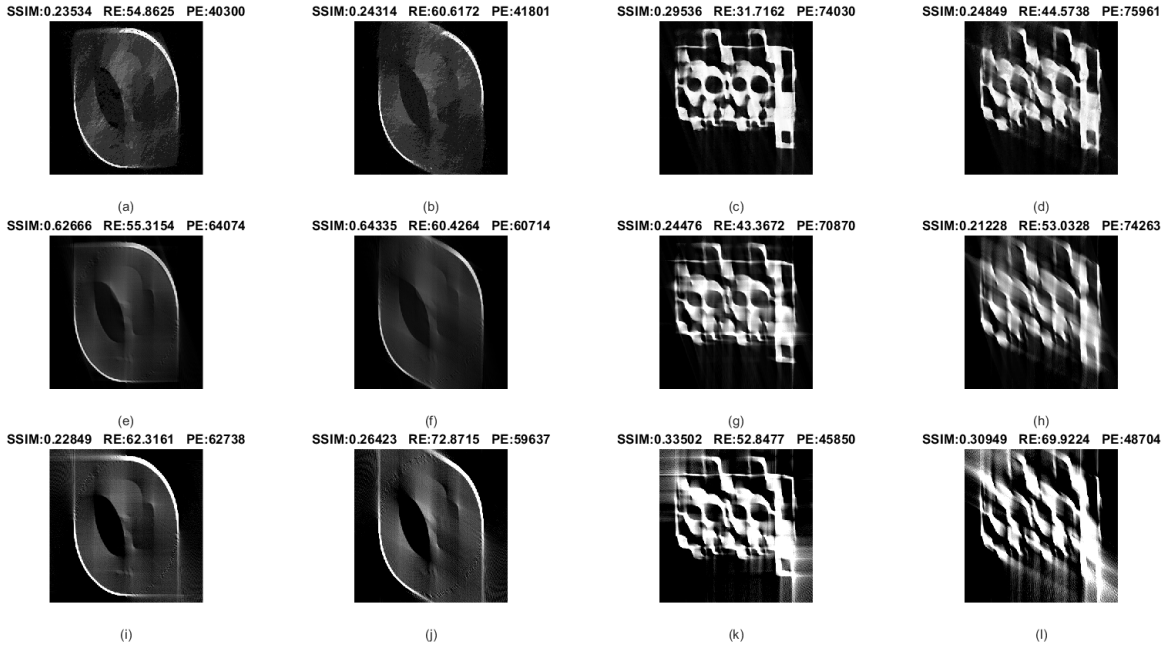


Figure 5.8: Comparison of DART (row 1), SART (row 2) and FBP (row 3) for Shepp-Logan phantom (columns 1 and 2) and phantom introduced in 5.4 (columns 3 and 4) using an angular range of 90° (column 1 and 3) and 60° (column 2 and 4). (a) DART 90° (b) DART 60° (c) DART 90° (d) DART 60° (e) SART 90° (f) SART 60° (g) SART 90° (h) SART 60° (f) FBP 90° (j) FBP 60° (k) FBP 90° (l) FBP 60°.

shows more detail than the reconstruction using DART, especially in the two small light grey circles present in the middle of the phantom between the black ovals.

More interesting however is to compare the methods' performance regarding noisy data with a limited amount of projections. As explained in Section 5.3.2 performance with a limited amount of projections is of importance, by incorporating noise into the limited projections experiment we come closer to reality. In Figure 5.10 reconstructions are shown with the same parameter settings and noise levels as in Figure 5.9, only know instead of sampling the projections in 1° intervals, only 10 projections are sampled over the entire full angle range of 180°. As was concluded in Section 5.3.2, FBP does not perform well with a little amount of projections. Therefore, even tough the method deals well with noise, with a limited amount of projections the performance lacks. The comparison between DART and SART will lead to the same conclusions as in all preceding subsections of this section. SART has more smoothed reconstructions and while DART performs better error wise, the reconstructions are pixel dominated and one should decide which method works better dependent on the application.

The same experiment was conducted using the phantom introduced in Figure 5.4. As the reconstructions show the same results and the same conclusions can be drawn from them, the results will not be displayed here to limit the number of images shown, rather the results can be found in Appendix C.

5.3.5. Varying grey levels

For the same reason why the experiment in Section 5.3.4 was conducted we will now perturb the grey levels, assumed to be prior knowledge, with noise. In practical applications the precise grey levels might not be known. Since only DART requires the usage of prior knowledge of the grey levels we only consider DART in this subsection we chose to not compare the performance to another reconstruction methods. Nevertheless, we are able to compare something as the noise will be added to the grey values in three different manners. We test the effect of overestimation, underestimation and a combination of both. In the overestimation the grey values where estimated higher than the actual values by adding a certain percentage of it's original value. In the underestimation the same method was used however now this value, calculated by taking the percentage of the original grey value in consideration, was deducted from the the original value. In the combined

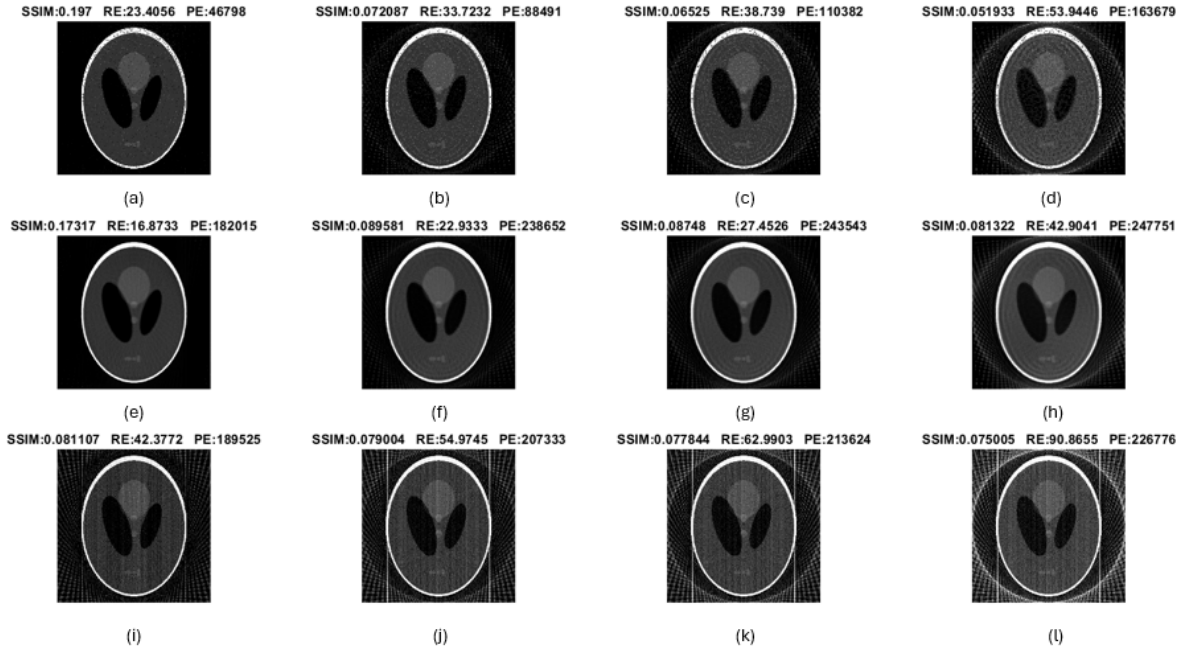


Figure 5.9: Comparison of DART (row 1), SART (row 2) and FBP (row 3) for Shepp-Logan phantom using data perturbed by a noise level of 0 (column 1), 5.6% (column 2), 8.4% (column 3) and 16.8% (column 4). (a) DART 0% noise (b) DART 5.6% noise (c) DART 8.4% noise (d) DART 16.9% noise (e) SART 0% noise (f) SART 5.6% noise (g) SART 8.4% noise (h) SART 16.9% noise (i) FBP 0% noise (j) FBP 5.6% noise (k) FBP 8.4% noise (l) FBP 16.9% noise.

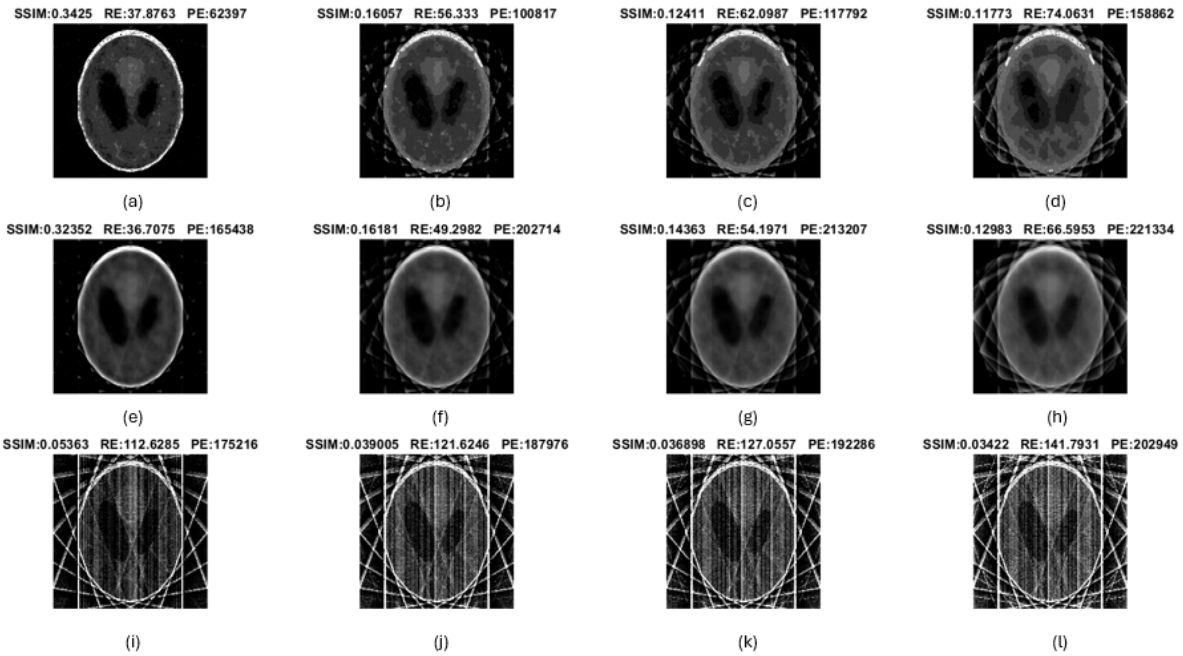


Figure 5.10: Comparison of DART (row 1), SART (row 2) and FBP (row 3) for Shepp-Logan phantom using data perturbed by a noise level of 0 (column 1), 5.6% (column 2), 8.4% (column 3) and 16.8% (column 4) of 10 projections. (a) DART 0% noise (b) DART 5.6% noise (c) DART 8.4% noise (d) DART 16.9% noise (e) SART 0% noise (f) SART 5.6% noise (g) SART 8.4% noise (h) SART 16.9% noise (i) FBP 0% noise (j) FBP 5.6% noise (k) FBP 8.4% noise (l) FBP 16.9% noise.

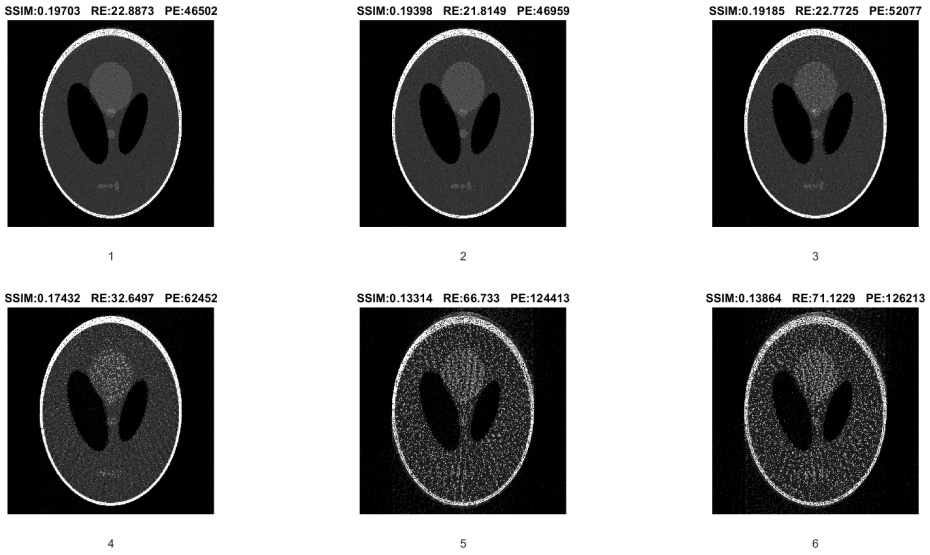


Figure 5.11: Comparison on the effect of different levels of overestimation within the grey values used in DART showed on the Shepp-Logan phantom. (1) overestimation of 0.02% (2) overestimation of 0.1 % (3) overestimation of 0.18 % (4) overestimation of 0.26 % (5) overestimation of 0.34 % (6) overestimation of 0.4.

approach, the sign of the percentage was sampled randomly such that a random mix of over and under estimation is used. For all three experiments the following percentages of over- or underestimation were used $[0.02, 0.1, 0.18, 0.26, 0.34, 0.4]$.

First, Figure 5.11 shows the reconstructions for the overestimation of the grey levels for six different percentages. As one might expect, the reconstruction performance of DART decreases as the percentage perturbing the grey values increases. Only above overestimating the grey values by 0.26% the overestimation becomes visually bothersome. However, when comparing this to Figure 5.12, we can conclude that overestimating the grey values results is better reconstructions. The errors of the reconstructions while underestimating the grey values are higher. Especially visually, starting from overestimating the gray values by 0.26%, the reconstruction becomes incomprehensible. This is due to the fact that underestimation of the grey levels the pixel values are closer to zero, thus darker and therefore it can lead to loss in detail and loss in contrast.

Finally, Figure 5.13 shows reconstructions dealing with both underestimations and overestimations of the grey values. By sometimes overestimating the grey level and sometimes underestimating the grey level we balance out the overall error as it could happen that the threshold between an underestimated value and an overestimated value is still correct. Whereas when considering for instance only underestimated values the threshold is shifted lower. The combination could even ensure that lighter areas are not overestimated and darker areas are not underestimated. The quality of the reconstruction thus depends on which values are underestimated and which values are overestimated, resulting in a potentially better performance in Figure 5.13(5) whilst the grey levels are estimated further away from the truth than in for example Figure 5.13(2). The same experiment was conducted using the phantom introduced in Figure 5.4, for these reconstruction the reader is referred to Appendix C.

5.3.6. Varying fix probability

One of the most influential parameters of DART is the fix probability p. As explained, using relatively low fix probabilities will result in higher computational time, therefore it was chosen to run this experiment using only three different values for p which were evaluated for six different noise levels, The noise is computed in the same manner as in Section 5.3.4. This experiment was only run using DART as the other mentioned methods do not include this parameter. The reconstructions were made using measurements from 180 projections ranging from 0 to π.

A lower fix probability, one of $p = 0.5$ as shown in Figure 5.16 results in a better reconstruction than higher

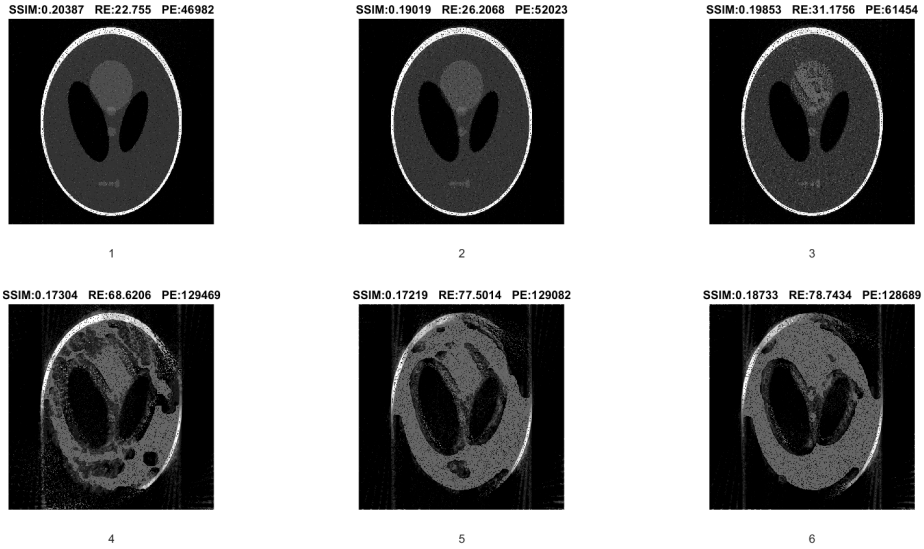


Figure 5.12: Comparison on the effect of different levels of underestimation within the grey values used in DART showed on the Shepp-Logan phantom. (1) underestimation of 0.02% (2) underestimation of 0.1 % (3) underestimation of 0.18 % (4) underestimation of 0.26 % (5) underestimation of 0.34 % (6) underestimation of 0.4.

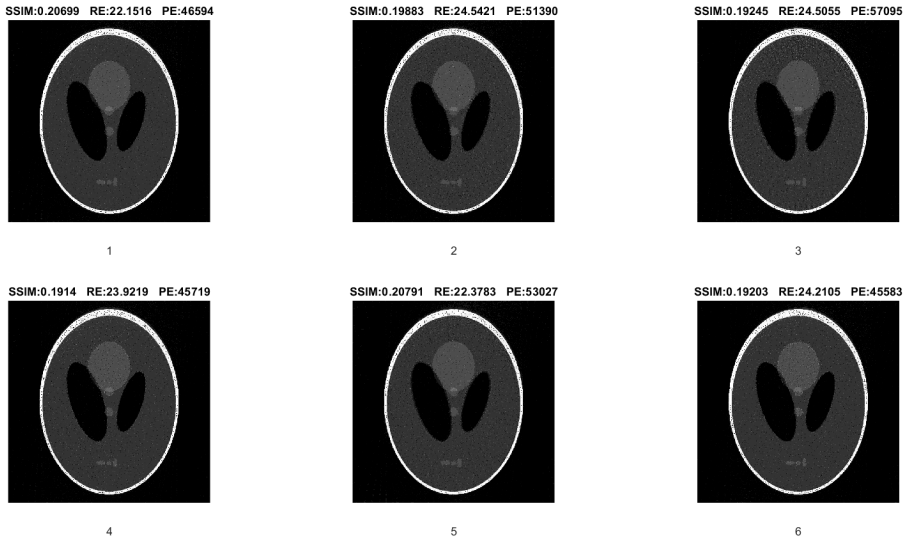


Figure 5.13: Comparison on the effect of different levels of a combination of underestimation and overestimation within the grey values used in DART showed on the Shepp-Logan phantom. (1) estimation error of 0.02% (2) estimation error of 0.1 % (3) estimation error of 0.18 % (4) estimation error of 0.26 % (5) estimation error of 0.34 % (6) estimation error of 0.4%.

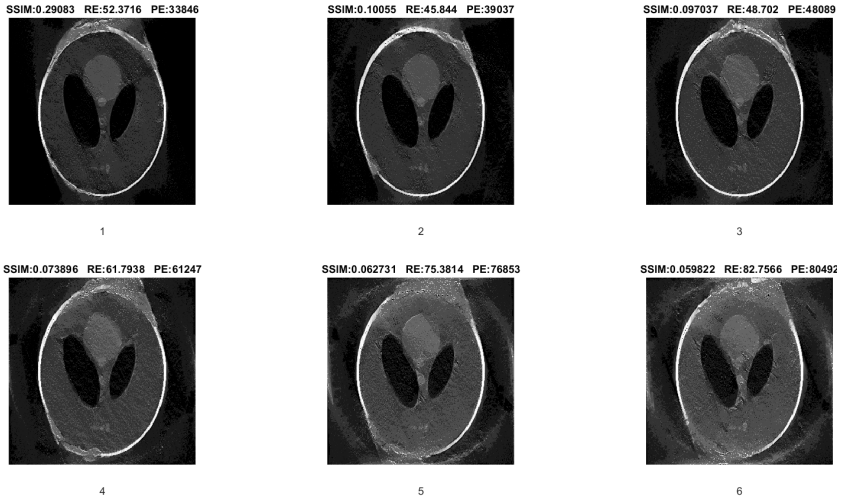


Figure 5.14: Comparison reconstructions of the Shepp-Logan phantom made using DART for different noise levels and a fix probability of $p = 0.9$. (1) noise level of 0% (2) noise level of 2.8% (3) noise level of 5.6%(4) noise level of 8.5%(5) noise level of 14.1%(6) noise level of 16.8%.

fix probabilities of $p = 0.7$ show in Figure 5.15 and $p = 0.9$ shown in Figure **??**. This is due to more free pixels means more flexibility in representing a smooth transition between different coloured regions. More free pixels also enables the algorithm to better adapt to noise. Increasing the amount of free pixels however also means increasing computation time. This trade-off can be made depending on the context, as I have limited computation power I might choose for a larger fix probability as opposed to one who might have more computational power and needs more detailed reconstructions. Although this experiment was run for only three different fix probabilities, from the results in combination with the reference paper [7] it can be concluded that the overall reconstruction quality improves as the fix probability decreases.

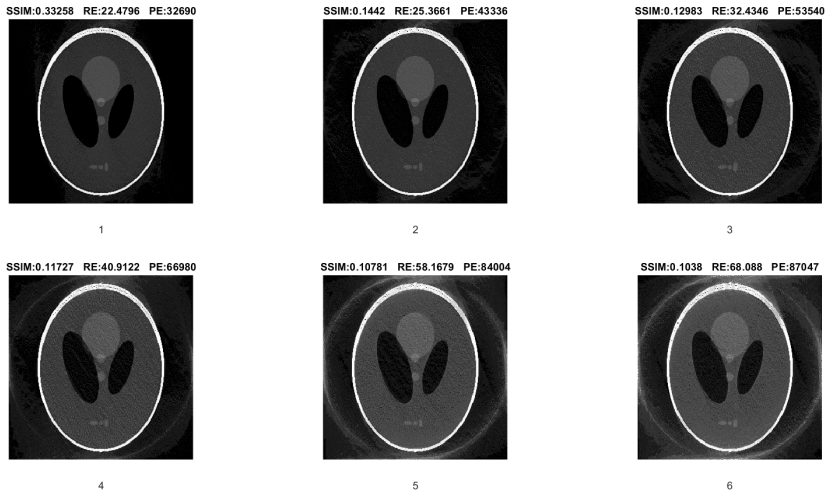


Figure 5.15: Comparison reconstructions of the Shepp-Logan phantom made using DART for different noise levels and a fix probability of $p = 0.7$. (1) noise level of 0% (2) noise level of 2.8% (3) noise level of 5.6%(4) noise level of 8.5%(5) noise level of 14.1%(6) noise level of 16.8%.

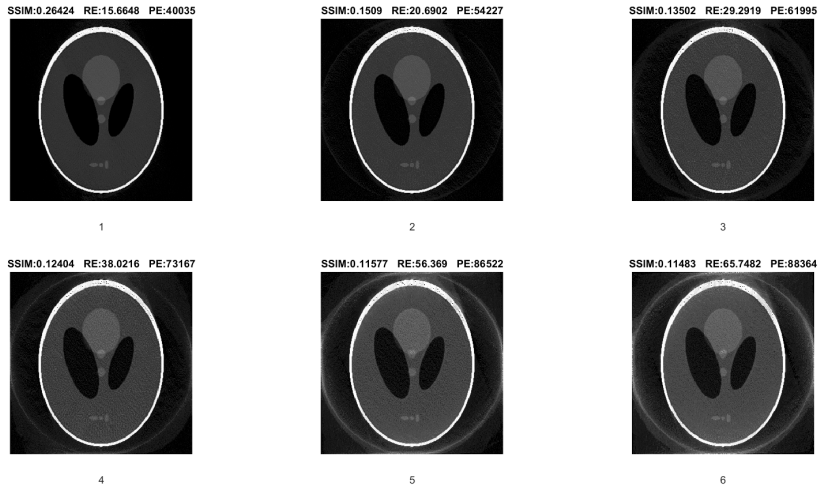


Figure 5.16: Comparison reconstructions of the Shepp-Logan phantom made using DART for different noise levels and a fix probability of $p = 0.5$. (1) noise level of 0% (2) noise level of 2.8% (3) noise level of 5.6%(4) noise level of 8.5%(5) noise level of 14.1%(6) noise level of 16.8%.

6

Conclusion and discussion

The aim of this thesis was to give an introduction to the mathematics of X-ray computed tomography and to reconstruct the internal structure of objects from measurement data using DART. For this a basic understanding of the sub-algorithm SART was needed. DART and SART both consider the discrete version of the X-ray computed tomography. Besides this filtered back-projection was introduced which models X-ray computed tomography as a continuous problem. The three methods were visualized by reconstructions of simulated data using the Shepp-Logan phantom. Each reconstruction was evaluated using multiple performance metrics and visually.

Filtered and unfiltered back-projection

Back-projection was only applied to noise free data. This reconstruction is blurry resulted through back-projecting the measurement values evenly throughout the reconstruction. Filtered back-projection was applied to both noise free data and to data perturbed by 5% noise. This data is ideal, meaning that there is enough measurement data made from an acceptable angular range. For the noiseless data filtered back-projection shows an almost perfect reconstruction. The reconstruction from the noisy sinogram is clearly of lower quality however the main characteristics of the Shepp-Logan phantom are still visible. During the analysis of DART multiple reconstructions using filtered back-projection were made as well to compare the performances. With these it us concluded that with limited projection data available the method however does not perform as well. In the frequency domain insufficient sampling leads to aliasing. Because the Nyquist sampling criterion is not met high-frequencies components will be misinterpreted as low-frequency components, resulting in streaking artifacts in the reconstructions.

DART and SART

DART and SART were evaluated with all kinds of different parameter settings, from noisy measurement data to limited data. Both DART and SART are robust when using limited measurements, therefore these methods have a preference over filtered back-projection. When looking solely at the performance metrics, DART overall performs better than SART. However, visually we are not always able to appoint a winner. Reconstructions from both methods have their advantages and disadvantages. While DART is more detailed, it also shows more noise. In comparison, SART looks more smoothed, but it therefore also lacks detail. This difference can be explained as follows; DART constraints the the pixel values to a discrete and limited set of grey values. Therefore it maintains sharp boundaries between different coloured regions. Additionally, DART includes the step to refine the boundaries, again enhancing the sharp edges. With this DART effectively maintains boundaries, even of smaller regions. In contrast, SART can be considered the more continuous approach as

the pixel values are not restricted to a discrete set of grey levels. SART instead aims to minimize the overall projection error in a way that often averages out the pixel values. This creates a more gradual and smooth change between different coloured regions, thus not displaying as sharp edges as DART. This smoothing effect also helps to reduce noise as it mitigates random fluctuations in pixel values. For DART, which was our main focus, it was concluded that a low fix probability works best. When considering the pre-determined grey levels we conclude that overestimation results in better reconstructions than underestimation. A combination of both however works best, a combination of both is also closest to reality and therefore we conclude that DART is robust when making errors in the pre-determination of the grey levels.

Recommendations further research

This report used simulated data in it's reconstructions, no real life data was used. Therefore it is recommended to considered real-life data and compare the reconstructions using real-life data to the simulated data reconstructions. Furthermore, parameter tuning could be performed more extensively to find the optimal parameters. These optimal parameters should thereafter also be used during the reconstructions, as in this thesis the used fix probability was not optimal. This decision was due to computational limitations.

The methods TSVD, Tikhonov regularisation and Total variation minimization were only introduced briefly due to time limitations and no reconstructions were made using these methods. These methods could potentially perform well in situations with limited data, therefore it is recommended to also generate reconstructions using these methods and compare them to the reconstructions from DART and SART.

Bibliography

[1] A Andersen. "Simultaneous Algebraic Reconstruction Technique (SART): A superior implementation of the ART algorithm". In: *Ultrasonic Imaging* 6.1 (1984), pp. 81–94. DOI: 10.1016/0161-7346(84)90008-7. URL: https://www.sciencedirect.com/science/article/pii/0161734684900087?via=ihub.

[2] *ASTRA Toolbox 2.1.0 documentation*. URL: https://astra-toolbox.com/docs/algs/SART.html.

[3] K.J. Batenburg and J. Sijbers. *DART: A Practical Reconstruction Algorithm for Discrete Tomography*. URL: https://ieeexplore.ieee.org/document/5738333.

[4] Daniel Bell and Mirjan Nadrljanski. "Computed tomography". In: *Radiopaedia.org* (2010). DOI: 10.53347/rid-9027. URL: https://radiopaedia.org/articles/computed-tomography?lang=us.

[5] *DART$_p$ython*. URL: https://github.com/OhGreat/DART_python/blob/main/experiment_scripts/gray_values_exp.py.

[6] Harvard Health. *Radiation risk from medical imaging*. en-US. 2021. URL: https://www.health.harvard.edu/cancer/radiation-risk-from-medical-imaging.

[7] D Ieronymakis and E Ieronymaki. *GitHub of OhGreat DART$_p$ython*. URL: https://github.com/OhGreat/DART_python/blob/main/DART_experimentation_report.pdf.

[8] "Image quality assessment: From error visibility to structural similarity". In: *IEEE TRANSACTIONS ON IMAGE PROCESSING* 13.4 (2004), pp. 600–601. URL: https://www.cns.nyu.edu/pub/eero/wang03-reprint.pdf.

[9] Jennifer L. Mueller and Samuli Siltanen. *Linear and Nonlinear Inverse Problems with Practical Applications*. 2012. DOI: 10.1137/1.9781611972344. URL: https://researchportal.helsinki.fi/en/publications/linear-and-nonlinear-inverse-problems-with-practical-applications.

[10] *phantom from the ASTRA Toolbox*. URL: https://github.com/astra-toolbox/astra-toolbox/blob/master/matlab/algorithms/DART/examples/cylinders.png.

[11] Jerry L. Prince and Jonathan M. Links. *Medical imaging signals and systems*. 2015. URL: https://iacl.ece.jhu.edu/images/0/06/Medical_Imaging_Signals_and_Systems_Pearson_2014.pdf.

[12] *Resolution and Size Limitations – UTCT – University of Texas*. URL: https://www.ctlab.geo.utexas.edu/about-ct/resolution-and-size-limitations/.

[13] Raymond A. Schulz, Jay A. Stein, and Norbert J. Pelc. "How CT happened: the early development of medical computed tomography". In: *Journal of Medical Imaging* 8.5 (2021). DOI: 10.1117/1.jmi.8.5.052110. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8555965/.

[14] Efrat Shefer et al. "State of the Art of CT Detectors and Sources: A Literature review". In: *Current Radiology Reports* 1.1 (2013), pp. 76–91. DOI: 10.1007/s40134-012-0006-4. URL: https://link.springer.com/article/10.1007/s40134-012-0006-4#citeas.

[15] A. C. Kak Slaney and Malcolm. *Principles of computerized tomographic imaging*. 1988. URL: https://engineering.purdue.edu/~malcolm/pct/pct-toc.html.

[16] C Stolk. *The Radon transform*. 2014. URL: https://staff.fnwi.uva.nl/c.c.stolk/FourierAnalysis2013/notes_Radon1.pdf.

[17] Kunio Tanabe. "Projection method for solving a singular system of linear equations and its applications". In: 17 (1971), pp. 203–214. DOI: 10.1007/bf01436376. URL: https://link.springer.com/article/10.1007/BF01436376.

[18] *Use of Kaczmarz's method in intelligent-particle swarm optimization*. URL: https://ieeexplore.ieee.org/abstract/document/6713898.

A

Matlab code SART

This Matlab code, which is used in Chapter 4, was adapted from [7] and uses the ASTRA Toolbox to generate reconstructions of the Shepp-Logan phantom using the SART algorithm.

A.1. SART function using the ASTRA Toolbox

This code includes the function to create SART reconstructions.

```
function [V, ssim_values, relative_errors, pixel_errors] = SART(d_high, d_low,
    noise_level, order, m, iter, J)
    % Generate high-resolution phantom
    V_exact_high = phantom(d_high);

    % Create high-resolution projection geometry
    proj_geom_high = astra_create_proj_geom('parallel', 1.0, d_high, linspace(0, pi, J)
    );
    vol_geom_high = astra_create_vol_geom(d_high, d_high);
    proj_id_high = astra_create_projector('linear', proj_geom_high, vol_geom_high);

    % Generate high-resolution sinogram
    [~, sinogram_high] = astra_create_sino(V_exact_high, proj_id_high);

    % Add Gaussian noise to the sinogram
    noise = noise_level * randn(size(sinogram_high));
    noisy_sinogram_high = sinogram_high + noise;

    % Create low-resolution projection geometry
    proj_geom_low = astra_create_proj_geom('parallel', 1.0, d_low, linspace(0, pi, J));
    vol_geom_low = astra_create_vol_geom(d_low, d_low);
    proj_id_low = astra_create_projector('linear', proj_geom_low, vol_geom_low);

    % Rescale the noisy sinogram to low resolution
    noisy_sinogram_low = imresize(noisy_sinogram_high, [size(noisy_sinogram_high, 1),
    d_low], 'bilinear', Dither=false,Antialiasing=false);

    % Perform ASTRA reconstruction
    recon_id = astra_mex_data2d('create', '-vol', vol_geom_low, 0);
    cfg = astra_struct('SART');
    cfg.ProjectorId = proj_id_low;
    cfg.ProjectionDataId = astra_mex_data2d('create', '-sino', proj_geom_low,
    noisy_sinogram_low);
    cfg.ReconstructionDataId = recon_id;
    cfg.option.ProjectionOrder = 'custom';
    cfg.option.ProjectionOrderList = order;
    cfg.option.MinConstraint = m;
```

```
    sart_id = astra_mex_algorithm('create', cfg);

    % Perform iterations
    astra_mex_algorithm('iterate', sart_id, iter);
    V = astra_mex_data2d('get', recon_id);

    % Clean up ASTRA resources
    astra_mex_algorithm('delete', sart_id);
    astra_mex_data2d('delete', recon_id);
    astra_mex_projector('delete', proj_id_high);
    astra_mex_projector('delete', proj_id_low);

    % Normalization
    max_val = max(V(:));
    normalised = V / max_val;

    % Resize exact high-resolution image for comparison
    V_exact_low = imresize(V_exact_high, [d_low, d_low], 'bilinear',Dither=false,
    Antialiasing=false);

    % Define margin for pixel comparison
    margin = 0.0050;

    % Calculate differences and errors
    diff_matrix_n = abs(V_exact_low - V) > margin;
    pixel_errors = sum(diff_matrix_n(:));
    ssim_values = ssim(V_exact_low, V);
    relative_errors = norm(V_exact_low - normalised, 'fro') / norm(V_exact_low, 'fro')
    * 100;
end
```

A.2. Generate Figure 4.11

Note that Figure 4.10 can be generated using the same code as below only using $d_{low} = 200$.

```
figure;

% Simple reconstruction with no noise
[V1, V1a, V1b, V1c] = SART(200, 100, 0, [0:5:175, 1:5:176, 2:5:177, 3:5:178, 4:5:179],
    -inf, 180, 180);
subplot(2,3, 1);
imshow(V1, []);
xlabel('(a)'); % Simple reconstruction
title(['SSIM:', num2str(V1a), '   RE:', num2str(V1b), '   PE:', num2str(V1c)]);

% Reconstruction with noise level 0.5
[V2, V2a, V2b, V2c] = SART(200, 100, 0.1, [0:5:175, 1:5:176, 2:5:177, 3:5:178,
    4:5:179], -inf, 180, 180);
subplot(2,3, 2);
imshow(V2, []);
xlabel('(b)'); % Noisy
title(['SSIM:', num2str(V2a), '   RE:', num2str(V2b), '   PE:', num2str(V2c)]);

% Reconstruction with noise level 0.1 and linear projection order
[V3, V3a, V3b, V3c] = SART(200, 100, 0.1, 0:179, -inf, 180, 180);
subplot(2,3, 3);
imshow(V3, []);
xlabel('(c)'); % Noise and linear projection order
title(['SSIM:', num2str(V3a), '   RE:', num2str(V3b), '   PE:', num2str(V3c)]);

% Reconstruction with noise level 0.1 and fewer angles
[V4, V4a, V4b, V4c] = SART(200, 100, 0.1, [0:5:40, 1:5:41, 2:5:42, 3:5:43, 4:5:44], -
    inf, 45, 45);
subplot(2,3, 4);
imshow(V4, []);
xlabel('(d)'); % Noise and fewer angles
title(['SSIM:', num2str(V4a), '   RE:', num2str(V4b), '   PE:', num2str(V4c)]);

```

```
% Reconstruction with noise level 0.1 and minimum constraint
[V5, V5a, V5b, V5c] = SART(200, 100, 0.1, [0:5:175, 1:5:176, 2:5:177, 3:5:178,
    4:5:179], 0, 180, 180);
subplot(2,3, 5);
imshow(V5, []);
xlabel('(e)'); % Noise and min constraint
title(['SSIM:', num2str(V5a), '   RE:', num2str(V5b), '   PE:', num2str(V5c)]);

% Reconstruction with noise level 0.1 and 50 iterations
[V6, V6a, V6b, V6c] = SART(200, 100, 0.1, [0:5:175, 1:5:176, 2:5:177, 3:5:178,
    4:5:179], -inf, 50, 180);
subplot(2,3, 6);
imshow(V6, []);
xlabel('(f)'); % Reconstruction with 50 iterations
title(['SSIM:', num2str(V6a), '   RE:', num2str(V6b), '   PE:', num2str(V6c)]);
```

B

Matlab code DART

This Matlab code, which is used in Chapter 5, was adapted from [7] and uses the ASTRA Toolbox to generate reconstructions DART algorithm.

B.1. DART algorithm

The following Matlab code shows the class which defines the DART algorithm.

```
classdef DART < handle
    properties
        gray_levels
        thresholds
        p
        rec_shape
        vol_geom
        all_neighbours_idx
        proj_geom
        projector_id
        sinogram
        sinogram_id
    end

    methods
        % constructor to initialize
        function obj = DART(gray_levels, p, rec_shape, proj_geom,vol_geom, projector_id
    , sinogram)
            obj.gray_levels = gray_levels;
            obj.thresholds = obj.update_gray_thresholds();
            obj.p = p;
            obj.rec_shape = rec_shape;
            obj.vol_geom = vol_geom;

            %all_neighbors_idx is a cell array with in each cell an array of indices
            obj.all_neighbours_idx = cell(rec_shape);
            for i = 1:rec_shape(1)
                for j = 1:rec_shape(2)
                    obj.all_neighbours_idx{i,j} = obj.pixel_neighborhood(rec_shape, i,
    j);
                end
            end

            obj.proj_geom = proj_geom;
            obj.projector_id = projector_id;
            obj.sinogram = sinogram;
            obj.sinogram_id = astra_mex_data2d('create', '-sino', proj_geom, sinogram);
```

```
36          end
37
38          function rec = run(obj, iters, rec_iter)
39              % Create initial reconstruction, zero vector
40              curr_rec = obj.SART(zeros(obj.rec_shape), [], rec_iter);
41
42              for i = 1:iters
43                  % Segment current reconstructed image
44                  segmented_img = obj.segment(curr_rec);
45                  % Calculate boundary pixels
46                  boundary_pixels = obj.boundary_pixels(segmented_img);
47                  % Calculate free pixels
48                  free_pixels = obj.free_pixels();
49                  % Mask of all free pixels, the boundary pixels + extra free
50                  % pixels from randomly picked with p
51                  free_pixels = boundary_pixels | free_pixels;
52                  % Take indexes of non-fixed pixels
53                  free_pixels_idx = find(free_pixels);
54                  % Fixed pixels
55                  fixed_pixels = ~free_pixels;
56                  % Take indexes of fixed pixels
57                  fixed_pixels_idx = find(fixed_pixels);
58                  % Create image to feed to reconstructor
59                  curr_rec(fixed_pixels_idx) = segmented_img(fixed_pixels_idx);
60                  % Run reconstruction algorithm on free pixels
61                  curr_rec = obj.SART(curr_rec, free_pixels, rec_iter);
62
63                  % Smoothing operation except on last iteration
64                  if i < iters
65                      smooth_rec = imgaussfilt(curr_rec, 1);
66                      curr_rec(free_pixels_idx) = smooth_rec(free_pixels_idx);
67                  end
68              end
69              rec = curr_rec;
70          end
71
72          function rec = SART(obj, rec, mask, iters)
73              if ~isempty(mask)
74                  free_pixels_idx = find(mask);
75                  fixed_rec = rec;
76                  fixed_rec(free_pixels_idx) = 0;
77                  [~, fixed_sino] = astra_create_sino(fixed_rec, obj.projector_id);
78                  free_sino = obj.sinogram - fixed_sino;
79                  free_sino_id = astra_mex_data2d('create', '-sino', obj.proj_geom,
    free_sino);
80                  rec_id = astra_mex_data2d('create', '-vol', obj.vol_geom, rec);
81              else
82                  rec_id = astra_mex_data2d('create', '-vol', obj.vol_geom, zeros(obj.
    rec_shape));
83              end
84
85              % Define configuration parameters
86              alg_cfg = astra_struct('SART');
87              alg_cfg.ProjectorId = obj.projector_id;
88
89              if ~isempty(mask)
90                  alg_cfg.ProjectionDataId = free_sino_id;
91              else
92                  alg_cfg.ProjectionDataId = obj.sinogram_id;
93              end
94
95              alg_cfg.ReconstructionDataId = rec_id;
96
97              alg_cfg.option.MinConstraint = 0;
98              alg_cfg.option.MaxConstraint = 255;
99              if ~isempty(mask)
100                 mask_id = astra_mex_data2d('create', '-vol', obj.vol_geom, mask);
101                 alg_cfg.option.ReconstructionMaskId = mask_id;
102             end
103
104             % Define algorithm
```

```
105             algorithm_id = astra_mex_algorithm('create', alg_cfg);
106             % Run the algorithm
107             astra_mex_algorithm('iterate', algorithm_id, iters);
108             % Get reconstructed values
109             rec = astra_mex_data2d('get', rec_id);
110             % Free memory
111             astra_mex_algorithm('delete', algorithm_id);
112         end
113
114         function thresholds = update_gray_thresholds(obj)
115             %first add black in there as lowerbound
116             thresholds = zeros(1,numel(obj.gray_levels));
117             for i = 1:numel(obj.gray_levels) - 1
118                 thresholds(i+1)=(obj.gray_levels(i)+obj.gray_levels(i+1))/2;
119             end
120             %also add white in there as upperbound
121             thresholds(end+1) = 255;
122         end
123
124         % Segment the image using the graylevels
125         function segmented_img = segment(obj, img)
126             segmented_img = zeros(size(img), 'uint8');
127             for i = 1:length(obj.thresholds)-1
128                 cond = (img >= obj.thresholds(i)) & (img < obj.thresholds(i+1));
129                 segmented_img(cond) = obj.gray_levels(i);
130             end
131         end
132
133         % Calculate neighborhood of pixel
134         function neighbours = pixel_neighborhood(~, img_shape, x, y)
135             neighbours = zeros(8,2);
136             count = 0;
137             % Look at pixels next to the evaluated pixel lying at (x,y), 8-connectivity
138             for i = (x-1):(x+1)
139                 for j = (y-1):(y+1)
140                     % Make sure the pixels we consider lie within the image
141                     if i > 1 && i <= img_shape(1) && j > 1 && j <= img_shape(2) && ~( i
    ==x && j==y)
142                         count = count +1;
143                         % Add a row to our list storing pixel with indices (i,j) as
    neighbor
144                         neighbours(count, :) = [i,j];
145                     end
146                 end
147             end
148             neighbours = neighbours(1:count, :);
149         end
150
151         function bool_mask = boundary_pixels(obj, img)
152             % Apply noise reduction to the image
153             img_smoothed = imgaussfilt(img, 1);
154             % Initialize the boundary mask
155             bool_mask = false(size(img));
156
157             % Define a threshold for edge detection
158             threshold = 10;
159
160             % Iterate over each pixel in the image
161             for x = 2:size(img, 1)-1
162                 for y = 2:size(img, 2)-1
163                     % Compute the difference between the central pixel and its
    neighbors
164                     diff = abs(img_smoothed(x, y) - img_smoothed(x-1:x+1, y-1:y+1));
165
166                     % If any neighbor's difference is greater than the threshold, mark
    the pixel as a boundary
167                     if any(diff(:) > threshold)
168                         bool_mask(x, y) = true;
169                     end
170                 end
171             end
```

```
        end

                function free_pixels = free_pixels(obj)
                    free_pixels = rand(obj.rec_shape) > obj.p;
                end
            end
        end
```

The following function is used to implement DART using the above class

```
function [rec, ssim_values, relative_errors, pixel_errors] = DART_crime(img, proj_geom,
     vol_geom, proj_id, sinogram, dart_iters, sart_iters,p,gray_levels)
    rec_shape = size(img);  % Assuming img is already defined in your workspace
    dart = DART(gray_levels, p, rec_shape, proj_geom, vol_geom, proj_id, sinogram);

    % Update gray levels and thresholds
    dart.gray_levels = single(unique(img));  % Find unique gray levels and convert to
    single precision
    %dart.thresholds = dart.update_gray_thresholds();  % Update thresholds
    rec = dart.run(dart_iters,sart_iters);

    rec(rec < 0) = 0;
    rec(rec > 255) = 255;
    % Define margin for pixel comparison
    margin = 0.0050;

    % Calculate differences and errors
    diff_matrix_n = abs(img - rec) > margin;
    pixel_errors = sum(diff_matrix_n(:));
    ssim_values = ssim(img, rec);
    relative_errors = norm(img - rec, 'fro') / norm(img, 'fro') * 100;
end
```

B.2. Function to generate FBP reconstructions

```
function [V, ssim_values, relative_errors, pixel_errors] = FBP_crime(img, vol_geom,
    proj_id, proj_geom, sino)
    % Create reconstruction data structure in ASTRA
    recon_id = astra_mex_data2d('create', '-vol', vol_geom, 0);

    % Configure the FBP algorithm
    cfg = astra_struct('FBP');
    cfg.ProjectorId = proj_id;
    cfg.ProjectionDataId = astra_mex_data2d('create', '-sino', proj_geom, sino);
    cfg.ReconstructionDataId = recon_id;
    cfg.option.MinConstraint = 0;
    cfg.option.MaxConstraint = 255;
    cfg.option.FilterType = 'ram-lak';
    fbp_id = astra_mex_algorithm('create', cfg);

    % Run the algorithm
    astra_mex_algorithm('run', fbp_id);

    % Retrieve the reconstructed volume
    V = astra_mex_data2d('get', recon_id);

    % Clip the values to enforce constraints
    V(V < 0) = 0;
    V(V > 255) = 255;

    % Clean up ASTRA resources
    astra_mex_algorithm('delete', fbp_id);
    astra_mex_data2d('delete', recon_id);
    astra_mex_data2d('delete', sino);

    % Define margin for pixel comparison
    margin = 0.0050;
```

```
32
33      % Calculate differences and errors
34      diff_matrix_n = abs(img - V) > margin;
35      pixel_errors = sum(diff_matrix_n(:));
36
37      % Structural Similarity Index (SSIM)
38      ssim_values = ssim(img, V);
39
40      % Relative Error Calculation
41      error_norm = norm(img - V, 'fro');
42      img_norm = norm(img, 'fro');
43      relative_errors = (error_norm / img_norm) * 100;
44  end
```

B.3. DART steps of reconstruction process as in Figure 5.1

```
1  figure;
2  d = 200;
3  phant = phantom(d);
4  img = phant *255;
5  angles = linspace(0, pi, 180);
6
7  subplot(3,3,1);
8  imshow(uint8(img));
9  xlabel('a) origional phantom');
10
11 rec_shape = size(img);
12 proj_geom = astra_create_proj_geom('parallel', 1, size(img, 1), angles);
13 vol_geom = astra_create_vol_geom(d,d);
14 projector_id = astra_create_projector('linear', proj_geom, vol_geom);
15 [~, sinogram] = astra_create_sino(img, projector_id);
16
17 gray_levels = single(unique(img));
18 dart = DART(gray_levels, 0.99, rec_shape, proj_geom, vol_geom, projector_id, sinogram);
19
20 % Update gray levels and thresholds
21 dart.gray_levels = single(unique(img));  % Find unique gray levels and convert to
       single precision
22 dart.thresholds = dart.update_gray_thresholds();  % Update thresholds
23
24 % Run the algorithm for 0 iterations
25 init_rec = dart.run(0, 100);
26
27 % Calculate the mean absolute pixel error
28 mean_abs_error = mean(abs(double(img(:)) - double(init_rec(:))));
29 fprintf('Mean absolute pixel error: %f\n', mean_abs_error);
30
31 % Display the image
32 subplot(3,3,2);
33 imshow(uint8(init_rec));
34 xlabel('b) initial reconstruction');
35 title(['PE:', num2str(mean_abs_error)]);
36
37 % Update thresholds
38 dart.thresholds = dart.update_gray_thresholds();
39
40 % Segmentation step
41 segmented_img = dart.segment(init_rec);
42
43 % Display initial gray levels
44 fprintf('Initial gray levels: ');
45 disp(dart.thresholds);
46
47 % Display gray levels in segmented image
48 unique_segmented_img_levels = unique(segmented_img);
49 fprintf('Gray levels in segmented image: ');
50 disp(unique_segmented_img_levels);
51
```

```
% Calculate and display mean absolute pixel error
mean_abs_error_segmented = mean(abs(double(img(:)) - double(segmented_img(:))));
fprintf('Mean absolute pixel error: %f\n', mean_abs_error_segmented);

% Display the segmented image
subplot(3,3,3);
imshow(uint8(segmented_img));
xlabel('c) segmentation');
title(['PE:', num2str(mean_abs_error_segmented)]);

% Calculate boundary pixels
boundaries = dart.boundary_pixels(segmented_img);

% Display the boundary pixels
subplot(3,3,4);
imshow(boundaries);
xlabel('d) boundary pixels');

% Calculate free pixels
free_pixels = dart.free_pixels();

% Display the free pixels
subplot(3,3,5);
imshow(free_pixels);
xlabel('e) free pixels');

% Combine boundary and free pixels
all_free_pixels = boundaries | free_pixels;

% Display the combined free pixels
subplot(3,3,6);
imshow(all_free_pixels);
xlabel('f) all free pixels');


% Fix gray levels and thresholds
dart.gray_levels = unique(img(:));
dart.gray_levels = cast(dart.gray_levels, 'double');
dart.thresholds = update_gray_thresholds(dart);

% Run the algorithm
full_rec = dart.run(10, 100);
mean_absolute_error_fin = mean(abs(img - full_rec), 'all');
disp(['Mean absolute pixel error: ', num2str(mean_absolute_error)]);

subplot(3,3,8);
imshow(uint8(full_rec));
xlabel('g) final reconstruction');
title(['PE:', num2str(mean_absolute_error_fin)]);
```

B.4. Tuning the parameters of DART as in Figure 5.5

```
d = 512;
% Initialize the Shepp-Logan phantom
img_phant = phantom(d);
img_phant = img_phant * 255;

% Initialize projections
proj_geom = astra_create_proj_geom('parallel', 1.0, d, linspace(0, pi, 90));
vol_geom = astra_create_vol_geom(d, d);
proj_id = astra_create_projector('linear', proj_geom, vol_geom);

% Create sinogram
[sinogram_id_phant, sinogram_phant] = astra_create_sino(img_phant, proj_id);
g = single(unique(img_phant));
%%
figure;
[D1, D1a, D1b, D1c] = DART_crime(img_phant, proj_geom, vol_geom, proj_id,
```

```
    sinogram_phant, 200, 3, 0.85, g);
subplot(2,3,1);
imshow(D1, []);
xlabel('a');
title(['SSIM:', num2str(D1a), '   RE:', num2str(D1b), '   PE:', num2str(D1c)]);
%%
[D2, D2a, D2b, D2c] = DART_crime(img_phant, proj_geom, vol_geom, proj_id,
    sinogram_phant, 100, 10, 0.85, g);
subplot(2,3,2);
imshow(D2, []);
xlabel('b');
title(['SSIM:', num2str(D2a), '   RE:', num2str(D2b), '   PE:', num2str(D2c)]);

%%
[D3, D3a, D3b, D3c] = DART_crime(img_phant, proj_geom, vol_geom, proj_id,
    sinogram_phant, 50, 100, 0.85, g);
subplot(2,3,3);
imshow(D3, []);
xlabel('c');
title(['SSIM:', num2str(D3a), '   RE:', num2str(D3b), '   PE:', num2str(D3c)]);

%%
[D4, D4a, D4b, D4c] = DART_crime(img_phant, proj_geom, vol_geom, proj_id,
    sinogram_phant, 10, 1000, 0.85, g);
subplot(2,3,4);
imshow(D4, []);
xlabel('d');
title(['SSIM:', num2str(D4a), '   RE:', num2str(D4b), '   PE:', num2str(D4c)]);

%%
[D5, D5a, D5b, D5c] = DART_crime(img_phant, proj_geom, vol_geom, proj_id,
    sinogram_phant, 10, 200, 0.85, g);
subplot(2,3,5);
imshow(D5, []);
xlabel('e');
title(['SSIM:', num2str(D5a), '   RE:', num2str(D5b), '   PE:', num2str(D5c)]);

%%
[D6, D6a, D6b, D6c] = DART_crime(img_phant, proj_geom, vol_geom, proj_id,
    sinogram_phant, 10, 200, 0.9, g);
subplot(2,3,6);
imshow(D6, []);
xlabel('f');
title(['SSIM:', num2str(D6a), '   RE:', num2str(D6b), '   PE:', num2str(D6c)]);
```

B.5. Varying the number of projections as in Figure 5.6

```
iters_sart = 200;
iters_dart = 10;
d = 300;

% Initialize phantoms
filename = 'cylinders.png';
img_cyl = imreadgs(filename);
img_cyl = imresize(img_cyl, [d, d]);
img_phant = phantom(d);
img_phant = img_phant * 255;

% Initialize projections
proj_geom_10 = astra_create_proj_geom('parallel', 1.0, d, linspace(0, pi, 10));
proj_geom_20 = astra_create_proj_geom('parallel', 1.0, d, linspace(0, pi, 20));

vol_geom = astra_create_vol_geom(d, d);
proj_id_10 = astra_create_projector('linear', proj_geom_10, vol_geom);
proj_id_20 = astra_create_projector('linear', proj_geom_20, vol_geom);

% Create sinograms
[sinogram_id_cyl_10, sinogram_cyl_10] = astra_create_sino(img_cyl, proj_id_10);
```

```
[sinogram_id_cyl_20, sinogram_cyl_20] = astra_create_sino(img_cyl, proj_id_20);
[sinogram_id_phant_10, sinogram_phant_10] = astra_create_sino(img_phant, proj_id_10);
[sinogram_id_phant_20, sinogram_phant_20] = astra_create_sino(img_phant, proj_id_20);

%%
figure;
[D1, D1a, D1b, D1c] = DART_crime(img_phant, proj_geom_20, vol_geom, proj_id_20,
    sinogram_phant_20, iters_dart, iters_sart);
subplot(3,4,1);
imshow(D1, []);
xlabel('(a)');
title(['SSIM:', num2str(D1a), '   RE:', num2str(D1b), '   PE:', num2str(D1c)]);

[D2, D2a, D2b, D2c] = DART_crime(img_phant, proj_geom_10, vol_geom, proj_id_10,
    sinogram_phant_10, iters_dart, iters_sart);
subplot(3,4,2);
imshow(D2, []);
xlabel('(b)');
title(['SSIM:', num2str(D2a), '   RE:', num2str(D2b), '   PE:', num2str(D2c)]);

[D3, D3a, D3b, D3c] = DART_crime(img_cyl, proj_geom_20, vol_geom, proj_id_20,
    sinogram_cyl_20, iters_dart, iters_sart);
subplot(3,4,3);
imshow(D3, []);
xlabel('(c)');
title(['SSIM:', num2str(D3a), '   RE:', num2str(D3b), '   PE:', num2str(D3c)]);

[D4, D4a, D4b, D4c] = DART_crime(img_cyl, proj_geom_10, vol_geom, proj_id_10,
    sinogram_cyl_10, iters_dart, iters_sart);
subplot(3,4,4);
imshow(D4, []);
xlabel('(d)');
title(['SSIM:', num2str(D4a), '   RE:', num2str(D4b), '   PE:', num2str(D4c)]);

%%
[V1, V1a, V1b, V1c] = SART(1,d, d, 0, 0, iters_sart, 20,1);
subplot(3,4,5);
imshow(V1, []);
xlabel('(e)');
title(['SSIM:', num2str(V1a), '   RE:', num2str(V1b), '   PE:', num2str(V1c)]);

[V2, V2a, V2b, V2c] = SART(1,d, d, 0, 0, iters_sart, 10,1);
subplot(3,4,6);
imshow(V2, []);
xlabel('(f)');
title(['SSIM:', num2str(V2a), '   RE:', num2str(V2b), '   PE:', num2str(V2c)]);

[V3, V3a, V3b, V3c] = SART(2,d, d, 0, 0, iters_sart, 20,1);
subplot(3,4,7);
imshow(V3, []);
xlabel('(g)');
title(['SSIM:', num2str(V3a), '   RE:', num2str(V3b), '   PE:', num2str(V3c)]);

[V4, V4a, V4b, V4c] = SART(2,d, d, 0, 0, iters_sart, 10,1);
subplot(3,4,8);
imshow(V4, []);
xlabel('(h)');
title(['SSIM:', num2str(V4a), '   RE:', num2str(V4b), '   PE:', num2str(V4c)]);

%%
[F1, F1a, F1b, F1c] = FBP_crime(img_phant, vol_geom, proj_id_20, sinogram_id_phant_20);
subplot(3,4,9);
imshow(F1, []);
xlabel('(i)');
title(['SSIM:', num2str(F1a), '   RE:', num2str(F1b), '   PE:', num2str(F1c)]);

[F2, F2a, F2b, F2c] = FBP_crime(img_phant, vol_geom, proj_id_10, sinogram_id_phant_10);
subplot(3,4,10);
imshow(F2, []);
xlabel('(j)');
title(['SSIM:', num2str(F2a), '   RE:', num2str(F2b), '   PE:', num2str(F2c)]);
```

```

[F3, F3a, F3b, F3c] = FBP_crime(img_cyl, vol_geom, proj_id_20, sinogram_id_cyl_20);
subplot(3,4,11);
imshow(F3, []);
xlabel('(k)');
title(['SSIM:', num2str(F3a), '   RE:', num2str(F3b), '   PE:', num2str(F3c)]);

[F4, F4a, F4b, F4c] = FBP_crime(img_cyl, vol_geom, proj_id_10, sinogram_id_cyl_10);
subplot(3,4,12);
imshow(F4, []);
xlabel('(l)');
title(['SSIM:', num2str(F4a), '   RE:', num2str(F4b), '   PE:', num2str(F4c)]);
```

B.6. Varying the number of angles as in Figure ??

```
iters_sart = 200;
iters_dart = 10;
d = 300;

% Initialize phantoms
filename = 'cylinders.png';
img_cyl = imreadgs(filename);
img_cyl = imresize(img_cyl, [d, d]);
img_phant = phantom(d);
img_phant = img_phant * 255;

% Initialize projections
proj_geom_20 = astra_create_proj_geom('parallel', 1.0, d, linspace(0, pi/2, 90));
proj_geom_10 = astra_create_proj_geom('parallel', 1.0, d, linspace(0, pi/3, 60));

vol_geom = astra_create_vol_geom(d, d);
proj_id_10 = astra_create_projector('linear', proj_geom_10, vol_geom);
proj_id_20 = astra_create_projector('linear', proj_geom_20, vol_geom);

% Create sinograms
[sinogram_id_cyl_10, sinogram_cyl_10] = astra_create_sino(img_cyl, proj_id_10);
[sinogram_id_cyl_20, sinogram_cyl_20] = astra_create_sino(img_cyl, proj_id_20);
[sinogram_id_phant_10, sinogram_phant_10] = astra_create_sino(img_phant, proj_id_10);
[sinogram_id_phant_20, sinogram_phant_20] = astra_create_sino(img_phant, proj_id_20);

%%
figure;
[V1, V1a, V1b, V1c] = DART_crime(img_phant, proj_geom_20, vol_geom, proj_id_20,
    sinogram_phant_20, iters_dart, iters_sart,0.85);
subplot(3,4,1);
imshow(V1, []);
xlabel('(a)');
title(['SSIM:', num2str(V1a), '   RE:', num2str(V1b), '   PE:', num2str(V1c)]);

[V2, V2a, V2b, V2c] = DART_crime(img_phant, proj_geom_10, vol_geom, proj_id_10,
    sinogram_phant_10, iters_dart, iters_sart,0.85);
subplot(3,4,2);
imshow(V2, []);
xlabel('(b)');
title(['SSIM:', num2str(V2a), '   RE:', num2str(V2b), '   PE:', num2str(V2c)]);

[V3, V3a, V3b, V3c] =  DART_crime(img_cyl, proj_geom_20, vol_geom, proj_id_20,
    sinogram_cyl_20, iters_dart, iters_sart,0.85);
subplot(3,4,3);
imshow(V3, []);
xlabel('(c)');
title(['SSIM:', num2str(V3a), '   RE:', num2str(V3b), '   PE:', num2str(V3c)]);

[V4, V4a, V4b, V4c] =  DART_crime(img_cyl, proj_geom_10, vol_geom, proj_id_10,
    sinogram_cyl_10, iters_dart, iters_sart,0.85);
subplot(3,4,4);
imshow(V4, []);
xlabel('(d)');
```

```
title(['SSIM:', num2str(V4a), '   RE:', num2str(V4b), '   PE:', num2str(V4c)]);

%%
[V1, V1a, V1b, V1c] = SART(1,d, d, 0, 0, iters_sart, 90,2);
subplot(3,4,5);
imshow(V1, []);
xlabel('(e)');
title(['SSIM:', num2str(V1a), '   RE:', num2str(V1b), '   PE:', num2str(V1c)]);

[V2, V2a, V2b, V2c] = SART(1,d, d, 0, 0, iters_sart, 60,3);
subplot(3,4,6);
imshow(V2, []);
xlabel('(f)');
title(['SSIM:', num2str(V2a), '   RE:', num2str(V2b), '   PE:', num2str(V2c)]);

[V3, V3a, V3b, V3c] = SART(2,d, d, 0, 0, iters_sart, 90,2);
subplot(3,4,7);
imshow(V3, []);
xlabel('(g)');
title(['SSIM:', num2str(V3a), '   RE:', num2str(V3b), '   PE:', num2str(V3c)]);

[V4, V4a, V4b, V4c] = SART(2,d, d, 0, 0, iters_sart, 60,3);
subplot(3,4,8);
imshow(V4, []);
xlabel('(h)');
title(['SSIM:', num2str(V4a), '   RE:', num2str(V4b), '   PE:', num2str(V4c)]);

%%
[F1, F1a, F1b, F1c] = FBP_crime(img_phant, vol_geom, proj_id_20,sinogram_id_phant_20);
subplot(3,4,9);
imshow(F1, []);
xlabel('(i)');
title(['SSIM:', num2str(F1a), '   RE:', num2str(F1b), '   PE:', num2str(F1c)]);

[F2, F2a, F2b, F2c] = FBP_crime(img_phant, vol_geom, proj_id_10,sinogram_id_phant_10);
subplot(3,4,10);
imshow(F2, []);
xlabel('(j)');
title(['SSIM:', num2str(F2a), '   RE:', num2str(F2b), '   PE:', num2str(F2c)]);

[F3, F3a, F3b, F3c] = FBP_crime(img_cyl, vol_geom, proj_id_20,sinogram_id_cyl_20);
subplot(3,4,11);
imshow(F3, []);
xlabel('(k)');
title(['SSIM:', num2str(F3a), '   RE:', num2str(F3b), '   PE:', num2str(F3c)]);

[F4, F4a, F4b, F4c] = FBP_crime(img_cyl, vol_geom, proj_id_10,sinogram_id_cyl_10);
subplot(3,4,12);
imshow(F4, []);
xlabel('(l)');
title(['SSIM:', num2str(F4a), '   RE:', num2str(F4b), '   PE:', num2str(F4c)]);
```

B.7. Noisy measurement reconstructions as in Figure 5.9 and 5.10

```
iters_sart = 200;
iters_dart = 10;
d = 512;

% Initialize phantoms
% filename = 'cylinders.png';
% img_phant = imreadgs(filename);
% img_phant = imresize(img_phant, [d, d]);
img_phant = phantom(d);
img_phant = img_phant * 255;

% Initialize projections
proj_geom = astra_create_proj_geom('parallel', 1.0, d, linspace(0, pi, 180));
vol_geom = astra_create_vol_geom(d, d);
```

```
proj_id = astra_create_projector('linear', proj_geom, vol_geom);
noises = [0, 3000, 4500, 9000];

% Create sinograms
% [sinogram_id_cyl, sinogram_cyl] = astra_create_sino(img_cyl, proj_id);
[sinogram_id_phant, sinogram_phant] = astra_create_sino(img_phant, proj_id);
%%
i = 1;
figure;
for noise = noises
    sinogram_phant_noisy = sinogram_phant + poissrnd(noise, size(sinogram_phant));
    grays = [0,255];

    [D1, D1a, D1b, D1c] = DART_crime(img_phant, proj_geom, vol_geom, proj_id,
    sinogram_phant_noisy, iters_dart, iters_sart, 0.9, grays);
    subplot(3,4,i);
    imshow(D1, []);
    xlabel(i);
    title(['SSIM:', num2str(D1a), '   RE:', num2str(D1b), '   PE:', num2str(D1c)]);

    [V1, V1a, V1b, V1c] = SART_crime(img_phant, vol_geom, proj_id, proj_geom,
    sinogram_phant_noisy, iters_sart);
    subplot(3,4,(i+4));
    imshow(V1, []);
    xlabel(i);
    title(['SSIM:', num2str(V1a), '   RE:', num2str(V1b), '   PE:', num2str(V1c)]);

    [F1, F1a, F1b, F1c] = FBP_crime(img_phant, vol_geom, proj_id, proj_geom,
    sinogram_phant_noisy);
    subplot(3,4,(8+i));
    imshow(F1, []);
    xlabel(8+i);
    title(['SSIM:', num2str(F1a), '   RE:', num2str(F1b), '   PE:', num2str(F1c)]);

    i = i + 1;
end

% Delete ASTRA objects
astra_mex_data3d('delete', sinogram_id_phant);
astra_mex_projector('delete', proj_id);
```

B.8. Varying grey levels as in Figures 5.12, 5.11 and 5.13

```
iters_sart = 200;
iters_dart = 10;
d = 512;

noises = [0.02, 0.1, 0.18, 0.26, 0.34, 0.4];

% Initialize phantoms
img_phant = phantom(d);
img_phant = img_phant * 255;

% filename = 'cylinders.png';
% img_phant = imreadgs(filename);  files
% img_phant = imresize(img_phant, [d, d]);

% Initialize projections
proj_geom = astra_create_proj_geom('parallel', 1.0, d, linspace(0, pi, 50));
vol_geom = astra_create_vol_geom(d, d);
proj_id = astra_create_projector('linear', proj_geom, vol_geom);

% Create sinograms
[sinogram_id_phant, sinogram_phant] = astra_create_sino(img_phant, proj_id);

%% overestimate
i = 1;
figure;
```

```
for noise = noises
    phant_grays = unique(img_phant(:));
    phant_grays(phant_grays==0)=1;
    noised_gray = phant_grays + (1 + phant_grays) * noise;

    [D1, D1a, D1b, D1c] = DART_crime(img_phant, proj_geom, vol_geom, proj_id,
    sinogram_phant, iters_dart, iters_sart, 0.9, noised_gray);
    subplot(2,3,i);
    imshow(D1, []);
    xlabel(i);
    title(['SSIM:', num2str(D1a), '   RE:', num2str(D1b), '   PE:', num2str(D1c)]);

    i = i + 1;
end
astra_mex_data2d('delete', sinogram_id_phant);
astra_mex_projector('delete', proj_id);
astra_mex_algorithm('clear');

%% underestimate
i = 1;
figure;
for noise = noises
    phant_grays = unique(img_phant(:));
    phant_grays(phant_grays==0)=1;
    noised_gray = phant_grays - (1 + phant_grays) * noise;

    [D1, D1a, D1b, D1c] = DART_crime(img_phant, proj_geom, vol_geom, proj_id,
    sinogram_phant, iters_dart, iters_sart, 0.9, noised_gray);
    subplot(2,3,i);
    imshow(D1, []);
    xlabel(i);
    title(['SSIM:', num2str(D1a), '   RE:', num2str(D1b), '   PE:', num2str(D1c)]);

    i = i + 1;
end
astra_mex_data2d('delete', sinogram_id_phant);
astra_mex_projector('delete', proj_id);
astra_mex_algorithm('clear');

%% over & underestimate
i = 1;
figure;
for noise = noises
    phant_grays = unique(img_phant(:));
    signs = 2 * (rand(size(phant_grays)) > 0.5) - 1;
    noised_gray = phant_grays + ((1 + phant_grays) .* signs * noise);

    [D1, D1a, D1b, D1c] = DART_crime(img_phant, proj_geom, vol_geom, proj_id,
    sinogram_phant, iters_dart, iters_sart, 0.9, noised_gray);
    subplot(2,3,i);
    imshow(D1, []);
    xlabel(i);
    title(['SSIM:', num2str(D1a), '   RE:', num2str(D1b), '   PE:', num2str(D1c)]);

    i = i + 1;
end
astra_mex_data2d('delete', sinogram_id_phant);
astra_mex_projector('delete', proj_id);
astra_mex_algorithm('clear');
```

B.9. Varying fix probability as in Figures 5.14, 5.15 and 5.16

```
iters_sart = 200;
iters_dart = 10;
d = 300;

% Initialize phantoms
% filename = 'cylinders.png';
```

```
% img_cyl = imreadgs(filename);  % Use imread for standard image files
% img_cyl = imresize(img_cyl, [d, d]);  % Use imresize for resizing
img_phant = phantom(d);
img_phant = img_phant * 255;

% Initialize projections
proj_geom = astra_create_proj_geom('parallel', 1.0, d, linspace(0, pi, 180));
vol_geom = astra_create_vol_geom(d, d);
proj_id = astra_create_projector('linear', proj_geom, vol_geom);

% Create sinograms
% [sinogram_id_cyl, sinogram_cyl] = astra_create_sino(img_cyl, proj_id);
[sinogram_id_phant, sinogram_phant] = astra_create_sino(img_phant, proj_id);
noises = [0, 1500, 3000, 4500, 7500, 9000];
grays = unique(img_phant(:));

%%
p = 0.5; % or 0.7, 0.9
figure;
i = 1;
for noise = noises
    sinogram_phant_noisy = sinogram_phant + poissrnd(noise, size(sinogram_phant));

    [D1, D1a, D1b, D1c] = DART_crime(img_phant, proj_geom, vol_geom, proj_id,
    sinogram_phant_noisy, iters_dart, iters_sart,p,grays);
    subplot(2,3,i);
    imshow(D1, []);
    xlabel(i);
    title(['SSIM:', num2str(D1a), '   RE:', num2str(D1b), '   PE:', num2str(D1c)]);

    i = i + 1;
end

% Delete ASTRA objects
astra_mex_data3d('delete', sinogram_id_phant);
astra_mex_projector('delete', proj_id);
```

C

Additional results Section 5.3

This appendix includes reconstructions of the phantom introduced in Figure 5.4 where the amount of noise both in the measurements and in the grey levels is varied as in Chapter 5.

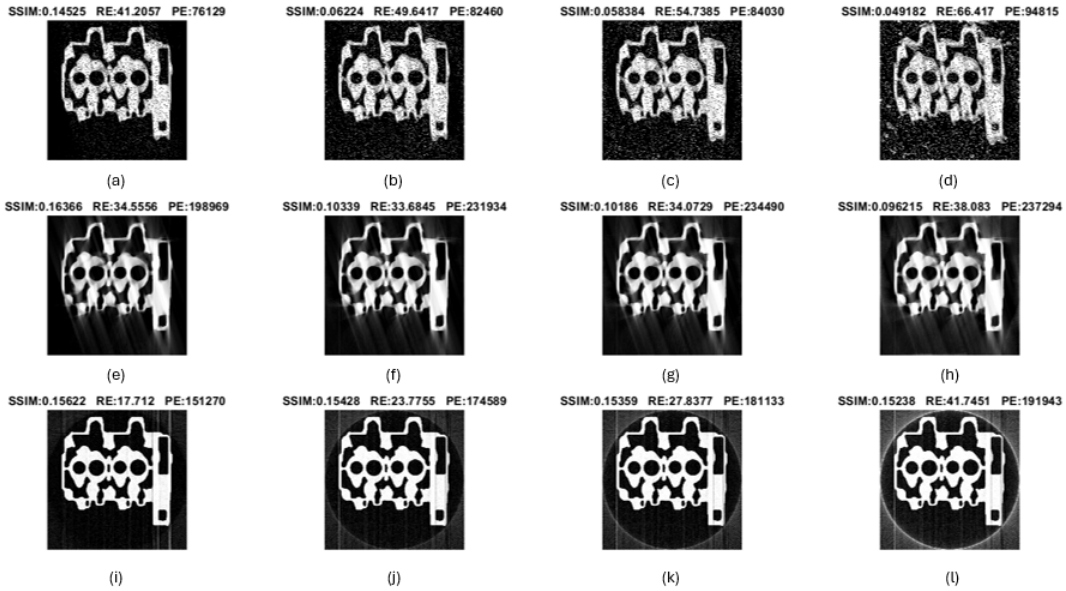


Figure C.1: Comparison of DART (row 1), SART (row 2) and FBP (row 3) for reconstruction of the phantom introduced in Figure 5.4 with measurements perturbed by a noise level of 0% (column 1), 5.6% (column 2), 8.4% (column 3) and 16.8% (column 4). (a) DART 0% noise (b) DART 5.6% noise (c) DART 8.4% noise (d) DART 16.9% noise (e) SART 0% noise (f) SART 5.6% noise (g) SART 8.4% noise (h) SART 16.9% noise (i) FBP 0% noise (j) FBP 5.6% noise (k) FBP 8.4% noise (l) FBP 16.9% noise.

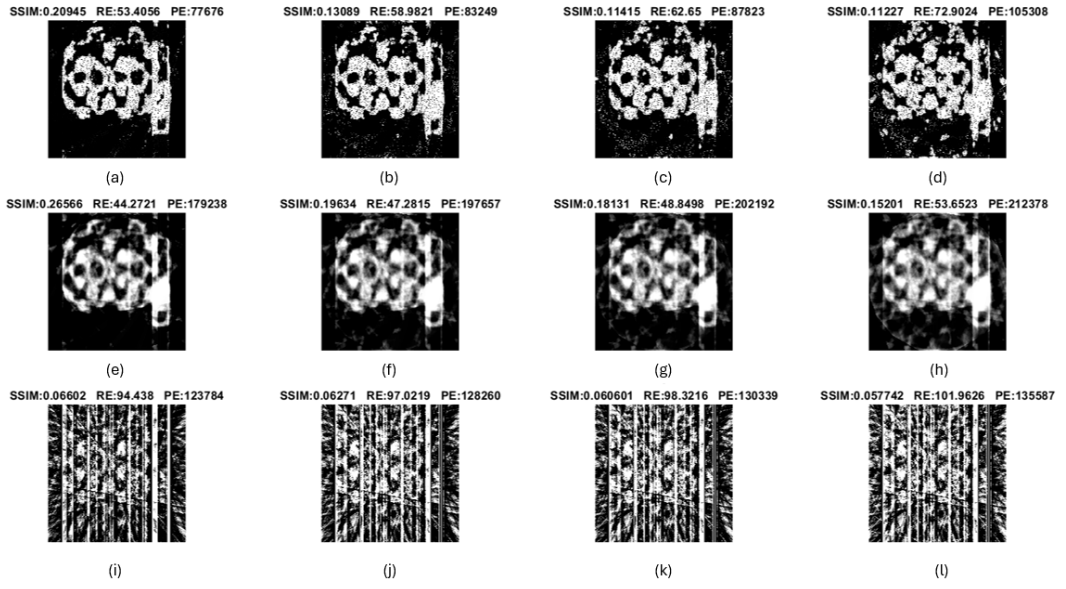


Figure C.2: Comparison of DART (row 1), SART (row 2) and FBP (row 3) for reconstruction of the phantom introduced in Figure 5.4 with measurements taken from 10 projections perturbed by a noise level of 0% (column 1), 5.6% (column 2), 8.4% (column 3) and 16.8% (column 4) of 10 projections. (a) DART 0% noise (b) DART 5.6% noise (c) DART 8.4% noise (d) DART 16.9% noise (e) SART 0% noise (f) SART 5.6% noise (g) SART 8.4% noise (h) SART 16.9% noise (i) FBP 0% noise (j) FBP 5.6% noise (k) FBP 8.4% noise (l) FBP 16.9% noise.

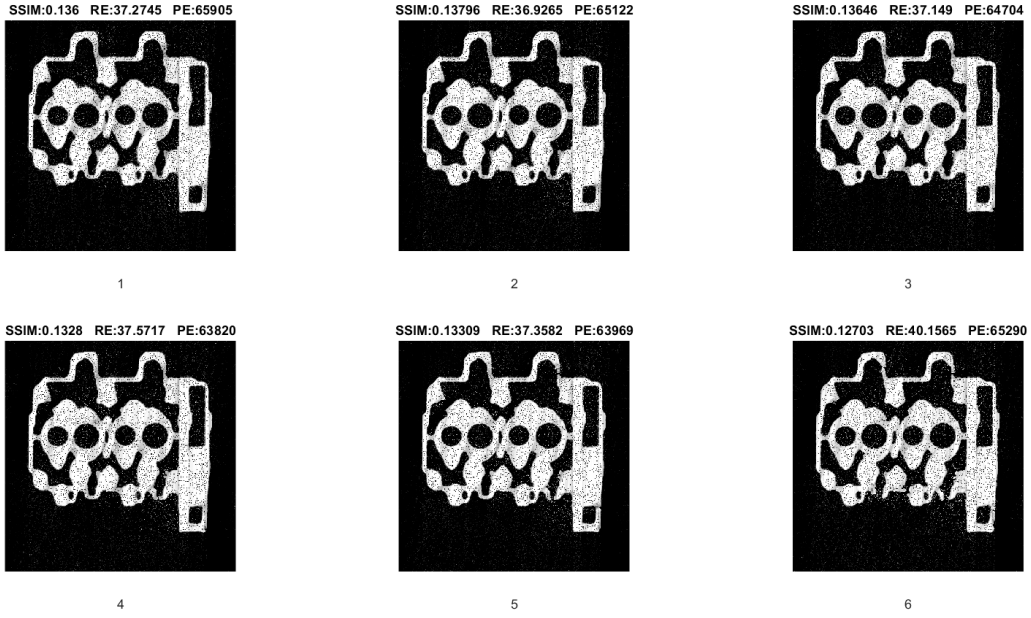


Figure C.3: Comparison on the effect of different levels of overestimation within the gray values used in DART showed on the phantom introduced in Figure 5.4. (1) overestimation of 0.02% (2) overestimation of 0.1 % (3) overestimation of 0.18 % (4) overestimation of 0.26 % (5) overestimation of 0.34 % (6) overestimation of 0.4.

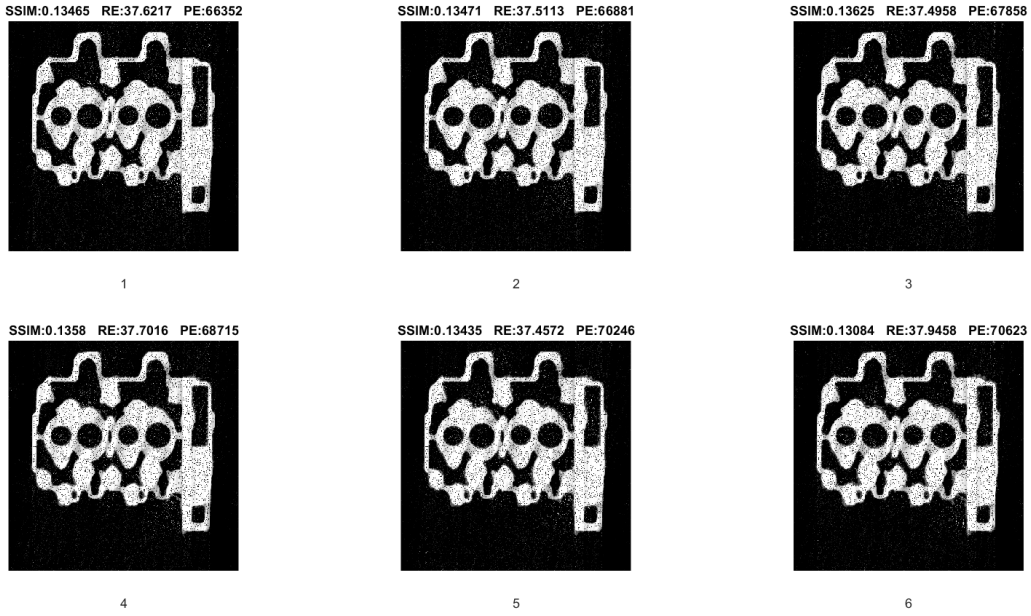


Figure C.4: Comparison on the effect of different levels of underestimation within the gray values used in DART showed on the phantom introduced in Figure 5.4 . (1) underestimation of 0.02% (2) underestimation of 0.1 % (3) underestimation of 0.18 % (4) underestimation of 0.26 % (5) underestimation of 0.34 % (6) underestimation of 0.4.

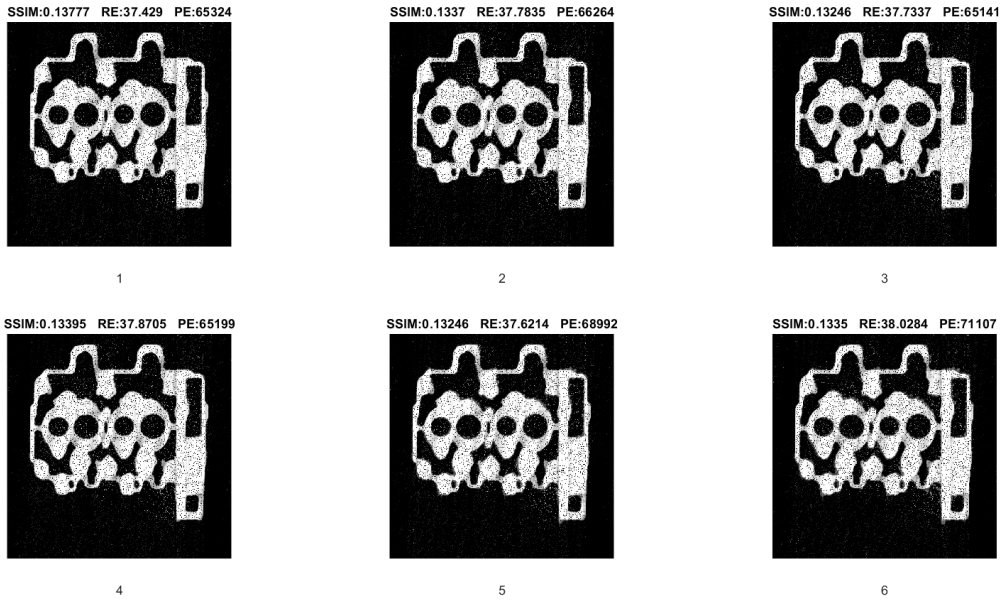


Figure C.5: Comparison on the effect of different levels of a combination of underestimation and overestimation within the grey values used in DART showed on the Shepp-Logan phantom. (1) estimation error of 0.02% (2) estimation error of 0.1 % (3) estimation error of 0.18 % (4) estimation error of 0.26 % (5) estimation error of 0.34 % (6) estimation error of 0.4%.