UNIVERSITY OF HELSINKI

Master's Thesis

Some fundamental concepts of discrete tomography

University of Helsinki

Erna Piila

May 2016

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta/Osasto — Fakultet/Sektion — Faculty	Laitos — Institution — Department
Faculty of Science	Department of Mathematics and Statistics

Tekijä — Författare — Author
Erna Piila

Työn nimi — Arbetets titel — Title

Some fundamental concepts of discrete tomography

Oppiaine — Läroämne — Subject
Mathematics

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Master's thesis	May 2016	68 p.

Tiivistelmä — Referat — Abstract

The word tomography refers to a variety of imaging methods where a penetrating wave is used to collect data about an unknown object of interest. The waves usually need to be sent through the object from a large number of different angles in order to have enough data for a successful reconstruction. The problems can be expressed in a form where the measured data is known to be equal to the unknown object (expressed as a function) multiplied by a known operator. Reconstructing either a two-, three-, or in case of dynamic tomography, four-dimensional image based on data is not a simple matter of inverting said operator. The measurement noise, which is always a factor in imaging situations, can be amplified greatly in the reconstruction, making the inverse problem in question ill-posed. To avoid this, some regularization method in which a stable, unique problem close to the original, ill-posed one, needs to be applied. A method called Tikhonov regularization is one of the most commonly used ones.

Discrete tomography differs from general tomography by limiting the objects or images being reconstructed to ones consisting of only a small set of different densities or colours. This a priori knowledge of the object makes it possible to make successful reconstructions based on a much smaller amount of data. Traditionally discrete tomography has only focused on making reconstructions of binary images but more recently algorithms have been developed that allow the number of different colours or densities to be as large as five. There are some very promising new algorithms in the field of discrete tomography but due to the requirements set by new applications, an ever-increasing number of researchers are working on new ones.

In this thesis a small, simulated example of tomographic reconstruction is made using both Tikhonov regularization and DART (discrete algebraic reconstruction technique), which is an algorithm of discrete tomography. Both methods give reasonably good results in all of the situations that were studied. It is found, however, that for an image fulfilling the requirements for using DART (small enough number of different colours), DART performs significantly better when the number of projection angles is decreased.

Avainsanat — Nyckelord — Keywords
Inverse problems, X-ray tomography, Discrete tomography

Säilytyspaikka — Förvaringsställe — Where deposited
Kumpula Campus Library

Muita tietoja — Övriga uppgifter — Additional information

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta/Osasto — Fakultet/Sektion — Faculty	Laitos — Institution — Department
Matemaattis-luonnontieteellinen	Matematiikan ja tilastotieteen laitos

Tekijä — Författare — Author

Erna Piila

Työn nimi — Arbetets titel — Title

Some fundamental concepts of discrete tomography

Oppiaine — Läroämne — Subject

Matematiikka

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Pro gradu -tutkielma	Toukokuu 2016	68 s.

Tiivistelmä — Referat — Abstract

Sanalla tomografia viitataan erilaisiin kuvantamismenetelmiin, joissa tuntemattomasta kappaleesta kerätään tietoa käyttäen läpäisevää aaltoa. Useimmissa tapauksissa aalto täytyy lähettää kappaleen lävitse monesta eri kulmasta, jotta saadaan kerättyä riittävästi tietoa hyvän rekonstruktion aikaansaamiseksi. Tämäntyyppiset ongelmat voidaan ilmaista muodossa, jossa mitattu data tiedetään yhtä suureksi kuin (funktion muodossa esitetty) tuntematon kappale kerrottuna tunnetulla operaattorilla. Joko kaksi-, kolmi- tai dynaamisen tomografian tapauksessa neliulotteisen rekonstruktion rakentaminen perustuen dataan ei onnistu yksinkertaisesti kääntämällä kyseinen operaattori. Mittaustilanteissa aina vaikuttavana tekijänä oleva kohina saattaa vahvistua rekonstruktiossa merkittävästi, tehden kyseessä olevasta inversio-ongelmasta huonostiasetetun. Jotta tämä pystytään välttämään, on käytettävä jotakin regularisaatiomenetelmää, jossa tutkitaan ongelmaa, joka on lähellä alkuperäistä, huonosti asetettua ongelmaa, mutta yksikäsitteinen ja stabiili. Tikhonov-regularisaationa tunnettu menetelmä on yksi näiden joukossa laajimmin käytetyistä.

Diskreetti tomografia poikkeaa yleisestä tomografiasta siten, että kappaleet tai kuvat joita yritetään rekonstruoida koostuvat siinä ainoastaan muutamista eri tiheyksistä tai väreistä. Tämä a priori-tieto mahdollistaa sen, että hyviä rekonstruktioita voidaan saada aikaan paljon pienemmän datamäärän perusteella. Alun perin diskreetti tomografia keskittyi ainoastaan binääristen kuvien rekonstruoimiseen, mutta viime vuosina on kehitetty uusia algoritmeja, jotka sallivat eri tiheyksien tai värien määrän olla jopa viisi. Jotkut uusista algoritmeista ovat hyvin lupaavia, mutta uusien sovellusten asettamien vaatimuksien takia yhä kasvava joukko tutkijoita työskentelee uusien algoritmien kehittämisen parissa.

Tässä tutkielmassa pieni, simuloitu esimerkki tomografisesta rekonstruktiosta on toteutettu käyttäen sekä Tikhonov regularisaatioa että DART-algoritmia, joka on eräs diskreetin tomografian algoritmi. Molemmat menetelmät toimivat hyvin kaikissa tutkielmassa käsitellyissä tilanteissa. Huomataan kuitenkin, että kuvalle, jossa eri värien määrä on riittävän pieni, DART antaa selkeästi parempia rekonstruktioita kun projektiokulmien määrää vähennetään.

Avainsanat — Nyckelord — Keywords

Inversio ongelmat, Röntgentomografia, Diskreetti tomografia

Säilytyspaikka — Förvaringsställe — Where deposited

Kumpulan tiedekirjasto

Muita tietoja — Övriga uppgifter — Additional information

ACKNOWLEDGEMENTS

I would like to thank professor Samuli Siltanen for being the supervisor of this thesis and Esa Niemi for being the other person to assess it as well as for all his practical help in the final days of this project. I would also like to thank Jyrki Piila, Hanne Kekkonen and Tero Suhonen for all their invaluable help and support during the excruciatingly long process of writing this thesis.

SOME FUNDAMENTAL CONCEPTS OF DISCRETE TOMOGRAPHY

ERNA PIILA

Contents

1. Introduction: About inverse problems

The type of problems where an object is known and the mission is to collect data about it are sometimes referred to as direct problems. The opposite of this are problems called inverse problems, where the object is unknown and the goal is to recover it as well as possible based on

Date: May 11, 2016.

known data, which is often in the form of observed measurements. In a literal sense of the word a very simple (although for the purposes of the mathematical field of inverse problems overly simplistic) practical example of an inverse problem would be a photograph and its negative. This is, however a *well-posed* problem as it is easy to go back and forth between the direct problem and the inverse one. It is not always trivial which is the direct problem and which the inverse one from a set of two problems that are inverse to each other. Sometimes it is merely a question of which one is simpler or was studied first. In the mathematical field of inverse problems where only *ill-posed* problems are ever really studied, out of the set of two problems that are inverse to each other one is well-posed and the other one is ill-posed. In this situation direct problem is the one that is well-posed [3], [4].

In the literal sense of the word inverse problems do often appear in mathematics even in situations where they are not necessarily studied as such. This is often the case with problems that are well-posed. In practice this means the kind of situation where we are trying to solve

$$Ax = m \tag{1.1}$$

where A is a given operator and m is measured data for x and A is invertible and its inverse is bounded. It is easy to see that

$$x = A^{-1}m. \tag{1.2}$$

For example Fourier- and Laplace-transforms are well-posed operators [1].

In [6], Jaques Hadamard defined well-posed problem as follows:

(i) There is at least one solution
(ii) The solution is unique (There is at most one solution)
(iii) The solution depends continuously on data

Inverse problem is thus ill-posed if one or more of these conditions do not apply. This means that it is not possible to move back and forth between the direct and inverse problems as it is with for example with Fourier- and Laplace-transforms and their respective inverses.

Ill-posedness in inverse problems in a sense in which they are studied in the mathematical field named thusly is most commonly caused by failing to fulfil the third criterion, which is often referred to as stability. Often even a relatively small disruption in data can lead to significant instability in the solution of an inverse problem. This difficulty is being addressed with various regularization methods, more on which later. As noise is inevitably present in all practical measurement situations, aforementioned equation

$$Ax = m$$

is in fact going to take the form

$$Ax + \varepsilon = m \tag{1.3}$$

where ε represents white noise. For an ill-posed problem the naïve reconstruction that was illustrated before is going to fail (below x represents the naïve reconstruction of x):

$$\mathrm{x} \approx A^{-1}m = A^{-1}(Ax + \varepsilon) = x + A^{-1}\varepsilon \tag{1.4}$$

When noise is non-existent, the naïve reconstruction gives perfect results. This is, however, never the case in real situations and the attempt for naïve reconstruction will often cause the noise to amplify to an extent where any results gained will be completely useless. This is an example of a so-called *inverse crime* [1].

Inverse problems are often approached by first testing the method that is going to be used for solving them with simulated data and only after that tackling the problem with real, measured data. This way the accuracy of the method can be evaluated and the possible problems are easier to recognize and to deal with. In practice most of the problems studied are continuous instead of being discrete as the data represented as a set of measured values might suggest. When an inverse problem is defined in terms of an infinate-dimensional function spaces and then discretized for practical purposes, a model error occurs. Since inverse problems are usually ill-posed, neglecting this error, which is often called inverse crime, may have serious consequences to the quality of the given solution. It is important that a different discretization is used while generating the data than what is used while performing the reconstruction. Using the same discretization would constitute an inverse crime as some reconstructions might provide perfect results with some particular discretization but fail completely with any other. Most often the object being measured is going to be constructed in a continuous way, meaning that for example changes in fabric density are going to happen smoothly, so any discretization is going to be artificial in any case. A very simple practical example of using a different discretization in this way could be in a problem of trying to recover a known function in which one parameter is time. To avoid inverse crime one would have to use different step length while generating the function than one does while reconstructing it. That is to say the points in time where the values are observed should be different in the reconstructed model than they are in the original one. Otherwise one might get results that are much better than they should be. Using the same grid is not the only way to commit an inverse crime. Also using the same mathematical model for both generating the simulated data and inverting it may lead to much better inversion results that would otherwise be obtained [1], [5].

As was mentioned before, the naïve reconstruction practically always fails when it is used for solving ill-posed inverse problems. In this introduction the nature of A is not specified and it is referred to only as a given operator. In a situation where A is a function, A^{-1} does not exist or is not continuous when the problem is ill-posed. In situations concerning discrete tomography, which are of particular interest in this thesis, or generally any finite dimensional discretization of an inverse problem, A is a matrix. In these cases the problem leading to failure of naïve reconstruction is most often the lack of stability, which is the result of inevitable noise in practical measurement. One possible way of obtaining good inversion results despite of measurement noise is by using some regularization method. The choice of methods depends on the nature of the inverse problem that is being solved as well as the tools available. For example MATLAB, which is a software commonly used for the numerical solving of inverse problems, works much better and faster with some regularization methods than it does with others. Also the speed with which the inversion results are needed can in many practical situations be a factor. Even with sophisticated mathematical software, processing amounts of data as large as some inversion problems require can take significant amounts of time. Some regularization methods may provide results much quicker than others, even if the results are not as good.

There are several choices of possible regularization methods. Some of the commonly used ones include Tikhonov regularization, Total Variation regularization (often referred to as TV-regularization), Landweber iteration and Truncated or Selective singular value decomposition (SVD). Even though different regularization methods should be used for different types of inverse problems, some of the methods are more often applicable than others. Truncated SVD and Tikhonov regularization are possibly the most commonly used ones. Tikhonov regularization is the method that is used for solving the practical problem studied in this thesis, and it is discussed later on. As good regularization methods as there are available, there are still issues that are waiting to be solved. Good parameter choices are often a challenge in many methods. For example the choice of parameter α in the formula

$$x^{approx} = \min x \left\{ \parallel Ax - m \parallel_2^2 + \alpha \parallel x \parallel_2^2 \right\} \tag{1.5}$$

for Tikhonov regularization is an open problem. There are possible methods for finding the value for α that gives the best results but they work uncertainly at best. Often the value is chosen based simply on what has in similar situations previously been found to give smallest error or what choice of parameter works best while the method was tried out with simulated data. While working with simulated data that is similar enough to the actual, measured data one can easily experiment with the value of α in order to find out which value (at least the ballpark) gives the best reconstruction [1], [3].

In this thesis the type of inverse problems that are mostly the object of interest are those revolving around tomography, where information such as the inner structure of an object is sought to recover from a set of projections. The object that is attempted to recover is seen as a function with a domain that is either discrete or continuous and a range that is a given discrete set of (usually) real numbers. The problem posed in tomography is to recover this function, sometimes even just partially or approximately, based on weighted sums in a discrete case or weighted integrals in a continuous case over subsets of its domain. In practical applications such as medical imaging it is common for these sums or integrals to only be known approximately as noise is always a factor when something is being measured in a practical situation. Computerized tomography (CT) which G. T. Herman and A. Kuba in their book Discrete Tomography: Foundations, Algorithms and Applications [2] refer to as general (in other words not discrete) tomography often deals with a very large number of projections. In discrete tomography, which is the main field of interest in this thesis, very few projections are in ideal situations used for recovering the function.

Discrete tomography with this particular name is a relatively young field, dating back to 1994 when Larry Shepp organized the first meeting devoted to it. The problems revolving around discrete tomography had obviously also been studied before that and it has been used for many practical applications much before the name of the field was established for example in fields of medicine, image-processing etcetera. A common problem in discrete tomography is to find out when a set of points on a plane is uniquely defined by for example vertical and horizontal projections and when possible trying to reconstruct it. This situation is often represented as a (in ideal situations square) matrix the elements of which are unknown. The problem is to recover this matrix based on its row and column sums. As using row and column sums only rarely gives a unique solution, additional information such as prior knowledge about the qualities of the reconstructed object or additional, for example diagonal projections are also often used. In the mathematical field of discrete tomography (as opposed to the words being used just to refer to non-continuous tomography) the images that are being reconstructed are often binary (situation consistent with aforementioned set of points on a plane), meaning that for each pixel there are only two possible values. Binary images translate into binary matrices (matrices, where only possible numbers are 0 and 1). This obviously makes the odds of finding a unique solution based on only small number of projections better. The number of projections needed for good reconstructions in different situations and the theory behind this are discussed later [2].

The main emphasis of this thesis is on x-ray tomography in situations where the attenuation values (which measure how much the intensity of the x-ray decreases when it passes through tissue) are known. The case where there are only two or three different values is of particular interest. In the first part of the thesis x-ray tomography is discussed both generally and in the kind of situation described above. The goal is to shed some light on how having prior information about the attenuation values could be used in tomographic reconstruction and to some extent also how this is relevant to practical situations in x-ray tomography. A practical problem of this type is solved in the last part of the thesis. It is done with simulated data to find out if the chosen method works well in this situation, in order to better find suitable parameters values as well as to make sure it is robust against noise. If the method is deemed good enough, a similar reconstruction could be made with real, measured data. The chosen method for the reconstruction is Tikhonov regularization with conjugate gradient method, the theory of which is also covered later in this thesis.

2. About tomography

2.1. **General things about tomography.** Even though x-ray tomography was historically the first method of medical imaging in use and also the method of tomography of the most interest in this thesis, it is by no means the only method. The word refers to any imaging method by a penetrating wave and the different methods are used for varying purposes including radiology, different fields of physics, biology and so forth. The history of tomography dates back to at least year 1895, when Wilhelm Röntgen discovered x-rays. X-ray tomography is a popular method of medical imaging to this day, even though since the 1970's several more computer-based techniques have begun challenging its position. The mathematical principles behind tomography were introduced by Johann Radon, after whom Radon transform that is discussed later on is named, in 1917 [8].

From a mathematical point of view the interesting part of any method of tomography is the reconstruction. In simple terms the method with which most tomographic machinery works is that there is a source, which sends off penetrating waves of some type to a detector, which measures what is left of the waves when they get there. The object of interest is placed between the source and the detector and the reconstruction of the object is made based on how much the waves have attenuated between leaving the source and reaching the detector. Different types of material have different attenuation values or in more simple terms they absorb different amounts of radiation, so the intensity of the wave reduces to a different degree depending on what kind of material it passes through. Generally if the object being reconstructed is not priorly known to be homogeneous and very regularly shaped, sending the waves from just one angle is not going to lead to a reconstruction that is in any way accurate. That is why several projections are generally required. The number of projections needed for a good reconstruction varies greatly depending on which tomography technique is in question. Different tomographic approaches can be used in situations where the number or the distribution of projections is inadequate in terms of classical tomography. Sparse tomography is a field dedicated to building reconstructions based on projections that are acquired at large intervals but where distances between projections are fairly even. Limited-angle tomography on the other hand seeks to build reconstructions from projections that are from less than full angle range, leaving parts of the object virtually invisible. As was already mentioned in the introduction, the number of projections commonly used in discrete tomography can be only a few whereas general tomography (which approaches the ideal situation of continuous tomography which is the basis for the mathematical theory) often uses hundreds of projections [1], [2].

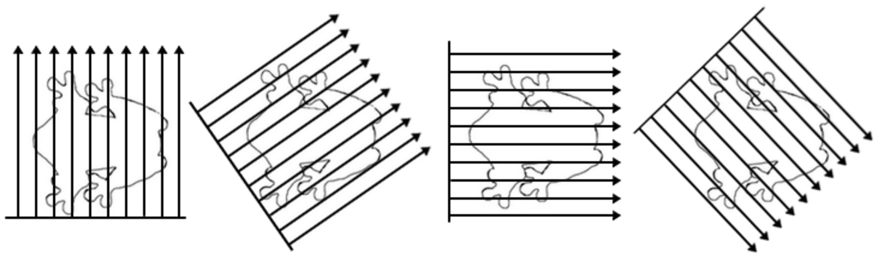


Figure 1. An unknown object being x-rayed from four different angles.

Mathematically tomographic reconstruction is an inverse problem where, as mentioned in the introduction, the object of interest is seen as a function. The mathematical theory is relatively different for discrete tomography and for general tomography (for example computerized tomography). The means for recovering the function are in discrete tomography based on line sums and in continuous tomography line integrals. Continuous tomography is a mathematical ideal that cannot happen in reality, some discretization always has to takes place in a practical imaging situation. When a method of tomography being used is one based on radiation (such as x-rays) and it is being used for medical purposes (or some other purpose where the object of interest is a living thing), it is necessary to keep the number of projections as small as possible to avoid excessive exposure to radiation. For the general public being aware of the dangers of radiation is a relatively new phenomenon; x-rays were not always only used for research purposes or when it was a medical necessity. Sometimes they were even used for very frivolous purposes; until 1970's an x-ray shoe fitter could be found in many shoe shops despite the warnings about dangers of radiation published in medical journals since the 1950's [1], [9].

2.2. **Mathematical theory of tomography.**

The mathematical theory of x-ray tomography is based on line integrals. The x-rays travel through the objects of interest in straight lines. The intensity of an x-ray is measured in a number of photons; a certain amount of photons leaves the x-ray source and a certain amount reaches the detector. Let us denote the intensity of the x-ray when it enters the object of interest by I_0 and when it exits it by I_1. Clearly $I_1 < I_0$ as the intensity of the radiation reduces when it passes through material. In the following example where the x-ray passes through several objects in a row, let the intensity of the radiation be I_{n-1} before and I_n after the n^{th} object. Now $I_0 > I_1 > ... > I_n$. Attenuation of radiation passing through material is exponential. Let us consider a situation where an x-ray passes through n identical, homogeneous objects, each of which absorbs 30% of the radiation. Now it is easy to see that $I_n = 0.7^n I_0$. Obviously not only the attenuation qualities of the material the x-ray passes through but also the length of the distance it travels in said material affect how much the intensity of the radiation reduces passing through the object.

Now let us denote that $I(x)$ is the intensity of the x-ray when the distance it has travelled within the object is x. Because the material the x-ray passes through is in most situations not homogeneous, we need to define a function that takes into account the different attenuation properties of the different materials. Let us mark this optical density function by $f(x)$. For a sufficiently small distance dx the reduced intensity $dI(x)$ is propotional to $f(x)$ by formula

$$dI(x) = -f(x)I(x)dx \tag{2.1}$$

Now if we consider that the intensity when the x-ray is entering the object is I_0 and exiting it I_1 where $I_0 > I_1$ as the intensity of the x-ray is attenuated when it passes through the object. The reduction in the

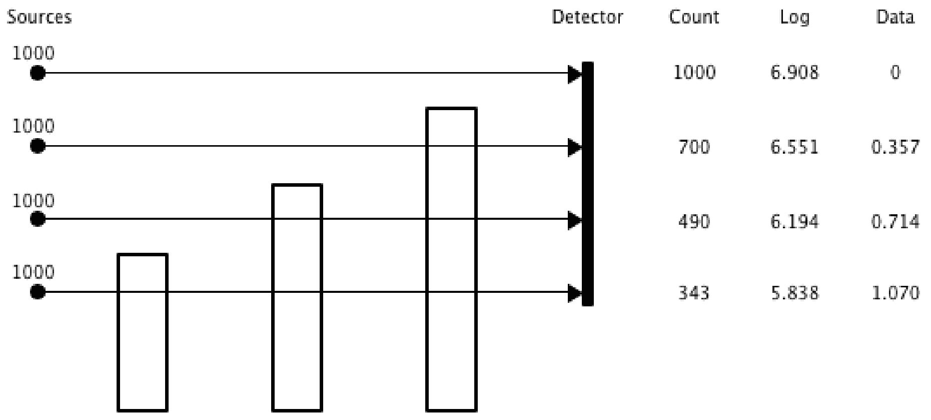


FIGURE 2. X-ray beams passing through three objects, each of which absorbs 30% of radiation that passes through it. The photon count of each of the beams when they leave the source is 1000. On the right of the image are the photon counts as measured by the detector, the logarithms of said photon counts and difference of the logarithm of the photon count to that of the unattenuated beam.

intensity is defined by

$$I_1 = I_0 e^{-R}, \text{where } R = \int f(x)\mathrm{d}x \tag{2.2}$$

Previous formula only gives information about the attenuation of a single x-ray, and in order to be able to make any tomographic reconstructions, multiple x-rays from various directions are almost always needed.

Let us next consider a two dimensional situation where an x-ray passes horizontally from left to right as demonstrated by the following picture:

The target (a two-dimensional slice of the object of interest) is placed in a square defined by $0 \leq x \leq 1$ and $0 \leq y \leq 1$. Let us consider the level on which the horizontal x-ray passes through the target object to be a constant that we denote by y_1. Now the initial intensity of the x-ray is $I_0 := I(0)$ and the intensity of the x-ray exiting the object is $I_1 := I(1)$. The intensity of the x-ray at point (x, y_1) is denoted by $I(x)$. The optical density function is defined much as before, except that now it is a function with two variables: $f(x, y)$. Also the formula that states the intensity's reduction's relation to the optical density remains essentially the same:

$$dI(x) = -f(x, y_1)I(x)dx \tag{2.3}$$

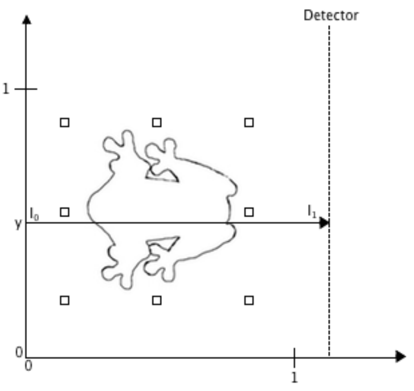


FIGURE 3. An x-ray beam passing through the object of interest (the little squares surrounding the object are not there on purpose).

The reduction in the intensity is otherwise defined exactly as before, but for the definition of R, which is now a function with two variables:

$$\int_0^1 f(x, y_1)dx = -\int_0^1 \frac{dI(x)}{I(x)}dx = -(\ln(I(1) - \ln(I(0))) = \ln I_0 - \ln I_1 \tag{2.4}$$

The expression on the left is the line integral along the x-ray from the source to the detector. Because I_0 is the intensity of the x-ray leaving the source and $I_1$is its intensity that is measured by the detector, both these values are known and the rightmost expression is easy to calculate (see figure 2). This is, however, a slightly idealized situation as in a real life measuring situation the data gathered from the detector is always noisy [1], [10].

2.2.1. *Radon transform.* The target is usually x-rayed from several directions and not all the x-rays pass through its centre. Let us denote the angle the normal vector of the x-ray makes with the x-axis with θ and the shortest distance of the x-ray passing through the target object (considering that the centre of the target object is placed on the point (0, 0) on the plane) from its centre with p. For compactly supported, continuous function $f : \mathbb{R}^2 \to \mathbb{R}$ that represents the unknown density the Radon transform is defined as follows:

$$\Re f(p, \theta) = \int_{-\infty}^{\infty} f(p\cos(\theta) + s\sin(\theta), p\sin(\theta) - s\cos(\theta))ds \tag{2.5}$$

where

$$\begin{bmatrix} p \\ s \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{2.6}$$

.

The Radon transform, the graphic representation of which is often referred to as a sinogram, is what x-ray machinery give as output data. The larger $\Re$ is, the more the radiation from that particular projection is attenuated. The mathematical problem of tomographic reconstruction is then to find the inverse of the function $\Re f(p, \theta)$ in order to reconstruct the target object. The problem with an inverse Radon transform is that it is not robust against measurement noise. To work around this problem different versions of finding a more stable approximations are often utilized. There are multiple computationally efficient inversion formulas available for the Radon transform. One of the most commonly used ones is called filtered back projection algorithm. For an easy way of inverting Radon transform we also need Fourier transform [1], [10], [11].

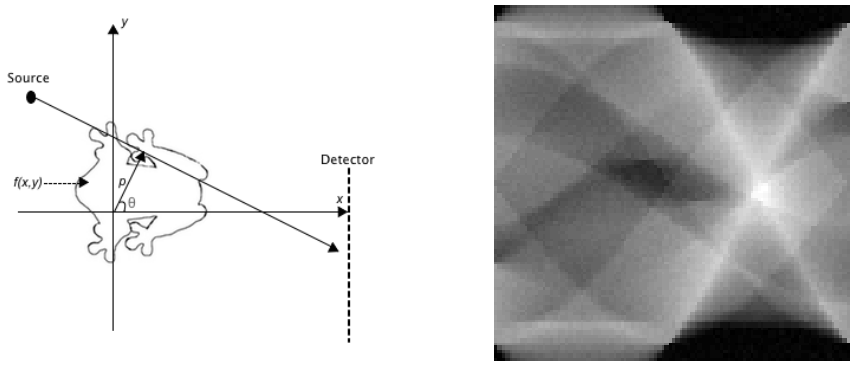


FIGURE 4. On the left an image of the geometry of the Radon transform and on the right an example of a sinogram.

2.2.2. *Fourier transform.* Fourier transform for $f : \mathbb{R}^n \to \mathbb{R}$ is defined by

$$\widehat{f}(\xi) = \int_{\mathbb{R}^n} f(x) e^{-i\xi \cdot x} dx \tag{2.7}$$

and its inverse by

$$f(x) = \frac{1}{(2\pi)^n} \int_{\mathbb{R}^n} \widehat{f}(\xi) e^{i\xi \cdot x} d\xi. \tag{2.8}$$

Radon transform is mostly used for reconstructing a two-dimensional ”slices” of a three-dimensional objects (sometimes three-dimensional tomographic reconstructions are also made), so the Fourier transform aiding with this is mostly used for $n = 2$. Sometimes three-dimensional tomographic reconstructions are also made [1], [10], [12].

2.2.3. *Fourier slice theorem.* To use this for inverting Radon transform we first need to find Fourier transform of a Radon transform where the angle θ is fixed and ξ is any real number. Let us denote this by $\widehat{\Re f}(\xi, \theta)$:

$$\widehat{\Re f}(\xi, \theta) = \int_{-\infty}^{\infty} \Re f(p, \theta) e^{-i\xi p} dp \tag{2.9}$$
$$= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(p\cos(\theta) + s\sin(\theta)), p\sin(\theta) - s\cos(\theta)) e^{-i\xi p} ds dp$$

Now because $(x, y) = (p\cos(\theta) + s\sin(\theta), p\sin(\theta) - s\cos(\theta))$ and $p = x\cos(\theta) + y\sin(\theta)$ a change of variables gives us

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-i\xi(x\cos(\theta)+y\sin(\theta))} dx dy = \widehat{f}(\xi\cos(\theta), \xi\sin(\theta)). \tag{2.10}$$

This result is called Fourier slice theorem. As can be seen it in this situation states that the one dimensional Fourier transform of the Radon transform equals the Fourier transform in two dimensions of the density function of which we originally took the Radon transform. This gives us a way of inverting the Radon transform. The Radon transform is known from detecting how much the x-ray attenuated between the source and the detector and its value is

$$\Re f(p, \theta) = \frac{\ln I_0}{\ln I_d} \tag{2.11}$$

where I_0 is the intensity of the x-ray leaving the source and I_d is its intensity arriving to the detector. Let us first apply inverse Fourier transform formula to the Fourier slice theorem:

$$f(x, y) = \frac{1}{4\pi^2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \widehat{f}(\alpha, \beta) e^{i(\alpha x + \beta y)} d\alpha d\beta \tag{2.12}$$
$$= \frac{1}{4\pi^2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \widehat{\Re f}(\xi, \theta) e^{i\xi(x\cos(\theta)+y\sin(\theta))} \mid \xi \mid d\xi d\theta$$
$$= \frac{1}{4\pi^2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \Re f(p, \theta) e^{i\xi(x\cos(\theta)+y\sin(\theta)-p)} \mid \xi \mid dp d\xi d\theta$$

The absolute value follows from the fact that

$$\widehat{\Re f}(-\xi, -\theta) = \widehat{\Re f}(\xi, \theta). \tag{2.13}$$

This method of inverting the Radon transform will in an ideal situation always recover the density function f when the Radon transform is known. It is, however, very unstable and even a small errors in the measurement can cause the reconstruction to become very inaccurate. There are more accurate and faster discrete methods available for reconstructing an image from its Radon transform [1], [10], [11].

2.3. **Reconstruction with discrete tomographic data.**

Just like in general tomography, the point of tomography with discrete data is to reconstruct the density function f, only whereas in continuous situation both the domain and the range of the function are continuous, in discrete case at least the range is discrete, specifically a finite set of real, nonnegative numbers. In a reconstructed model this is always the case, no matter how high the resolution an image that is recovered is always formed of pixels. The target object is certainly not always discrete so in order to apply methods of tomography with discrete data the x-rayed area has to be artificially divided into "pixels". When working with simulated data an image can also be further discretized (the resolution lowered) in order to make the number of calculations necessary smaller. For high-resolution images it can take a long time for even reasonably powerful computers to run the programs. To attempt discrete tomographic reconstruction in two dimensions, each pixel of the target area is numbered (going column by column, for example) and each pixel is assumed to have a constant value. These nonnegative values are denoted with $\boldsymbol{f}_j, j = 1, \ldots, n$ where n is the number of pixels. The measurement of the line integrals along the x-rays can now be approximated by sums where a_{ij} is the distance the x-ray denoted by L_i travels within the j^{th} pixel.

$$\boldsymbol{m}_i = \int_{L_i} f(x,y)ds \approx \sum_{j=1}^{n} a_{ij}\boldsymbol{f}_j \tag{2.14}$$

On each sum only the pixels that are intersected by the x-ray in question are included [1], [2].

For k measurements in $\boldsymbol{m} \in \mathbb{R}^k$ the previous sum gives the equation

$$A\boldsymbol{f} = \boldsymbol{m} \tag{2.15}$$

where a_{ij} form the matrix A in a way that is explained in the following example:

2.16. **Example.** Let us divide a square shaped area within which the object we are trying to reconstruct is into (for the sake of being able to still keep the matrix A small enough for the example to be demonstrative) 9 pixels. Within each of the pixels the attenuation value remains constant. Let the number of projections be 2, one horizontal and one that for one pixel that it travels horizontally from right to left travels vertically down three pixels. Let the number of x-ray beams for each projection be 3. Now the number of measurements is 6. The pixels are numbered column by column from left to right.

Now the distance each of the x-ray beams giving the measurements m_1, m_2 and m_3 travels within each pixel it passes through is $\frac{\sqrt{10}}{3}$. For the beams with measurements m_4, m_5 and m_6 the distance each beam travels within each pixel is 1. The number of columns in the measurement matrix A is the number of pixels in the image and the number of rows the number of measurements. The equation $A\boldsymbol{f} = \boldsymbol{m}$

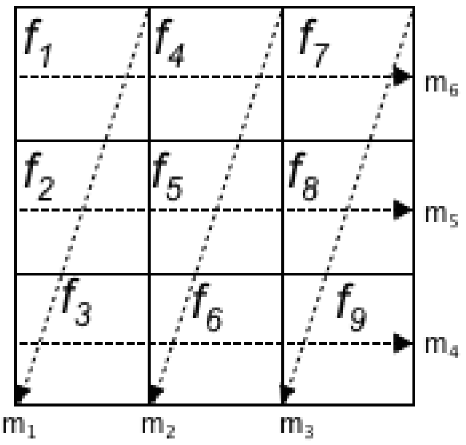


FIGURE 5. In this example the length of the side of each pixel is considered to be 1.

now takes the form
(2.17)

$$\begin{bmatrix} \frac{\sqrt{10}}{3} & \frac{\sqrt{10}}{3} & \frac{\sqrt{10}}{3} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\sqrt{10}}{3} & \frac{\sqrt{10}}{3} & \frac{\sqrt{10}}{3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{\sqrt{10}}{3} & \frac{\sqrt{10}}{3} & \frac{\sqrt{10}}{3} \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \\ f_9 \end{bmatrix} = \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ m_5 \\ m_6 \end{bmatrix}$$

Many more projections are usually needed for the solution to the equation to be unique. The number of projections needed depends on the situation, what level of accuracy is expected of the reconstruction and the chosen reconstruction method. There are methods with which by having some prior information about the image or by making assumptions about it one can use much lower number of projections and still get a good results. Of course one does not always have the kind of prior information needed and wrong initial assumptions are going to compromise the chances of getting any useful results at all. Normally the projections are chosen by taking them at regular intervals over 180 degrees (because after that the projections are just going to be repeated). As the number of rows in the matrix A is the number of the parallel x-ray beams times the number of projections and the number of beams is in real situations also much larger, the size of A increases quickly as the number of projections is increased. This is the main reason why running certain reconstruction algorithms can sometimes be very time consuming [1], [2].

It is easy to see that the naïve inversion $\boldsymbol{f} = A^{-1}\boldsymbol{m}$ will not work in this situation as A is not a square matrix and can thus not be invertible. An inversion can be attempted using some generalized inverse such as *Moore-Penrose pseudoinverse* for A, often denoted by A^{+}. A simple way of finding this pseudoinverse is described in definition 3.6. Pseudoinverses can also be found by finding a *left inverse* or a *right inverse* in the following manner: If A^TA is invertible, this pseudoinverse is given by formula $(A^TA)^{-1}A^T$. If on the other hand AA^T is invertible, the formula that gives the pseudoinverse is $A^T(AA^T)^{-1}$. Using this the measurement matrix not being square does not prevent us from trying the naïve inversion:

$$\boldsymbol{f} \approx (A^TA)^{-1}A^T\boldsymbol{m} \text{ or } \boldsymbol{f} \approx A^T(AA^T)^{-1}\boldsymbol{m} \tag{2.18}$$

This type of inversion method as well as the normal naïve inversion if the matrix A were invertible and such method could be used can give almost perfect inversion results when applied to noise free data. As will be further discussed in section 3.2, these methods are not robust against the inevitable measurement noise and adding even small amount of noise can make the reconstruction useless. The failure of naïve reconstruction is discussed more later on. Sensitivity to noise is not the only problem with naïve inversion. While working with simulated data trying to find $\boldsymbol{f}$ simply by multiplying the measurement vector $\boldsymbol{m}$ by the inverse or pseudoinverse of A is an example of an inverse crime. In this situation the inverse crime is constituted of not changing the discretization of the image between creating the data and reconstructing it. To avoid the inverse crime in the previous example's situation one could change the unrealistically crude 3x3 discretization to one (for example) twice as fine, 6x6 (which obviously would still be unrealistically crude). In practice this means twice as many x-ray beams from each measurement angle. The resulting data is now crime free and naïve reconstruction, doomed to fail as it may be, can be applied [1], [13].

3. Tomographic reconstruction

As mentioned earlier, there are a number of different methods of reconstruction that are widely used in tomography. Different applications have different requirements for the quality of reconstruction as well as for how quickly the reconstruction needs to be acquired. Often there can also be limits set by the obtained data, it can be very noisy, the number of projections can be small or the angle of projections limited. Due to its noise amplifying qualities, simply inverting the Radon transform will not give satisfactory reconstructions in real life situations where the tomographic reconstruction problems are always ill-posed. In order to obtain good reconstructions even from noisy, imperfect data, different regularization methods are used. The basic idea behind regularization is that instead of treating a problem that due to its ill-posedness is not uniquely solvable, we try to find a nearby problem, for example by introducing additional information, that has a unique solution. By solving that, we can then find an approximate solution to the original problem.

For a standard form inverse problem $A\boldsymbol{f}+\varepsilon = \boldsymbol{m}$ where the measurement noise $\varepsilon > 0$, the best solution that can be hoped for is to find $\boldsymbol{f}$ approximately. Due to the measurement noise, the naïve way of trying to solve the problem by inverting A, given that it even is invertible and its inverse continuous, and getting the crude approximate solution of $\boldsymbol{f} \approx A^{-1}\boldsymbol{m}$ is certainly going to fail as even if the measurement noise is taken into account, the method will most likely amplify the noise. Following is a brief introduction to some of the most commonly used methods for finding noise robust solutions.

3.1. **Singular value decomposition.**

Even though previously discussed mathematical theory of tomography is based on a continuous model $\mathcal{A}f + \varepsilon = \boldsymbol{m}$, in practice it is necessary to approximate the continuous situation with a discrete model of the form $A\boldsymbol{f} + \varepsilon = \boldsymbol{m}$, where A is a matrix, $\boldsymbol{f} \in \mathbb{R}^n$ and $\boldsymbol{m} \in \mathbb{R}^k$. A factorization of a matrix called *singular value decomposition* (SVD) is not in itself a regularization method but it is a useful tool for many methods that employ least squares fitting of data. In linear algebra an orthogonal matrix U is a matrix such that

$$U^TU = UU^T = I \tag{3.1}$$

and any matrix $A \in \mathbb{R}^{k\times n}$ can be written in the form

$$A = UDV^T \tag{3.2}$$

where both $U \in \mathbb{R}^{k\times k}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal matrices and $D \in \mathbb{R}^{k\times n}$ is a diagonal matrix. If the matrix A had complex values, instead of orthogonal matrices U and V would be unitary matrices and instead of the normal transpose of V the last term would be its conjugate transpose. UDV^T is called the singular value decomposition of A and the non-zero elements of D, denoted by d_j, are called the *singular values* of A [1], [14], [16].

If $k = n$ and the matrix D is square-shaped, the elements d_j are placed on the diagonal from the upper left corner of the matrix (element d_{11} in standard notation, denoted d_1 in a diagonal matrix) to the lower right corner (element d_{kk} in standard notation, denoted simply by d_k in case of a diagonal matrix). The rest of the elements in the matrix D are zero. If $k \neq n$ the structure of a diagonal matrix is a bit less intuitively clear. If $k > n$, the singular values d_j are situated diagonally from d_1 in the upper left corner to d_n in the spot that in standard notation would be d_{nn} and all the elements on the rows from row r_{n+1} to the row r_k are zero. If $k < n$, the singular values run diagonally from d_1 to d_k in exactly the same manner, the element d_k being on the spot that would normally be denoted with d_{kk}. All the elements in the columns form column c_{k+1} to column c_n are zero. Singular values are non-negative and in decreasing order:

$$d_1 \geq d_2 \geq \cdots \geq d_{min(k,n)} \geq 0 \tag{3.3}$$

To calculate a singular value decomposition, which is not necessarily unique, for a matrix one needs to find the eigenvalues and eigenvectors of AA^T and A^TA, the eigenvectors of which make up the columns of U and V respectively. The singular values are square roots of the eigenvalues of AA^T and A^TA.

A non-zero vector $\boldsymbol{v}$ is an eigenvector of a square matrix W if

$$W\boldsymbol{v} = \lambda\boldsymbol{v} \tag{3.4}$$

where λ is a scalar called the eigenvalue that corresponds to the eigenvector $\boldsymbol{v}$. To find the needed eigenvalues and eigenvectors one must first calculate matrices AA^T and A^TA and then find the value of eigenvalue λ from the following equation:

$$(AA^T)\boldsymbol{v} = \lambda\boldsymbol{v} \Rightarrow (AA^T)\boldsymbol{v} - \lambda\boldsymbol{v} = 0 \Rightarrow (AA^T - \lambda I)\boldsymbol{v} = 0$$

or correspondingly for A^TA:

$$(A^TA)\boldsymbol{v} = \lambda\boldsymbol{v} \Rightarrow (A^TA - \lambda I)\boldsymbol{v} = 0.$$

This equation only has a unique solution when the determinant $\mid AA^T - \lambda I \mid= 0$ (and correspondingly $\mid A^TA - \lambda I \mid= 0$) from which the potential eigenvalues can be found. The number of eigenvalues depends on the size of the square matrices AA^T and A^TA, each matrix gives as many eigenvalues as the number of its rows (or columns). All the eigenvalues are not necessarily separate.

Now the eigenvectors associated with each of the eigenvalues can be found by placing the values of λ into the equation

$$(AA^T - \lambda I)\boldsymbol{v} = 0 \text{ (or}(A^TA - \lambda I)\boldsymbol{v} = 0)$$

The eigenvectors gained from the matrix AA^T form the columns of the matrix U and the vectors gained from the matrix A^TA form the columns of the matrix V in the singular value decomposition. Finally the singular values d_n are square roots of the eigenvalues, placed on the diagonal from the upper left to the lower right corner of the matrix D

in a decreasing order [1], [14], [16].

In practice the singular value decomposition is of course never calculated by hand, even for a very small measurement matrix it would take a ridiculously long time. It can, however, be easily done using any number of mathematics softwares. Ill-posedness of a reconstruction problem can be detected from the singular value decomposition of the measurement matrix, a problem is ill-posed if the singular values gradually decrease to zero and if the ratio between the largest and the smallest non-zero singular values is very large.

3.2. **Truncated singular value decomposition.** Singular value decomposition in itself is a nice method of detecting ill-posedness in inverse problem and it can be used to calculate a *pseudoinverse* of a matrix, but to attempt the actual regularized inversion, one needs more tools. SVD is used as a part of several regularization methods but most closely related to it is truncated singular value decomposition, often called just truncated SVD or even just TSVD. TSVD is also probably the most straightforward and easiest to implement method of regularized inversion.

In truncated SVD only t biggest singular values are taken into account and the rest of the values are set equal to zero. This means also using only the first t columns of matrices U and V. Larger singular values contribute more to forming the matrix A than the smaller ones. Depending on how many of the singular values are discarded, calculating TSVD is much more economical than calculating the full SVD. The value of t chosen can depend on what properties are prioritized most in the situation at hand. The more singular values are cut off, the quicker the SVD for the remaining part is to calculate, but at the same time the approximation of the original matrix gets less accurate. Sometimes some of the singular values are significantly larger that the rest of them, this may offer a very natural point after which the rest of the singular values can be discarded. Often the choice just comes down to what is valued more, time or accuracy [1], [15].

TSVD can be used for solving the linear least squares problem

$$\min \parallel A\boldsymbol{f} - \boldsymbol{m} \parallel^2 \tag{3.5}$$

where A is a matrix, $\boldsymbol{f} \in \mathbb{R}^n$ and $\boldsymbol{m} \in \mathbb{R}^k$. By using TSVD additional requirements are imposed to the solution to make the norm as small as possible. This way the effects of noise can ideally be lessened, as the smallest singular values that in TSVD are discarded are often result of noise in the measurement. To find the minimum norm solution, Moore-Penrose pseudoinverse of A can be used. The pseudoinverse can be calculated using singular value decomposition and it is usually denoted by A^+.

3.6. **Definition.** If $A = UDV^T$ is the singular value decomposition of A, then $A^+ = VD^+U^T$, where

$$D^+ = \begin{bmatrix} \frac{1}{d_1} & 0 & \dots & 0 & \dots & 0 \\ 0 & \frac{1}{d_2} & & & & \vdots \\ \vdots & & \ddots & & & \\ & & & \frac{1}{d_r} & & \\ \vdots & & & & 0 & \vdots \\ 0 & \dots & & & \dots & 0 \end{bmatrix} \in \mathbb{R}^{n\times k} \tag{3.7}$$

and r is the largest index with which the singular value d_n is still greater than zero [13].

Simply using the pseudoinverse to find $\boldsymbol{f}$ will generally not lead to satisfactory results especially if the singular values approach zero quickly or $1/d_1$ is much smaller than $1/d_r$ as these are indicators that the problem is ill-posedness and thus the solution is most likely not stable. Using truncated SVD will help with this by discarding the elements that lead to amplification of noise in the reconstruction. Truncated singular value decomposition is defined similarly to pseudoinverse A^+ but by replacing all the reciprocals of the singular values in D^+ from an index α onwards with zero:

3.8. **Definition.** Truncated SVD, denoted by A^+_α, is defined as follows:
(3.9)

$$A^+_\alpha = VD^+_\alpha U^T, \text{ where} D^+ = \begin{bmatrix} \frac{1}{d_1} & 0 & \dots & 0 & \dots & 0 \\ 0 & \frac{1}{d_2} & & & & \vdots \\ \vdots & & \ddots & & & \\ & & & \frac{1}{d_{r_\alpha}} & & \\ \vdots & & & & 0 & \vdots \\ 0 & \dots & & & \dots & 0 \end{bmatrix} \in \mathbb{R}^{n\times k}$$

and r_α is smallest of the values of r with which the corresponding singular value is still greater than α:

$$r_\alpha = \min\Big\{r, \max\{j \mid 1 \le j \le \min(k,n), d_j > \alpha\}\Big\} \tag{3.10}$$

The reconstruction function corresponding to this, denoted by $\mathcal{L}_\alpha$, can now be defined by the formula

$$\mathcal{L}_\alpha(\boldsymbol{m}) = VD^+_\alpha U^T \boldsymbol{m} = \sum_{i=1}^{r_\alpha} \frac{\boldsymbol{u_i}^T \boldsymbol{m}}{d_i} \boldsymbol{v_i} \tag{3.11}$$

The bigger the value of α is, the fewer singular values are kept and the less the measurement noise gets amplified in the reconstruction. The problem with TSVD is knowing where to cut as keeping too few singular values leads the solution not being an accurate approximation of the original matrix [1], [15].

3.3. **Tikhonov regularization.**

Tikhonov regularization is possibly the most widely used regularization method of ill-posed inverse problems. It is named after Andrey Tikhonov but has in fact been independently invented by many different people. As was already discussed earlier in this chapter, a solution to a standard form inverse problem $A\boldsymbol{f} + \varepsilon = \boldsymbol{m}$ can't be found by simply finding the inverse or the pseudoinverse of A without causing the measurement noise to amplify. A noise robust solution can be sought using ordinary least squares-method, by trying to minimize the expression

$$\| A\boldsymbol{f} - \boldsymbol{m} \|^2 . \tag{3.12}$$

Throughout this thesis $\| \cdot \|$ denotes the Euclidean (ℓ_2) norm. This method often leads to an underdetermined system of equations and can thus not be uniquely solved. Adding a *regularization term* to this will ideally lead to the solution $\boldsymbol{f}$ that fits data $\boldsymbol{m}$ that has the kind of properties we expect the solution to have. To find a Tikhonov regularized solution we instead of trying to find a solution to the not uniquely solvable problem, try to find $T_\alpha(\boldsymbol{m})$ that minimizes the expression

$$\| AT_\alpha(\boldsymbol{m}) - \boldsymbol{m} \|^2 + \alpha \| T_\alpha(\boldsymbol{m}) \|^2 \tag{3.13}$$

where $\alpha > 0$ is a *regularization parameter* that needs to be chosen and $\alpha \| T_\alpha(\boldsymbol{m}) \|^2$ is the regularization term. The role of the regularization term is to make the solution stable and this can be achieved with a good choice of parameter α. Stabilizing quality of the regularization term means that even though the original problem is ill-posed, the regularized problem has to be well-posed and thus measurement noise in data should not affect the minimizer much. The problem with Tikhonov regularization is that a sure method for finding the best possible value for α is yet to be discovered. Too small value of α will affect the noise robustness properties of the regularized solution and too large value will lead to the solution not being a good approximation of the original problem [1], [17].

3.14. **Theorem.** *Let A be a $k \times n$ matrix. The solution $T_\alpha(\boldsymbol{m})$ to the Tikhonov regularized problem is given by*

$$T_\alpha(\boldsymbol{m}) = V\mathcal{D}_\alpha^+ U^T \boldsymbol{m}, \tag{3.15}$$

where $A = UDV^T$ is the singular value decomposition and

$$\mathcal{D}_\alpha^+ = diag(\frac{d_1}{d_1^2 + \alpha}, \ldots, \frac{d_{min(k,n)}}{d_{min(k,n)}^2 + \alpha}) \in \mathbb{R}^{n \times k} \tag{3.16}$$

Proof of this will be omitted but can be found for example in [1].

Using the column vectors u_i and v_i of matrices U and V, $T_\alpha(m)$ can now be written in the form

(3.17)

$$T_\alpha(\boldsymbol{m}) = V\mathcal{D}_\alpha^+ U^T(\boldsymbol{m}) = \sum_{i=1}^{r}(\frac{d_i^2}{d_i^2+\alpha})\frac{\boldsymbol{u_i}^T\boldsymbol{m}}{d_i}\boldsymbol{v_i} = \sum_{i=1}^{r}(\frac{d_i}{d_i^2+\alpha})(\boldsymbol{u_i}^T\boldsymbol{m})\boldsymbol{v_i}.$$

If the singular value decomposition of A is known, Tikhonov-regularization is in many cases a computationally fairly attractive method. When it comes to very large-scale problems, calculating the SVD even by computer can, however, be very time-consuming. In those situations it is more effective to use a method of finding a Tikhonov regularized solution that does not require calculating the SVD of the matrix A. An alternative way of presenting a Tikhonov regularization problem is as a system of linear equations:

$$(A^TA + \alpha I)\boldsymbol{f} = A^T\boldsymbol{m} \tag{3.18}$$

Now the solution isn't based on SVD and it is given by

$$T_\alpha(\boldsymbol{m}) = (A^TA + \alpha I)^{-1}A^T(\boldsymbol{m}). \tag{3.19}$$

3.3.1. *Generalized Tikhonov regularization.* One of the things that makes Tikhonov-regularization such a widely used method is that a priori information about the solution can be introduced to the method, thus making finding a unique solution easier. We may for example know that f is smooth or, as we know in the practical example in this thesis, that all the values of f come from a certain small set of values. A situation like this in relation to tomography might occur, at least in idealized sense, for example if a body part being x-rayed is known to consist of certain tissue types, each of which is known to have certain known attenuation value. The expression that is being minimized in Tikhonov regularization can be modified based on the a priori knowledge. For example in case of a smooth function $\boldsymbol{f}$, we seek to find $T_\alpha(\boldsymbol{m})$ that minimizes the expression

$$\| AT_\alpha(\boldsymbol{m}) - \boldsymbol{m} \|^2 + \alpha \| LT_\alpha(\boldsymbol{m}) \|^2 \tag{3.20}$$

where L is a discretization of a differential operator often called a *regularization matrix.* For $L = I$ this would clearly be exactly the same as 3.13, which in literature is sometimes referred to as zeroth-order Tikhonov regularization. If we have a prior estimate of $\boldsymbol{f}$, denoted by f^*, the expression being minimized takes the form

$$\| AT_\alpha(\boldsymbol{m}) - \boldsymbol{m} \|^2 + \alpha \| L(T_\alpha(\boldsymbol{m}) - \boldsymbol{f}^*) \|^2 . \tag{3.21}$$

With a regularization matrix, the formula 3.19 for the solution of the regularized problem looks like

$$T_\alpha(\boldsymbol{m}) = (A^TA + \alpha L^TL)^{-1}A^T(\boldsymbol{m}). \tag{3.22}$$

A version to which a priori information can be introduced is called *generalized Tikhonov regularization.* The role of the regularization matrix is to try and reduce the error in the reconstruction without dampening the important features of the available estimate of $\boldsymbol{f}$. In addition to the identity matrix I, some common choices for L include finite difference matrices such as

$$L = \begin{bmatrix} 1 & -1 & 0 & & \dots & 0 \\ 0 & 1 & -1 & 0 & \dots & 0 \\ \vdots & & \ddots & \ddots & & \\ \vdots & & & 1 & -1 & \vdots \\ 0 & \dots & & \dots & 1 & -1 \end{bmatrix} \in \mathbb{R}^{(n-1)\times n} \tag{3.23}$$

When L is not an identity matrix or an identity matrix multiplied by scalar, it provides smoothing for the solution. For some methods of choosing a good regularization matrix and additional information about generalized Tikhonov regularization, see [20], also [1], [18].

3.3.2. *Choice of regularization parameter.* As was mentioned before, a sure method for finding the best value for the parameter α is an open problem. There are, however, a few methods that can be utilized in certain situations. One of the best-known methods is the so-called *L-curve method.* L-curve as a method of investigating how different parameter values affect a regularized system goes back to Miller and Lawson & Hanson. Later it has been studied extensively by Hansen and O'Leary and several others.

To use the L-curve method, one first needs to choose a fairly large number of possible values for regularization parameter α such that $0 < \alpha_1 < \alpha_2 < \dots < \alpha_M < \infty$. The values of expressions

$$\| AT_\alpha(\boldsymbol{m}) - \boldsymbol{m} \| \text{ and } \| LT_\alpha(\boldsymbol{m}) \| \tag{3.24}$$

are then calculated for each of the values of α and the results are plotted on a *log-log scale.* The resulting curve is hopefully relatively smooth and by its shape resembles capital L (hence the name L-curve method). The best possible choice for the value of α is supposedly found as close as possible to the part that represents the corner of L. Intuitively it is relatively easy to see why L-curve method would produce a good parameter value α, as it graphically demonstrates the trade-off between minimizing the residual $AT_\alpha(\boldsymbol{m}) - \boldsymbol{m}$ and minimizing the term $LT_\alpha(\boldsymbol{m})$ in L^2 norm. Ideally the "corner" between the vertical and horizontal parts of the L-shape, which is being emphasized by the log-log scale, offers a natural point of compromise between these two qualities. Problems with this method of course arise when, despite the log-log scale, there is no distinctive L-shape. It is noteworthy that all the possible regularized solutions must lie above or on the Tikhonov L-curve [1], [17], [21].

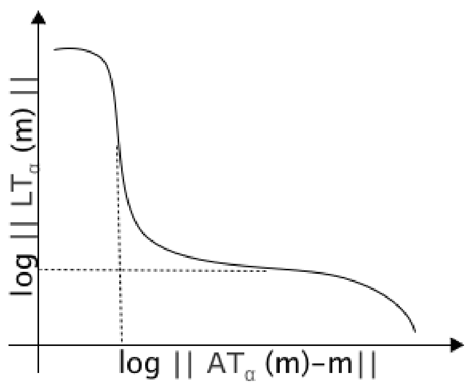


FIGURE 6. L-curve. An ideal situation, where the curve resembles capital L enough that it can easily be used to find as good a value for α as possible.

Some of the other well-known methods include *general cross-validation (GCV)* and *Morozov's discrepancy principle.* Compared to L-curve method, GCV produces an estimate for a good parameter value more rarely than L-curve does, but estimates from GCV are a little more accurate as L-curve tends to over-smooth the solutions slightly. The problem with Morozov's discrepancy principle is that it doesn't work for generalized Tikhonov regularization; it can only be used to find a good parameter value in form 3.19 situations. Finding a method that would always produce an optimal value for α is an unsolved problem. One practical method of finding at least a reasonable approximation is to first test the regularization algorithm with simulated data that resembles the actual situation. Knowing what the reconstruction should look like enables calculating the errors that different choices of α produce. It is also possible to look at the parameter values that have been previously used by others in similar situations and use an average of those. For more information about GVC, discrepancy principle and other methods of finding a good estimate optimal parameter value, see for example [1], [17], [21].

3.3.3. *Large problems and conjugate gradient method.* Calculations with large matrices even with good mathematics-softwares are very time-consuming and often require powerful computers. For an even remotely realistic-sized tomography problem, even storing the resulting matrix A in computer memory would be problematic. In very large-scale problems using methods that utilize SVD is extremely uneconomical. Luckily, this problem can be surpassed when Tikhonov regularization is used in the form 3.22. We can use it in a matrix-free way in a sense that only a matrix-vector product is required by using some iterative solution method such as *conjugate gradient method* or *generalized minimal residual method.* System of linear equations

$$(A^TA + \alpha L^TL)\boldsymbol{f} = A^T\boldsymbol{m}$$

can be solved without actually having to construct the matrices A and L. In order to do that, it is helpful to have available computational routines that produce aforementioned matrix-vector products (for for example `Matlab`) that we shall call `Amult` and `Lmult` that for arbitrary vector $\boldsymbol{z} \in \mathbb{R}^n$ give

$$\texttt{Amult}(\boldsymbol{z}) = A\boldsymbol{z} \in \mathbb{R}^k \text{ and } \texttt{Lmult}(\boldsymbol{z}) = L\boldsymbol{z} \in \mathbb{R}^{k'} \tag{3.25}$$

and correspondingly for the transposes of the matrices A and L routines `ATmult` and `LTmult` that with vectors $\boldsymbol{v} \in \mathbb{R}^k$ and $\boldsymbol{w} \in \mathbb{R}^{k'}$ as arguments return

$$\texttt{ATmult}(\boldsymbol{v}) = A^T\boldsymbol{v} \in \mathbb{R}^n \text{ and } \texttt{LTmult}(\boldsymbol{w}) = L^T\boldsymbol{w} \in \mathbb{R}^n. \tag{3.26}$$

There are various methods that can be used for solving systems of linear equations. When implemented iteratively, conjugate gradient method is particularly well suited for solving very large, sparse problems. By using it it is possible to numerically find solutions even when the size of the problem makes it impossible by direct methods. The method is based on a correction made to the direction of the gradient after each iterative step. It can be implemented when the matrix of the system is symmetric, that is $M = M^T$, and positive-definite. Matrix $M \in \mathbb{R}^{n\times n}$ is positive-definite if $\boldsymbol{c}^TM\boldsymbol{c}$ is positive for every non-zero column vector $\boldsymbol{c} \in \mathbb{R}^n$ [1], [19].

Conjugate gradient method was originally developed for the purpose of minimizing quadratic function

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^TM\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x}, \tag{3.27}$$

which is equivalent to solving

$$M\boldsymbol{x} = \boldsymbol{b}. \tag{3.28}$$

Gradient of the equation 3.27 is a vector-valued function

$$f'(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1} f(\boldsymbol{x}) \\ \frac{\partial}{\partial x_2} f(\boldsymbol{x}) \\ \vdots \\ \frac{\partial}{\partial x_n} f(\boldsymbol{x}) \end{bmatrix} \tag{3.29}$$

that for any given point x points in the direction of the greatest increase of the function $f(\boldsymbol{x})$. Equation 3.27 can thus be minimized by setting $f'(\boldsymbol{x})$ to zero, the minimizer then also being solution to 3.28.

3.30. **Definition.** Two non zero-vectors $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ are said to be conjugate with respect to symmetric matrix M, if $\boldsymbol{d}_1^T M \boldsymbol{d}_2 = 0$ or in other words if they are orthogonal for inner product $\langle \boldsymbol{d}_1, \boldsymbol{d}_2 \rangle_M = \boldsymbol{d}_1^T M \boldsymbol{d}_2$. A finite set $D = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \dots, \boldsymbol{d}_n\}$ is said to be a set of mutually conjugate vectors if $\boldsymbol{d}_i^T M \boldsymbol{d}_j = 0$ for all $i \neq j$.

3.31. **Lemma.** *If matrix $M \in \mathbb{R}^{n \times n}$ is positive-definite and $D = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \dots, \boldsymbol{d}_n\}$ is a set of non-zero, mutually conjugate vectors, then these vectors are linearly independent.*

Proof: Let $\alpha_0, \alpha_1, \dots, \alpha_k$ be scalars such that

$$\alpha_0 \boldsymbol{d}_0 + \alpha_1 \boldsymbol{d}_1 + \cdots + \alpha_k \boldsymbol{d}_k = 0. \tag{3.32}$$

Because D is a set of mutually conjugate vectors and thus $\boldsymbol{d}_i^T M \boldsymbol{d}_j = 0$ for all $i \neq j$, multiplying the equality 3.32 by $\boldsymbol{d}_j^T M$, $0 \leq j \leq k$ yields

$$\alpha_j \boldsymbol{d}_j^T M \boldsymbol{d}_j = 0.$$

But $M = M^T$ 0 and $\boldsymbol{d}_j \neq 0$. Therefore $\alpha_j = 0$ for $j = 0, 1, \dots, k$ and the vectors in D are linearly independent.□

Let us denote the solution of 3.27 and 3.28 by $\boldsymbol{x}^*$. If $D = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \dots, \boldsymbol{d}_n\}$ is a set of mutually conjugate vectors, $\boldsymbol{x}^*$ can be expressed in the following way:

$$\boldsymbol{x}^* = \sum_{i=0}^{n-1} \alpha_i \boldsymbol{d}_i \tag{3.33}$$

The term α_i in the previous expression can be calculated as follows:

$$\boldsymbol{x}^* = \sum_{i=0}^{n-1} \alpha_i \boldsymbol{d}_i \Leftrightarrow M\boldsymbol{x}^* = \textstyle\sum_{i=0}^{n-1} \alpha_i M \boldsymbol{d}_i \Leftrightarrow \boldsymbol{d}_j^T M \boldsymbol{x}^* = \sum_{i=0}^{n-1} \alpha_i \boldsymbol{d}_j^T M \boldsymbol{d}_i$$

$$\Leftrightarrow \boldsymbol{d}_j^T \boldsymbol{b} = \textstyle\sum_{i=0}^{n-1} \alpha_i \boldsymbol{d}_j^T M \boldsymbol{d}_i \tag{3.34}$$

Now recall that $\langle \boldsymbol{d}_i, \boldsymbol{d}_j \rangle_M = \boldsymbol{d}_i^T M \boldsymbol{d}_j$ and that because vectors $\boldsymbol{d}_i$ in the set D are mutually conjugate with respect to M, all the terms in the sum in the following expression except for the jth one are zero.

$$\boldsymbol{d}_j \boldsymbol{b} = \langle \boldsymbol{d}_j, \boldsymbol{b} \rangle = \sum_{i=0}^{n-1} \alpha_i \langle \boldsymbol{d}_i, \boldsymbol{d}_j \rangle_M = \alpha_j \langle \boldsymbol{d}_j, \boldsymbol{d}_j \rangle_M$$

$$\Rightarrow \alpha_j = \frac{\langle \boldsymbol{d}_j, \boldsymbol{b} \rangle}{\langle \boldsymbol{d}_j, \boldsymbol{d}_j \rangle_M} = \frac{\boldsymbol{d}_j^T \boldsymbol{b}}{\boldsymbol{d}_j^T M \boldsymbol{d}_j} \tag{3.35}$$

Using a set of n mutually conjugate direction vectors we can now find an expression for $\boldsymbol{x}^*$ using only them and elements that are known to us:

$$\boldsymbol{x}^* = \sum_{j=0}^{n-1} \frac{\langle \boldsymbol{d}_j, \boldsymbol{b} \rangle}{\langle \boldsymbol{d}_j, \boldsymbol{d}_j \rangle_M} \boldsymbol{d}_j = \sum_{j=0}^{n-1} \frac{\boldsymbol{d}_j^T \boldsymbol{b}}{\boldsymbol{d}_j^T M \boldsymbol{d}_j} \boldsymbol{d}_j \tag{3.36}$$

Let us denote the kth conjugate direction vector by $\boldsymbol{d}_k$ and kth gradient vector by $\boldsymbol{g}_k$:

$$\boldsymbol{d}_k = -\boldsymbol{g}_k = \boldsymbol{b} - M\boldsymbol{x}_k \tag{3.37}$$

In conjugate direction methods, the solution is sought by with each step choosing a point

$$\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k. \tag{3.38}$$

3.39. **Algorithm.** *The conjugate direction algorithm* Let $\{\boldsymbol{d}_i\}_{i=0}^{n-1}$ be a set of mutually conjugate vectors. For any $\boldsymbol{x}_0 \in \mathbb{R}^n$ the sequence $\{\boldsymbol{x}_k\}$ generated according to

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k, \quad k \geq 0$$

with

$$\alpha_k = -\frac{\boldsymbol{g}_k^T \boldsymbol{d}_k}{\boldsymbol{d}_k^T M \boldsymbol{d}_k} = \frac{\langle \boldsymbol{g}_k, \boldsymbol{d}_k \rangle}{\langle \boldsymbol{d}_k, \boldsymbol{d}_k \rangle_M} \tag{3.40}$$

converges to a unique solution $\boldsymbol{x}^*$ of $M\boldsymbol{x} = \boldsymbol{b}$ after n steps.

The problem with using conjugate direction algorithm is that all the values in the set of mutually conjugate vectors D have to be found before its implementation. This can be difficult in practice as the matrix M is often so large that simply storing the data is problematic. For more information about using conjugate gradient as a direct method, see for example [22] (original source), [1], [23], [24].

When conjugate gradient method is implemented iteratively, it is not necessary to determine the direction vectors beforehand. They are chosen one at the time by using the previous choice to determine the next. This is much more economical than obtaining all the vectors in D beforehand, as the previous vectors can now be discarded after each iteration step.

To use the iterative conjugate gradient method, one first needs to choose an initial guess for the value of solution. Let us denote this guess by $\boldsymbol{x}_0$. If there is no a priori information about the solution that would give a reason for a different guess, let $\boldsymbol{x}_0 = 0$. Every new direction vector is determined by taking the current negative gradient vector (as we are trying to minimize 3.27, negative gradient takes us to the direction of the greatest decrease) and adding to it a linear combination of the previous direction vectors. All the direction vectors are mutually conjugate with respect to M. After each iteration step the current position can be evaluated by finding the value of $f(\boldsymbol{x})$ for the current value of $\boldsymbol{x}$ ($\boldsymbol{x}_k$ after the kth step). The value of $f(\boldsymbol{x})$ should be getting smaller with each step [1], [23], [24].

3.41. **Algorithm.** *The conjugate gradient algorithm:*

For any $\boldsymbol{x}_0 \in \mathbb{R}^n$, let $\boldsymbol{d}_0 = -\boldsymbol{g}_0 = \boldsymbol{b} - M\boldsymbol{x}_0$. For $k = 0, 1, 2, \ldots$

$$\begin{aligned} \alpha_k &= -\tfrac{\boldsymbol{g}_k^T \boldsymbol{d}_k}{\boldsymbol{d}_k^T M \boldsymbol{d}_k} \\ \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k \\ \boldsymbol{g}_{k+1} &= M\boldsymbol{x}_{k+1} - \boldsymbol{b} \\ \beta_k^T &= \tfrac{\boldsymbol{g}_{k+1} M \boldsymbol{d}_k}{\boldsymbol{d}_k^T M \boldsymbol{d}_k} \\ \boldsymbol{d}_{k+1} &= -\boldsymbol{g}_{k+1} + \beta_k \boldsymbol{d}_k \end{aligned}$$

Stop if $\boldsymbol{g}_{k+1} = 0$ or sufficiently small. The resulting $\boldsymbol{x}_{k+1}$ is an approximation of the solution of 3.27 and 3.28.

In the following example a small image is reconstructed using Tikhonov regularization implemented with conjugate gradient method.

3.42. **Example.** The simulated data used in this example resembles a common problem of discrete tomography. It is an image of a pawn on a checked background consisting of 3 different grey values (black, grey and white). It was created with Matlab using a code that can be found in appendix of this thesis. All the codes used can be found in the appendix.

The process for this experiment was conducted as follows: The beginning part of the experiment is very much like example 2.16. The first step was to create a $[-1, 1] \times [-1, 1]$ square. For a fixed value N, which represents the number of projections, lines representing x-rays were then placed through the square so that $\cos(\theta)x + \sin(\theta)y = s$ where values of θ run from $-\frac{\pi}{2}$ to $\frac{\pi}{2} - \frac{\pi}{N}$ with a step length of $\frac{\pi}{N}$ and the values of s run from -1.1 to 1.1 with a step length of $\frac{2.2}{N-1}$. There are N different values for both θ and s, which leads to a total number of measurements of N^2. We number them so that the lines in the first direction, which is the direction $-\frac{\pi}{2}$, are marked with $m_1, \ldots, m_n$ so that they correspond to the values of s $-1.1, \ldots, 1.1$ in that order. The lines in the second direction $-\frac{\pi}{2} + \frac{\pi}{N}$ are marked with $m_{N+1}, \ldots, m_{2N}$ and correspond again to the values of s $-1.1, \ldots, 1.1$ in that order and so on. The square was then divided into N^2 similar squares and they were numbered so that the upper left corner is subsquare 1, the one under it is subsquare 2 and so on, this way subsquare $N + 1$ is on the right side of the first subsquare and the last subsquare N^2 in the lower right corner. An approximate picture of the situation with the value of $N = 4$ below.

To form the coefficient matrix A, we note that the number of columns in the matrix is the number of pixels, N^2, and the number of rows is the number of measurements (in other words the number of x-ray beams,

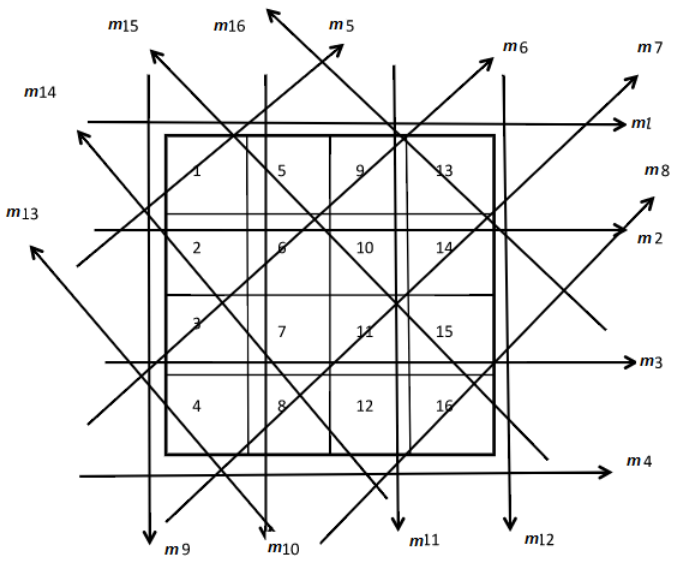


FIGURE 7. An approximate picture showing the directions and placements of the lines $m_1, \dots, m_{N^2}$ and how the pixels the square is divided into are numbered with the choice $N = 2$.

represented by arrows in the image above times the number of projection angles), also N^2. Each component $A_{(i,j)}$ is the distance that the line m_i travels within the subsquare j. The code for creating the matrix is Appendix A, note that the choice for value N can be changed in the first row.

The next order of business was to find simulated data suitable in a sense that it only consists of a small number of grey values. Any random shape would have worked but for reasons that are for a large part frivolous (it looks much nicer) but also practical (it makes detecting mistakes in the reconstruction with naked eye easier) a particular black and white image was constructed. An image of a pawn was constructed into the $[-1, 1] \times [-1, 1]$ square in the following way:
The areas in the set

$$\{(x, y) : [(x > 0 \land y > 0) \lor (x \leq 0 \land y \leq 0)] \land x^2 + y^2 > (\frac{9}{10})^2\}$$

are left white. The areas in the combination of the sets

$$\{(x, y) : [(x \leq 0 \land y > 0) \lor (x > 0 \land y \leq 0)] \land x^2 + y^2 > (\tfrac{9}{10})^2\},$$
$$\{(x, y) : x^2 + (y - (\tfrac{1}{2})^2 \leq (\tfrac{1}{4})^2\},$$
$$\{(x, y) : x^2 + 4(y + \tfrac{1}{2})^2 \leq (\tfrac{1}{2})^2, y \geq -\tfrac{1}{2}\} \text{ and}$$
$$\{(x, y) : \tfrac{y}{4} - \tfrac{1}{8} \leq x \leq -\tfrac{y}{4} + \tfrac{1}{8}, -\tfrac{1}{2} \leq y \leq \tfrac{1}{2}\}$$

are black. What remains of the square is grey, more specifically the colour halfway between black and white. The code that can be found in Appendix B forms $N \times N$ matrix $\boldsymbol{F}$ for an input value of N (which has to be of the form $4k$). The component $F(i,j)$ of the matrix gets value 1 (white), 0 (black) or $\frac{1}{2}$ (grey) based on which of the sets $(x,y) = (\frac{2j-N}{N}, \frac{2i-N}{N})$ above it belongs to. The code also forms a vector $\boldsymbol{f}$ of N^2 components that reads the components of $\boldsymbol{F}$ from down to up and from left to right: $f((j-1)*N+i) = F(N+1-i,j), i = 1,\dots,N, j = 1,\dots,N$. For the chosen value of N a number $N1 = 2*N$ is then calculated. This part of the process helps avoid the dreaded inverse crime. For this number a $N1 \times N1$ matrix named $\boldsymbol{ro}$ and a vector $\boldsymbol{roo}$ of $N1^2$ components are calculated by the same method as $\boldsymbol{F}$ and $\boldsymbol{f}$ before.

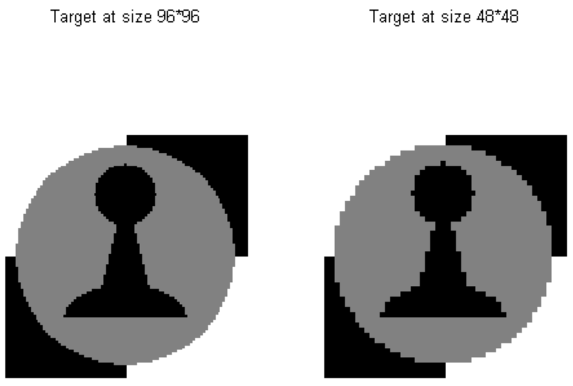


FIGURE 8. The image constructed with the code Pawn.m on the left with resolution $N = 96$ and on the right with resolution $N = 48$.

Next step in the process was to simulate the measurement data by using the code matrixA.m (Appendix A) to construct a $96^2 \times 96^2$ matrix $A96$. The code Pawn.m (Appendix B) is then used to construct the vector $\boldsymbol{roo}$, which is the vector constructed of the image of a pawn with resolution $N = 96$. Now the noise free measurement data is gained by using multiplication: $m96 = A96 * \boldsymbol{roo}$. This would of course not be possible if the measurement data would be real instead of simulated, so to make the situation more realistic we then add some noise to the measurement data. The noise is simulated to each component individually by using Matlab's rand-generator so that $L2$-norm relative error is about 2 percent. We now have a new vector of 96^2 components that we will denote by $m96n$. To avoid inverse crime we will be looking for the reconstruction using a $48^2 \times 48^2$ matrix $A48$ that is constructed using the code matrixA.m with the choice of $N = 48$. To do this we need to interpolate the measurement data from $m96n$ to a same type of vector of 48^2 components $m48n$. Each component depends on parameters θ and s and the choice of $N = 96$ contains all the measurement angles θ ($\theta = -\frac{\pi}{2} : \frac{\pi}{N} : \frac{\pi}{2} - \frac{\pi}{N}$) of the choice $N = 48$. The distance components s are trickier as they don't fall into same places as they do with resolution $N = 96$. With the choice N the jth component of the vector s is

$$s_j^N = -1.1 + \frac{2.2(j-1)}{N} \tag{3.43}$$

and it is easy to prove that

$$s_j^{48} = s_{2j-1}^{96} + \frac{j-1}{47}(s_{2j}^{96} - s_{2j-1}^{96}) \tag{3.44}$$

which is why in the code measurements.m (Appendix C) the vector $m48n$ is formed using the vector $m96n$ and the previous formula. To draw the images the vectors $m96$, $m96n$ and $m48n$ are then turned into matrices $M96$, $M96N$ and $M48N$. See the figure 9.

Finally, the system $(A48^T \times A48 + \lambda I)x = A48^T \times m48n$ (the system 3.18 in Tikhonov-regularization) is solved using a choice of $\lambda = 0.023$ (a value which, in all honesty, is an approximation found by the method of trial and error). The initial value for iteration in the code conj.m (Appendix D) is chosen to be a vector of 48^2 components in which all the components have the value 0.5. The code gives an output of reconstructions after 2 and 49 iterations. It then changes the latter image into a version in which the value of each pixel is rounded to whichever of the values $0, 0.5$ and 1 is the closest, a process which is similar to the segmentation done in the beginning of using the DART-algorithm. This mimics a typical situation in discrete tomography, where even though the image is not known before the reconstruction like it is in this simulated example, the possible grey values are often known. Using this method we get the reconstruction (Figure 10) in which the relative error after 49 iterations is 16% where it in the data with noise added to it was 2% . This is a fairly good result given the partial nature of measurement data.

This method, as was demonstrated before, gives relatively good results, does not take too long to run and does not require the computer to be particularly powerful. Smoothing of the image typically caused by the regularization term can be seen particularly in the reconstructed image after two rounds of iteration. This is due to the tendency of the l_2-norm to favour smooth functions. Using for example *total variation regularization* in which the norm is l_1, it is likely that we would have

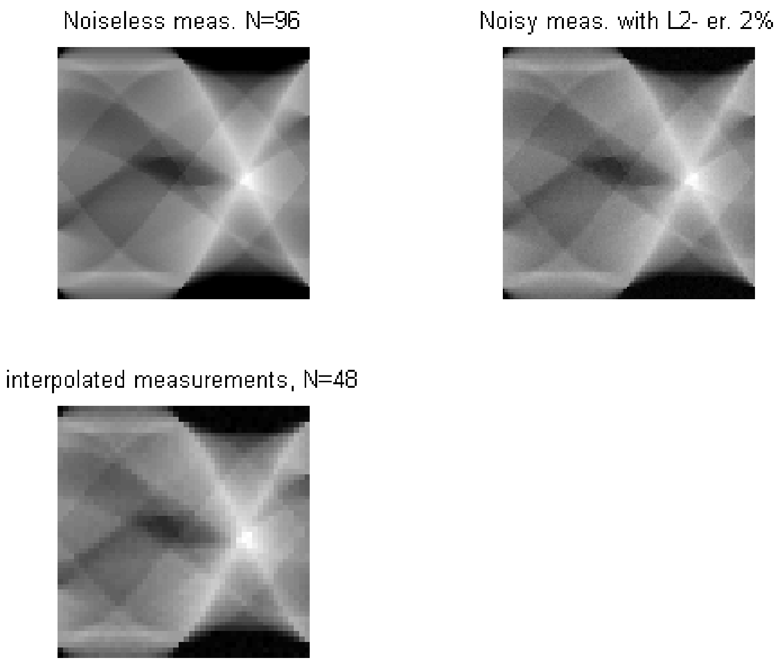


FIGURE 9. Image on the upper left-hand-side is the sinogram of the measurement data with $N = 96$. On the right is the sinogram of the same data but with added noise. On the bottom is the sinogram for the interpolated measurement data.

ended up with an image resembling the lowest one (in which all the pixel values are forced to the closest grey value found in the original image) straight away. This method seems to be a decent choice when a reconstruction is wanted as quickly as possible and some minor flaws in the reconstruction will not have very dramatic effects. As can be seen from the reconstructed image 10, some pixels have been misplaced, which could very well be an issue if this type of reconstruction would be used for example as a guideline for performing a surgery. It is highly likely, that a method of reconstruction aimed especially for discrete tomography would yield better results in a situation like this.

Tikhonov regularization is a quick and effective method and it is easy to see why it is probably the most popular of all the reconstruction methods. Problems more severe than a few misplaced pixels arise, however, when the data from which the reconstruction is attempted is incomplete. In the following example almost the same small image (just altered so that the same code could more easily be used) is reconstructed based on only 20 projections. For an image this simple

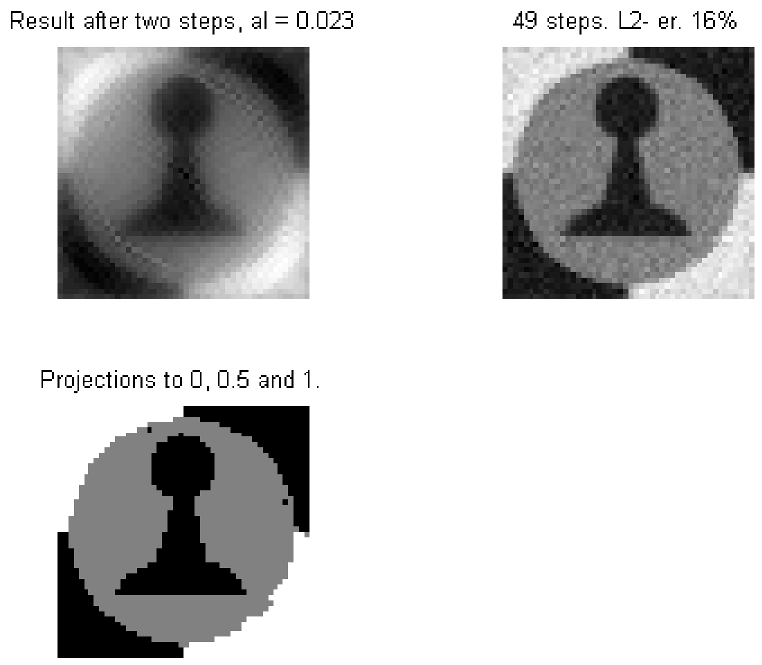

Result after two steps, al = 0.023
49 steps. L2- er. 16%
Projections to 0, 0.5 and 1.

FIGURE 10. The image in the upper left corner is the reconstruction after two iterations, on its right side is the reconstruction after 49 iterations and on the bottom is version of the reconstruction (after 49 iterations) in which all the pixels are rounded to whichever of the values 0, 0.5 and 1 is closest to it.

and a shape which is this close to convexity the reconstruction is likely to still be recognizable, but for a more complex shape even 20 equally distributed projections would not be likely to be enough to give at all usable results.

3.45. **Example.** The same codes that can be found in the appendix were used here with the following alterations:

The reduced number of projections was achieved by leaving out projections in every other direction and then making the image being reconstructed slightly smaller (as in the original code the number of projections is tied to the size of the image) in order to get the number of projections to be exactly the same as in the third reconstruction example later on. In that example the number of projections is 20, a number which was chosen based on it being one of the numbers used in the original papers on DART-algorithm [32] and [33].

Now for the fixed value N lines that represent x-rays $cos(\theta)x + sin(\theta)y = s$ are placed so that the values of θ run again from $-\frac{\pi}{2}$ to $\frac{\pi}{2} - \frac{\pi}{N}$ but wth a step length that is now $\frac{2\pi}{N}$, twice what it was before. As there are as still N values for s, just as there were in the previous example, the measurement matrix is now of the size $\frac{N^2}{2} \times N^2$. We denote this by AHN, H representing the fact that this measurement matrix is half the size of the measurement matrix in the previous example. This matrix is formed of the rows from 1 to N, from $2N+1$ to $3N$, from $4N+1$ to $5N$ and so on of the original matrix. At this point, in order to get the number of projections to be 20, we choose the value of $N = 40$, which of course in this code leads both the original image and the image used for the reconstruction to be a little smaller than they were before, see figure 11.

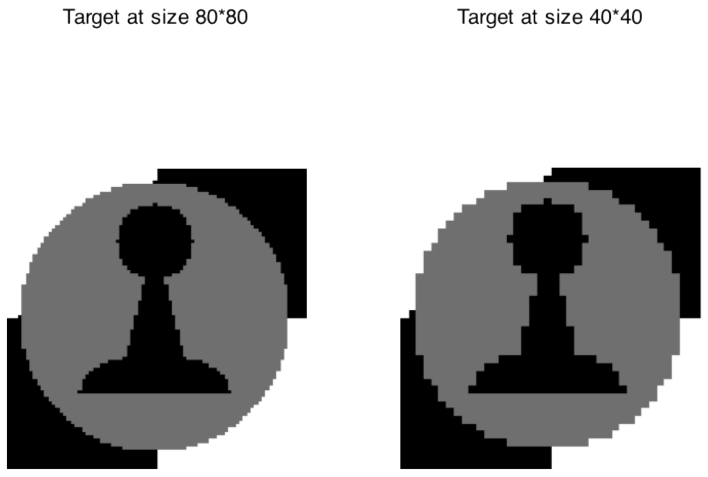


FIGURE 11. On the left hand side target image with the choice of $N = 80$, on the right the one used for the reconstruction, in which $N = 40$.

From here on the example closely follows the previous one. We next simulate the measurement data by multiplying the 80×80 image with a vector gained from the matrix AHN (see previous example for more details). Then we add 2% of noise and interpolate the measurement data to match the 40×40 image that is in order to avoid inverse crime used for the reconstruction. In the figure 12 the reconstructions are being shown after 2 and 49 rounds of iteration. Just as in previous example, the bottom image is one where the pixel values are rounded to the closest of the values that are known to appear in the image. The relative error after 49 rounds of iteration (top right corner) is now 20% compared to previous example's 16%, the reconstruction is visibly worse but still (depending on the application it is used for) fairly good.

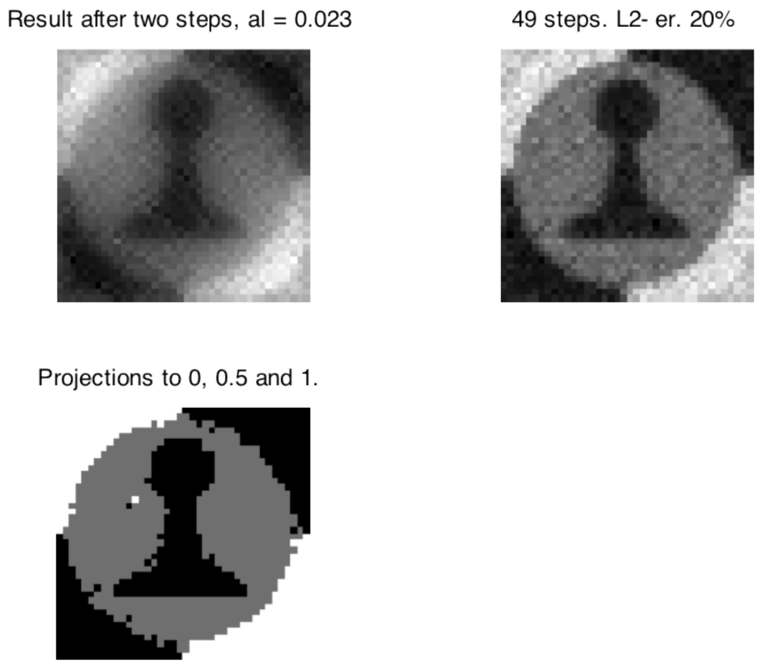

Result after two steps, al = 0.023
49 steps. L2- er. 20%
Projections to 0, 0.5 and 1.

FIGURE 12. The image in the upper left corner is the reconstruction after two iterations, on its right side is the reconstruction after 49 iterations and on the bottom is version of the reconstruction (after 49 iterations) in which all the pixels are rounded to whichever of the values 0, 0.5 and 1 is closest to it.

As can be seen, for a situation as simple as this, Tikhonov regularization still performs quite well. However, for a situation where the image being reconstructed is more complex, there are even fewer projections or the projection angle is limited, some far more sever problems start to arise. Clearly some other reconstruction methods need to be studied.

4. Discrete tomography

4.1. **About discrete tomography.**

The problems dealing with discrete tomography have been seen in literature since at least 1957 *(an article published by H.J. Ryser in Canadian Journal of Mathematics,* [25]*)*. However, discrete tomography as a separate field of study worthy of its own theory has only been around since the first mini-symposium carrying the name was organized in 1994. The main reason for there being a demand for such theory is that due to the lack of uniqueness; most conventional methods of tomography such as filtered back projection and algebraic reconstruction techniques (ART) usually fail when only a small amount of projections is available. Since its introduction discrete tomography as a special field has for a great part focused on the reconstruction of binary images, images that are comparable to binary matrices where ones are seen as white and zeroes are seen as black. Binary images or -matrices can also be seen as two-dimensional special case of reconstructing subsets of integer lattices. This kind of a situation occurs for example in reconstruction of nanocrystals at atomic resolution. Even though a binary image is considerably easier to reconstruct than an image of which nothing is known a priori, it is still often necessary to make further assumptions about it for the number of projections in use to suffice for finding a unique solution. In many algorithms of discrete tomography, local weight functions are utilized if the image being reconstructed is known to be relatively smooth, in other words containing large areas of connected ones or zeroes. In iterative methods where the reconstruction is attained pixel by pixel, larger weight can be assigned to pixels that are surrounded by pixels with the same value [2], [34].

Even though binary images are the main area of interest in discrete tomography, some slightly varied situations can still be studied under the name. For the purposes of this master's thesis as well as many situations where practical measurement takes place, one of the most interesting ones is making reconstructions of functions that, rather than having just values $\{0, 1\}$, have some other small set of values $\{p_1, p_2, \ldots, p_l\}$. By using the prior knowledge about the possible values occurring in the reconstructed image, using algorithms of discrete tomography it is often possible to greatly reduce the number of projections needed for a good reconstruction.

As discrete functions seem like special cases of general functions, it is easy to assume that discrete tomography could use the same algorithms as general tomography. This turns out not to be the case, some integral questions about consistency, existence and uniqueness cannot be answered using only the results of general tomography. Knowing that the function f is discrete also offers hope that it can be reconstructed from less data than would be needed to reconstruct general function. Discrete tomography is mostly based on discrete mathematics, but the theory is also strongly connected to that of geometry and combinatorics [2].

4.2. **Reconstruction from row- and column sums.** A very simple reconstruction problem in discrete tomography resembles a Sudoku puzzle but unlike most Sudokus, the solutions are usually not unique. Essentially the problem is very much like the example 2.16 before, but now the only projections in use are the horizontal one and the vertical one. Knowing only the row and column sums one should now be able to reconstruct the matrix. What makes it easier is the prior knowledge about the possible values in $\boldsymbol{f}$, in most typical discrete tomography cases the possible values being 0 and 1. The same idea can be easily visualized using so called *lattice sets* (or *discrete sets*), which are finite subsets of the integer lattice $\mathbb{Z}^2$ (or in a more general case $\mathbb{Z}^d$). In commonly used notation in discrete tomography we denote the finite subset of $\mathbb{Z}^2$ that is attempted to reconstruct by F and state that if a point of $\mathbb{Z}^2$ is an element of F, it has value one and if it is not an element of F, it has value zero. F can now be seen as a function that attaches either value one or value zero to every point in $\mathbb{Z}^2$. Usually we do not consider the whole $\mathbb{Z}^2$ but only a rectangle $A \subset \mathbb{Z}^2$ that includes every point that has value one. F can then be represented as a function $F : A \to \{0, 1\}$, as can be seen for example in [2] and [34].

In the following example the task is to reconstruct a lattice set of $\mathbb{Z}^2$ from x-rays in two *lattice directions*, vertical and horizontal. In these projections we don't consider all the possible lines (there being an infinite number of them) but only the lines that pass through integer points. These lines, often denoted by ℓ, in a lattice direction are called lattice lines if they pass through at least one point in $\mathbb{Z}^2$ (or in a more general case $\mathbb{Z}^d$). An x-ray of the set in these directions gives the number of points in the set, usually called the *line sum* of F along these lines, on each vertical or horizontal line.

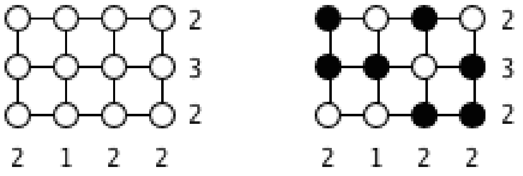


FIGURE 13. On the left an image of a lattice set, where potential locations for points are marked with circles and the line- and column sums are given at the end of each line and column. On the right the circles are filled in to demonstrate one possible reconstruction based on given line sums.

4.1. **Example.** On the left is the reconstruction problem that gives the line sums of each (non-zero) line of each of the two projections. In

$$(4.4)\qquad \begin{cases} x_1 + x_4 = 2 \\ x_6 + x_7 + x_8 = 3 \\ x_9 + x_{11} = 2 \\ x_1 + x_9 = 2 \\ x_6 = 1 \\ x_7 + x_{11} = 2 \\ x_4 + x_8 = 2 \end{cases} .$$

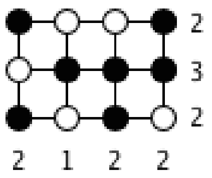


FIGURE 14. Circles filled in to demonstrate the reconstruction described on the left.

other words it tells how many of the integer points on each vertical or horizontal line are elements of F. On the right is one possible version of F based on the line sums in these two lattice directions. This kind of discrete reconstruction problem can be viewed as the following linear equation system:

$$(4.2)\qquad P\boldsymbol{x} = \boldsymbol{m}, \text{where } \boldsymbol{x} \in \{0,1\}^N, P \in \{0,1\}^{M\times N} \text{and } \boldsymbol{m} \in \mathbb{N}_0^M$$

where N is the number of lattice points in the rectangle that includes all the lattice points with value one and M is the number of lattice lines. P is now a matrix that describes whether a point with value one is on the lattice line ℓ. The line sums based on which the reconstruction is made are expressed in the vector $\boldsymbol{m}$. With the two dimensional example we are looking at the reconstruction of F on the right can be expressed in the form

$$(4.3)\qquad \begin{cases} x_1 + x_3 = 2 \\ x_5 + x_6 + x_8 = 3 \\ x_{11} + x_{12} = 2 \\ x_1 + x_5 = 2 \\ x_6 = 1 \\ x_3 + x_{11} = 2 \\ x_8 + x_{12} = 2 \end{cases} .$$

In this way of writing the expression only the variables with the value one are written. The whole reconstruction problem can be expressed similarly by writing all the variables in each line- and column sum, then the problem of determining which of them have value one and which value zero remains. The binary constrain $\boldsymbol{x} \in \{0,1\}^N$ often makes solving these systems much more simple. However, the solutions are often not unique. In the situation above it is easy to see that the same vector $\boldsymbol{m}$ can be achieved with for example the reconstruction and the corresponding system below:

Only in very rare and usually very simple situations are two projections enough for finding a unique solution. Even in a situation this simple additional projections or other additional information is needed.

With very few projections and very little information the problem is not finding a unique solution, but sometimes, often when a larger number of projections are available, it can be that no solutions that satisfy the projections are found.

4.2.1. *Existence and uniqueness of reconstructions from two projections.* The first reconstruction algorithm in discrete tomography was introduced by Ryser in 1957 in [25], when he described a method for reconstructing matrices of zeroes and ones from two projections. He also gave necessary and sufficient conditions on the projections for the existence of this reconstruction. In the next definition let us consider vertical and horizontal projections of a binary matrix (a matrix of zeroes and ones).

4.5. **Definition.** Let $R = (r_1, \ldots, r_m)$ be a sequence of row sums or a *row sum vector* and let $C = (c_1, \ldots, c_n)$ be a sequence of column sums or a *column sum vector*. Clearly

$$\sum_{i=1}^{m} r_i = \sum_{j=1}^{n} c_j \tag{4.6}$$

as both sums are equal to the number of ones in the binary matrix. The class of all binary matrices $A = (a_{ij})$ that share the same row- and column sums is often denoted by $\mathfrak{U}(R, C)$.

$$\sum_{j=1}^{n} a_{ij} = r_i, \quad i = 1, \ldots, m \tag{4.7}$$

$$\sum_{i=1}^{m} a_{ij} = c_j, \quad j = 1, \ldots, n \tag{4.8}$$

Ryser also described the following condition that can be used to find out whether there is a possible reconstruction with given row- and column sums, or in other words to find out if the class of matrices $\mathfrak{U}(R, C)$ is nonempty:

Let us denote a matrix in which on every row $i, i = 1, \ldots, m$ there are r_i ones followed by $n - r_i$ zeroes by $\overline{A}$. This kind of a matrix is called *maximal.* A row sum vector of a maximal matrix determines it uniquely. Now let us denote its column sum vector by $\overline{C} = (\overline{c_1}, \ldots, \overline{c_n})$. Clearly rearranging the terms on each row have no effect on the row sum, so $R = \overline{R}$. Let us then denote the non-increasing permutations of the elements of the row- and column sum vectors R and C by R', that is $r'_1 \geq r'_2 \geq \cdots \geq r'_m$ and C', that is $c'_1 \geq c'_2 \geq \cdots \geq c'_n$.

4.9. **Theorem.** *Let R and C be the row- and column sum vectors of a binary matrix. The class $\mathfrak{U}(R, C)$ is nonempty if and only if*

$$\sum_{j=l}^{n} c'_j \geq \sum_{j=l}^{n} \overline{c}_j \text{ for } 2 \leq l \leq n. \tag{4.10}$$

(For proof, see [25] or [2].)

Using this method for the previous example 4.1 , we can easily see that the solution exists, just like we already saw by actually finding two possible solutions. Rearranging the elements of the column sum vector C in a non-increasing order we get $C' = (c'_1, c'_2, c'_3, c'_4) = (2, 2, 2, 1)$. To find $\overline{C}$ we first need to find the maximal matrix with given row sums, this happens by stacking all the ones on each row on the left. Now it is easy to see that $\overline{C} = (\overline{c_1}, \overline{c_2}, \overline{c_3}, \overline{c_4}) = (3, 3, 1, 0)$. To use the theorem 4.9 we only need the elements from the second element onwards on each of the vectors. If the vectors were very long, calculating the sums and making the comparisons for all $l \geq 2$ would of course be very time consuming. In this situation, however, it is easy to see that

$$\begin{aligned}
\sum_{j=2}^{4} c'_j &= 2 + 2 + 1 = 5 &&\geq \sum_{j=2}^{4} \overline{c_j} = 3 + 1 + 0 = 4, \\
\sum_{j=3}^{4} c'_j &= 2 + 1 = 3 &&\geq \sum_{j=3}^{4} \overline{c_j} = 1 + 0 \text{ and} \\
\sum_{j=4}^{4} c'_j &= 1 &&\geq \sum_{j=4}^{4} \overline{c_j} = 0
\end{aligned}$$

$$\text{so for all } 2 \leq l \leq n, \ \sum_{j=l}^{n} c'_j \geq \sum_{j=l}^{n} \overline{c}_j.$$

There is an algorithm that reconstructs a binary matrix based on its row- and column sum vectors (see for example Herman & Kuba, [2]), but determination of the precise number of matrices in $\mathfrak{U}(R, C)$ remains an open problem.

The question of uniqueness in respect to reconstructing binary matrices from two projections is really not much more complicated than the question of existence, but at least using the original method that Ryser provided we first need to have one reconstruction. A binary matrix is said to be unique if and only if there exists no other binary matrix with the same row- and column sums. For finding out if a reconstruction is unique, we need an element called *switching component*:

4.11. **Definition.** In a binary matrix A a *switching component* is a 2x2 submatrix of either one of the two forms

$$A_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \text{ or } A_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{4.12}$$

Switching operation is one that changes a type A_1 submatrix into type A_2 or vice versa. All the other elements in matrix A remain unchanged.

Switching operation clearly does not change the row- and column sums and thus a binary matrix A is not unique if it has at least one switching component. The reverse of the statement seems less intuitive but is nevertheless true.

4.13. **Theorem.** *A binary matrix is non-unique with respect to its row and column sums if and only if it has a switching component.*

In addition to non-uniqueness Ryser also showed that a binary matrix A in with a switching component can be transformed into any other matrix with the same row- and column sums with a finite number of switching operations (Proof: Ryser, [25]). Interestingly, Kong and Herman in chapter 3 of [2] later proved that no similar result is true for any grid that has grid lines in three or more directions. The questions of uniqueness and reconstruction have to be approached in a different way when dealing with discrete tomographic reconstruction problems with three or more projections.

In the example 4.1 it is easy to see that the first suggested reconstruction has three switching components and is thus not unique, as we already saw by finding two different reconstructions with the given row and column sums. A version of the previous theorem 4.9 can also be used to find out if a matrix is unique. By considering equality instead of the possible inequality we get a situation that with binary matrices is equivalent to the non-increasing permutation of elements in column sum vector C being equal to the column sum vector of the maximal matrix $\overline{A}$; $C' = \overline{C}$:

$$\sum_{j=l}^{n} c'_j = \sum_{j=l}^{n} \overline{c}_j \text{ for } 2 \leq l \leq n. \tag{4.14}$$

The equivalence is a result from all of the terms in vectors C' and $\overline{C}$ having to be the same from the second element on for the sums to be equal for $2 \leq l \leq n$, clearly the first element has to be the same in both vectors then also. A maximal matrix $\overline{A}$ is always unique as the ones are stacked on the left on each row and so a zero can never be placed on the left side of a one and $\overline{A}$ can thus not have a switching component. The same goes for any matrix A that has been obtained from $\overline{A}$ by a permutation of columns. As all the column sums have to be the same for the terms to be equal, this is obviously the case if $C' = \overline{C}$ and thus if 4.14 is true for matrix A, it is unique with respect to its row- and column sums. This condition was most likely first found by Y.R. Wang and published in 1975 in [26].

Knowing that a binary matrix satisfies 4.14 and is thus unique, we can more easily reconstruct it using for example the following algorithm according to [2]:

4.15. **Algorithm.** As an input we have a pair of vectors, (R,C), that are as previously the row- and colums sums of a binary matrix. To use this algorithm, the vectors must satisfy 4.14. Now the matrix can be reconstructed using the following three steps:

```
Step 1. $A = 0$; (zero matrix)
Step 2. Find $i_1, i_2, ..., i_m$ such that $r_{i_1} \geq r_{i_2} \geq \cdots \geq r_{i_m}$;
Step 3.
        For $j = 1$ to $n$,
        for $k = 1$ to $c_j$,
            $a_{i_k j} = 1$;
```

Output: Matrix A.

4.16. **Example.** Let us try this out in practice. We must first have the row- and column sums of a matrix A that satisfies 4.14, that is to say a unique matrix. In practice this means that the column sum vector of the matrix A has to have the exact same elements as the column sum vector of the maximal matrix $\overline{A}$. For example the pair of vectors $R = (3, 1, 3)$ and $C = (2, 0, 3, 2)$ satisfies this condition as now $C' = (3, 2, 2, 0)$ and $\overline{C} = (3, 2, 2, 0)$ and clearly the sums $\sum_{j=l}^{4} c'_j$ and $\sum_{j=l}^{4} \bar{c}_j$ are equal for all $2 \leq l \leq n$. We start the reconstruction from a zero matrix of the right size, in this case $3x4$:
1.

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

For the second step we need to find i_1, i_2 and i_3 such that the elements of the row sum vector are in a non-increasing order. If $i_1 = 1$, $i_2 = 3$ and $i_3 = 2$ we see that
2.

$$r_{i_1} = r_1 = 3,\ r_{i_2} = r_3 = 3 \text{ and } r_{i_3} = r_2 = 1$$

For the third and the last step we need to be replacing zeroes in the matrix A above with ones so that for $j = 1, 2, 3, 4$ and for $k = 1, \ldots, c_j$ the elements $a_{i_k j} = 1$. Let's look at all the values of j one by one:
3.

$$j = 1 : k = 1, \ldots, c_1 = 1, 2. \text{ Now } a_{i_1 1} = a_{11} = 1 \text{ and } a_{i_2 1} = a_{31} = 1.$$

$j = 2 : k = 1, \dots, c_2 = 1$ to 0. No results.

$j = 3 : k = 1 \dots c_3 = 1, 2, 3$. Now $a_{i_1 3} = a_{13} = 1, a_{i_2 3} = a_{33} = 1$ and $a_{i_3 3} = a_{23} = 1$.

$j = 4 : k = 1, \dots, c_4 = 1, 2$. Now $a_{i_1 4} = a_{14} = 1$ and $a_{i_2 4} = a_{34} = 1$.

From this we get an output of a matrix

$$A = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

which as we can see corresponds to the row- and column sums of $R = (3, 1, 3)$ and $C = (2, 0, 3, 2)$.

A binary matrix that is bigger in size being unique in respect to its row- and column sums is of course a fairly unlikely situation to happen, so this algorithm cannot in most situations be used. In the book Discrete Tomography: Foundations, Algorithms and Applications ([2]) Herman and Kuba give a version of this algorithm that can be used in a general situation. They also give a proof that the algorithm in fact gives a matrix belonging to $\mathfrak{U}(R, C)$ as output. That algorithm, however, is too complicated to be used as an example that is done by hand in order to demonstrate the different steps.

4.2.2. *Convexity and other prior information.* Whether or not a binary matrix is convex is an important factor in both reconstructing from two projections or more. Convexity means the same thing it always does, that you can draw a straight line from any point of the object to any other point without crossing a boundary or leaving the object. A lattice set A that is a subset of $\mathbb{Z}^n$ is said to be convex if $A = \mathrm{conv}A \cap \mathbb{Z}^n$. For binary matrices different levels of convexity are described based on whether ones (or in case of binary images white pixels) are all following each other consecutively. The formal definition is as follows:

4.17. **Definition.** A binary matrix is h-convex (horizontally convex) if in the rows the ones follow each other consecutively. A binary matrix is v-convex (vertically convex) if in the columns ones follow each other consecutively. A binary matrix is hv-convex if it is both h-convex and v-convex.

When a binary matrix or a binary image is simply called convex, this generally refers to it being hv-convex but in a way where it only contains one convex body. Hv-convex set can have several, which is demonstrated in the rightmost image below. In the image below convexity is demonstrated using binary images where ones are seen as white and zeroes are seen as black instead of binary matrices as it is easier to see whether an a white shape in black background is convex even with a quick glance.

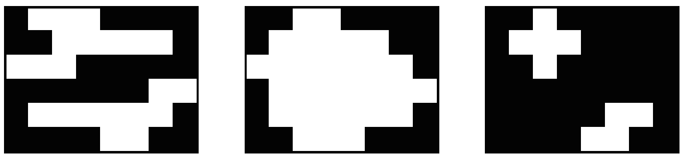


FIGURE 15. The leftmost image is h-convex, the one in the middle is hv-convex and the one on the right hand side is also hv-convex but consists of two separate hv-convex bodies.

With the binary images where the zeroes and ones of binary matrices are expressed as black and white pixels, the convexity can also be studied based on *boundary length.* When the length of the side of one pixel is considered to be one, the boundary of a binary image is defined as the set of line segments that separate the black pixels from the white pixels. A binary matrix that has m nonzero row sums and n nonzero column sums is hv-convex if its boundary length is $2m + 2n$. One problem with this method of looking at convexity is that clearly an image and its negative have the same boundary lengths or in other words, there is no way differentiating between an image, where the convex bodies are formed by ones from an image where the convex bodies are formed by zeroes. Based on line sums it is not always possible to know whether there exists an hv-convex reconstruction, but if the line sums and the boundary length are known, relatively good assumptions of the shapes of the objects formed by white pixels in the image can sometimes be made. If for example the line sums are large (relative to the size of the binary image in question) but the boundary length is small, it is easy to guess that the white pixels (with value one) form some sort of a solid object [28], [34].

4.3. **Discrete tomography with larger number of projections.**

By knowing the length of the boundary or by having some other prior knowledge of the binary matrix (or the binary image or lattice set) in addition to the row- and column sums one can in some cases increase their chances of finding a unique reconstruction. In most cases, however, additional projections are needed. Problematically, the computationally effective reconstruction strategies for binary images from two projections suggested by among others Ryser and Gale (whose strategy was based on network flow problems) become NP-hard in situations where the number of projections is larger. New algorithms are needed for situations where the number of projections is larger than two but still significantly smaller than the number of projections needed for adequate reconstructions in CT. Combinatorics-based reconstruction strategies such as the ones proposed by Ryser are not the only possible approach; some of the different points of view from which DT problems can also be solved are statistical, continuous optimization and *discrete algebraic reconstruction technique* (DART), which is an algorithm based on continuous method with a discretization step. DART-algorithm is discussed more later on. In the following chapters, whenever a particular algorithm is introduced, this thesis follows the notation of the original author.

In 1997, less than 3 years after the first mini-symposium on discrete tomography was organized, R. J. Gardner and P. Gritzmann proved that in $\mathbb{Z}^2$ there are certain prescribed sets of four lattice directions that any convex lattice set can be distinguished from any other such set based on its projections in these directions. They also showed that this cannot be achieved with a number of projections smaller than four. In order to provide uniqueness, the four lattice directions have to be such that their slopes do not yield a cross ratio of 4/3, 3/2, 2, 3 or 4. For more information on this, see [27]. Using only four projections in a real measurement situation would, however, be very risky as even a small change in suitable four directions can lead to a set of projections that does not determine the object being reconstructed uniquely. The result showing that certain set of four lattice directions is enough to determine a convex lattice set in $\mathbb{Z}^2$ can be easily extended to $\mathbb{Z}^n, n \geq 2$. In their paper Gardner and Gritzmann also showed that any seven mutually nonparallel lattice directions suffice to determining any convex subset of $\mathbb{Z}^2$ from any other such set uniquely and that also in this case seven is the smallest possible number with which this can be achieved [27].

The number of projections required for determining a convex polygon in $\mathbb{Z}^2$ can be reduced to three by using an iterative method called *successive determination* first proposed by Edelsbrunner and Skiena in [28] where the previous projections are utilized to help decide the best directions for following x-rays. For a more general case, Gardner and Gritzmann proved that any finite subset of $\mathbb{Z}^n$ can by using successive determination be distinguished from any other such set by $n/(n-k)$ projections on $(n-k)$-dimensional lattice subspaces. The results proposed by Gardner and Gritzmann only apply to determining the convex lattice sets, finding an algorithm with which any convex set in $\mathbb{Z}^2$ could be reconstructed is a problem to which they did not find a solution. Six years later S. Brunetti and A. Daurat continued the work of Gardner and Gritzmann by presenting an algorithm with which convex subsets of $\mathbb{Z}^2$ can be reconstructed based on their x-rays in a suitably chosen set of four lattice directions or any set of seven mutually nonparallel lattice directions [29].

Many algorithms of discrete tomography require the object that is being reconstructed to have certain properties such as being convex in the direction of the x-rays in same manner as h-, v- and hv-convexity described in 4.17. It can be more generally defined as follows:

4.18. **Definition.** A lattice set F is line-convex with respect to direction p if intersections of all lines of p with the set F are the sets of points with integer coordinates of straight line segments.

For $\mathbb{Z}^2$ this means that on each line of the direction p, all the integer pairs that form the coordinates that are on the line segment from the point where the line enters F to the point where it exits it are in F. In addition to h-, v-, and hv-convexities that are defined as before, we say that F is *d-convex* (respectively hd-, vd- or hvd-convex) if it is convex with respect to diagonal direction, meaning direction directed by vector $(1, 1)$.

4.19. **Definition.** A lattice set F is convex if is the intersection between $\mathbb{Z}^2$ and its convex hull, denoted by $Conv(F)$, which is the smallest of all the convex sets containing F.

Another property that many algorithms of discrete tomography expect of the binary matrix, -image or the lattice set being reconstructed is *connectivity*:

4.20. **Definition.** A 4-path is a finite sequence of points $(M_0, M_1, \ldots, M_n) \in \mathbb{Z}^2$ such that $M_{i+1} - M_i$ is in the set $\{(\pm 1, 0), (0, \pm 1)\}$. A lattice set F is 4-connected if for any $A, B \in F$ there is a 4-path from point A to point B. A 4-connected lattice is also called a *polymino*.

Even though polyminoes and thus 4-connectivity is the most important and the most restrictive type of connectivity at least for the purposes of discrete tomography, sometimes 8- and 6-connectivities are also used. They are defined like 4-connectivity, except that the following point can in 8-connectivity's case be any of the surrounding dots and in 6-connectivity's case the one on left, right, up, down, lower left or upper right. More formally:

4.21. **Definition.** An 8-path (respectively a 6-path) is a finite sequence of points $(M_0, M_1, \ldots, M_n) \in \mathbb{Z}^2$ such that $M_{i+1} - M_i$ is in the set $\{(\pm 1, 0), (0, \pm 1), (\pm 1, \pm 1)\}$, (respectively $\{(\pm 1, 0), (0, \pm 1), (1, 1), (-1, -1)\}$).

8-connected and 6-connected lattice sets are defined just as a polymino but for 8-path and 6-path in place of the 4-path [29].

4.4. **Some reconstruction algorithms for discrete tomography.**

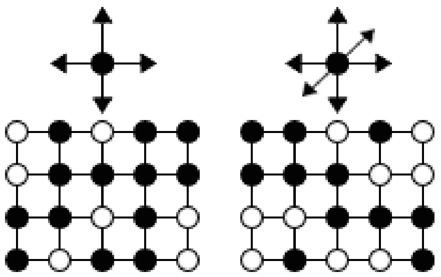


FIGURE 16. Above are directions of a 4-path (left) and a 6-path(right) and below a polymino and a 6-connected lattice set respectively.

4.4.1. *A reconstruction algorithms proposed by Brunetti, Daurat and Del Lungo.* In [30] Brunetti and Daurat offer an algorithm for reconstructing convex lattice sets for x-rays from two directions, but for a situation with x-rays from more than two directions they only gave the following result:

4.22. **Theorem.** *Let $\mathfrak{D} = \{v_1, \ldots, v_d\}$ be a set of directions such that $d \geq 7$ or one of the cross-ratios of the slopes of four directions arranged in an increasing order is not in $\{4/3, 3/2, 2, 3, 4\}$. We have d vectors $\big(p_{v_i}(min_{v_i}), \ldots, p_{v_i}(max_{v_i})\big)_{1\leq i\leq d}$. A convex set such that $X_{v_j}F(j) = p_{v_i}(j)$ for all $i \in [1, d]$ and $j \in [min_{v_i}, max_{v_i}]$ can be reconstructed in $O(d^2n^5)$ time.*

For proof, see [30].

Soon after, they wrote paper [31] in collaboration with A. Del Lungo, in which they present an algorithm, which can be used to reconstruct convex lattice sets based on approximate x-rays. This is a realistic situation as in practice measurement noise and inaccuracies lead to only approximate x-rays ever being available. It is, however very slow to use for sets that are simply convex and thus they present a class called *Q-convex* (or *strongly Q-convex*, which is a version where Q-convexity is generalized to any set of directions) lattice sets.

4.23. **Definition.** Let $\mathfrak{D}$ be a set of to independent linear lattice directions p and q on $\mathbb{Q}^2$. Let us assume that $p(x, y) = ax + by$ and $q(x, y) = cx + dy$ where $a, b, c, d \in \mathbb{Z}, ad - bc \neq 0$, $\gcd(a, b) = 1$ and $\gcd(c, d) = 1$. Let $M = (x_M, y_M)$. We now denote $p(x_M, y_M) = p(M)$. Now x-ray of a lattice set in the direction $p(M)$ (a constant) is the following function $X_pF : \mathbb{Z} \to \mathbb{N}_0$:

$$X_pF(i) = \text{card}(\{N \in F : p(N) = i\}). \tag{4.24}$$

Now function X_pF gives the number of lattice points of F on each line parallel to direction p. Let us now define four zones around a point

$M \in \mathbb{Z}^2$:

$$\begin{aligned} (4.25)\quad Z_0(M) &= \{N \in \mathbb{Z}^2 : p(N) \leq p(M)\text{and}q(N) \leq q(M)\} \\ Z_1(M) &= \{N \in \mathbb{Z}^2 : p(N) \geq p(M)\text{and}q(N) \leq q(M)\} \\ Z_1(M) &= \{N \in \mathbb{Z}^2 : p(N) \geq p(M)\text{and}q(N) \geq q(M)\} \\ Z_1(M) &= \{N \in \mathbb{Z}^2 : p(N) \leq p(M)\text{and}q(N) \leq q(M)\} \end{aligned}$$

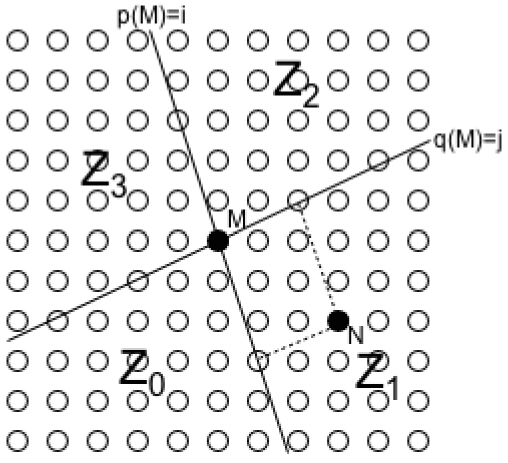


FIGURE 17. Lines $p(M) = i$ and $q(M) = j$ and zones $Z_0, \dots, Z_3$.

Using the previous definitions we can now define Q-convexity:

4.26. **Definition.** A lattice set F is quadrant convex or Q-convex around $\mathfrak{D} = \{p, q\}$ if $Z_k(M) \cap F \neq \emptyset$ for all $k = 0, 1, 2, 3$ implies $M \in F$.

In other words this means that if there is at least one point of F in all four zones $Z_k(M), k = \{0, 1, 2, 3\}$, the point M has to belong to F. A Q-convex set around two directions is always also line convex with respect to those directions. Let $P = (p_1, \dots, p_n), P' = (p'_1, \dots, p'_n), Q = (q_1, \dots, q_m)$ and $Q' = (q'_1, \dots, q'_m)$ be vectors whose elements are non-negative integers and the first and the last element in each vector are positive. A Q-convex set F around $\mathfrak{D}$ with x-rays such that $p_i \leq X_pF(i) \leq p'_i$ and $q_j \leq X_qF(j) \leq q'_j$ for all i, j is contained in the following lattice parallelogram:

$$(4.27)\qquad \Delta = \{N \in \mathbb{Z}^2 : 1 \leq p(N) \leq n \text{ and } 1 \leq q(N) \leq m\}$$

Let us denote the point $M \in \Delta$ the intersection of the p-line $p(M) = i$ and q-line $q(M) = j$ by $M = (i, j)$. Let $K = \{1, 2, 3, 4\}$. To use the algorithm of Brunetti, Daurat and Del Lungo, four Boolean variables $V_k(M)$ have to be associated one with each zone $Z_k(M)$. Then a 2-satisfiability formula is built on these variables so that there is a Q-convex set F around $\mathfrak{D}$ such that $p_i \leq X_pF(i) \leq p'_i$ for $i = 1, \dots, n$

and $q_j \leq X_qF(j) \leq q'_j$ for $j = 1, \dots, m$ if and only if the 2SAT formula is satisfiable. If an evaluation V of the variables $V_k(M)$ satisfying the 2SAT-formula exists, F is defined by the function ϕ:

$$F = \phi(V) \text{ iff } F = \{M \in \Delta : \overline{V_k(M)} \text{ is true } \forall\ k \in K\} \tag{4.28}$$

where $\overline{V_k(M)}$ is true if and only if $V_k(M)$ is false. The constraints of the 2SAT formula are Q-convexity, *lower bound* and *upper bound* on the x-rays. With lower bound the purpose is to express that in the direction p the x-ray values of the lattice set are greater than some prescribed integers. For line $p(M) = i$ must be $X_pF(i) \geq l$ for the variables $V_k(M)$ to satisfy the lower bound constraint. The upper bound constraint is satisfied if in the direction q the x-ray values are smaller than some prescribed integers, for line $X_q(M) = j$ must be $X_qF(j) \leq l$.To see how Q-convexity, upper- and lower bound can be expressed with Boolean variables, see [31].

4.29. **Algorithm.** Now let us fix four bases A, B, C and D such that $p(A) = 1, p(B) = n, q(C) = 1$ and $q(D) = m$. The following 2SAT-formula named APPROX presented by Brunetti, Daurat and Del Lungo is satisfiable if and only if there is a Q-convex set F around $\mathfrak{D}$ containing the bases A, B, C and D with x-rays along p and q such that $p_i \leq X_pF(i) \leq p'_i$ for $i = 1, \dots, n$ and $q_j \leq X_qF(j) \leq q'_j$ for $j = 1, \dots, m$:

$$\begin{aligned} &\text{APPROX}(P, P', Q, Q', A, B, C, D) = \\ \text{QCONV} \wedge &\bigwedge_{1 \leq i \leq n} \big((\text{LB}(p, i, p_i) \wedge \text{UB}(p, i, p'_i, C, D)\big) \\ \wedge &\bigwedge_{1 \leq j \leq m} \big(\text{LB}(q, j, q_j) \wedge \text{UB}(q, j, q'_j, A, B)\big) \end{aligned} \tag{4.30}$$

where P, P', Q and Q' are as before, QCONV stands for Q-convexity, LB for lower bound and UB for upper bound. These conditions can be expressed in the following way:

4.31. **Lemma.** *If an evaluation V of the variables $V_k(M)$ satisfies the formula QCONV, then $F = \phi(V)$ is Q-convex around $\mathfrak{D}$.*

If an evaluation V of the variables $V_k(M)$ satisfies QCONV$\wedge$LB(p, i, l)*, then the x-ray of $F = \phi(V)$ along p is such that $X_pF(i) \geq l$.*

If an evaluation V of the variables $V_k(M)$ satisfies QCONV $\wedge$UB(q, j, l, A, B)*, then $F = \phi(V)$ contains the bases A and B and its x-ray of along q is such that $X_qF(j) \leq l$.*

For proof and formal definitions of QCONV, LB(p, i, l) and UB(q, j, l, A, B), see [31].

The algorithm chooses bases A, B, C and D and builds the formula 4.30. If the formula is satisfiable, the solution is given by $F = \phi(V)$. If the reconstruction attempt fails, the algorithm chooses new four bases and repeats the procedure. For proof and additional information about how to use the algorithm, see [31].

In a situation where $P = P'$ and $Q = Q'$ and $\sum_{i=1}^{n} p_i = \sum_{j=1}^{n} q_j$ the problem of finding suitable a Q-convex set F around $\mathfrak{D}$ becomes *exact consistency* problem. For solving that, Brunetti, Daurat and Del Lungo propose the following algorithm:

4.32. **Algorithm.** Let A and B be two bases such that $p(A) = 1$ and $p(B) = n$. The following formula named EXACT is satisfiable if and only if there is a Q-convex set F around $\mathfrak{D}$ containing bases A and B and with x-rays along p and q such that $X_pF(i) = p_i$ for $i = 1, \dots, n$ and $X_qF(j) = q_j$ for $j = 1, \dots, m$.
(4.33)
$$\text{EXACT}(P, Q, A, B) = QCONV \bigwedge_{1 \le i \le n} LB(p, i, p_i) \bigwedge_{1 \le j \le m} UB(q, j, q_j, A, B)$$

For proof and additional information, see [31]. This algorithm may be attractive in theoretical situations, especially as in both algorithms the number of possible positions of the bases is what defines the maximum number of reconstruction attempts and it is obviously much lower in the second algorithm. The conditions of the algorithm 4.32 are, however, not always met and with practical measurements the previous formula 4.29 is most likely more applicable.

For an algorithm that can reconstruct a lattice set from x-rays from more than two directions, additional conditions are required. First we need to define Q-convexity around larger number of directions:

Let U be a set of d directions $\{\vec{u_h} = (a_h, b_h)_{h=1}^{d}\}$ where a_h and b_h are a pair of coprime integers and $b_h \ge 0$. The vector $\vec{u_h} = (a_h, b_h)$ can be expressed in linear form in the following way: $u_h(x, y) = b_h x - a_h y$. Given two directions $u_i, u_j \in U$ the four zones $Z_k^{(i,j)}(M)$ are defined as in 4.25 around every point $M \in \mathbb{Z}^2$. Altogether there are $2d(d-1)$ zones around each $M \in \mathbb{Z}^2$. Let us choose $2d$ of these zones. The point M on a line in the direction u_h divides it into two semi-lines that have origin in M:

$$s_h^+(M) = \{N \in \mathbb{Z}^2 : u_h(N) = u_h(M) \text{ and } \vec{u_h} \cdot (\vec{ON}) \ge \vec{u_h} \cdot (\vec{OM})\}$$

$$s_h^-(M) = \{N \in \mathbb{Z}^2 : u_h(N) = u_h(M) \text{ and } \vec{u_h} \cdot (\vec{ON}) \le \vec{u_h} \cdot (\vec{OM})\}$$

where O stands for any point of origin and $''\cdot''$ is a scalar product of two vectors.

4.34. **Definition.** An almost-semi-plane along a set of directions U is a zone $Z_k^{(i,j)}(M)$ such that for each direction u_h of U only one of the two semi-lines $s_h^+(M)$ and $s_h^-(M)$ is contained in it.

Brunetti, Daurat and Del Lungo denote the almost-semi-planes surrounding M in the following way:

Let $\prod_0(M)$ be the almost-semi-plane that contains $s_h^+(M)$ for each $h = 1, \ldots, d$. The other almost-semi-planes encountered clockwise around M from $\prod_0(M)$ are $\prod_1(M), \ldots, \prod_{2d-1}(M)$. Using almost-semi-planes Q-convexity can now be defined for more than two directions:

4.35. **Definition.** A lattice set F is strongly Q-convex around a set of directions U if and only if for each $M \notin F$ there exists an almost-semi-plane $\prod_k(M)$ around M such that $F \cap \prod_k(M) = \emptyset$.

The reconstruction algorithm for more than two directions now tries to find out whether there is a strongly Q-convex set F around U such that

$$p_{h,i} \leq X_{u_h}F(i) \leq p'_{h,i} \text{ for } i = 1, \ldots, n_h \text{ and } d = 1, \ldots, h \tag{4.36}$$

where $P_1 = (p_{1,1}, \ldots, p_{1,n_1}), P'_1 = (p'_{1,1}, \ldots, p'_{1,n_1}), \ldots, P_d = (p_{d,1}, \ldots, p_{d,n_d})$ and $P'_d = (p'_{d,1}, \ldots, p'_{d,n_d})$ are $2d$ vectors whose elements are non-negative integers and the first and the last element of each vector are positive.

Similarly to 4.27, a Q-convex set F around U with x-rays such that $p_{h,i} \leq X_{u_h}F(i) \leq p'_{h,i}$ is contained in a lattice polygon:

$$\Delta = \{N \in \mathbb{Z}^2 : 1 \leq u_h(N) \leq n_h \ \forall\ 1 \leq h \leq d\} \tag{4.37}$$

With the difference of changing $Z_k(M)$ to $\prod_k(M)$, the same strategy as before can now be used to find if there is a strongly Q-convex set as described in 4.36. A 2SAT formula is built on Boolean variables $V_k(M)$ where $K = \{0, 1, \ldots, 2d-1\}$ and $M \in \Delta$ so that the described strongly Q-convex set F around U exists if and only if the formula is satisfiable. If there is an evaluation V of the variables $V_k(M)$ satisfying the 2SAT formula, the solution is given by $F = \phi(V)$ where ϕ is defined as before. The formula APPROX can easily be generalized to a situation with several x-rays. The constraints of APPROX are the same as before with the exception of QCONV being replaced by SQCONV, strong Q-convexity.

The lower bound for strongly Q-convex sets does not differ much from the Q-convex case, but the formula by which upper bound is expressed depends on a much larger number of bases:

$$\mathrm{SUB}(p, i, p'_i, A_1, B_1, \ldots, A_{h-1}, B_{h-1}, A_{h+1}, B_{h+1}, A_d, B_d)$$

A_j and B_j are the bases in the direction u_j. While the reader is referred to the original paper of Brunetti, Daurat and Del Lungo [31] for the formal definition of the Boolean expression for SUB, the following lemma explains the bases that appear in the formula:

4.38. **Lemma.** *Let K be a set of indices $\{0, 1, \ldots, 2d-1\}$ and $K_h^+ = \{t \in K : s_h^+ \subset \prod_t(M)\}$. For any $M \in \Delta$ such that $u_h(M) = i$ there exists at most one $k \in K_h^+$ such that for any $k' \in K_h^+ \backslash \{k\}$ the almost semi-plane $\prod_{k'}(M)$ contains one of the points $A_1, B_1, \ldots, A_{h-1}, B_{h-1}, A_{h+1}, B_{h+1}, A_d, B_d$.*

For proof, see [31].

The algorithm as proposed by Brunetti, Daurat and Del Lungo for reconstructing strongly Q-convex lattice sets from more than two projections can now be written in the following way:

4.39. **Algorithm.** Let $P_1, P_1', \ldots, P_d, P_d'$ be as before. Let us fix $2d$ bases $A_1, B_1, \ldots, A_d, B_d$ where each pair of bases corresponds to one of the d directions. The following formula is satisfiable if and only if there is a strongly Q-convex set F around U that has x-rays along u_h as described in 4.36 and that contains the bases $A_1, B_1, \ldots, A_d, B_d$:

$$\begin{aligned}(4.40)\qquad &\mathrm{APPROX}(P_1, P_1', \ldots, P_d, P_d', A_1, B_1, \ldots, A_d, B_d) \\ &= \mathrm{SQCONV} \wedge \bigwedge_{1 \le h \le d, 1 \le i \le n_h} \mathrm{SLB}(u_h, i, p_i) \\ &\wedge \mathrm{SUB}(p, i, p_i', A_1, B_1, \ldots, A_{h-1}, B_{h-1}, A_{h+1}, B_{h+1}, A_d, B_d)\end{aligned}$$

For the ways SQCONV, SLB and SUB are expressed with Boolean variables, see [31].

The algorithm works by choosing d pairs of bases for which it builds the 2SAT formula APPROX. As before, the number of different positions the bases can take defines the maximum number of attempts it can take to reconstruct F, which in this case is n^{2d}.

4.4.2. *Discrete algebraic reconstruction technique (DART).* The discrete algebraic reconstruction technique, often referred to as the DART-algorithm, was proposed by Batenburg and Sijbers in 2007 in their paper [32]. In 2011 they wrote a new, more elaborate paper [33] on the subject, giving a description of the algorithm along with practical examples validating it, which was needed as DART is a heuristic algorithm without a guarantee of convergence. In the latter paper they also went on to show that when only a small number of projection images is available or the angular range of the projections is small, DART-algorithm gives more accurate reconstructions than alternative methods. The algorithm is also robust against noise. It works by first solving a problem as if it were a problem of continuous tomography using some *algebraic reconstruction method*, which is a group of iterative reconstruction techniques, and then discretizing reconstructed values by setting them to closest one of the possible values the image is known to attain.

DART-algorithm differs from many other algorithms of discrete tomography in a few important ways. First of all, the images reconstructed using DART do not necessarily need to be binary, it works with any set of allowed grey values $\{p_1, p_2, \ldots, p_\ell\}$, although it works best when the number of grey values is not larger than five. Even though there needs to be prior knowledge about the grey values and simply knowing that the number of different values is small enough will not suffice for the purposes of DART-algorithm, it is robust with respect to errors in estimation of the values. Secondly, even though DART-algorithm gives good results when the number of projections is small, it can be used also when the number of projections is in dozens or even hundreds. Thirdly, as has been mentioned earlier, many algorithms of discrete tomography require the object to be of a particular shape or to have specific convexity or connectedness properties. Many algorithms can only reconstruct for example hv-convex or Q-convex objects. With DART there are no particular requirements for the shape of the object being reconstructed.

0: The initial step To use DART-algorithm, one first needs to execute a fixed number of iterations with some algebraic reconstruction method (the group of algebraic reconstruction methods is often called ARMs). This part is the initial step. A range of different algebraic reconstruction methods can be used for this. The one Batenburg and Sijbers use in their paper is called SART (short for Simultaneous Algebraic Reconstruction Technique), which is an algorithm in which the reconstruction is updated with each projection angle separately. Following is a very brief description of SART:

Let $\boldsymbol{W} \in \mathbb{R}^{m\times n}_{\geq 0}$ be a linear operator called the projection matrix that maps the image $\boldsymbol{x} = (x_i) \in \mathbb{R}^n$ to $\boldsymbol{p}$, which is a vector of measured data:

$$\boldsymbol{W}\boldsymbol{x} = \boldsymbol{p} \tag{4.41}$$

$\boldsymbol{W}$ and $\boldsymbol{p}$ can be decomposed into d blocks of k rows in the following way:

$$\boldsymbol{W} = \begin{pmatrix} \boldsymbol{W}^1 \\ \vdots \\ \boldsymbol{W}^d \end{pmatrix}, \boldsymbol{p} = \begin{pmatrix} \boldsymbol{p}^1 \\ \vdots \\ \boldsymbol{p}^d \end{pmatrix} \tag{4.42}$$

Each of the blocks $\boldsymbol{W}^t = (w_{ij}^t)$ represents the projection operator from one angle and $\boldsymbol{p}^t$ the corresponding projection data. Now let $\gamma_j^t = \sum_{i=1}^k w_{ij}^t$ for $j = 1, \dots, n$ and $t = 1, \dots, d$ and let $\beta_i^t = \sum_{j=1}^n w_{ij}^t$ for $i = 1, \dots, m$ and $t = 1, \dots, d$. We denote the set of all permutations of the numbers $1, \dots, d$ by S_d. Let σ be a random element of S_d. To use the SART-algorithm, we first need to make and initial guess $\boldsymbol{x} = \boldsymbol{x}^{(0)}$. The algorithm computes a new estimate $\boldsymbol{x}^{(s)}, s = 1, 2, \dots$ iteratively from the previous estimate $\boldsymbol{x}^{(s-1)}$ using the formula

$$\boldsymbol{x}_j^{(s)} = \boldsymbol{x}_j^{(s-1)} + \lambda \frac{1}{\gamma_j^{\sigma(s)}} \sum_{i=1}^{k} \frac{w_{ij}^{\sigma(s)} r_i^{(s)}}{\beta_i^{\sigma(s)}}, \ j = 1, \dots, n \tag{4.43}$$

where $\boldsymbol{r}^{(s)} = \boldsymbol{p}^{\sigma(s)} - \boldsymbol{W}^{\sigma(s)} \boldsymbol{x}^{(s-1)}$ and λ is a relaxation factor [33].

1: The segmentation The next step in using the DART-algorithm is to get an image that consists of pixels of only the allowed grey values $\{p_1, p_2, \dots, p_\ell\}$. This is achieved by *segmenting* the image. Let $\boldsymbol{x}^{(t-1)}$ be the current reconstruction in the beginning of the tth iteration. The segmented reconstruction $\boldsymbol{s}^{(t)} \in \mathbb{R}^n$ is computed from $\boldsymbol{x}^{(t-1)}$ so that each pixel has one of the allowed grey values $\{p_1, p_2, \dots, p_\ell\}$. A common way to do this is by simply rounding the pixel values to whichever of the grey values is closest to it. While there are several ways to get the segmentation, one more formal way to do this is by defining a threshold τ_i by the following formula:

$$\tau_i = \frac{p_i + p_{i+1}}{2}, i = 1, 2, \dots, \ell - 1 \tag{4.44}$$

Using this, we can define a threshold function $r : \mathbb{R} \to \{p_1, p_2, \dots, p_\ell\}$:

$$r(v) = \begin{cases} p_1, & v < \tau_1 \\ p_2, & \tau_1 \le v < \tau_2 \\ & \dots \\ p_n, & v \le \tau_{\ell-1} \end{cases} . \tag{4.45}$$

2: Selection of free pixels After the segmentation step the image is further divided into two groups of pixels: free pixels that are denoted by $U^{(t)}$ and fixed pixels that are denoted by $F^{(t)}$. The set $U^{(t)}$ consists of pixels that are adjacent to at least one pixel that has different grey value. One can either choose the adjacent pixels of pixel i to be the ones 4-connected to it or the ones 8-connected to it. DART used the 8-connected neighbourhood of pixel i. This neighbourhood is denoted by $N(i) \subset \{1, 2, \dots, n\}$. Set F is formed of the pixels that are not free. Note that $U^{(t)} \cap F^{(t)} = \emptyset$ and all the pixels in the image are in $U^{(t)} \cup F^{(t)}$. Edges or boundaries of the object are formed of the free pixels, as they are the part where the change from one density or a grey value to another happens. The boundary pixels, denoted by $B^{(t)} \subset \{1, 2, \dots, n\}$, are calculated from $\boldsymbol{s}^{(t)}$.

If the shape of the object is such that there are for example holes in it, the edges of the holes should of course be also formed of free pixels. This, however, does not always happen which leads to DART-algorithm overlooking possible holes in the structure of the object. To avoid this, a subset of free pixels that are not connected to current edge must be selected with each iteration step and updated along with pixels that form the current edge. This is done by supplementing the set $U^{(t)}$ with some random pixels from F so that each fixed pixel is freed independently of neighbouring pixels with a probability of $1-p$ where $0 < p \leq 1$. This process also helps with problems caused by noisy measurement data and grey level errors. Only the set of free pixels $U^{(t)}$ is subjected to update with each iteration step of DART. First, only the boundary pixels are selected as free pixels: $U^{(t)} = B^{(t)}$. Some fixed pixels are then added to $U^{(t)}$ by a randomized procedure as described above. With each new DART-update the selection process will change. This allows changes to happen in the areas of the image that are not close to the current edge.

3: Arm with fixed pixels: In the following, we consider the operation of fixing pixels more thoroughly. Let

$$\begin{pmatrix} | & & | \\ \boldsymbol{w}_1 & \dots & \boldsymbol{w}_n \\ | & & | \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} p_1 \\ \vdots \\ p_m \end{pmatrix} \tag{4.46}$$

be a linear system where $\boldsymbol{w}_i$ denotes the ith column vector of $\boldsymbol{W}$. Next step in using the DART-algorithm is to define an operation that fixes the variable x_i at value $v_i \in \mathbb{R}$. The goal is to be able to compute $x_i \boldsymbol{w}_i \in \mathbb{R}^m$ beforehand so that the variable x_i can be removed from $\boldsymbol{x}$, the column $\boldsymbol{w}_i$ from $\boldsymbol{W}$ and $v_i \boldsymbol{w}_i$ can be subtracted from the right-hand-side. We now get a new system:

$$\begin{pmatrix} | & & | & | & & | \\ \boldsymbol{w}_1 & \dots & \boldsymbol{w}_{i-1} & \boldsymbol{w}_{i+1} & \dots & \boldsymbol{w}_n \\ | & & | & | & & | \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_{i-1} \\ x_{i+1} \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} p_1 \\ \vdots \\ p_m \end{pmatrix} - v_i \boldsymbol{w}_i \tag{4.47}$$

This new system has the same number of equations as the original system 4.46 but the number of variables has decreased by one.

With each DART-iteration, pixels i in the set $F^{(t)}$ are fixed in their values $s_i^{(t)}$. This further reduces the number of variables from n to

$n- \mid F^{(t)} \mid$. Now the following version of the system that only has free pixels as variables can be solved with a constant number of iterations using the chosen algebraic reconstruction method:

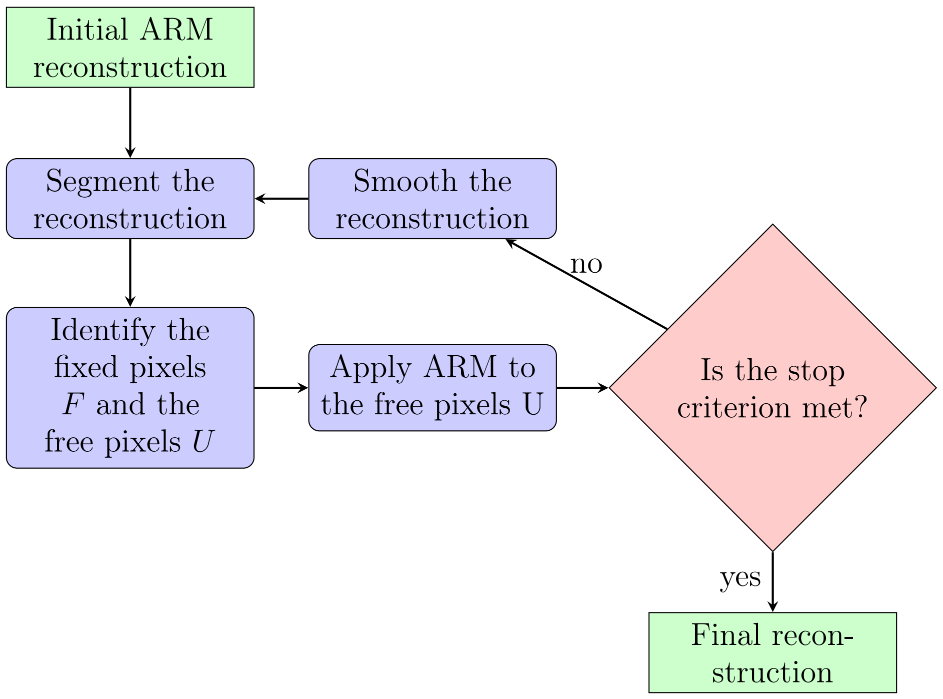
$$\tilde{\boldsymbol{W}}\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{p}} \tag{4.48}$$

Given that the grey values assigned for the fixed pixels are close to being correct with respect to the unknown object being reconstructed, solving the system 4.48 should improve the values of the remaining free pixels compared to solving the underdetermined system 4.46.

4: Smoothing operation: The measurement noise now only affects the free pixels, causing heavy fluctuations on their values. To deal with this problem, we need a smoothing operation. In order to regularize the system, after applying the algebraic reconstruction method, a Gaussian smoothing filter with radius of one is applied to the boundary pixels.

5: Termination criterion: The last step in using the DART-algorithm is setting a termination criterion. While many practical examples performed by Batenburg and Sijbers demonstrate that in the relevant situations they studied the algorithm rapidly converges towards an accurate reconstruction, due to the heuristic nature of the algorithm, it cannot be formally stated under what conditions it converges. The termination criterion can be either a fixed number of iterations or the *total projection error* $E : \mathbb{R}^n \to \mathbb{R}$ defined as

$$E(\boldsymbol{x}) = \parallel \boldsymbol{W}\boldsymbol{x} - \boldsymbol{p} \parallel_2 \tag{4.49}$$


Initial ARM reconstruction
Segment the reconstruction
Smooth the reconstruction
no
Identify the fixed pixels F and the free pixels U
Apply ARM to the free pixels U
Is the stop criterion met?
yes
Final reconstruction

Above: Flowchart of the DART-algorithm.

Following is the DART-algorithm in pseudo code as presented by Batenburg and Sijbers in [33].

4.50. **Algorithm.** *DART-algorithm*

```
Compute a start reconstruction $\boldsymbol{x}^{(0)}$ using ARM;
$t := 0$;
**while** (stop criterion is not met) **do**;
**begin**
        t:=t+1;
        Compute the segmented image $\boldsymbol{s}^{(t)} = r(\boldsymbol{x}^{(t-1)})$;
        Compute the set $B^{(t)}$ of the boundary pixels of $\boldsymbol{s}^{(t)}$;
        Compute the set $U^{(t)}$ of the free pixels of $\boldsymbol{s}^{(t)}$;
        Compute the set $F^{(t)} = \{1, \dots, n\} \setminus U^{(t)}$ of fixed pixels;
        Compute the image $\boldsymbol{y}^{(t)}$ from $\boldsymbol{x}^{(t-1)}$ and $\boldsymbol{s}^{(t)}$, setting
            $y_i^{(t)} := s_i^{(t)}$ if $i \in F^{(t)}$ and $y_i^{(t)} := x_i^{(t-1)}$ otherwise;
        Using $\boldsymbol{y}^{(t)}$ as the start solution, compute the
            ARM-reconstruction $\boldsymbol{x}^{(t)}$, while keeping the pixels
            in $F^{(t)}$ fixed;
        Apply a smoothing operation to the pixels in $U^{(t)}$;
**end**
```

The results gained by using DART depend on various things. One of the most important ones is the choice of algebraic reconstruction method used; in addition to SART, which was described earlier, some common choices include SIRT (short for Simultaneous Iterative Reconstruction Technique) and ART (short for Algebraic Reconstruction Technique). Other things that affect the success of the DART-algorithm are the number and placement of the projection angles, the number most often being much smaller than in general tomography, and the choice of p, which gives the probability with which each pixel is freed in each iteration. When $p = 1$ the set of free pixels U is not supplemented with random pixels and DART is most often not able to reconstruct an image with a hole, but the results improve dramatically when p is even slightly smaller than 1. The more noise there is, the smaller p needs to be in order to gain good reconstructions. If the level of the noise is low a value of p closer to 1 can be chosen as then a good reconstruction may be gained with fewer computations. For examples of how the level of noise and the fix probability work together, see [35]. As was mentioned earlier, DART also works best when the number of grey values is not larger than five. The number of DART-iterations is obviously also an extremely important factor with respect to quality of the reconstructions [32], [33], [35].

There are two main methods for implementing the DART-algorithm: ray-driven and pixel (or in 3D-case voxel)-driven. Ray-driven methods visit all the pixels lying on a projection line, adding together the values of each pixel to form a total projection of each line. The projection lines are visited sequentially. In pixel-driven scheme the total projections are collected simultaneously, visiting the pixels one by one. The pixel-driven method works better for the DART-algorithm as many of the pixels are fixed in each iteration step and only the pixels in the set of free pixels U need to be calculated in each iteration. If the set of fixed pixels F is large compared to the total number of pixels, the running time of the ARM-iterations can be reduced significantly by the use of pixel-driven scheme. If the value of each pixel i is denoted by $v_i \in \mathbb{R}$ and ith column vector of the projection matrix $\boldsymbol{W}$ is denoted by $\boldsymbol{w}_i \in \mathbb{R}^m$, the projection $\boldsymbol{q}_i \in \mathbb{R}^m$ of each of the pixels i is calculated in the following way:

$$\boldsymbol{q}_i = v_i \boldsymbol{w}_i \tag{4.51}$$

In the practical experiments carried out by Batenburg and Sijbers in [33], the pixel-driven method was used on a number of phantoms of the size 512×512 pixels so that each projection for each angle consists of 512 detector values, one value for each pixel. Parallel beam geometry was used in these simulated experiments and the number of grey values was between two and five, except for the well-known Shepp-Logan phantom, in which the number of grey values is six. In these simulated situations the images were not contaminated by noise. Compared to the other algorithms studied (Filtered Back Projection, Total Variation Minimization, SART), DART-algorithm consistently gave more accurate results with a smaller number of projections. Near-perfect reconstructions were acquired depending on the complexity of the image with a number of projections ranging from 8 to 40, 40 being the number of projections used for reconstruction of the Shepp-Logan phantom in which the number of grey values is larger than the suggested maximum number for the DART-algorithm.

DART-algorithm proved even more useful compared to its continuous counterparts when the reconstructions were attempted with a limited angle of projections. DART's ability to utilize prior knowledge of the grey values found in the image seems to yield reconstructions that are vastly better than those gained by algorithms of continuous tomography. Even though SART also gave relatively good results with incomplete angular range, DART was the only one out of the algorithms tested in [33] that gave close to perfect reconstructions with as little as 80 degree projection angle.

In the same paper a real-life problem acquired by x-ray scanning a diamond was also studied. In this situation the scanned object was obviously not pixelized, so additional discretization step was added to the procedure. As always when working with real data, the sinogram was contaminated by noise. With real data, the grey levels are also often only known approximately, which can make reconstructions with DART less accurate. as was mentioned earlier, the choice of the probability p grows in importance when the data is noisier and also when there are errors in the assumed values of the grey levels. The reason why a lower fix probability yields better results when the data is noisy is that all the noise is distributed between free pixels in the ARM-iterations. The fewer free pixels there are, the more their value is determined exclusively by noise. The same is true for errors in the grey values; The smaller the number of free pixels is, the more they will be affected by the projection error resulting from over-or under estimating the values. With the estimation of grey values up to 10% off from the actual value, only 0.5% of the pixels were misclassified with both values $p = 0.5$ and $p = 0.85$.

The main problem with the DART-algorithm is its heuristic nature. There is no certain way to predict its behaviour in terms of convergence. Even though practical trials have shown that DART converges in a smooth way, it cannot be guaranteed that it converges towards a correct solution. Setting the fix probability p close but not equal to 1 has given the best convergence rate but as was discussed in the previous paragraph, larger fix probabilities make the algorithm less stable when dealing with noisy data or errors in estimation of the grey values. In practice, even though DART-algorithm often converges slower than its continuous counterparts and the number of iterations needed is thus larger, the fact that the ARM-iterations only need to be performed on a subset of pixels increases the speed of the process. Its ability to find excellent reconstructions from a much smaller number of projections makes it an effective tomographic tool [33].

In order to demonstrate the usefulness of DART-algorithm, similar small image that was used in examples 3.42 and 3.45 is reconstructed in the following example implementing it with same number of projections as was used in 3.45. The code used for the reconstruction is from Astra Toolbox (which can be downloaded for free online). The image used in the other examples had to be altered some in order for it to fulfill the requirements of the said code.

4.52. **Example.** The readymade code for a reconstruction using DART required the image that is being reconstructed to be 512 times 512 pixels, so significantly larger than the image used in previous examples. It also required the edges of the image to be black for a reason that remains a mystery to the author. It is possible that some other colour would have worked as well but the code was run with the same image with the colour of the edges being white and that did not work. It also appears that not all of the area of the image is active measurement area, the code loses everything happening closer to the edges of the image that is not black. Actual size of the active measurement area is somewhere between 512 times 512 pixels and 300 times 300 pixels, which is the size of the image in the middle of the black frames. The small image used in examples 3.42 and 3.45 was altered to meet the requirements of the code using an image processing software. As the code is not author's own, it remains somewhat unclear whether any measures have been taken to avoid inverse crime during the reconstruction. In the figure 18 are the original picture and the reconstruction without added noise. As could be expected from the examples in [32], [33] and [35], the reconstruction from the noise free data is nearly perfect, only 137 pixels that are not correct. Due to not knowing whether the amount of misplaced pixels ought to be compared to the total number of pixels in the image or to that of the active measurement area, the size of which remains unclear, we only give the number of pixels the colour of which is incorrect compared to the original picture and the reader can decide if they would rather compare it to the total amount of pixels, 262144, or maybe the number of pixels inside the black frame, 90000. The number of projection angles used for the reconstructions was 20, same as in example 3.45.

As is often a problem while working with codes that are not one's own, it is also unclear to the author, in which part of the code is it possible to change the fix probability. By altering a variable named "masking random" in the code, results similar to what can be assumed would be gained by altering fix probability were aquired. The writer of the code describes this variable as "percentage of random points", which is here assumed to mean the percentage of the pixels that can change, that is free pixels. This would mean that this variable can be written as $1 - p$ (the fix probability subtracted from 1). In the figure 19 the reconstructions of the image can be seen with two different levels of gaussian noise, 1% and 2%, and with the values of masking random 0.1, 0.5 and 0.9. Even though the reconstructions are still quite good, clearly the role of noise is quite significant while using the DART-algorithm. This is not surprising considering the important role of apriori knowledge about the grey values found in the picture. It is possible that the effects of noise could have been somewhat lessened

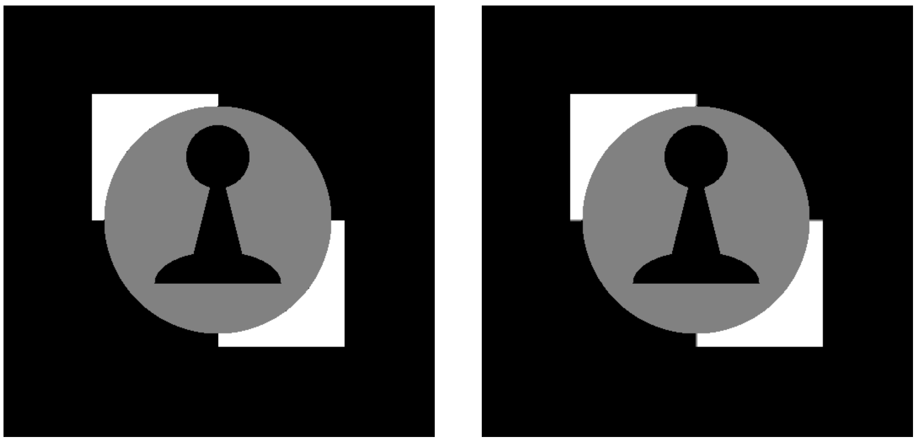


Figure 18. On the left the original image used for the reconstruction and on the right DART-reconstruction from 20 projection angles with no added noise. There are only 137 pixels that are not the colour they are supposed to be in the reconstruction.

by for example tampering with the smoothing filter, but the results gained here are relatively similar to the results gained in similar experiments in for example [35], so there is no reason to believe that there is anything significantly wrong with the filter provided in the code.

The whole Astra Toolbox, from which the code used for these reconstructions can be found, can be downloaded from this website: https://sourceforge.net/projects/astra-toolbox/ (5.5.2016).

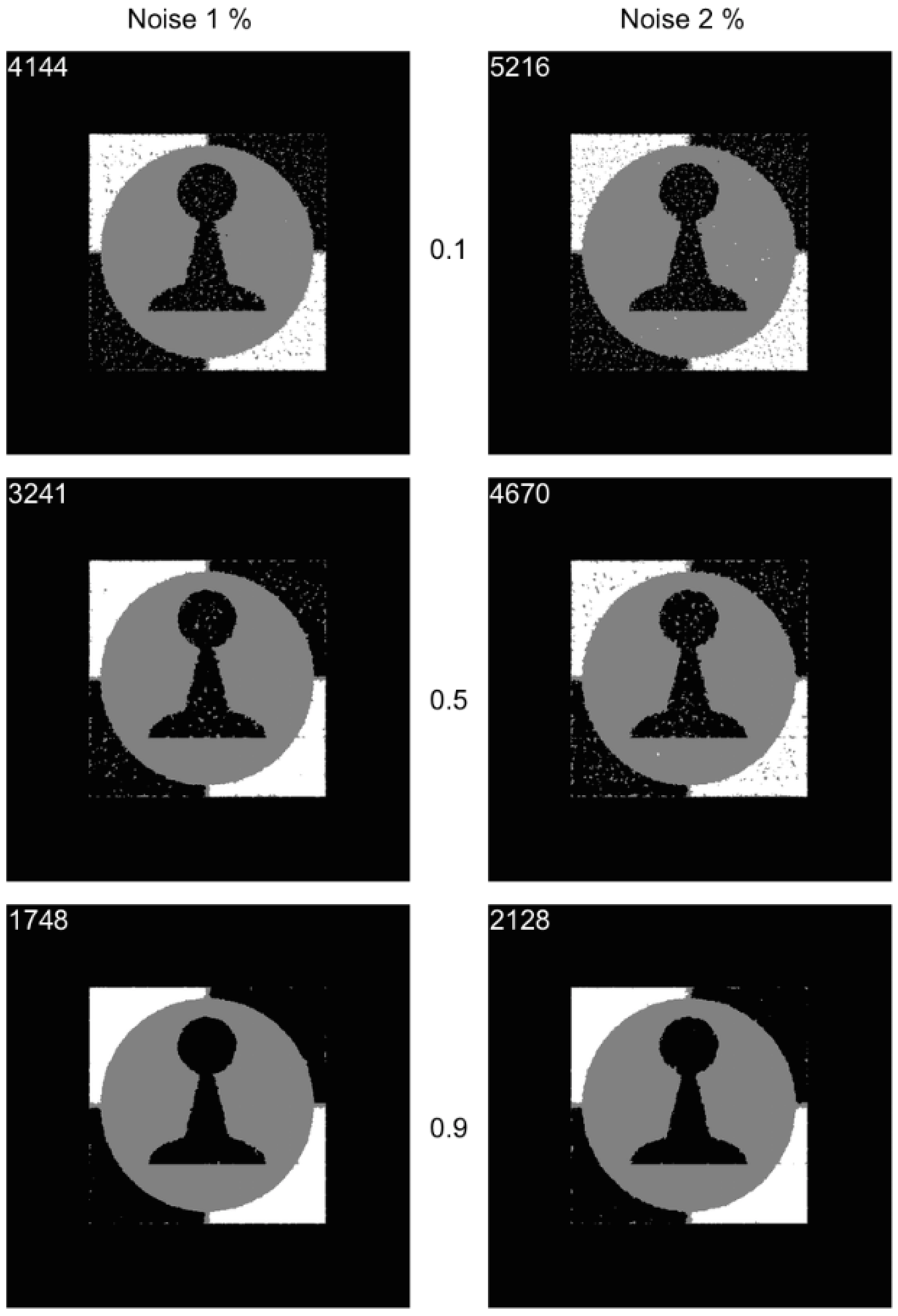


Figure 19. On the left side are the reconstructions with 1% of added gaussian noise, on the right side the level of noise is 2%. The values of masking random from up to down are 0.1, 0.5 and 0.9. The white number in the upper left corner of each image is the number of pixels that are wrong in the reconstruction.

4.4.3. *Monte Carlo algorithms.* Another heuristic group of algorithms commonly used to solve discrete tomographic reconstruction problems is called *Monte Carlo algorithms.* It is related to group of *Las Vegas algorithms*, but where Las Vegas algorithms always give the correct answer with a running time that is a random variable, Monte Carlo algorithms produce a solution in a fixed number of steps but there is a possibility, usually with probability $< 1/3$, that the solution is incorrect. Las Vegas algorithm that is terminated early thus not guaranteeing the correct answer is in fact Monte Carlo algorithm. Both Monte Carlo- and Las Vegas algorithms belong to a larger group called randomized algorithms, algorithms that make random choices during execution. It is a group that was originally developed to be a tool of computational number theory but has since found uses in many other fields as well. The reason for the attractiveness of randomized algorithms is their simplicity and speed compared to most alternative methods. Most Monte Carlo algorithms can be solved by integration. There are two basic approaches to Monte Carlo algorithms: direct sampling and Markov-chain sampling, often referred to as MCMC (Markov-chain Monte Carlo). As the more simple method of direct sampling rarely works, most situations call for MCMC.

Following is merely a brief introduction to Monte Carlo methods giving some explanation about the general idea and the basic consepts without attempting to get into theory of it on an any deeper level. The basic idea behind Monte Carlo methods is that given a sufficiently large set X and a distribution $p(X)$ over it, we can approximate the distribution by drawing a set of N independent and identically distributed samples:

$$p_N(x) = \frac{1}{N}\sum_{i=1}^{N} 1(x^{(i)} = x) \underset{N \to \infty}{\longrightarrow} = p(x) \tag{4.53}$$

The same samples can also be used to calculate the expectations:

$$E_N(f) = \frac{1}{N}\sum_{i=1}^{N} f(x^{(i)}) \underset{N \to \infty}{\longrightarrow} E(f) = \sum_{x} f(x)p(x) \tag{4.54}$$

These expectations can be used to solve some complicated multi-dimensional integrals. The sampling is what distinguishes Monte Carlo methods from general integration schemes in which the object of interest is the one-dimensional integral

$$I = \int_a^b f(x)dx \approx \sum_{l=0}^{M-1} \delta \cdot f((x_l + x_{l+1})/2) \tag{4.55}$$

where M is the number of slices of the width δ. Approximating integral with a sum converges towards the correct solution when $M \to \infty$ but this method becomes more difficult to use and less accurate when the integral that needs to be computed is multi-dimensional [38], [39], [40].

A direct sampling (sometimes also called simple sampling) Monte Carlo method can be used to overcome these problems. Let's say we want to find out the area of an irregularly shaped patch of grass in the desert. We first draw a square an area of which we can of course calculate around the patch and arm ourselves with a large number of pebbles. We then start randomly throwing the pebbles into the square. We can then estimate the area of the patch by calculating what percentage of the pebbles landed inside the patch. As the number of the pebbles thrown gets larger, the percentage converges towards the percentage of the area of the square that the patch occupies. A small number of pebbles is enough to give a decent estimate for the area but sadly this method proves to be useless in most practical situations such as the one described next.

MCMC-method we get out of this when we realize that our upper body strength is simply not enough to throw the pebbles all the way to the other side of the patch. We start at a random location, throw the pebble into a random direction, walk to the pebble and throw a new one to a random direction from there and so on. This method works well until we throw a pebble out of the square. When this happens, this throw is *rejected*, the thrower doesn't move and the previous throw is counted twice, meaning that a second pebble is placed next to or if possible on top of the previous pebble. After a large amount of throws the pebbles should be scattered around the squares with piles of pebbles close to the edges and especially corners of the square due to the rejected throws.

Rejection method ensures that the path generated by flinging pebbles, or in less practical terms the Markov chain, is reversible. It is often referred to as *detailed balance condition*. It is fundamental to the *Metropolis algorithm* that is introduced in a little bit and it is explained in slightly more formal terms in that context. Unlike the direct sampling method of simply flinging pebbles around from one spot, MCMC-method works in most situations. It is, however, much less economical and often requires the kind of time and resources that are simply not available. Even a long Markov-chain often produces only a small number of independent samples, which makes the result very approximative, and the programs based on them are usually extremely sensitive to noise and other small irregularities. Determining the step length or in terms of pebbles and grass patch the length of each throw can also be problematic. If the step length is too short, we end up sampling only a very small area of the grass patch as it takes a huge number of pebble flings to get far enough from the starting point, in other words convergence towards a good estimate is very slow. If, however, the step length is too long we end up throwing the rocks outside of the square all the time, leading to a very large number of rejected throws, which also leads to very slow convergence.

The value of π can also be estimated using the same idea so that the patch of grass is in fact a circle the radius of which is 1. This situation is denoted using a probability density π, which in this case is the square

$$\pi(x,y) = \begin{cases} 1, \text{ if } \mid x \mid\leq 1 \text{ and } \mid y \mid\leq 1 \\ \quad 0 \text{ otherwise} \end{cases} \tag{4.56}$$

and a function f, which here is the circle

$$f(x,y) = \begin{cases} 1, \text{ if } x^2+y^2<1 \\ \quad 0 \text{ otherwise} \end{cases} \tag{4.57}$$

If the probability that the pebble is thrown from location a to location b is expressed by the formula $p(a \to b)$ and $\pi(a)$ stands for the stationary probability for the pebble to be at location a, location in this context meaning a certain area of the square, for example the upper left corner if the square is divided into 9 smaller squares (meaning that if a is inside the location a, $\pi(a) = 1$, otherwise $\pi(a) = 0$) and the probability for the pebble to be at each location is equal, the aforementioned detailed balance condition can be written in the following way:

$$\begin{aligned} \pi(a)p(a \to b) &= \pi(b)p(b \to a) \\ \pi(a)p(a \to c) &= \pi(c)p(c \to a) \ \text{ etc.} \end{aligned} \tag{4.58}$$

Source: [38], [39], [40].

4.4.4. *Metropolis algorithm.* One of the most well-known and commonly used Monte Carlo algorithms that utilizes Markov chains is called the Metropolis algorithm. It was developed in 1953 by the Rosenbluth and Teller families within a nuclear weapons program. It differs from the scenario described above where the probability for the pebbles to be at each location is considered equal in a way that it allows some locations to be more likely than others. The probability for a move of the pebble from location a to location b not to be rejected in Metropolis algorithm is

$$p(a \to b) = \min\left[1, \frac{\pi(b)}{\pi(a)}\right]. \tag{4.59}$$

In practice this means, that if you at first stand at a location a that we assume to be inside the square so that $\pi(a) = 1$ and then throw a pebble to a random direction so that it lands in location b, there are two things that can happen: either location b is inside the square, so that $\pi(b) = 1$ or it is not, in which case $\pi(b) = 0$. In the first case you will accept the move and walk to your pebble, in this situation the formula 4.59 tells you to accept the move with probability 1. In the second scenario, the formula tells you that you should reject the move as $\frac{\pi(b)}{\pi(a)} = 0$ and thus the probability with which you move from a to b is 0. When the values of $\pi(a)$ and $\pi(b)$ are not equal, one needs to consider the two possible situations $\pi(a) < \pi(b)$ and $\pi(a) > \pi(b)$ separately in terms of the acceptance probability but the situation is still very simple [39], [40].

The basic idea is to evaluate the properties of any substance that may be seen as being composed of interacting individual molecules by generating a Markov chain of successive states $s_1 \to s_2 \to \dots$. The new state is generated from the previous one using a transition probability $P(s_n \to s_{n+1})$ which is designed so that it occurs with a probability given by the *equilibrium Boltzmann distribution* which gives the probability $\mathcal{P}_k(s)$ with which state s occurs at the kth time step. The sequence of states or samples can then be used much as in the pebbles in the allegory before to calculate areas or integrals.

Metropolis-algorithm and MCMC-methods in general can be used to solve inverse problems in Bayesian or discrete settings. In Bayesian inversion a priori-information is added to the measurement data and for example the tomographic reconstruction is calculated based on both these elements. Discrete tomography in turn can often be considered from a Bayesian perspective, as it is often the case that neighbouring pixels can be assumed to be more likely to have the same values. If the noise and the prior distribution (i.e. the distribution that assigns probabilities to vectors based on how likely they are in the light of the a priori information) are assumed to be Gaussian, then the maximum a posteriori estimate and conditional mean estimate are the same and coincide with the Tikhonov-regularized solution with δ representing the level of noise [37], [38], [39], [40].

4.5. **Applications and development in the field of discrete tomography.** Discrete tomography is a relatively young field, so it is hardly surprising that it is still constantly developing. Existing algorithms are improved and new algorithms coined at a quick pace. In addition to development in algorithms, new applications to discrete tomography are found regularly. Many of the applications are of course medical, but it is not the only field where discrete tomography has proven its usefulness. In the following chapter some relatively new algorithms as well as some common applications are briefly listed without going into specifics.

4.5.1. *Medical imaging and cardiac angiography.* Using discrete tomography for medical purposes is only possible in special circumstances as human body is not a homogenous object and it cannot even be seen as consisting of only a few tissue densities. It is, however, possible to use different methods to enhance a certain region compared to the surrounding tissue. The method often used when a heart is studied is injecting contrast material in the blood vessels, making it possible to study and reconstruct a 3D-object consisting of the blood vessels that now have a certain specific attenuation value and everything else, which in this situation is considered background material. Something that needs to be taken into account while making reconstructions of blood vessels, or for that matter anything else, close to the heart is the added difficulty posed by the movement of the heart.

Angiography is the technique of medical imaging that is used to visualize the insides of blood vessels. In *cardiac angiography* or *cardiography* the focus is on aortas and the heart ventricles. The resulting images are either *cardiac angiograms*, *ventriculograms*, or *arteriograms* depending on which part of the heart contrast material was used to opacify. In [41] the authors state that a section through the heart can be seen as a binary image so that white is assigned to those parts or pixels that contain contrast material. They use *Gibbs priors* in order to have a better idea of the character of the image to limit the class of possible solutions, thus allowing them to reconstruct the binary image in question from only horizontal-, vertical- and one diagonal projection. In their take of a *Gibbs distribution*, the colour of a pixel influences the *local energy function* of only itself and its 8 neighbours. Local energy function in turn is defined in such a way that it encourages certain configurations such as clusters of black or white pixels. In this case the purpose of the Gibbs distribution is to point the solutions to a direction where if a pixel is of certain colour, it is likely that so are the pixels that are 8-adjacent to it. Using a modified Metropolis-algorithm on noiseless data, the reconstructions were nearly perfect. This is of course a situation that would never occur with real measurement data. Even with added noise, their results suggest that if an image is a typical member of a class of images with a certain Gibbs distribution, three projections are often enough to narrow down the possible solutions to only those that are close to the original image. [41], [42]

4.5.2. *Dynamic tomography.* Dynamic tomography, also known as 4D-tomography, is a field that is interested in developing algorithms that can be used for reconstructing usually three-dimensional objects that evolve continuously with time. It is something that we often come across in medical imaging, for example in tomographic reconstruction of heart, like was mentioned in the previous section, or lungs. In many non-medical applications the movement is not just something that makes the reconstruction of the structure that is the main object of interest more difficult. It can in fact be the main reason for the need of reconstructions. In [43], the example for a typical yet notoriously difficult problem in dynamic tomography is mapping the displacement of one immiscible fluid with another inside some porous material, a situation that is of practical interest in oil production. Most commonly used reconstruction techniques of general tomography assume that the sample is static. Movement of the sample will lead to blurred reconstructions. The necessary scan time is much shorter when making reconstructions using algorithms of discrete tomography because of the a priori information about the gray values present in the sample and the fewer projections that are needed because of that. This means that the sample doesn't have time to move as much, which leads to better reconstructions. This way a priori information that the sample consists of only a small number of gray values can be used to improve the reconstruction of the sample at any given time. The authors of [43] present an algorithm where a priori information that the sample consists of only two gray levels is used in order to successfully map two-phase fluid flow in porous media using existing x-ray micro-CT equipment. This is achieved by assuming that the porous structure (such as rock with capillaries) is static, the change from one moment to the next is localized to a small area and that the static structure steers the behaviour of the dynamic component. For more information about the algorithm and the relationship between discrete- and dynamic tomography, see [43].

4.5.3. *Belief-propagation reconstruction.* The correspondence between discrete tomography and Bayesian approach was already briefly mentioned in the chapter about Monte Carlo-methods. Often in discrete tomography the images being reconstructed are such that neighbouring pixels are more likely to have the same value. In [44], a fast and accurate belief-propagation-based algorithm is presented for situations like this. The algorithm estimates the marginal probability for each pixel value by relying on belief propagation (BP), which is a message-passing algorithm that even though it is not always exact, has given very accurate results for the problem at hand. With noise free data the algorithm produced a perfect reconstruction and with data that was corrupted with a small amount of Gaussian noise the results were perfect when the number of projections was increased. Even with a moderate amount of noise, the algorithm still outperformed for example ones based on convex relaxation of a binary tomography problem. The main problem of the algorithm is the lack of speed: the number of iterations needed is very high which makes the algorithm slow and thus unsuitable for purposes, where the results are needed fast. The authors still clearly feel that the algorithm has great potential and that its implementation can possibly be sped up making it in every aspect one of the best algorithms for discrete tomography in Bayesian settings [44].

The future promises great advances in the field of discrete tomography. Many promising algorithms are still under development and a continuously increasing number of researchers are working on new ones. There is also plenty of demand for new algorithms in many different fields. For example coloured 3D x-ray images are something that can be expected to appear for widespread medical use in not too distant future and algorithms of discrete tomography can be used to improve the process of acquiring them.

Appendix A.
matrixA.m

```
clear all
N=96; theta=-pi/2:pi/N:pi/2-pi/N; s=-1.1:2.2/(N-1)
    :1.1; d=2/N;
% part of matrix that corresponds to value
% theta=-pi/2.

for i=1:N,
    for j=1:N^2,
% Let's construct that
        j1=floor((j-1)/N); j2=j-1-N*j1; ss=s(i); b
            =1-(j2+1)*d;
        if b<= -ss && -ss < b+d, A(i,j)=d;
        else A(i,j)=0;
        end,
    end,

end,
% Then the rows corresponding to values 0<theta<-pi
    /2

for i=N+1:N^2/2,
    for j=1:N^2,
        i1=floor((i-1)/N); i2=i-1-N*i1; j1=floor((j
            -1)/N); j2=j-1-N*j1;
        Co=cos(theta(i1+1)); Si=sin(theta(i1+1));
            ss=s(i2+1);
        a=-1+j1*d; b=1-(j2+1)*d;
        k1=(ss-a*Co)/Si; k2=(ss-(a+d)*Co)/Si; k3=(
            ss-b*Si)/Co; k4=(ss-(b+d)*Si)/Co;
        if b<=k1 && k1<=b+d, t1=1; else t1=0; end
        if b<=k2 && k2<=b+d, t2=1; else t2=0; end
        if a<=k3 && k3<=a+d, t3=1; else t3=0; end
        if a<=k4 && k4<=a+d, t4=1; else t4=0; end

        if t1==1 && t2==1, A(i,j)=sqrt(d^2+(k2-k1)
            ^2);
        elseif t1==1 && t3==1, A(i,j)=sqrt((k3-a)
            ^2+(k1-b)^2);
```

```
          elseif t1==1 && t4==1, A(i,j)=sqrt((k4-a)
             ^2+(b+d-k1)^2);
          elseif t2==1 && t3==1, A(i,j)=sqrt((a+d-k3)
             ^2+(k2-b)^2);
          elseif t2==1 && t4==1, A(i,j)=sqrt((a+d-k4)
             ^2+(b+d-k2)^2);
          elseif t3==1 && t4==1, A(i,j)=sqrt((k4-k3)
             ^2+(d^2));
          else A(i,j)=0;
          end
     end
end

% Next the rows that correspond to value theta =0.

for i=N^2/2+1:N^2/2+N,
     for j=1:N^2;
    i1=floor((i-1)/N); i2=i-1-N*i1; j1=floor((j-1)/N
       ); j2=j-1-N*j1;
    a=-1+j1*d; ss=s(i2+1);
    if ss>=a && ss<a+d, A(i,j)=d; else A(i,j)=0; end
     end
end

% Finally the remaining rows corresponding to cases
     where 0<theta<pi/2

for i=N^2/2+N+1:N^2,
     for j=1:N^2,
          i1=floor((i-1)/N); i2=i-1-N*i1; j1=floor((j
             -1)/N); j2=j-1-N*j1;
          Co=cos(theta(i1+1)); Si=sin(theta(i1+1));
             ss=s(i2+1);
          a=-1+j1*d; b=1-(j2+1)*d;
          k1=(ss-a*Co)/Si; k2=(ss-(a+d)*Co)/Si; k3=(
             ss-b*Si)/Co; k4=(ss-(b+d)*Si)/Co;
          if b<=k1 && k1<=b+d, t1=1; else t1=0; end
          if b<=k2 && k2<=b+d, t2=1; else t2=0; end
          if a<=k3 && k3<=a+d, t3=1; else t3=0; end
          if a<=k4 && k4<=a+d, t4=1; else t4=0; end
```

```
        if t1==1 && t2==1, A(i,j)=sqrt(d^2+(k2-k1)
            ^2);
        elseif t1==1 && t3==1, A(i,j)=sqrt((k3-a)
            ^2+(k1-b)^2);
        elseif t1==1 && t4==1, A(i,j)=sqrt((k4-a)
            ^2+(b+d-k1)^2);
        elseif t2==1 && t3==1, A(i,j)=sqrt((a+d-k3)
            ^2+(k2-b)^2);
        elseif t2==1 && t4==1, A(i,j)=sqrt((a+d-k4)
            ^2+(b+d-k2)^2);
        elseif t3==1 && t4==1, A(i,j)=sqrt((k4-k3)
            ^2+(d^2));
        else A(i,j)=0;
        end
    end
end
```

Appendix B.
PAWN.M

```
% Give N of the form N=4*k
clear f
N=48; N1=2*N; k=N1/4;
for i=1:N1,
    for j=1:N1,
        if (i<=2*k && j<=2*k && (i-2*k)^2+(j-2*k)
            ^2>(1.8*k)^2) || ...
                (i>2*k && j>2*k && (i-2*k)^2+(j-2*k
                    )^2>(1.8*k)^2),
            ro(i,j)=1;
        elseif (i>2*k && j<=2*k && (i-2*k)^2+(j-2*k
            )^2>(1.8*k)^2) || ...
                (i<=2*k && j>2*k && (i-2*k)^2+(j-2*
                    k)^2>(1.8*k)^2) || ...
                (i-2*k)^2+(j-2*k)^2>=(1.8*k)^2 && (
                    i-2*k)^2+(j-2*k)^2<=(1.9*k)^2 ||
                      ...
                (i-k)^2+(j-2*k)^2<=(0.5*k)^2 || ...
                4*(i-3*k)^2+(j-2*k)^2<=k^2 && i<=3*
                    k || ...
                2*k-1/4*(i-k)<=j && j<=2*k+1/4*(i-k
                    ) && i>=k && i<=3*k,
            ro(i,j)=0;
        else ro(i,j)=0.5;
        end,
    end,
end
 figure(1), imagesc(ro), axis equal, axis off,
    colormap gray
 for i=1:N1,
     for j=1:N1,
            roo((j-1)*N1+i)=ro(N1+1-i,j);
     end,
 end
 k=N/4;

 for i=1:N,
    for j=1:N,
        if (i<=2*k && j<=2*k && (i-2*k)^2+(j-2*k)
            ^2>(1.8*k)^2) || ...
                  (i>2*k && j>2*k && (i-2*k)^2+(j-2*k
                     )^2>(1.8*k)^2),
             F(i,j)=1;
        elseif (i>2*k && j<=2*k && (i-2*k)^2+(j-2*k
            )^2>(1.8*k)^2) || ...
                  (i<=2*k && j>2*k && (i-2*k)^2+(j-2*
                     k)^2>(1.8*k)^2) || ...
                  (i-2*k)^2+(j-2*k)^2>=(1.8*k)^2 && (
                     i-2*k)^2+(j-2*k)^2<=(1.9*k)^2 ||
                      ...
                  (i-k)^2+(j-2*k)^2<=(0.5*k)^2 || ...
                  4*(i-3*k)^2+(j-2*k)^2<=k^2 && i<=3*
                     k || ...
                  2*k-1/4*(i-k)<=j && j<=2*k+1/4*(i-k
                     ) && i>=k && i<=3*k,
             F(i,j)=0;
        else F(i,j)=0.5;
        end,
    end,
end
 figure(1),
 subplot(1,2,1)
 imagesc(ro), axis equal, axis off, colormap gray
 title(['Target at size ', num2str(N1) ,'*',
    num2str(N1)])
 subplot(1,2,2)
 imagesc(F), axis equal, axis off, colormap gray
 title(['Target at size ', num2str(N) ,'*', num2str
    (N)])

 for i=1:N,
     for j=1:N,
           f((j-1)*N+i)=F(N+1-i,j);
     end,
 end
 f=f';
```

Appendix C. MEASUREMENTS.M

```
% We first construct the non-noisy measurements
    m_96 using matrix A96 and
% f_96. Then we sample it with app. 2% relative L2-
    norm error and finally
% we solve m_48 using interpolation. Load first A96
     and drive pawn.m
% with N=48.

N=48;
m96=A96*(roo)'; m96n=m96+0.01*max(abs(m96))*randn
    ((2*N)^2,1);

error1=sqrt(sum((m96-m96n).^2))/sqrt(sum(m96.^2));
for i=1:N,
     for j=1:N,
          m48n((i-1)*N+j)= m96n((2*i-2)*2*N+2*j-1)+
               ...
                (j-1)/(N-1)*(m96n((2*i-2)*2*N+2*j)-m96n
                    ((2*i-2)*2*N+2*j-1));
     end,
end
m48n=m48n';


N1=2*N;
for i=1:N1^2, k=ceil(i/N1); j=i-N1*(k-1);

   M96(k,N1+1-j)=m96(N1^2+1-i); M96N(k,N1+1-j)=m96n(
       N1^2+1-i);
end

for i=1:N^2,   k=ceil(i/N); j=i-N*(k-1);

   M48N(k,N+1-j)=m48n(N^2+1-i)  ;
end


  figure(2),
  subplot(2,2,1),
imagesc(M96'),
axis equal, axis off
title(['Noiseless meas. N=' num2str(N1)])
subplot(2,2,2),
imagesc(M96N'),
axis equal, axis off
colormap gray
title(['Noisy meas. with L2- er. ',num2str(round(
    error1*100)),'%'] )
subplot(2,2,3),
imagesc(M48N'),
axis equal, axis off
title(['interpolated measurements, N=', num2str(N)
    ])
```

Appendix D.
CONJ.M

```
% Load A48, pawn.m and measurements.m, both with N
    =48

N=48;
clear X g dd a1 a2 a3
a1=0.023;
Q=A48'*A48+a1*eye(N^2); b=A48'*m48n;
X(:,1)=0.5*ones(N^2,1);
g(:,1)=Q*X(:,1)-b; dd(:,1)=-g(:,1);
for k=1:49
     alfa=-g(:,k)'*dd(:,k)/(dd(:,k)'*Q*dd(:,k));
    X(:,k+1)=X(:,k)+alfa*dd(:,k);
    g(:,k+1)=Q*X(:,k+1)-b;
    beta=g(:,k+1)'*Q*dd(:,k)/(dd(:,k)'*Q*dd(:,k));
    dd(:,k+1)=-g(:,k+1)+beta*dd(:,k);
end,



% plotting the results

answer1=X(:,3);
answer2=X(:,50);
aid1=answer2<0.33; aid3=answer2>0.66; aid2=1-aid1-
    aid3;
answer3=aid3+0.5*aid2;

for i=1:N^2, k=ceil(i/N); j=i-N*(k-1); a1(k,N+1-j)=
    answer1(i);
  a2(k,N+1-j)=answer2(i);   a3(k,N+1-j)=answer3(i);

end

error2=sqrt(sum((f-answer2).^2))/sqrt(sum(f.^2));

 figure(3),
 subplot(2,2,1),
 imagesc(a1'),
 axis equal, axis off
```

```
title(['Result after two steps, al = ', num2str(al
   )] )
subplot(2,2,2),
imagesc(a2'),
axis equal, axis off
title(['49 steps. L2- er. ',num2str(round(error2
   *100)),'%'] )
subplot(2,2,3),
imagesc(a3'),
axis equal, axis off
colormap gray
title('Projections to 0, 0.5 and 1.')
```

References

[1] Jennifer L. Mueller, Samuli Siltanen: Linear and Nonlinear Inverse Problems with Practical Applications, Computational Science & Engineering, SIAM (2012)

[2] G.T. Herman, A. Kuba (Eds.), Discrete Tomography: Foundations, Algorithms and Applications, Birkhäuser, Boston (1999)

[3] H.W. Engl, M. Hanke, A. Neubauer, Regularization of Inverse Problems, Kluwer Academic Publishers, Dordrecht (1996) (Paperback edition 2000)

[4] A. Kirsch: Introduction to the Mathematical Theory of Inverse Problems, Springer-Verlag, New York, 1996.

[5] A. Wirgin: The Inverse Crime. arXiv:math-ph/0401050

[6] Jacques Hadamard. 1902. Sur les problèmes aux dérivées partielles et leur signification physique. Princeton University Bulletin. Vol. 13, p.49-52.

[7] H.W. Engl, C.W. Groetsch (eds.), Inverse and Ill-Posed Problems, Academic Press, Orlando, 1987.

[8] R. Gagliardi, B.L. McClennan, A History of the Radiological Sciences: Diagnosis, Radiology Centennial, Inc. (1996)

[9] E.D. Dyson, Shoe-fitting X-Ray Fluoroscopes Radiation Measurements and Hazards, British Medical Journal, Aug. 4 1956, p. 269

[10] C. Budd, C. Mitchell, Saving Lives: The Mathematics of Tomography, Plus Magazine May 31. 2008

[11] C. Høilund, The Radon Transform, Aalborg University, VGIS, 07gr721, November 12, 2007

[12] A. Meaney, Design and Construction of an X-ray Computed Tomography Imaging System. 2015, Master's thesis, University of Helsinki

[13] B.-I. Adi and T.N.E. Greville. Generalized Inverses: Theory and Applications. John Wiley and Sons, New York, 1974.

[14] William Green, Jr., course materials for 10.34 Numerical Methods Applied to Chemical Engineering, Fall 2006. MIT OpenCourseWare (http://ocw.mit.edu), Massachusetts Institute of Technology. Downloaded on 11 February 2016.

[15] P.C. Hansen. The truncatedSVD as a method for regularization, BIT Numerical Mathematics 27 (4), 534-553. 1987

[16] P.C. Hansen. Regularization Tools - A Matlab Package for Analysis and Solution of Discrete Ill-posed Problems. Numer. Algo. 46 (2007), pp. 189-194

[17] D. Calvetti, S. Morigi, L. Reichel, F. Sgallari, Tikhonov regularization and the L-curve for large discrete ill-posed problems, Journal of Computational and Applied Mathematics, 123, Issues 1-2, 1 November 2000, p 423-446, ISSN 0377-0427

[18] Flemming, Jens. Generalized Tikhonov regularization: basic theory and comprehensive results on convergence rates. 2011, Doctoral thesis, Chemintz University of Technology

[19] B.H. Fotland. A matrix-free method for regularization with unrestricted variables. 2008, Master's thesis, University of Bergen

[20] Noschese, Silvia, and Lothar Reichel. "Inverse problems for regularization matrices." Numerical Algorithms 60.4 (2012): 531-544.

[21] P.C. Hansen. The L-curve and its use in the numerical treatment of inverse problems. IMM, Department of Mathematical Modelling, Technical Universityof Denmark, 1999.

[22] M. R. Hestenes and E. Stiefel (1952). "Methods of conjugate gradients for solving linear systems," /. Res. N.B.S., Vol. 49, p. 409

[23] M.M. Mäkelä. Optimointialgoritmit, Turun yliopisto 2014. Lecture notes.

[24] J.R. Shewchuk. An Introduction to the Conjugate Gradient Method Without the Agonizing Pain Edition 1 1/4. 1994, Carnegie Mellon University. Lecture notes.

[25] H.J. Ryser, Combinatorial Properties of Matrices of Zeroes and Ones, Canad. J. Math. 9(1957), 371-377

[26] Y. R. Wang, "On data retrieval from unambiguous bit matrices", Int. J. Comput. Inform. Sci., 1975

[27] R.J. Gardner, P. Gritzmann Discrete tomography: determination of finite sets by X-rays Trans. Amer. Math. Soc., 349 (1997), pp. 2271-2295

[28] Herbert Edelsbrunner and Steven S. Skiena, Probing convex polygons with X-rays, SIAM J. Comput. 17 (1988), no. 5, 870-882. MR 961045 (89i:52002)

[29] S. Brunetti, A. Daurat. Reconstruction of discrete sets from two or more X-rays in any direction. Rémy Malgouyres. 2000, Université de Caen, pp.241-258. <hal-00023032>

[30] S. Brunetti and A. Daurat, "An algorithm for reconstructing lattice convex sets", Theoret. Comput. Sci., Vol. 304, pp. 35?57, 2003.

[31] S. Brunetti, A. Daurat, A. Del Lungo. Approximate X-rays reconstruction of special lattice sets. Pure Mathematics and Applications, 2000, 11, pp.409-425. <hal-00023034>.

[32] K. J Batenburg and J. Sijbers. Dart: A Fast Heuristic Algebraic Reconstruction Algorithm for Discrete Tomography, pages IV - 133-IV - 136. IEEE, 2007.

[33] K. J. Batenburg and J. Sijbers. DART: A practical reconstruction algorithm for discrete tomography. IEEE Transactions on Image Processing, 20(9):2542-2553, 2011.

[34] Dalen, B.E., van. Discrete tomography with two directions. 2011, Doctoral thesis, Leiden University

[35] F. Tabak. Robust Algorithms for Discrete Tomography. 2012, Master's thesis, Delft University of Technology

[36] Srivastava, Tanuja, Shiv Kumar Verma, and Divyesh Patel. "Reconstruction of binary images from two orthogonal projections." IJTS 21.2 (2012): 105-114.

[37] Motwani, Rajeev; Raghavan, Prabhakar (1995). Randomized Algorithms. New York: Cambridge University Press. ISBN 0-521-47465-5.

[38] Krauth, Werner, Introduction to Monte Carlo Algorithms. arXiv:cond-mat/9612186v2

[39] Krauth, Werner, Statistical Mechanics: Algorithms and Computations (Oxford University Press, 2006)

[40] Katzkraber, Helmut G, Introduction to Monte Carlo Methods. arXiv:0905.1629

[41] Bruno M. Carvalho , Gabor T. Herman , Samuel Matej , Claudia Salzberg , Eilat Vardi, Binary Tomography for Triplane Cardiography, Proceedings of the 16th International Conference on Information Processing in Medical Imaging, p.29-41, June 28-July 02, 1999

[42] Herman, G. T. and Kuba, A. (2003). Discrete tomography in medical imaging. Proceedings of the IEEE, 91(10), 1612-1626.

[43] Glenn R. Myers, Andrew M. Kingston, Trond K. Varslot, Michael L. Turner, and Adrian P. Sheppard, Dynamic tomography with a priori information, Appl. Opt. 50, 3685-3690 (2011)

[44] E. Gouillart, F. Krzakala, M. Mezard and L. Zdebrova 2013 Belief propagation reconstruction for discrete tomography Inverse Problems 29 035003 IOPscience