NTNU
Norwegian University of
Science and Technology

Exploring avenus for compressed sensing in X-ray tomography

Magnus Bro Kolstø

Master of Science in Physics and Mathematics
Submission date: June 2018
Supervisor: Ragnvald H. Mathiesen, IFY
Co-supervisor: Dag Werner Breiby, IFY

Norwegian University of Science and Technology
Department of Physics

Sammendrag

I denne oppgaven ble forskjellige måter å redusere mengden data som trengs for tomografiske rekonstruksjoner utforsket. L-BFGS-algoritmen ble testet med en SIRT back-projection som gradient. Dette resulterte i raskere og bedre konvergens enn SIRT og SART ved få projeksjoner. OWL-QN varianten av L-BFGS ble brukt for å implementere compressive sensing med utnytting av høy informasjonstetthet gitt av DCT, DWT, det reciprocale rom og i prøven selv. Problemer med dårlige søke-retninger og konvergens begrenset resultatet, men å bruke DCT og prøven selv ga lovende resultater. Bruk av stokastiske distribusjoner for å velge ut data fra projeksoner, med hensikt å redusere data-strømmen, ga lovende resultat, spesielt når compressive sensing med DCT ble brukt. En enkel analyse av å bruke nøkkelbilder på en data-strøm av projeksjoner virker lovende, men støy kan vise seg å være et problem.

Abstract

In this paper different ways to reduce the amount of data associated with tomographic scans were explored. The L-BFGS algorithm was tested using a SIRT back-projection as its gradient. It showed faster and better convergence than SIRT and SART for low numbers of projections. The OWL-QN variant was tested to implement compressive sensing using sparsity in DCT, DWT, a Fourier-grid and the sample. There were problems with bad search directions and slow convergence limiting the results, but using DCT and the sample itself showed promise. Using random sampling to reduce the size of projection data-streams also had promising results, especially when combined with compressive sensing using DCT. Simple analysis of the feasibility of keyframing the data-stream were positive, but noise could prove a problem.

Introduction

Computed tomography is one of the most common methods to accurately study the internal structure of objects in a non-invasive way. X-ray tomography was one of the first, and still one of the most popular, applications for these techniques. Since the first practical applications within medicine in the 70's it has also become a near and dear tool for scientists wishing to discover ever smaller nuances of the materials they are studying. Massive improvements in the scanning equipment has been achieved since then, making scans both faster and of higher resolution. At the biggest facilities, those using synchrotrons to get enough juice, the shear amount of data is starting to become a problem. This combined with the desire to reduce exposure and do ever faster scans means there is a great desire to reduce the data needed to do a decent reconstruction. In later years the increased computer power available has increased interest in more computationally heavy techniques, which until now have been in the shadow of less demanding, but also less accurate, algorithms. In this regard compressive sensing is a relevant idea. The idea is exploiting the inherent compressibility of the information describing a sample, and from this be able to reduce the data needed for a reconstruction.

In this paper several techniques will be explored to attempt to make good reconstructions from smaller amounts of data. To do this the OWL-QN variation of the L-BFGS algorithm will be used to perform compressive sensing. Using the sparsity of the sample itself, as well as the common compression domains of the discrete cosine transform and the discrete wavelet transform, will be explored. In addition to this the possibility of using a representation of the sample in reciprocal space as the reference point for the compressive sensing will be tried.

Some techniques for reducing the amount of data with minimal loss of information will also be explored. Removing data from projections according to a couple of distributions of random points will be compared to removing the corresponding amount of information as whole projections. Using compressive sensing on these random points, in an attempt to exploit the randomness, is also going to be attempted. A simple analysis of using the principles of keyframing will be done. That is to only store the whole projections at given points and otherwise only register the difference to from the previous projection.

Contents

1 Introduction to X-ray tomography

Tomography is a technique to make a 3-dimensional image of an objects using projections of the object onto 2-dimensional images. These projections correspond to line-integrals taken over the object following a given projection-geometry. To be able to make this 3-dimensional image projections from different angles are needed. Generally more projections give a higher spatial resolution. The projections are commonly made with a beam that is only partially attenuated by the object. To satisfy the condition that the projections are collections of line-integrals the beam should have a known and well defined attenuation for the sample of interest. In this paper X-ray beams are used. The data from the projections are plugged into one of several different reconstruction algorithms to produce the desired 3-dimensional image of the object. In some cases only a 2-dimensional slice of the object is desired. This only necessitates a 1-dimensional projection. Since a lot of the concepts are the same for this simplified case, it is used to illustrate the general concepts where applicable.

1.1 X-ray imaging

What makes X-ray tomography possible is the relationship between the attenuation coefficient and the transmitted intensity through an object. The transmitted intensity is given by

$$I_T = I_0 e^{\int \mu(x)\,\mathrm{d}x}, \tag{1}$$

where I_0 is the incoming intensity and $\mu(x)$ is a function describing the attenuation along the ray. I_T is the intensity measured at the detector.

The attenuation coefficient can be described as $\mu = \mu_s + \mu_a$, where μ_s is the scattering coefficient and μ_a is the absorption coefficient. In conventional tomography μ_s and μ_a are not individually discernible, but the scattered rays can cause noise in the measurements.

μ is not only dependent on the material, it is also dependent on the frequency of the beam going through the material. Since the X-rays used in tomography are not a single frequency, but a combination of bremsstrahlung[1] and specific frequencies for the metal in the source. Since different frequencies have different attenuation the frequency composition of the beam changes as it passes through

the material. The more attenuated frequencies disappear faster and the net effect is a reduced attenuation as the beam passes through the object. This effect is called beam hardening.

1.2 Noise in X-ray imaging

When taking X-ray images there will always be an element of noise in the resulting images. This noise stems from several different sources. Some of the noise is inherent in the physics, and will be there even under ideal circumstances. Other sources of noise might be due to imperfections in the measuring equipment or environment that one are not able to remove.

To analyse the noise it is often useful to use the signal to noise ratio (SNR). In image analysis it is common to use

$$\mathrm{SNR} = \frac{\mu_{\mathrm{sig}}}{\sigma_{\mathrm{noise}}},$$

where μ_{sig} is the average of the signal and σ_{noise} is the standard deviation of the noise (often measured in the background).

1.2.1 Shot noise

The shot noise is a consequence of the discreteness of the photons and the way they are detected. This effectively creates a counting noise, a Poisson noise[2]. The Poisson distribution approximates the normal distribution[3] for high numbers of photons, which is usual for X-ray tomography, and the standard deviation is then given by $\sqrt{I}$, where I is the expected intensity. This further gives

$$\mathrm{SNR} = \frac{I}{\sqrt{I}} = \sqrt{I}.$$

From this we can see that the shot noise is quite small, and usually dominated by other sources of noise. It can however be noticeable, especially with low intensity and high electronic amplification and other sources of noise are low.

1.2.2 Thermal noise

Thermal noise is just a constant background noise over the detector. As the names suggests a large portion of this is from signal generated in the detector due to it having a temperature above absolute zero, and thus having movement. Another source of this is background radiation hitting the detector. Although this source of noise doesn't need to be the same over the whole detector, with proper warm up procedures it can be assumed to be constant over time and independent of the intensity. As this source of noise is constant it can be minimized efficiently by taking high intensity images. The thermal noise contributes to limiting how little exposure you can get away with when taking X-ray images.

1.2.3 Scattering noise

Some of the attenuation that happens to the X-rays is from scattering. A part of this scattering will reach the detector, causing noise in the image. This scattering noise is complicated since it might have been scattered several time and the amount of the attenuation that causes scattering is not directly measurable in the scan itself. Luckily this scattering-effect is usually negligible in real systems.

1.2.4 Cross-talk

Cross-talk is the effect of each pixel not being perfectly isolated from each other, and some of the signal from one pixel leaking over to the neighbouring pixels. This can be looked upon as similar to the effects of a larger spot size, but more noisy and discrete in nature. This can cause problems when trying to reduce the amount of data.

1.3 Projection geometries

There are several different ways to make a projection of an object. The geometries of interest are the ones that can easily be produced in a measuring apparatus. In conventional tomography only the unattenuated beam is measured, which by its nature consists of straight rays from the source. The source of the beams has to have some size and shape in space. Since we want each

point on the projection to represent a single line through space, the rays of the beam would either have to be parallel or the source so small that it can be approximated as the rays originating in a point. Using these criteria three common projection geometries appear. The parallel-beam geometry, the fan-beam-geometry and the cone-beam geometry.

To simplify both measurements and the algorithms used for reconstruction, the projections are usually taken around a single axis of rotation.

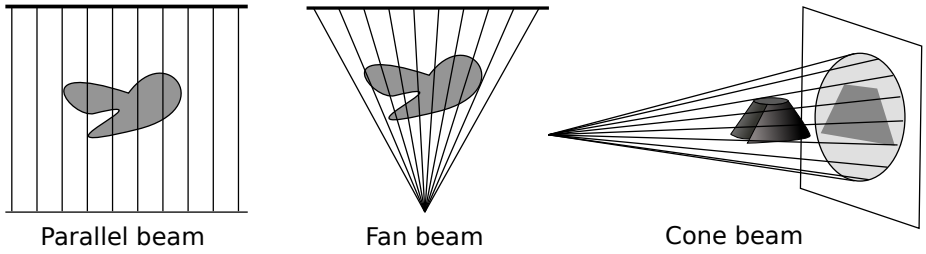


Figure 1: Illustration of three common types of projection geometries, the parallel beam, fan-beam and cone-beam geometry

The Parallel-beam geometry consists, as the name implies, of projections using purely parallel rays. This means that the (virtual) source of each ray is shifted the same distance parallel to the projection plane as its corresponding coordinate in the projection. This is the easiest type of projection to work with, but it comes with a cost. Big parallel sources are expensive and difficult to manufacture, so often small non-parallel sources, scanning parts of a projection at a time, are used. This leads to increased imaging time and often increased radiation exposure to the sample.

The fan-beam geometry uses a line-source. This means the rays are only parallel in one direction and diverging in the other. This makes the wave-front take an approximately cylindrical form. Usually the line-source is placed parallel with the axes of rotation, this allows the measured data to easily be sliced into different heights and allows for simpler reconstruction. As for the parallel-beam the fan-beam source is sometimes a virtual source made from a moving point source. The scanning in this case is however a lot quicker and simpler to implement as it only needs to move along one dimension. The disadvantage of the fan beam is an increased sensitivity to systematic error (the length from the source to the detector and the pixel size are important for reconstruction and

can introduce error) and a more complicated reconstruction.

The cone-beam geometry is a natural continuation of the fan beam. The cone-beam uses a point source, so there is no parallel requirement. This greatly speeds up and simplifies the imaging equipment, but complicates the reconstruction significantly. With cone beams projections are sometimes taken around two different axis of rotation to make more accurate reconstructions possible.

All of these geometries are of course ideal simplifications to be able to work with the measurements. The divergence of the assumed parallel beams or the real size of a focal spot defines an optical resolution that usually takes effect long before other optical limitations come into play. To avoid the effect of this experiments are usually set up so the resolution of the detector is lower than this optical resolution.

1.4 Rebinning fan-beam projections into parallel-beam projections

Sometimes there arises a need to transform projections made with a fan-beam geometry into a series of projections with a parallel beam geometry. For the analytical case, with infinitesimal angles between the projections and infinitesimal pixel size, this is a loss-less process. In real cases, made discrete by pixels and a limited amount of projections, some interpolation is required. The larger the angles between projections and the size of the pixels, the larger the gaps are in the interpolation and thus a greater error is acquired.

The method consists of shifting each pixel a certain angle given by the geometry and it's distance from the centre. The pixel also get shifted towards the center-ray (the fan-beam ray normal on the projection plane) of the projection. From figure 2 the following equations for this shift can be deduced

$$\alpha = \tan^{-1}\left(\frac{np_{\text{size}}}{s}\right),$$

$$\theta_p = \theta_f + \alpha = \theta_f + \tan^{-1}\left(\frac{np_{\text{size}}}{s}\right), \tag{2}$$

$$u_p = u_f \cos\alpha = u_f \frac{s}{\sqrt{s^2 + (np_{\text{size}})^2}} \tag{3}$$

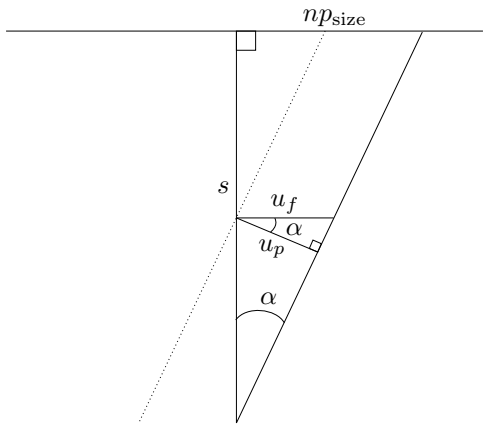


Figure 2: Illustration of the geometry used in rebinning.

where s is the distance from the source to the detector, np_{size} is the distance from the centre of the projection given in number of pixels n and the size of each pixel p_{size}. α is the angle between the given ray and the normal of the detector. u_f is the distance to the center for the fan-beam and u_p is the distance to the center for the parallel beam equivalent

1.4.1 Magnification effect

An effect from fan and cone geometry that is not properly accounted for in rebinning is the magnification effect. This is caused by the pixel field of view changing over the object. So a feature appears bigger from certain angles than others. Rebinning accounts for size adjustment from magnification, but not for the intensity effect.

1.5 Continuous scanning

When scanning an object there has to be a decision on which angles to scan the object from. For parallel beam projections only 180 degrees of rotation is strictly needed, since the projection from the exact opposite side would be a mirror image. Thus spreading over a larger angle adds no other information

than potential error reduction. For the fan-beam we can, by reversing the fan to parallel rebinning in the previous section, deduce that a section of 180 degrees plus the total angle of the fan will give nearly complete information about the object. It is not entirely perfect, as the rebinning is only perfect for the imagined continuous case, but it gives an idea about the limits. For the cone-beam the full 360 degrees of rotation is needed to maximise the information since the rays diverge in all directions.

In many cases the speed of the scan has a high priority. This is usually either to reduce the exposure of the sample or to capture samples that are changing over time. In addition to reducing the number of projections this goal is accomplished with continues scanning, i.e. the sample is continuously rotated during exposure and the projections taken continuously. This makes each projection represent the sum of an angle section $\Delta\theta$ instead of a single angle.

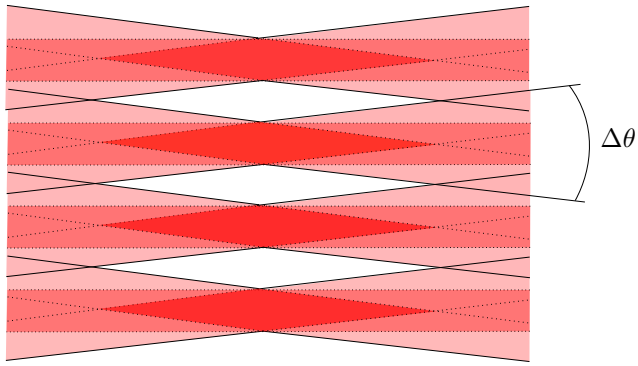


Figure 3: Illustration of what happens when you take a continuous sample over a certain angle $\Delta\theta$. The result gives a core that is always in view (the dark red and white diamonds) and the rest of the volume is smeared out more and more with increased distance from the center line. The medium red illustrates the area for a stationary projection

f(x)

Using continuous scanning comes with some disadvantages. As can be seen in figure 3 the sampling volume for each pixel bleeds into its neighbours. Only along a line through the center of rotation is the volume unchanged, and only a relatively small diamond shape centred on this line is always viewed by the same pixel. Since the neighbouring pixels make up for the lost intensity the

net effect is that the contribution of sample features removed from this line get smeared out. Since this gets repeated all around the sample the effect on the reconstruction is a greater and greater loss of detail as you move away from the center. This loss will logically be worst in the tangential direction and smallest in the radial direction.

From this it is evident that for a given number of projections, having them spread out across a smaller angle is desirable. So a 180° parallel projection series is not only equal but, from this perspective, better than a 360° series. In some cases it may still be desirable to take full 360° series, like when taking several series continuously without intermediate breaks, but since the only difference between the two while scanning is the rotational speed, there are few reasons to not prefer 180°. For non-parallel projection geometries a 180° series is not an option, and thus the effect of continuous scanning would be more noticeable.

2 Foundations of tomographic reconstruction

In the heart of tomographic reconstruction lies the radon transform, or rather the inverse of the radon transform, as will be shown in this section. Different methods for making this inversion, and filling in the gaps of "missing" information, is what defines a reconstruction algorithm.

2.1 Radon transform

The 2D radon transform is defined as the the line-integrals over a sample plane at different angles θ and distances

$$u = y\sin\theta + x\cos\theta \tag{4}$$

from the centre. Mathematically expressed as

$$R(u,\theta) = \iint_{-\infty}^{\infty} f(x,y)\cdot\delta(y\sin\theta + x\cos\theta - u)\mathrm{d}x\mathrm{d}y \tag{5}$$

where $R(u,\theta)$ is the result of the radon transform, $f(x,y)$ is the object-function and $\delta()$ is the dirac-delta function[4].

The image resulting from the 2D radon transform, with axis u and θ, is called a sinogram. This name comes from the observation that features in the original image-function take the form of sinusoids in $R(u, \theta)$.

The radon transform can be generalized to three dimensions. Where the generalized radon-transform turns the line-integral into a hypersurface integral, always keeping the resulting sinogram a two dimensional image. Further generalizing, the transform can be made by letting the integral go over a affine subspace with one dimension. This keeps the integral performed as a line-integral, which is the most relevant form for tomography. This configuration makes the sinogram three-dimensional and therefore allows for a complete inversion.

The sinograms from the radon transform appears to conform to the idea of tomographic imaging put forward in section 1, where each projection equals the sinogram at their set θ. This means that finding the object-function from a set of tomographic projections is just a matter of finding the inverse of a radon transform.

2.2 Fourier slice theorem

The Fourier slice theorem in two dimensions states that taking the two-dimensional Fourier transform of an object-function, and then taking a slice of this function through the origin at a given angle, is equivalent to projecting the object function onto a line at the same angle, and then taking the 1D Fourier transform of this line.

If we define F_N as the mathematical operator for the N-dimensional Fourier-transform, P_N as the operator making a N-dimensional projection and S_N as the central slicing operator down to N-dimensions, the theorem can be expressed as

$$P_1 F_1 = F_2 S_1. \tag{6}$$

This can be shown quite easily mathematically. The projection can be described by equation (5). This equation can be rewritten to the rotated coordinate system using $u = y \sin\theta + x \cos\theta$, $v = y \cos\theta - x \sin\theta$, which gives

$$R(u,\theta) = \int_{-\infty}^{\infty} f(u,v)\,\mathrm{d}v. \tag{7}$$

Taking the Fourier transform of this results in

$$P(k_u,\theta) = \int_{-\infty}^{\infty} R(u,\theta)e^{-2\pi u k_u}\,\mathrm{d}u, \tag{8}$$

where $P(k_u,\theta)$ is the one-dimensional Fourier transform of the projection with angle θ.

The 2D Fourier transform F of the object-function in the rotated coordinate system (u,v) is defined as

$$F(k_u,k_v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(u,v)e^{-2\pi(uk_u+vk_v)}\,\mathrm{d}u\,\mathrm{d}v, \tag{9}$$

and taking the center slice by setting $k_v = 0$ we get

$$F(k_u,0) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(u,v)e^{-2\pi u k_u}\,\mathrm{d}u\,\mathrm{d}v \tag{10}$$

$$= \int_{-\infty}^{\infty}\left(\int_{-\infty}^{\infty} f(u,v)\,\mathrm{d}v\right)e^{-2\pi u k_u}\,\mathrm{d}u \tag{11}$$

which, using (7), is equal to (8).

This proof can be generalized to higher dimensions. The theorem can then be written with the operators previously defined in (6) as

$$P_m F_m = F_N S_m, \tag{12}$$

where N is the number of dimensions in the object function, and m is some lower dimension.

3 Compressive sensing

The main idea in compressive sensing is to exploit the fact that the information contained in most objects can be described in simple terms. For objects made

from homogeneous materials you only need to describe the edges between the homogeneous parts, for uniform objects in a grid you only need to describe one object and the shape of the grid. Compressive sensing is a technique to exploit such simplifications.

The prerequisite for compressive sensing is that you have some kind of domain that describes your object sparsely. Being described sparsely means that a large portion of the information describing the object are zeros, i.e. not directly contributing to the description of the object. This is called increasing the energy density, a very important concept in data compression. For such a sparse domain to be usable you need to be able to do a transformation from this domain to the domain that contains your data.

Having such a known sparse domain, compressive sensing is about finding the least amount of points in this domain that when transformed satisfy the sample points. In other words you try to reduce the L_0-norm, the amount of non-zero values, in this domain. Doing is this is called applying a L_0-regulation in that domain to the problem. The L_0-norm is however hard to use in calculations due to its discontinuity, instead the L_1-norm is normally used.

$$L_1 = \|\vec{x}\|_1 = \sum_{i=1}^{n} |x_i|$$

The L_1-norm has been shown to be a good approximation of the L_0 norm in practical application. L_1-regulation is therefore the norm within compressive sensing.

It is apparent that the sparse domain used can not be a one to one transformation from the domain of the sample points, e.g. the same domain, even if this domain also happens to be sparse. The norm minimization would then simply result in the sample points themselves.

To process of implementing compressive sensing is two-fold. First you need to find a suitable domain that is sparse for the samples of interest. Second you need an efficient and robust algorithm to minimize the norm.

3.1 Basis pursuit denoising

The problem which we try to solve in compressive sensing is called the basis pursuit, and is defined as

$$\min_x \|x\|_1 \quad \text{subject to} \quad y = Ax$$

And this works quite well when we know there exists exact solutions to the problem, and we desire the simplest of these. But when there are no exact solutions, like when noise is added or you have an imperfect model of the projection geometry (basically when dealing with real systems), this model is not the best choice. For these situation we want to be able to adjust the focus on fidelity, how closely the solution fits, and the sparsity of the solution. This problem is defined as

$$\min_x \frac{1}{2}\|y - Ax\|_2^2 + \lambda\|x\|_1 \tag{13}$$

where λ is a variable controlling the focus on sparsity versus fidelity. The exact value of λ depends on the nature of the problem. It should be large enough to make the sparsity term in the same order of magnitude as the fidelity term at the solution. So if your system has a low fidelity, i.e the fidelity term will always remain somewhat large, you need to compensate with a large λ if you want to make a sparse solution.

3.2 General transformations

3.2.1 Minimizing in image-space

In some cases the object itself is so sparse that doing the minimization in the image-space itself can lead to a significant gain. Since this can lead to quite significant value-based artefacts in the image it is more suited for qualitative, rather than quantitative, tomography. Especially discrete tomography, where only the segmentation is of interest, can take advantage of this kind of compression.

3.2.2 Discrete Fourier transform

The Fourier-transform is tightly integrated into tomography through the Fourier-slice theorem. This separates it from the other transformations, as the reference

values for the minimization can be transferred to the transformation. These values can be used directly with some sort of interpolation algorithm, which could again use compressive sensing to improve the interpolation results. This would use some other kind of transform to actually do the L_1 minimization in, as doing it in the reference space will not work.

$$X_k = \sum_{n=0}^{N-1} x_n[\cos(2\pi kn/N) - i \cdot \sin(2\pi kn/N)]$$

The fourier slice theorem also tells us about a limitation of using the discrete Fourier transform. Since each projection translates to a hyperspace in reciprocal space, the points outside of these do not really contribute to the projection result. So even if the DFT increases the energy density of the reconstruction, it does this in a space which is already limited by the projections.

3.2.3 Discrete cosine transform

The discrete cosine transform (DCT) is a Fourier related transform that uses only cosine functions of different frequencies to describe the data points. It is quite popular for compression as it gives a high energy density for lots of common data sets. The transform has eight different variants depending on how the boundary conditions and symmetry points are defined, but the most commonly used is the type-II DCT described by

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n + \frac{1}{2}\right)k\right] \quad k = 0, \ldots, N-1$$

where $X_0, \ldots, X_{N-1}$ are the real numbers resulting from transforming the N real numbers $x_0, \ldots x_{N-1}$. Making the transform matrix orthogonal is achieved by multiplying X_0 by $1/\sqrt{(2)}$ and the whole matrix by $\sqrt{\frac{2}{N}}$. Given this the resulting inverse transform is given by

$$x_k = \frac{1}{\sqrt{2}}X_0 + \sqrt{\frac{2}{N}}\sum_{n=0}^{N-1} X_n \cos\left[\frac{\pi}{N}\left(k + \frac{1}{2}\right)n\right] \quad k = 0, \ldots, N-1$$

3.2.4 Discrete wavelet transform

The discrete wavelet transform(DWT) is a popular transform in data compression due to it's high energy density. Applying the DWT to an image results in four new images with a fourth the amount of pixels each. The first image, called the approximation, appears as a downscaled version of the original image. The other images, named the details, contains the local change around pixels, the edginess, in the up-down, left-right and diagonal direction.

These four images are made by first dividing the image into 2×2 blocks. Each block $/(A_2)$ is then transformed using a 2×2 wavelet matrix H_2,

$$B_2 = H_2 A_2 H_2^T.$$

In this paper the Haar wavelet is used, which has the 2×2 matrix

$$H_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

This results in

$$B_2 = \begin{bmatrix} a & d_1 \\ d_2 & d_3 \end{bmatrix}.$$

Disregarding the normalization constants, a is the sum of A_2, d_1 is the left side of A_2 subtracted by the right side, d_2 is the top of subtracted by the bottom and d_3 is the diagonal from the top left subtracted by the other diagonal. This means a can represent the average, d_1 edge from left to right, d_2 edges from top to bottom and d_3 sort of the edges along the diagonal from the bottom left and up.

The inverse transform is simply given by

$$A_2 = H_2^T B_2 H_2.$$

The DWT can also be applied in several levels by simply applying the transform again to the approximation matrix from the previous transform. The transform can now be described as a level n DWT, where n is the number of times the transform has been applied.

3.2.5 Combining transformations

In some cases the object might be a bit sparse in several transformations, and a way to exploit more than one transformation at the time would be useful. One way to do this would be to let the result of one minimization be the starting point for the other, thus implementing them in sequence. One could also iterate between these two transforms at different rates according to priority. A potential problem with this approach is that defects amplified in one of the minimizations are not necessarily discouraged by the other. With highly under-determined systems, where these algorithms are most relevant, there are a lot of freedom for defects. Since these defects were created in another minimization attempt removing them is likely to show little measurable improvement.

Taking this into account it might be better to let the minimization in both domains run on a clean slate and combine the results in a weighted average. This ensures that domain-specific artefacts average out instead of adding to each other. It also gives a lot of freedom to weight different domains according to their strengths and weaknesses.

Some transformations give natural opportunities to combine with other transforms. The DFT gives the opportunity to have the reference points in the Fourier domain for then to transform into another domain to do the L_1 minimization. Similarly the DWT offers the opportunity to apply another transform to the approximation matrix created, compacting this less energy dense part of the transformation.

3.3 Interslice information

When doing a reconstruction in two dimensions all the voxels are more or less connected. They are connected in the sense that changing one voxel affects how the others are calculated (this connectedness gets weaker with fewer projections though). For a three dimensional reconstruction the voxels are not connected in the same way. For the parallel case they are separated into independent slices. For the cone-beam there are not clear slices in the same way, but the pixels at different heights in a projection are still disconnected. When using a sufficient amount of projections this non-connectedness does not matter, as there is not a lot of ambiguity within the slices themselves, but when the number

of projections are reduced the slices get less defined, and using the information between the frames becomes more important.

In a real sample there is always some kind of implied connection between the slices. This connectedness can be exploited to create better reconstructions. For compressive sensing this comes quite natural when you use three-dimensional transformations, as it implies some kind of systematic sparsity in this third domain.

4 Analytical reconstruction

4.1 Filtered back projection (FBP)

The filtered back projection is one of the simplest algorithms for inverting the radon transforms. For the analytical case it gives a perfect reconstruction, but for under sampled discrete data sets it offers little flexibility. Its rigidity also extends to the projection geometry used, as it is hard to adapt to new geometries. To give a high quality reconstruction it is also dependent on the projections being evenly spaced, although this is rarely a problem.

The method uses the Fourier slice theorem to perform the inverse radon-transform. First the Fourier transforms of the projections are taken. According to equation (12) these are equal to slices in the reciprocal space of the desired object, so they get placed there accordingly. Now the inverse Fourier transform is taken to produce the object function. In the practical case some interpolation is needed to fit the slice-values onto the discrete grid determined by the resolution.

The inverse Fourier transform in two dimensions is defined as

$$f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(k_x,k_y)e^{i2\pi(xk_x+yk_y)}\mathrm{d}k_x\mathrm{d}k_y \tag{14}$$

Rewriting (14) to polar coordinates using $k_x = k\cos\theta$, $k_y = k\sin\theta$, and $\mathrm{d}k_x\,\mathrm{d}k_y = k\,\mathrm{d}k\,\mathrm{d}\theta$ we get

$$f(x,y) = \int_{0}^{2\pi}\int_{-\infty}^{\infty} F(k,\theta)e^{i2\pi k(x\cos\theta+y\sin\theta)}k\,\mathrm{d}k\,\mathrm{d}\theta.$$

Using the definition of u in (4) and the Fourier slice theorem through (8) we get

$$f(x,y) = \int_0^{2\pi} \int_{-\infty}^{\infty} P(k,\theta)\, k\, e^{i2\pi uk} \mathrm{d}k\, \mathrm{d}\theta.$$

Finally using the symmetry property $P(k,\theta) = P(-k,\theta+\pi)$ yields

$$f(x,y) = \int_0^{\pi} \left[\int_{-\infty}^{\infty} P(k,\theta)\, |k|\, e^{i2\pi uk} \mathrm{d}k \right] \mathrm{d}\theta. \tag{15}$$

The factor $|k|$ in the inverse Fourier transform is known as the filtering kernel. It arises from the circular geometry in created in Fourier space. This effectively decreases the intensity linearly when moving away from the center, i.e. for higher frequencies. In this simple form $|k|$ is a ramp or Ram-Lak filter with cut-off frequency at the Nyquist-frequency (see (16)). Without it the resulting reconstruction would look low contrast and faded.

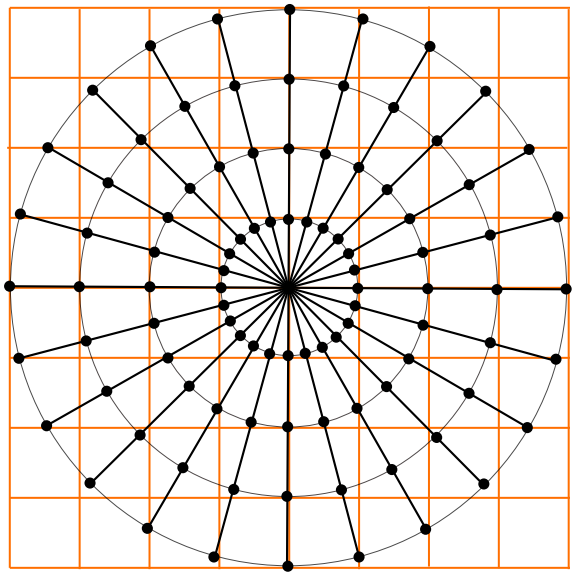


Figure 4: Illustration on how the Fourier transform of the projections (in black) are placed in relation to a square reciprocal grid. The decrease in data density with increasing frequency (radius) is apparent.

There is, however, a problem with this simple filter. Since it is a high-pass filter, all the noise in the high frequencies get amplified. Further worsening this effect is an increased spacing between projection-slices in reciprocal space at higher frequencies. This adds increased interpolation error to the amplified areas. To compensate for this it is normal to combine the ramp filter with a low-pass filter of some sort, reducing the noise in the image.

4.2 Sampling with analytical reconstruction

For a discrete dataset the highest frequency that is possible to withdraw from the data is given by the Nyquist-frequency

$$f_N = \frac{1}{2}\frac{1}{\Delta x}, \tag{16}$$

where Δx is the distance between data points. In the case of tomography

$$\Delta x = \frac{\Delta x_D}{M},$$

where Δx_D is the detector pixel size and M is a magnification factor determined by the projection geometry. For parallel geometry $M = 1$, and for fan-beam and cone-beam M is determined by the relationship between the source-to-detector distance and the source-to-object distance.

The radial frequency resolution in reciprocal space is then given by

$$\varepsilon = \frac{2f_N}{N_D} = \frac{1}{\Delta x N_D}, \tag{17}$$

where N_D is the number of pixels in the detector (normal to the rotation axis).

In the reciprocal space the angular distance between projections is given by

$$\Delta\theta = \frac{\pi}{N_p},$$

where N_p is the number of projections. Using this we can find the angular resolution for a desired frequency f

$$\Delta f = f\Delta\theta = \frac{1}{2\Delta x}\frac{\pi}{N_p} \tag{18}$$

The limit for a good reconstruction is usually set so that the worst angular resolution is about the same as the the radial resolution[5]. Using (17) and (18) this can be expressed as

$$\begin{aligned} \Delta f &\approx \varepsilon \\ \frac{1}{\Delta x N_D} &\approx \frac{1}{2\Delta x}\frac{\pi}{N_p} \\ N_p &\approx \frac{\pi}{2} N_D \end{aligned}$$

The system is thus oversampled in the angular domain, and its resolution limited by N_D, when

$$N_p > \frac{\pi}{2} N_D \tag{19}$$

5 Iterative reconstruction

5.1 Algebraic reconstruction technique (ART)

The continuous radon transform given in (5) is not always the most practical way of representing the relationship between measured data and the object sampled. In real applications the images taken are made out of discrete pixels. In consequence, the object-function is thought of as a collection of voxels, the three-dimensional equivalent to pixels.

The pixel values for a projection can then be viewed as the sum of the values of the voxels along the ray(s) illuminating that pixel, weighted according to how much each voxel contributes to the pixel. For a volume of N voxels a pixel p is described by

$$w_1 v_1 + w_2 v_2 + ... + w_N v_N = p$$

where w_i is the weight corresponding to the j^{th} voxel v_j. Extending this to all M pixels from all the projections we obtain a set of linear equations expressed in matrix notation as

$$\underbrace{W}_{M\times N} \times \underbrace{V}_{N\times 1} = \underbrace{P}_{M\times 1} \tag{20}$$

where W is a $M \times N$ matrix with the weights of each voxel to each pixel, V is a $N \times 1$ matrix with the values of each voxel and P is a $M \times 1$ matrix with the pixel values. In relation to the continuous radon transform V is the object function, P all the projections collected and W is the transformation operator for a given projection geometry.

With this representation it is easy to change W to represent different projection geometries or to compensate for known abnormalities in the ray-paths, etc.

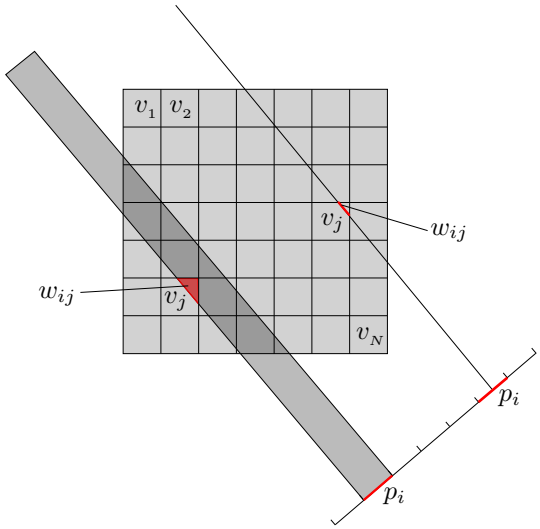


Figure 5: Illustration of two common methods to calculate the weight w_{ij} of a voxel v_j on a pixel p_i.

There are different ways to calculate the weights. It is a balance between having increased accuracy or fewer number of non-zero weights leading to increased calculation speed. The simplest method is to use a one dimensional ray, and calculate the length of the ray in each voxel. A slightly more convoluted method is to use a cuboid representing the volume affecting the pixel, and then set the weight of a voxel as the volume of the intersection between this cuboid and the voxel. These methods are illustrated in figure 5.

Since the number of pixels involved (M) is usually less than the number of voxels involved (N), making the problem under-determined, and since both numbers usually are quite large, using a matrix inversion algorithm to solve this

problem is infeasible. What is more feasible is to look upon this problem as a minimization problem. By starting with a guessed or empty volume V^0 and using this with W to make a set of virtual projections P_V, an error function ϵ between the real projections and the virtual projections can be made. Using a gradient of this error function, V can be iterated towards one of its local minima, which is hopefully close to the global minimum.

This process can be sped up by using subsets S of P and calculating the gradient for one subset at a time. This reduces the computational time and strain for each update, making for a faster iteration. There are however several downsides. For small subsets the gradient gets unprecise. This leads to a a more erratic iteration due to the correction overshooting. The minimum reached will also be less stable, as the minimum for one subset does not necessarily equal that of another, and you end up oscillating around the minimum of the complete data set. This is especially true when noise and other defects are present in the data, but can also happen for clean data where there exists several local minima. Choosing subsets that are large enough to give a good representation of the all the data, but still significantly smaller than the complete data set, is a way to get good reconstructions with less computing necessary.

Algorithm 5.1: Generic iterative reconstruction

```
1  V ← V^0
2  Split P into subsets S
3  while ⟨ε⟩ > threshold:
4          foreach S do:
5                  foreach p ∈ S do:
6                          compute virtual projection
7                          compute ε
8                  compute gradient of ε
9                  update V from gradient
```

More specifically the algorithm for updating a voxel in ART can be described by

$$v_j^{k+1} = v_j^k + \lambda \frac{\sum_{\{i|p_i \in S\}} \frac{\left(p_i - \sum_{n=1}^{N} w_{in} v_n^k\right) w_{ij}}{\sum_{n=1}^{N} w_{in}}}{\sum_{\{i|p_i \in S\}} w_{ij}} \tag{21}$$

Here v_j^k represents the value for the j_{th} voxel after k iterations. $\lambda \in (0, 1]$ is a relaxation factor which is used to reduce the amount each iteration corrects. This is to avoid over-correcting on noisy or otherwise imperfect data (i.e., real data). Finding a good value for λ gives both a faster and, for smaller subsets S, more stable convergence. $\sum_{n=1}^{N} w_{in} v_n^k$ is the back projection part, i.e., it

calculates the the value a pixel would have when projected from the current iteration of the reconstruction. The difference between this and the measured pixel value p_i is the error function in this approach. This error function is normalized though multiplying by w_{ij}, the weight of the contribution of v_j to p_i, and dividing by $\sum_{n=1}^{N} w_{in}$, the sum of all the weights affecting p_i. This together adjusts for how big a part of p_i comes from the contribution of v_j. This weighted error is then summed over all the pixels in S and divided by $\sum_{\{i|p_i \in S\}} w_{ij}$, the sum off all the weights connecting v_j to the pixels in S. The sum makes the adjustment of v_j account for all the errors it is contributing to in S and the division normalizes this sum.

5.1.1 Simultaneous iterative reconstruction technique(SIRT)

In the simultaneous iterative reconstruction technique all the pixels of all the projections are used as the "subset". From (21) this yields

$$v_j^{k+1} = v_j^k + \lambda \frac{\sum_{\{i|p_i \in P\}} \frac{\left(p_i - \sum_{n=1}^{N} w_{in} v_n^k\right) w_{ij}}{\sum_{n=1}^{N} w_{in}}}{\sum_{\{i|p_i \in P\}} w_{ij}}, \tag{22}$$

where P is the set of all pixels in all projections.

This gives a high quality reconstruction with a stable end-point, but a very slow convergence. SIRT is also quite computationally heavy, as it requires the entire data set to be computed at the same time over a great number of iterations.

5.1.2 Simultaneous algorithmic reconstruction technique(SART)

In the simultaneous algorithmic reconstruction technique each subset consist of all the pixels in a single projection. From (21) this yields

$$v_j^{k+1} = v_j^k + \lambda \frac{\sum_{p_i \in P_\phi} \left(\frac{p_i - \sum_{n=1}^{N} w_{in} v_n^k}{\sum_{n=1}^{N} w_{in}} \right) w_{ij}}{\sum_{p_i \in P_\phi} w_{ij}}$$

where P_ϕ is the set of all pixels of a single projection.

This is a compromise between the fastness of early ART algorithms that used a single pixel for each subset, and SIRT. Using all the pixels in a projection

ensures, at least for the common projection geometries, that all voxels are represented in each subset, which should make the convergence more stable. With a sensibly choice of relaxation factor SART produces results comparable to SIRT but with a much faster convergence[7].

5.1.3 Sampling with ART

With the iterative approach a system, given no duplicate information, should have enough data to determine all it's voxels if the number of pixels is greater than or equal to the number of voxels. $M \geq N$. The number of voxels in a circular cylinder in the reconstruction with a detector size of N_D has been shown[5] to follow

$$N \approx \frac{\pi}{4} N_D^2, \text{ for a reasonable large } N_D.$$

since $M = N_p N_D$, where N_p is the number of projections,

$$N_p > \frac{\pi}{4} N_D.$$

This is half the number of projections theoreticized for FBP in (19).

5.2 Making reference points in reciprocal space

Defining good reference points in reciprocal space from a series of projections requires some effort. After using the Fourier-slice theorem to place the projections into reciprocal space, the data needs to be placed onto a Fourier grid. For the purpose of iterative reconstruction only points that can be determined with a good degree of certainty should be used. Making this criteria too strict will lead to loss of data, but using too many points will lead to gridding error becoming prominent. To determine these points the method described in the recently developed GENFIRE [11, 12] algorithm is used.

GENFIRE uses a Fourier grid oversampled by a factor O, enabling filling in points on the grid using a stricter criteria without loosing information. The

way points are set can then be described by

$$F_{\text{obs}}\left(\vec{k}\right) = \sum_{\{j|D_j<D_{\text{lim}}\}} \overbrace{\frac{D_j^{-1}}{\sum_{\{j|D_j<D_{\text{lim}}\}} D_j^{-1}}}^{\text{weighting function}} \underbrace{\sum_{x=-\frac{N}{2}}^{\frac{N}{2}-1} \sum_{y=-\frac{N}{2}}^{\frac{N}{2}-1} f_{\text{obs}}^{j}(x,y) e^{\frac{-2\pi i(xu_j+yv_j)}{NO}}}_{\text{DFT for } (u_j,v_j)} \tag{23}$$

where D_j is the perpendicular distance from a point $\vec{k}$ on the reference Fourier grid F_{obs} to the Fourier plane of the j^{th} projection. (u_j, v_j) is the point on this Fourier plane that D_j is measured too. A projection only affects the value of $F_{\text{obs}}\left(\vec{k}\right)$ if D_j is less than a set value D_{lim}. If $D_j \geq D_{\text{lim}}$ for all j the point is deemed unknown. If $D_j < D_{\text{lim}}$ then the discrete Fourier transform (DFT)[13] is calculated of the j^{th} projection $f_{\text{obs}}^{j}(x,y)$ for (u_j, v_j). DFT is preferred because (u_j, v_j) generally falls between the descrete points found through the fast fourier transform(FFT)[14], and using DFT is more accurate than interpolating between those given by the FFT. The results for each projection are then weighted according to the inverse of D_j and summed together. Using the inverse gives the weighting a strong preference for short D_j's, i.e close to a fully determinable point.

5.3 L-BFGS

Limited-memory BFGS was developed as an approximation of the Broyden-Fletcher-Goldfarb-Shanno (BFGS)[10] optimization algorithm using a limited amount of memory. It is a quasi-Newton method and thus bases itself on an estimate of the inverse Hessian matrix(H_k), but instead of storing the dense $N \times N$ matrix it stores a few vectors to approximate (H_k). This makes its memory usage linear, and thus well suited to use with large data-sets.

5.3.1 Basis from Newton's method

BFGS has it's origin in Newton's method. Newton's method bases itself on the second order Taylor expansion around the current guessed solution $\boldsymbol{x}_k$

$$f(\boldsymbol{x}_k + \Delta\boldsymbol{x}) \approx f(\boldsymbol{x}_k) + \Delta\boldsymbol{x}^T \nabla f(\boldsymbol{x}_k) + \frac{1}{2}\Delta\boldsymbol{x}^T B_k \Delta\boldsymbol{x},$$

where $B_k \equiv \nabla^2 f(\boldsymbol{x}_k)$ is the Hessian matrix. This gives a good approximation of f in the close proximity of $\boldsymbol{x}_k$. Taking the derivative with regard to $\Delta\boldsymbol{x}$ on both sides gives

$$\nabla f(\boldsymbol{x}_k + \Delta\boldsymbol{x}) \approx \nabla f(\boldsymbol{x}_k) + B_k \Delta\boldsymbol{x}.$$

The goal of the optimization is to reach a critical point, meaning the gradient is zero. Setting the left side gradient to zero means that moving $\Delta\boldsymbol{x}$ should get us to where there appears to be a critical point according to the local approximation. This would therefore make a good directional vector $\boldsymbol{p}_k$ for the next guess. Moving the terms around and multiplying by the inverse Hessian $H_k \equiv B_k^{-1}$ gives us

$$\boldsymbol{p}_k = -H_k g_k,$$

where $g_k \equiv \nabla f(\boldsymbol{x}_k)$.

Since calculating H_k exactly can be quite an expensive or even impossible task the quasi-Newton family, which BFGS is a part of, approximates H_k in different ways.

5.3.2 Approximating the inverse Hessian matrix

The challenge for quasi-Newton optimization methods is to find a good approximation of the inverse Hessian matrix. BFGS accomplishes this by storing the two vectors $s_k \equiv x_{k+1} - x_k$ and $y_k \equiv g_{k+1} - g_k$ for the last m iterations of the algorithm. With these two vectors, H_k and the helping vector given by

$$\rho_k = \frac{1}{y_k^T s_k},$$

H_{k+1} can be approximated as follows

$$H_{k+1} = (I - \rho_k s_k y_k^T) H_k (I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T$$

An advantage of this approximation it that if H_k is positive/negative definite, then H_{k+1} has the same quality. This matters since H_k needs to be negative definite for minimization and visa versa.

For L-BFGS we first define the series $q_{k-m}, \ldots, q_k$ as $q_k \equiv g_k$ and $q_i \equiv (I - \rho_i y_i s_i^T) q_{i+1}$. A recursive calculation of q_i can then be expressed as $q_i = q_{i+1} - \alpha_i y_i$, where $\alpha_i \equiv \rho_i s_i^T q_{i+1}$. Using this a sequence $z_{k-m}, \ldots, z_k$ can be defined

as $z_i \equiv H_i q_i$. Using $z_{k-m} = H_k^0 q_{k-m}$ and $\beta_i \equiv \rho_i y_i^T z_i$, z_i can be recursively defined as $z_{i+1} = z_i + (\alpha_i - \beta_i)s_i$. z_k is then the approximation of the direction vector p_k.

z_i allows for the direction to be calculated without using the space inefficient H_k-matrix directly. The initial H_k^0 matrix needs to be positive definite for minimization. Often a simple diagonal matrix is used to simplify calculations.

5.3.3 Wolfe conditions

Line search is a technique to find a good step length When a good descent direction $\boldsymbol{p}_k$ is found you have to find a good step size (α_k) along this direction. Ideally you want to solve the sub-problem

$$\min_{\alpha_k} f(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k).$$

In a case like ours, where the loss function is quite costly, it is more efficient to find a step size that is "good enough". The Wolfe conditions are a couple of tests to decide to use a given step size or not.

i) $f(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k) \leq f(\boldsymbol{x}_k) + c_1 \alpha_k \boldsymbol{p}_k^T \nabla f(\boldsymbol{x}_k)$

ii) $-\boldsymbol{p}_k^T \nabla f(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k) \leq -c_2 \boldsymbol{p}_k^T \nabla f(\boldsymbol{x}_k)$

The first condition makes sure that the step size decreases the function sufficiently. c_1 regulates this condition and should generally be small,$c_1 = 10^{-4}$ is a typical value. The second condition makes sure that the curvature decreases sufficiently and is regulated by c_2. c_2 should be less than 1, but otherwise quite large. $c_2 = 0.9$ is a typical value for quasi-Newton methods.

5.3.4 OWL-QN

The Orthant-wise limited-memory quasi-Newton (OWL-QN) is a variant of L-BFGS designed to minimize functions on the form

$$f(\vec{x}) = f_{\text{loss}}(\vec{x}) + C\|\vec{x}\|_1,$$

where $f_{\text{loss}}(\vec{x})$ is a differentiable and convex loss function, and $\vec{x}$ is the solution vector for the problem. This allows fitting L_1-regularized problems.

The way this works is by using an active-set approach. For each iteration the sign of each component is estimated and fixed for the following step. Fixing the sign makes $\|\vec{x}\|_1$ a smooth linear term that can be handled by L-BFGS. After the step some components are allowed to change sign and the process is repeated.

5.3.5 L-BFGS-B

The L-BFGS-B is an extension of L-BFGS to handle simple bounds to the variables on the form $x_{\min} \leq x_i \leq x_{\max}$, where $x_{\min}$ and $x_{\max}$ are variable specific constraints. For each iteration these constraints are used to identify fixed and free variables. Only the free variables are used with the L-BFGS method to further minimize the function.

5.3.6 Limitations of L-BFGS

The main limit, apart from needing a gradient, is the constraints. Although the bounds of L-BFGS-B are on par with other algorithms, I have yet to find an implementation that combines both L-BFGS-B and OWL-QN. As both use active sets, which would need to be combined, this is not a straight forward operation. So to get compressive sensing through OWL-QN you would need to sacrifice the powerful effect of employing bounds. For compressive sensing it would also be useful to apply bounds that are defined in another space than $\boldsymbol{x}$, e.g if the minimization is done in a DCT-transformed space, it would be useful to apply bounds in real space.

Bounds can be forced but without using proper active sets this would make the calculation of H_k extra inaccurate. Using a low value of m would minimize this effect, but would again sacrifice some of the advantages of using L-BFGS. Anyhow the algorithm will be unstable in some way or another.

5.3.7 Using L-BFGS for tomographic reconstruction

To use L-BFGS you need to have a way to calculate the loss function and the gradient. For tomographic reconstruction the techniques from the ART algorithms can be used. We can define the loss function as

$$f_{\text{loss}}(\boldsymbol{x}) = \|W\boldsymbol{x} - P\|_2$$

which is the norm of the difference between the calculated projections and the actual projections. For doing the basis pursuit de-noising with OWL-QN the better function to use would be

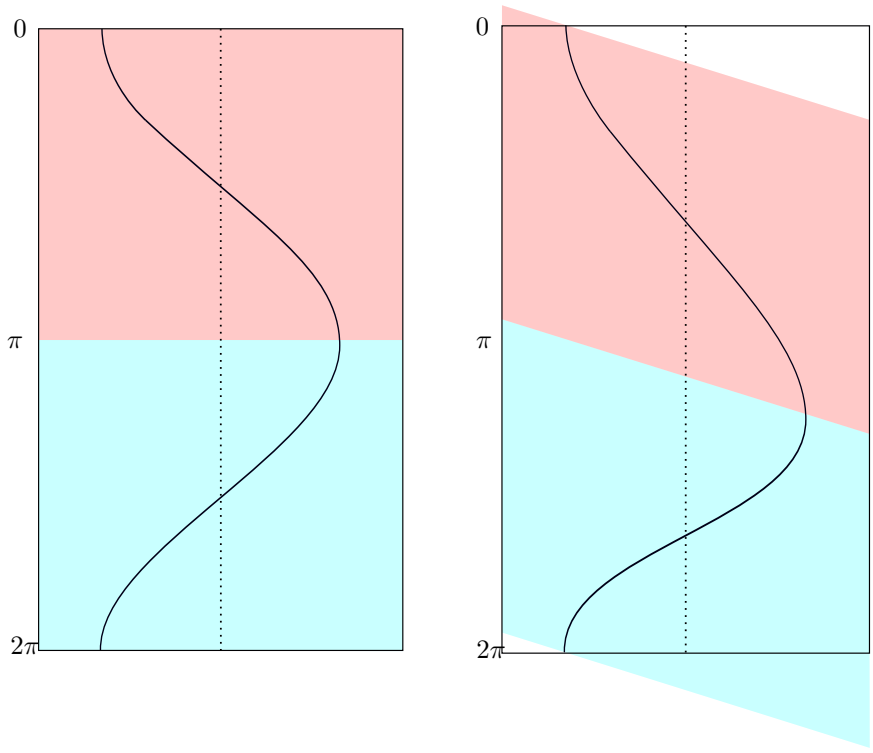
$$f_{\text{loss}}(\boldsymbol{x}) = \frac{1}{2}\|W\boldsymbol{x} - P\|_2^2$$

to match the expression given in (13).

The gradient is slightly more complicated to produce. Luckily the algorithm works quite well with just an approximate gradient, and one from the ART family can be used. The natural choice would be to use the SIRT gradient described in (22), as it should be the most accurate of the gradients. It also removes the complication of having to deal with the complexity of the other ART-gradients only using parts of the data at a time.

6 A closer look at the response of forward projections

If we define an impulse as the dirac-delta, that is an infinitesimal point with the integral sum of one, we can theoretical describe any object by putting together lots of these impulses. By studying the response of the radon-transform to this fundamental building block, different properties of the transform can be described. Since the discretization of the transform happens at the detector it makes sense to start with the analytical case. In that way the discretization can be calculated from that and be better compared to the ideal case. To simplify the following discussion it is assumed that there is only one axis of rotation. This is what most set-ups use so this exercise should still be useful. Translation is also assumed to be absent, but can easily be imagined by shifting the sinograms.



(a) Parallel-beam impulse response **(b)** Fan-beam impulse response

Figure 6: Impulse Responses for parallel and fan geometries. The red and blue areas represent the same data.

There are a few general features for the analytical case. The response is obviously different for every point the impulse inhabits. If this was not the inverse transform would not have determined solution even in the analytical case. The response is continuous and can only inhabit one point at each angle, so it takes the form of a line. Rotating the point around the center only shifts the angle of of the response, so it can be viewed as a phase shift. The impulses' distance from the center affect the maximum distance from the center-line in the response and can be viewed as an amplitude-equivalent. Amplitude is not the only effect, since the shape of the response morphs at greater radii for fan- and cone-beams..

The simplest case is with the parallel-beam. Here the response can easily be shown to be a sine-curve wavelength of 2π radians and amplitude of the radius to the center as shown in figure 6a. If the parallel-beam is normal on the axis of rotation the sine-curve is confined to a single plane. If not, a sine-curve along the height of the detector, 90° out of phase with the previous sine-curve, is added to the response.

Moving on to the fan-beam geometry the response becomes a little more complicated. From section 1.4 it can be related to the parallel response by rotation resulting in a skew and a stretch as viewed in figure 6. This skewness has a couple of effects. Firstly it removes the nice angular symmetry of the parallel case. The direction of rotation now matters. This lack of symmetry matters when taking measurements. For the parallel case shifting 180° will produce a mirrored version of the same response, so 180° of measurement describes the whole perfectly. This is not the case for the fan beam. Even though the red and blue areas in figure 6 represent the same information it is no longer easily accessible.

At first glance it might seem like the fan-beam has compressed the information in the blue area into a smaller part, and in a way it has. But since a real object consists of lots of these responses, and the compressed area does not stay in the same place, we need to use the most stretched part to contain all the information. From section 1.4 this equals 180° plus the angle of the fan. When making the response discrete there is one further complication. The change per angle is, on its quickest, faster than the parallel case. This means a higher angular resolution, that is the number of projections, is needed to fully resolve the response. Since the same information also is contained in a more stretched out part, this should not matter when using single angle projections over 360°.

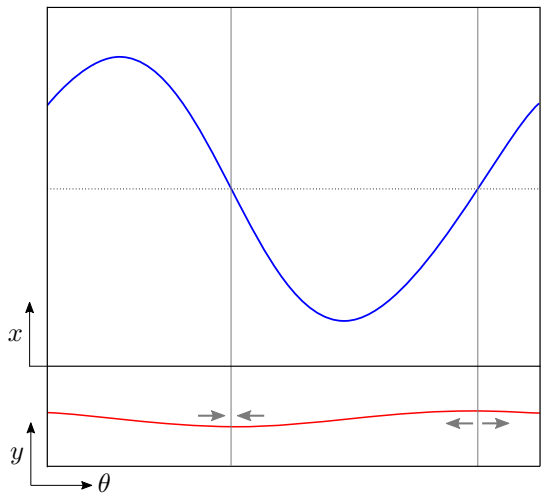


Figure 7: Cone-beam impulse response. x is the direction on the detector perpendicular to the axis of rotation. y is along the axis of rotation. The blue curve is equal to the corresponding fan-beam curve. The red curve here represents a point quite far from the center of the beam along y, the curve flattens out towards the centre

The response of the cone-beam is in many ways similar to the fan-beam. The only difference is in the response along the height of the detector. In figure 7 the response along the height seems to be a sinus-curve, similar to parallel lines not perpendicular on the detector, which would be a good approximation for the local area. The sine curve is however squeezed towards one of the peaks and away from the other in the same places the fan-beam response is squeezed and stretched.

6.1 The consequences of having a real sample

Previously we have accepted that the object being scanned can take any shape and value, but in the real world this is not the case. In the real world the values we are measuring are limited to specific ranges. This is information we can use both before and after reconstruction.

In most common applications tomographic measurements are either measuring some kind of attenuation, or some kind of emission. In neither of these cases does negative values make sense physically, and we have an absolute minimum limit at zero.

This minimum limit is important when the number of projections are reduced, as it helps counter the increased dimensional freedom caused by this. Since the projections are sums over the reconstruction the forced postivity also creates a sort of soft upper limit. With more knowledge about the sample a stricter upper limit can be implemented.

The forced positivity has consequences for what shapes are possible in the sinograms, and this can theoretically be used to pre-process noisy sinograms. Since a point in real space equals a curve of known shape on the sinogram, and the sinogram is a sum of such curves, the non-negativity makes it possible to create a continuity requirement. For very noisy sinograms this could be useful,

The more powerful application of this fact is in the reconstruction itself. Limiting the range of the variables severely limits the degrees of freedom of an under-determined solution. In addition to this it will reduce the degrees of freedom for iterative algorithms trying to minimize a solution, which leads to better and possibly faster convergence.

6.2 Having real measurements

In the real world you can't capture a complete continuous sinogram, you have to make your data discrete. First and foremost you have to divide the measurements into a given amount of projections. Each projection needs a certain amount of exposure to give a good measurement, so the amount of projections you can have are governed by the exposure provided per time and the amount of time you can utilize. There are two main ways of doing this. The first is a continuous scan, meaning every angle is binned into a projection. This can be achieved by the sample rotating during each projections exposure. Even though an uneven rotation (which would lead to uneven bin-sizes) could perceivably have some use, the added complexity in both measurement and reconstruction makes it impractical. An even rotation, and thus even bins, are what is normally used.

You could also imagine having gaps between the bins. This would lead to some

angles being discarded. This scenario might seem useless at first glance, since it implies discarding perfectly useful data, but in situations where you are limited by your data throughput this might still be useful.

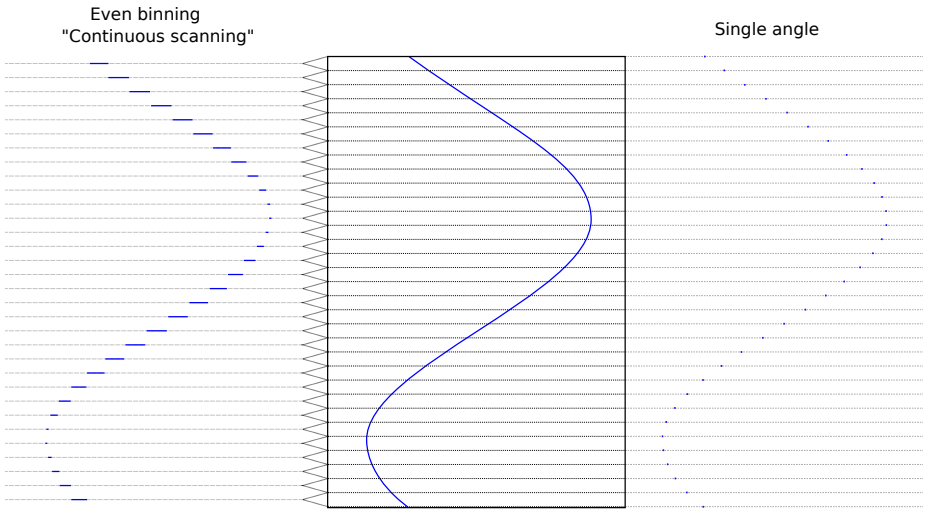


Figure 8: The two main ways to make the angular domain discrete. Either binning the angles with continuous scanning, or using single angles for each projection.

The other way of making a limited amount of projections is to take only the contributions from a single angle for each projection, i.e the sample does not move relative to the detector during the exposure. The advantage of this method is a higher certainty of the spatial placement of your data, the disadvantage is increased scanning time, as the system has to accelerate to and from rest between each exposure.

When the number of projections grow larger the difference between these two methods reduces, and at some point turns smaller than the spatial resolution of the scanning equipment used. Even before this point single angle projections are often assumed during reconstruction, as these offer simpler and in many cases more efficient reconstruction algorithms. The error introduced from this simplification is rarely significant before reaching a number of projections significantly less than this meeting point.

6.3 Exploring the dimensional freedom

Using discrete projections introduces freedom, in that multiple samples can result in the same set of projections. This means that a reconstruction-algorithm has different potential solutions, from which the correct one is impossible to distinguish without additional information. These solutions could all be connected through some set of vectors, or they could be in different groups or even completely isolated. Knowing in what way the solutions are connected could be of help to identify areas of interest for a reconstruction algorithm.

So what can be said about these solutions? Well, the vectors would be dependent on the geometry of the problem. They would not be dependent on the sample itself, unless non-linear corrections are introduced. If we start with changing a small dirac-delta (δ) in the positive direction, this would have to be compensated for in every projection. So for N_p projections we need N_p different $-\delta$. These would then have to be compensated for with additional $+\delta$, N_p in total, following the same logic. This also satisfies maintaining the total sum over space, which is represented in the sum over the projections. $2N_p$ would then be the minimum amount of components of a vector in the solution space.

To test if this minimum is reachable one can place two δ, not overlapping in any projection, and draw the projection lines passing through these points. The points where these lines cross represent places where $-\delta$ should be placed for them to cancel out both δ at same time. This is shown for parallel projections in figure 9. The resulting intersections are 2-fold rotational symmetric. Choosing only intersections that do not share any projection line creates regular N_p-sided polygons of different sizes. These sizes correspond to the distance between the two δ being the length of either a side or a diagonal. They can be viewed as the same solution, but with the two original points representing different points on this solution.

Placing in the rest of the δs of the vector is now easy and the result is seen in figure 10. It takes the form of two regular N_p-sided polygons, one with positive Δ and one with negative Δ, sharing the same center. The polygons are offset on each other by an angle. This angle is given by the offset of the polygon sides from the projection-lines, and can be varied continuously.

These vectors could be used to create more intelligent algorithms for solving the inverse radon transform, but for now let us find a few key points to take away from this. Firstly the number of vectors, and thus dimensions of the solution

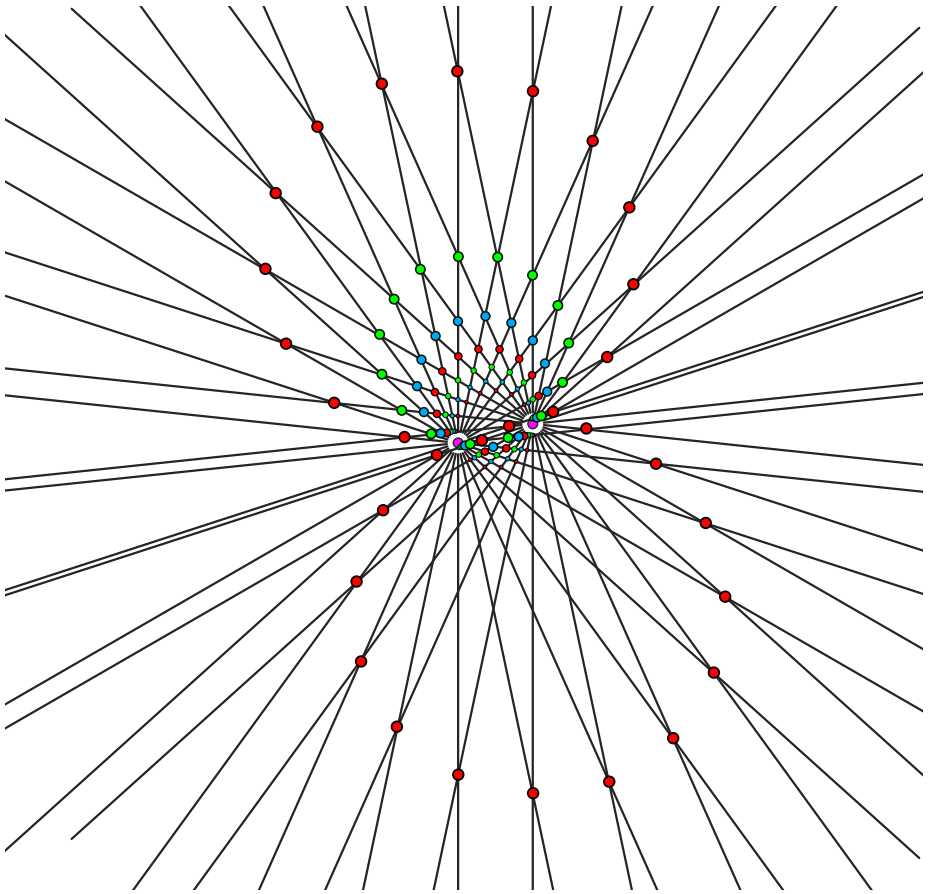


Figure 9: The result from placing two points (marked in purple), drawing the projection lines from both of them and marking the intersections. This is for parallel geometry. Outside the red points the lines are parallel in pairs and otherwise diverging (zooming out far enough they would appear as the projection lines from a single point.

space, increases drastically when reducing the number of projections. Secondly, the fact that the vectors has to contain both positive and negative values means that placing minimum and maximum constraints on the solutions should be quite effective at limiting the solution space.

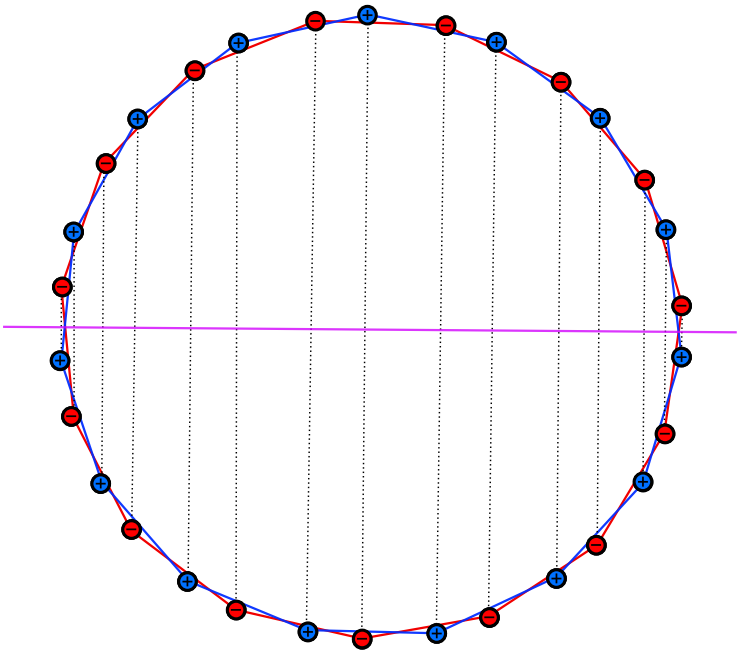


Figure 10: Solution to finding a vector in the solution space or parallel projections. The dotted lines represent the connections between positive and negative points along a projection. The purple line is the symmetry axis for this projection.

If this is all one connected solution space or several disconnected ones is beyond this paper. We know that all the solution spaces have the same shape, as we can apply these vectors to all of them. This also means the space appears the same from all positions. The space is also the same for any non-solution, which can affect the way iterative algorithms move through the problem space.

For the fan and cone-beam case things get a bit more complicated. For the fan-beam this minimum can not be achieved in all positions, and it is likely that there are fewer possible vectors in total. This would lead to a less dense solution space, and likely more separated. The cone-beam further adds to this with the spreading of projection-rays along the axis of rotation.

In real life the projections are made out of discrete pixels, and the solution is described by discrete voxels. The pixelization gives some leeway, but using vox-

els breaks the exactness of these vectors. They still provide a way for minimal change, and are useful for thinking about the problem. Because of the discreteness the size of the vectors is restricted. The minimum distance between a positive and negative point is now the size of a voxel. This means the smallest vector is a regular polygon with $2N_p$ sides 1 voxel-size in length.

6.4 Creating weights for continuous scanning

To adapt ART based algorithms to better function with continuous scanning you need to modify the weight matrix to better represent this geometry. There are two philosophies here, similar to the line and strip ways of calculating weights in normal ART.

The simplest is to adapt the line-weighting method, modifying the line-weight to make the contributions from voxels along the line more accurate. That means less weight on the voxels far from the rotation line. The idea with this method is to reduce the effect of voxels on projections were their contribution is uncertain, and let them be controlled more in projections where their contribution is more certain. The problem with this approach is that the sum of all the weights still must be one. This means the weight that is reduced in the more uncertain areas migrates towards the more certain areas, which we don't want to change. The other apparent defect is that a lot of voxels contributing to a pixel is not taken into account directly.

When the angle bins become too large we need to take into account the rest of the voxels. The idea is to give each voxel as accurate representation of its contributions as possible. This could be accomplished by finding an analytical expression for how the weights distribute, but this would be different for every projection-geometry and have to be made discrete in a flexible and accurate way. A simpler approach is to make a weight matrix for an oversampled number of projections using an established method, to then compact this matrix into the desired amount of projections. This requires answering the key question of how many projections should be used for this oversampled matrix.

This question is a rephrasing of the previously asked question of how many projections are necessary for the continuous and single angle projections to be the same for a certain discreteness. This can be answered by looking at where the impulse responses have the most change per angle. If this part of the curve

does not pass over more than one pixel-width from one projection to the next, the effect of continuous scanning should be negligible. For the parallel geometry this change is given by the derivative of the sine curve

$$\begin{aligned} f(x) &= R\,sin(kx+\phi) \\ \frac{\partial f(x)}{\partial x} &= Rk\,cos(kx+\phi) \end{aligned}$$

where R is the extent of the region of interest in number of pixels, typical $\frac{N_D}{2}$, and $k = \frac{2\pi}{N_p}$, where N_p is the number of projections. Setting the maximum of this derivative equal to one gives us

$$\begin{aligned} \frac{2R\pi}{N_p} &= 1 \\ N_p &= 2R\pi \\ N_p &= N_D\,\pi. \\ &. \end{aligned}$$

This approximates the derivative as a straight line around this point, which is a good approximation as long as N_p is not overly small.

So for 360° measurements at least $N_D\pi$ should ideally be used, and for 180° measurements $N_D\frac{\pi}{2}$ would be sufficient. 180° and 360° are not equivalent, as is the case with single angle projections.

When using a fan beam it is apparent from figure 6 that the maximum change per angle is greater. For this case it is easier to use the real space equivalent of the above derivation. This can be phrased as what is the minimum amount of rotation needed for a point to pass over a pixels field of view.

From figure 11 it is apparent that

$$\sin\frac{\theta}{2} = \frac{p_s}{2R},$$

and since $\theta = \frac{2\pi}{N_p}$ and $N_D = 2R$ we get

$$N = \frac{\pi}{\arcsin\left(\frac{p_s}{N_D}\right)} \approx \frac{\pi N_D}{p_s} \quad \text{since } \frac{p_s}{N_D} \ll 1.$$

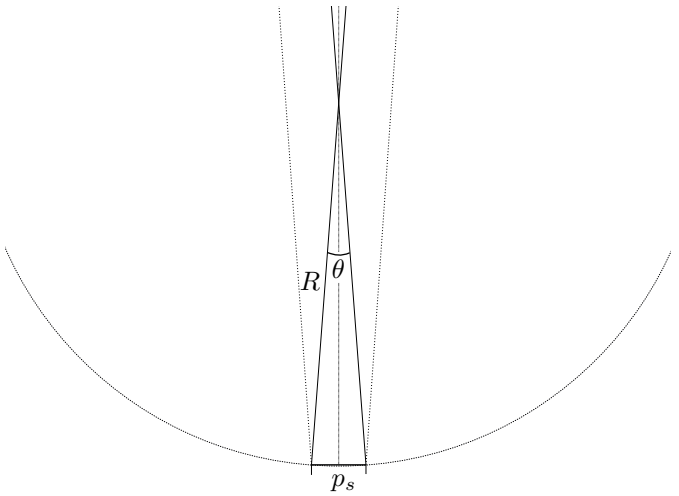


Figure 11: The representation of how large an angle θ a line must rotate to move the over the entire field of view of a pixel p_s at a radius R. This is a good measurement for when the effects of continuous scanning start to appear.

p_s is the size of a pixel's field of view divided by the voxel size. For parallel projections $p_s = 1$, giving us the previously found value for parallel projections. For a fan-beam geometry the size of the field of view (FOV) changes by how far from the source you measure. The voxel size (V_s) is set to be the field of view at the origin of rotation (ideally also the middle of the object scanned). We can then use equal triangles to find the FOV at a distance x from the origin.

$$\frac{\text{FOV}}{\text{SO} + x} = \frac{V_s}{\text{SO}}$$
$$\frac{\text{FOV}}{V_s} = \frac{\text{SO} + x}{\text{SO}},$$

where SO is the distance from the source to the origin of rotation.

The relevant FOV is at the point closest to the source that is still in the volume of interest. Typically this is $\frac{N_D}{2} V_s$ closer to the source than the origin. With $V_s = p_{\text{size}} \frac{SO}{SD}$ we get

$$p_s = 1 - \frac{N_D}{2} \frac{p_{\text{size}}}{\text{SD}}$$

where p_{size} is the size of the pixels and SD is the distance from the source to the detector.

Since the cone-geometry impulse reaction has the most change in the vertical direction when it has the least in the horizontal, we can assume this limit holds for the cone-beam too.

7 Usable information

To recreate how an object looks we need a certain amount of data about the object. How much depends on how detail is desired in the reconstruction and how powerful the information is. A sphere can be described as a point in space with a radius, or it can be approximately described with a lot of points on the sphere itself. If an object is known to be a combination of spheres a lot less data is needed to describe the object than if we only know there is an object. For tomography the question becomes twofold, what a priori information can be easily gathered and what information can be efficiently used.

Usable information can be separated into two categories, set-up information and sample information. The set-up information is the information connecting the measured data to the real world. Set-up information is required to be able to use the measured data, and thus do tomographic reconstruction, but the detail of this data can be varied. Sample information is information specific to the sample being scanned, and can be used with smarter algorithms to improve the results of reconstruction.

7.1 Set-up description

This is a description of the geometry of the projections and what angles they are taken from. Mathematically it is as accurate a description as possible of the transformation matrix W from (20). This is all the information that does not change depending on the specificities of the sample. In addition to the angles and projection geometry this can also include the effects of continuous scanning, translation of the sample and other known global movement of the sample.

Usually the geometry is given by just a few numbers, like the distance from the source to the detector, and the size and orientation of the detector. This leads

to highly condensed information. It is possible to adapt a more fine-detailed description, and W could be tweaked down to the voxel-level. This is however rarely productive, as stochastic sources of error will have a large effect on this level.

Having the information so condensed can have some advantages, as it presents clear vectors to apply corrections to. Measuring equipment is never perfect, and often there will be some jitter in the assumed path of movement. In many algorithms, like GENFIRE, potential jitter is sought corrected by allowing a slight variation in some of the set-up parameters.

Information on how the data is degraded, like noise and optical resolution, could also be viewed as set-up information. This information is not the most useful, but it could give hints on what to expect from reconstructions and how to filter the result.

7.2 Sample information

In many cases a lot of information about a sample is already known at the time of scanning. This information can be used to constraint or regulate a reconstruction to find better solutions. This is mostly useful when the system is under-determined fusing the measuring data alone.

Often the type of material is known, and thus the range of possible attenuation coefficients. Sometimes the ratios between the materials are known, giving a sort of distribution of the possible attenuation coefficients. This information allows for constraints to be placed on the possible values for a reconstruction, which is fairly simple to implement. Using a known distribution of attenuation coefficients is harder, and is for now left to the post-reconstruction analysis.

More specific knowledge about the sample could also be known. The sample could for instance be made out of parts with an approximately known shape. Or the expected sizes of the volumes inhabited by different materials could be known. Many samples also form regular patterns. This information is however harder to use than the attenuation information. If a domain where this information is sparse exists, compressive sensing can be used to exploit it. A lot of this information can not directly be used in the reconstruction, but must be applied in a segmentation step or in the analysis after.

7.3 Reducing measuring-data

There are two main reasons to reduce the amount of data/projections you are gathering. Either you want to reduce exposure or you want to reduce scanning time (or both). In the first case you are exposure bound, which means you are limited in the amount of data you can collect at the detector, i.e how many projections you can make. If fast scanning time is desired under these conditions continuous scanning with few projections is the only real option. In the second case you can be bound by exposure, but also by data throughput. The data throughput problem can also be solved by reducing the number of projections taken, but another way to solve it is by compressing the data on the fly.

7.3.1 Discarding data

The simplest of these on-the-fly compression techniques is to simply discard some of the projections. While first capturing and then not storing whole projections might seem counter-productive, it has some merit. Because of the delays associated with stopping and accelerating the rotation, and since disturbing the X-ray beam might introduce fluctuations, it is easier and faster to just scan continuously. Doing this scan with a lot of projections means the effect of it being continuous is minimized for each projection. Discarding the additional projections is then faster than taking stationary projections, and more accurate than taking less projections in a continuous scan.

A further obvious improvement is to not store regions known to be empty, i.e. not storing values outside a region of interest in the projection. These values would then be set to the expected value for $\mu = 0$. This would of course be the same as just having a smaller detector, and this method would only give significant reductions when the detector is under-utilized. If, however, the object is much wider in one direction than the other, necessitating such a large detector, a variable region of interest could be used. If the user defines a three-dimensional region of interest around the object this could be projected on to the different projections, creating a varying region of interest on the projections.

When first discarding data there is no reason to limit one self to just conventional projections. Picking the same amount of pixels according to some other pattern allows for maximizing the relevant data and to minimize and control the defects in the reconstruction. Using pixels bound to projections is not ideal

because of the angular regularity. Spreading the pixels equally over all the measured projections appears as an obvious choice for a general pattern. To avoid systematic artefacts a uniform random distribution over the projections could be used. Specific sample-information could be used to change the distribution for improved results.

The uniform random pattern seems like a good starting point, but it can be improved upon with two simple observations. Firstly, a greater part of the object is generally represented towards the center of the sinogram than towards the edges. If we only view the cylinder-shaped region that represents the voxels visible from all projections, the edge pixels only contain information about one voxels while the ones in the center contains the whole diameter. The second observation is that the voxels viewed by the edge pixels changes more slowly from projection to projection, meaning the sampling rate can be lowered here.

From the first observation a distribution based on a circle seems like a good choice. This can be done by the simple formula

$$P(r) = \frac{2}{\pi R^2}\sqrt{R^2 - r^2}, \quad r \in [-R, R]. \tag{24}$$

where $P(r)$ is the probability at the distance r from the center of a detector with width $2R$. The distribution simply takes the form of a half-circle with radius R.

The above distribution does not directly take into account the effect from the reduced speed towards the edges. The effect of this is quite a bit harder to calculate, and would at least partially depend on the shape and position of the measured sample. This means there is probably room for an improved general distribution, and an avenue for using sample-information.

7.3.2 Keyframing

Keyframing is a concept borrowed from video-compression. The basic idea is to store whole frames only at some points. In between these frames only the difference from the previous frame is stored. More modern algorithms also store differences to following frames, but since this breaks the on-the-fly nature, it is of limited interest here. If there is only small differences from frame to frame this can potentially save a lot of space.

In tomography the difference from one projection to the next is usually quite low, and should work quite well with keyframing. One problem is how to tackle

noise. One way is to treat differences under a calculated noise level as no change. This might introduce artefacts in the image, as there is an abrupt introduction of change when a value hits this limit. This can possibly be solved with applying dithering, that is adding artificial noise, to the areas not changing. Another technique is to apply some kind of soft limit.

Even though one keyframe could probably suffice in tomography, it would probably be wise to have several keyframes at given intervals. This will work both to increase the robustness of the data set and to avoid the effects of noise and other defects building up.

8 Method

8.1 Algorithm implementations

For the implementations of the SIRT and SART algorithms the ASTRA-toolbox[15, 16, 17], tomopy[18, 20] was used. For constructing the Fourier-grid the fillInFourierGrid() function from GENFIRE-Python[21] is used with some slight modifications to fit the 2D-case. Two different implementations of L-BFGS are used. fmin_l_bfgs_b() from scipy.optimize[22] for L-BFGS-B and the pylbfgs wrapper for liblbfgs[19] for OWL-QN. For a full list of packages used, with version numbers, see appendix A.

The loss-function in the L-BFGS implementations are calculated by first taking an inverse transform, if the L_1-regulation is performed in another domain, and then taking the forward projection. Half the 2-norm squared of the difference between the forward projections and the input projections are then used. The gradient is calculated by taking the back-projection of the same difference and then applying any additional transforms used. The improvised lower limit implementation for OWL-QN is applied by simply setting the solution values, when properly transformed, to zero. For L-BFGS-B proper bounds were used

For the SIRT and SART the ASTRA CUDA algorithms are used. For the forward and back projections in the L-BFGS implementations the OpTomo frontend of ASTRA is used. SIRT and SART are calculated with a minimum constraint of 0, using the inbuilt constraint mechanism, if not specified otherwise. For SART N_p calls are treated as one iteration, so all the projections are used in one iteration.

The parallel geometry is used for calculating all the artificial data, and the cone geometry is used for the real data.

8.2 The data-sets

The data used in this project is acquired at a Nikon XT H 225 ST micro-CT machine located at NTNU, Norway. This machine uses a fixed cone-beam projected on to a flat detector, while the sample can be rotated and translated in the beam. The X-ray beam of the machine is neither monochromatic nor coherent. The detector has 2000×2000 pixels, with the pixel spacing given as 0.2 mm. The distance from the detector to the virtual source, the spot where the spherical approximation of the wave has its origin, is given as 1127.464 mm. The tilt of the detector is given as 0. There is no standard deviation or other error given for these measurements.

8.2.1 data-set 1 (toothbrush)

The sample used in this data set is the head of toothbrush cut in such a way that the remaining hairs are mainly parallel (figure 12). This was placed so the hairs were closely aligned with the axis of rotation during scanning. The hairs are gathered in a total of 26 bundles, 15 forming a regular grid, and the rest in a narrowing grid next to them. Each bundle is about 1.5 mm across, slightly widening towards the end, and 2 to 2.5 mm apart, center to center. Each hair is around 0.15 mm in diameter and are seemingly randomly but tightly distributed within the bundles.

Figure 12: The cut toothbrush used for data-set 1

The sample was scanned over 360°, while continuously rotating, for a total of 3142 projections. Each projection had an exposure time of 2000 ms and where amplified by 18 dB. The projections were corrected with a shading correction using averages of images set to 0, 25 % and 100 % of the white target at 60 000(16-bit), while the black target was 0. The scan was performed at 135 kV and 100 µA with a tungsten reflective target.

This sample is chosen to have some distinct frequencies in the rotated plane, but to be simple along the axis of rotation. The distinct frequencies are intended to give some information about the sample to exploit in different algorithms, as well as having some additional information to analyse the performance of the algorithms with.

8.2.2 Data-set 2

This sample consists of a 0.7 mm capillary tube filled with small steel spheres with radii given as 27 to 31 µm and then topped of with glycerol left to seep down between the spheres.

The sample was scanned over 360°, while continuously rotating, for a total of 3142 projections. Each projection had an exposure time of 4000 ms and where amplified by 18 dB. A similarly configured shading correction as in data-set 1 was used. The scan was performed at 85 kV and 85 µA with a tungsten transmission target.

This data-set was originally intended to be the main data-set used for testing. The hope was to create an interesting three-phase problem with some regularities to exploit in all dimensions. Due to complications during data-gathering, the resulting data was deemed not good enough for the reconstruction comparisons. The data is however, with some precautions, usable for techniques where the projections are only compared to other projections. For these purposes it serves as representing a more symmetrical and dense sample than in data-set 1.

8.3 Making artificial data

Some artificial data is used to test different aspects of the algorithms. It would be better with real data to really test the robustness of some of these algorithms,

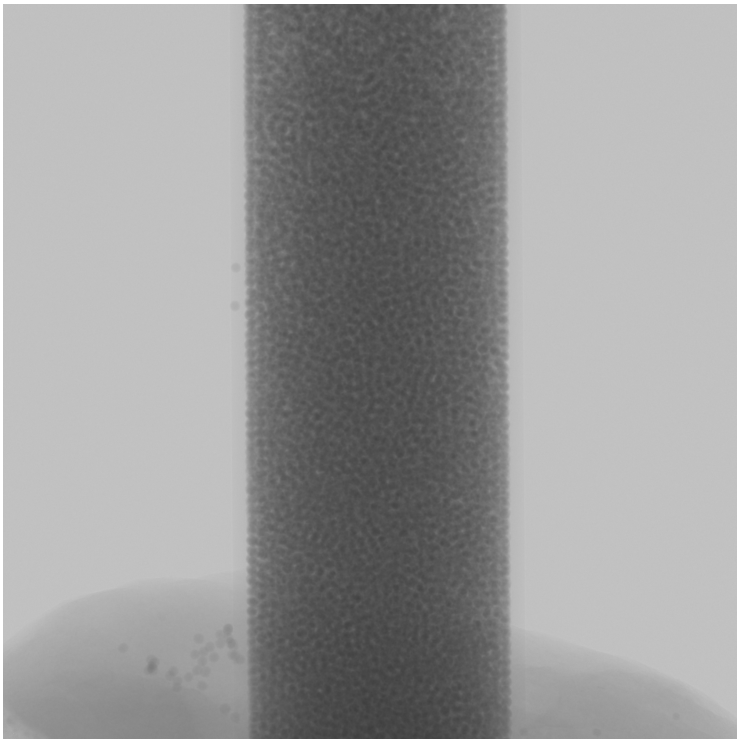


Figure 13: Example projection from data-set 2

but due to complications with capturing data, an artificial data set representing a three-phase problem was deemed necessary. In addition to this a reference image for data-set 1 is also created. These artificial data sets are created by using the ASTRA-toolbox forward-projection on the reference images below, and then adding noise to the resulting projections.

8.3.1 Central slice from data-set 1

To make representation of the sample in the central slice of data-set 1, a reconstruction from ASTRA's FBP_CUDA algorithm using the fanflat geometry was used as a starting point. This reconstruction was then segmented using a

simple threshold segmentation. After the segmentation a few obvious detached pixels were cleaned up giving the result seen in fig. 14

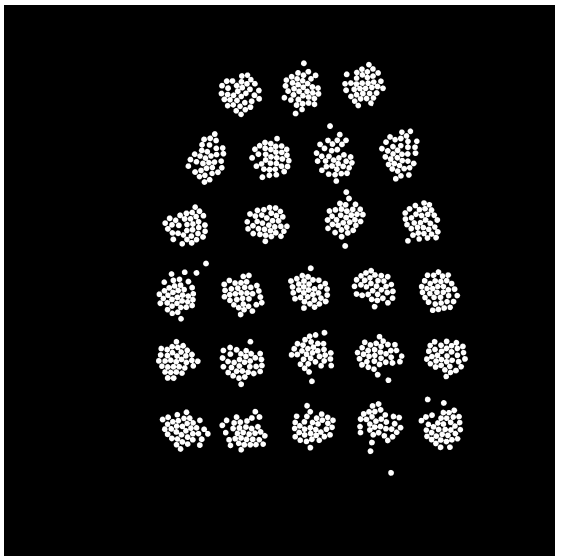


Figure 14: Full 2000×2000 resolution

This reference image creates a useful comparison for the real data, and also serves as the basis for creating the three-phase problem.

8.3.2 Three-phase sample

There are multiple reasons a three-phase problem is desired. A three phase problem is quite a common problem in tomography, so it gives a better representation of typical samples. Secondly it makes the sample less sparse, which makes it a harder problem for compressive sensing. Lastly it demands a better distinction between the levels to properly discern details, so it should give a better idea of how the algorithms collapse.

To quickly create a usable three-phase problem the reference image from dataset 1, described above, was used as a starting point. First a circular, slightly of center, background with a value about half-way between the two segments was

added. To this background some holes of various sizes are added. The result can be seen in figure 15. The small holes, and the overlapping in different ways with the hairs, should give some interesting details to see effects on. This data set is only a 2D-slice, but serves as a efficient initial probe into the algorithms.

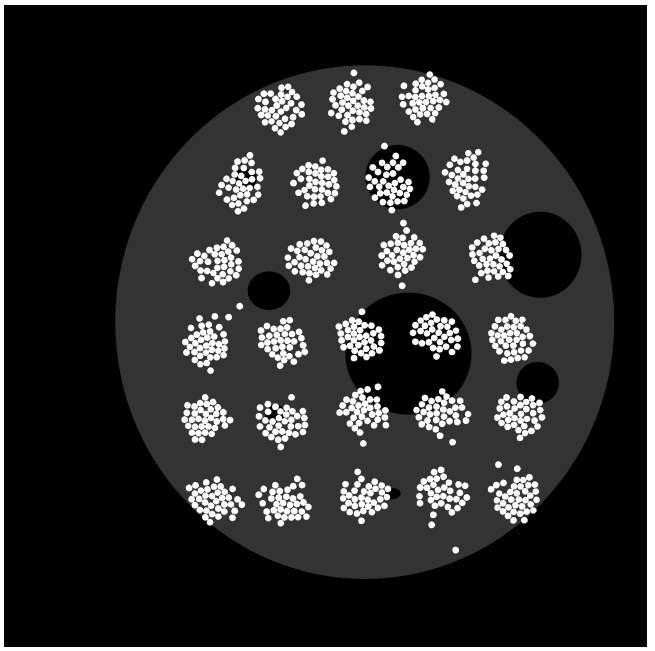


Figure 15: The artificial 2-dimensional data set used when studying the algorithms on a three phase problem.

8.3.3 Adding noise to the data

Noise is added to this artificial sinograms using the add_noise_to_sino() function from the ASTRA-toolbox. This function scales the sinogram to the range 0 to 1 and then uses it as the exponent in (1) to create a detector intensity. Poisson noise is then added to this modified sinogram before it is transformed back. After the noise is added the sinogram is made to be strictly positive, as the noise added to the empty regions creates negative values.

This is a very simplified form for noise, but should work to compare the algorithms to each other. The scaling of the sinogram ensures that the variation in intensity is lower, making the variation in noise lower. This ensures a constant background noise with a slight intensity based variation on top, which should adequately simulate a combination of thermal and shot noise.

9 Results

Only the parallel-beam geometry was used for making the artificial data in the following results, but the fan-beam geometry rendered similar result where tested.

9.1 Comparison compression

To have something useful to compare with the reference sample-images are compressed using the PNG lossless compression. The resulting file-size, compared with the original, gives an estimate on how much information is actually needed to replicate the data. This can then be translated into the number of projections this would represent given an ideal algorithm. A lossy compression would maybe be more logical, as a perfect reconstruction is not really sought, but the go-to JPEG-compression rendered less compressed results with all but the most lossy settings, leading to the choice of the PNG-algorithm.

The sample images used are both 2000×2000 pixels. The original detector used had a bit-depth of 16 bit. This leads to an original image size of 8 MB. In sample 1 only a bit-depth of 1 is really needed and in sample 2 only 2 bits are needed. This gives a bit-depth reduced size of 500 kB and 1 MB, without taking the shape of the sample into account.

Sample	PNG size	ratio of full size	ratio of reduced bit-depth
Central-slice toothbrush	47.4 kB	0.6 %	9.5 %
Three phase sample	72.4 kB	0.9 %	7.2 %

Table 1: The reference compression ratio for the two sample images

Obviously the first ratio in table 1 is way too low, resulting in 18 and 28 projections out of the full 3142 projections. This would require working a lot of

a priori information about the sample into the reconstruction, as well as the absence of noise. The second ratio seems more sensible, and would equal a case where we don't have the information about the segmentation. This equals 298 and 228 projections. Taking some kind of segmentation into account, but not being able to exploit it fully, one would expect to be able to go to a slightly lower number of projections than this and still retain a decent amount of information.

9.2 Comparing the convergence of SIRT, SART and L-BFGS-B

To compare the convergence of the algorithms the 2-norm of the difference between the original projections and the projections from the solution, determined $f(x)$, is used. The algorithms themselves use the square of this for minimization, but using this measure gives a more intuitive picture. $f(x)$ is not necessarily a good measure for the quality of the reconstruction, especially when the system is so under-determined as this, but it gives a good view of the efficiency of the convergence of the algorithm.

Algorithm	$f(x)$	# iterations	bounds
SIRT	1.5e5	1000	$[0, \infty)$
SIRT	9.1e4	2000	$[0, \infty)$
SIRT	5.9e4	2000	$(-\infty, \infty)$
SIRT	5.6e4	4000	$[0, \infty)$
SART	3.5e3	4000	$[0, \infty)$
SART	3.1e3	8000	$[0, \infty)$
L-BFGS-B	2.1e3	2031	$[0, \infty)$

Table 2: Data from reconstructing the three-phase problem using 120 parallel projections with no noise added.

In table 2 the resulting $f(x)$ after a number of iterations is illustrated. For this data no noise has been added, so an ideal algorithm should be able to reach $f(x) = 0$ at some point. This is with an all zero starting point, which has $f(x) = 4.5e7$. The unconstrained algorithm converges slightly faster, but not fast enough for this to be the source of the problem. SIRT has as expected the slowest convergence, but SART is not much quicker. L-BFGS-B is clearly the fastest. L-BFGS-B has slightly more function calls than shown, but the search direction is mostly well scaled, so not by a lot. This indicates that the

ART based algorithms struggle with their convergence with a low number of projections.

Algorithm	$f(x)$	# iterations	bounds
SIRT	2.2e5	1000	$[0, \infty)$
SIRT	1.7e5	2000	$[0, \infty)$
SIRT	1.4e5	4000	$[0, \infty)$
SIRT	1.3e5	8000	$[0, \infty)$
SART	1.9e5	4000	$[0, \infty)$
L-BFGS-B	1.1e5	1755(1852)	$[0, \infty)$

Table 3: Data from reconstructing the three phase problem using 120 parallel projections with noise added.

When adding a bit of noise (SNR=74) to the data the result is as seen in table 3. This time the iterations seem to flatten out earlier, which is a logical response to the noise. The difference between the different algorithms is also much smaller. SIRT seems to converge slightly faster than SART this time, which is interesting. L-BFGS-B still has the fastest convergence.

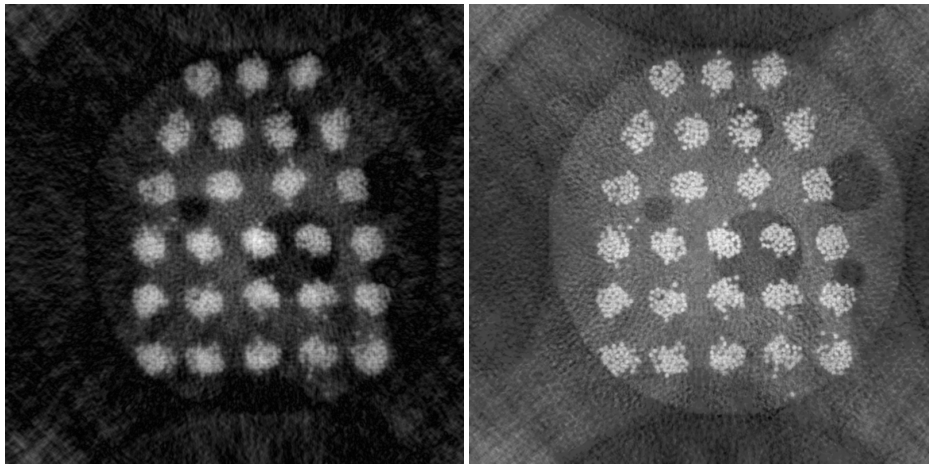
9.3 Compressive sensing

9.3.1 Using a Fourier grid

Putting the reference points on a Fourier grid, and then doing the L_1-regulation on different transformations on this, was tested. Both using the DCT and real space were tested, but both showed poor results. The result of the DCT reconstruction is shown in figure 16. The amount of detail compared to just taking the IFFT of the grid is quite good, but nowhere near just using plain SIRT or any of the other algorithms described here.

9.3.2 L_1 in real space

Since the sample itself needs to be quite sparse when using the L_1-regularization on the real space representation the three-phase sample is not used in this case. Instead the central-slice from data-set 2 and its reference image is used. For



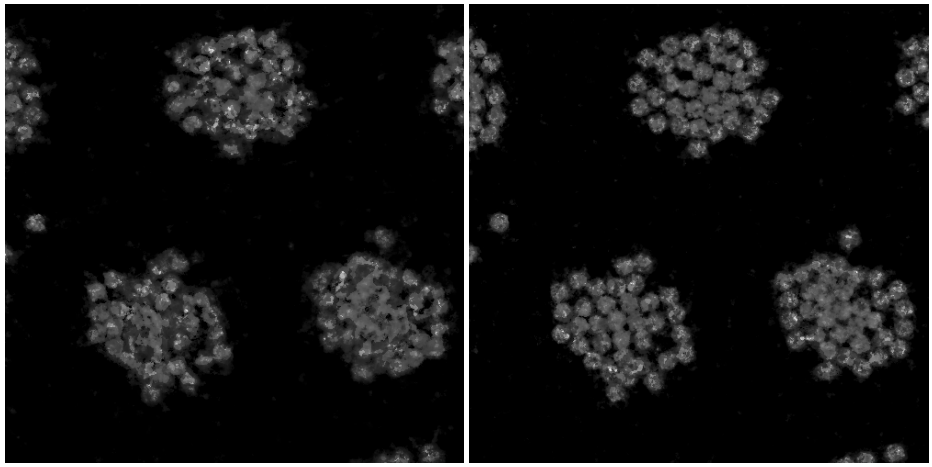
(a) IFFT of the grid **(b)** L_1-regulation in DCT-transform

Figure 16: The result of making a Fourier grid for reference-points for for compressive sensing

these results OWL-QN was used with the standard value of orthantwise-constant (C) of 5, which seemed to render acceptable results.

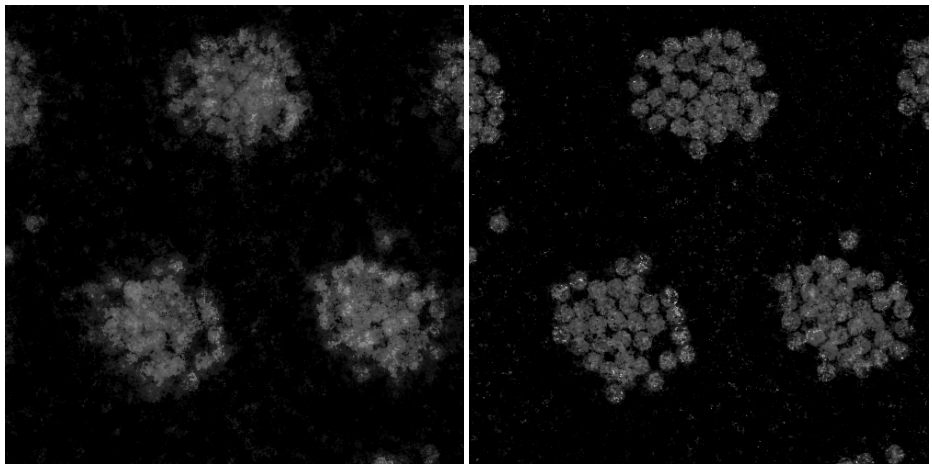
Testing the algorithm without noise on projections of the reference image created quite good results, as seen in figure 17. The algorithm creates results that preserve most details down to just 60 projections, about 2 % of the projections of a full series. At 40 projections the bundles are still well defined, but the individual hairs are starting to disappear. In both images intensity artefacts appear.

With the real data, seen in figure 18b, the reconstruction breaks down earlier than this. At 99 projections, around 3 % of the projections, the hairs are still nicely separable. Below this the distinctness of the hairs quickly degrade and at 50 projections. The bundles also seem to separate into two distinct levels, with the central level having more intensity.



(a) 40 projections **(b)** 60 projections

Figure 17: The result from OWL-QN of noiseless artificial data from set 1



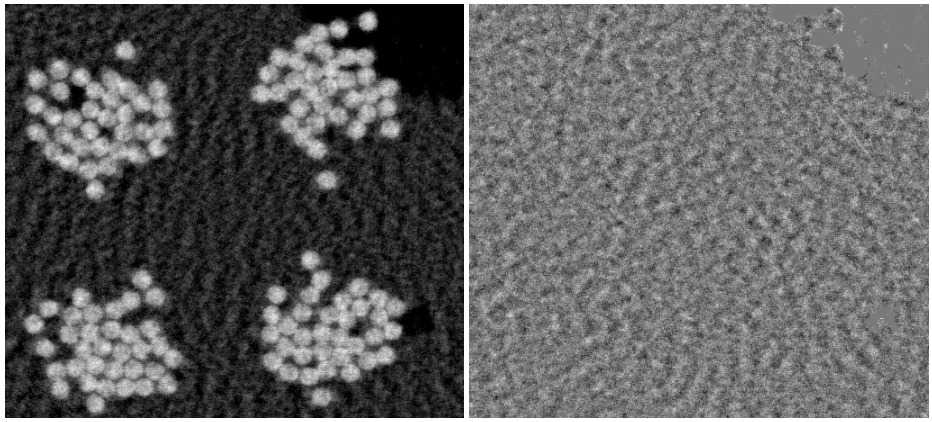
(a) 50 projections **(b)** 99 projections

Figure 18: The result of OWL-QN of the mid-slice from data-set 1

9.3.3 L_1 using DCT transform

Dividing into smaller blocks to apply DCT to resulted in quite severe edge effects between the blocks, so the DCT was applied to the whole reconstruction at once. A lower limit of zero was applied. Even though this resulted in the OWL-QN terminating due to a bad search direction, which is far from ideal, it still rendered better results than not using a lower limit. Thee following reconstruction are done on data with added noise giving an average SNR of 53.

The reconstruction was found to be sensitive to the starting position for the minimization. Starting with an all constant array the constant seems to make no difference. Starting with a non-even array, features seem to carry easily over to the result. It is therefore advised to start with a constant array.

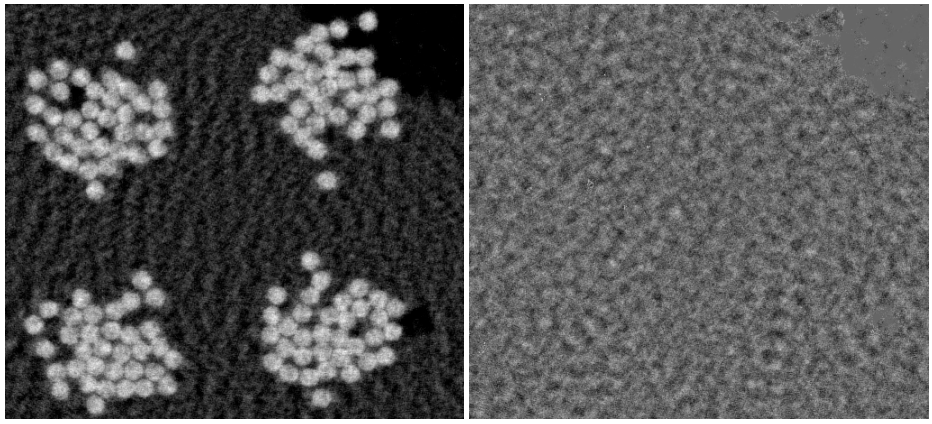


(a) OWL-QN with DCT **(b)** Difference from SIRT

Figure 19: The comparison between OWL-QN using $C = 5$ and a SIRT reconstruction of 2000 iterations

Seen in figure 19 is the result of using OWL-QN with a DCT transform and $C = 5$. In In the difference to a SIRT reconstruction of 2000 iterations only noise is really discernible. This noise is in the disfavour of the OWL-QN implementation, probably because of early termination from a bad search direction.

As seen in figure 20, which is representative of the whole reconstruction, the effect of applying the circular ROI has a small effect. The difference image



(a) Circular ROI applied **(b)** Difference from without ROI

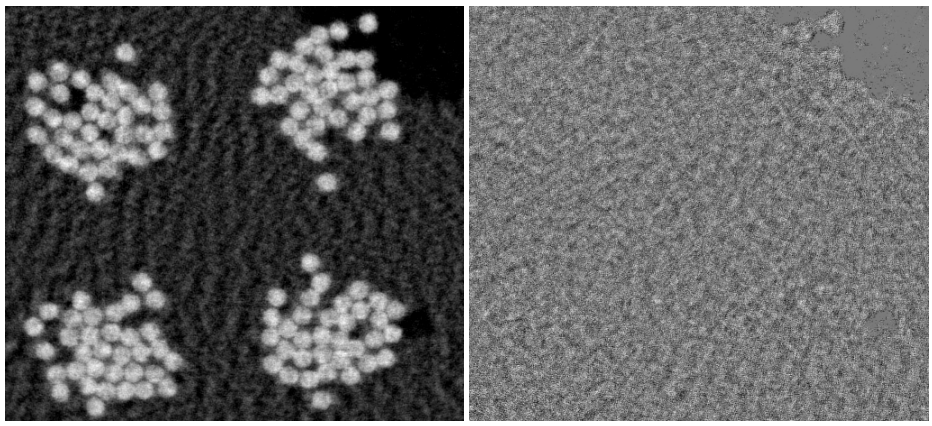
Figure 20: The effect of adding a circular region of interest around the center of rotation with a diameter of 0.97 the width of the image

shows that the change is mostly in the reduction of noise, with almost no detail being added. The only detail clearly visible in the difference is around the zero regions, since these are already quite removed of noise. The noise difference seems to be dominated by frequencies similar to those represented in the data, but does not seem to reduce the frequency defects in the image significantly.

A low C of 5 seems to render a result quite similar to a non-compressive results. Raising it resulted in blurring, which with a good constant could be used for noise reduction. A constant around $C = 10^4$ seemed to give the best balance.

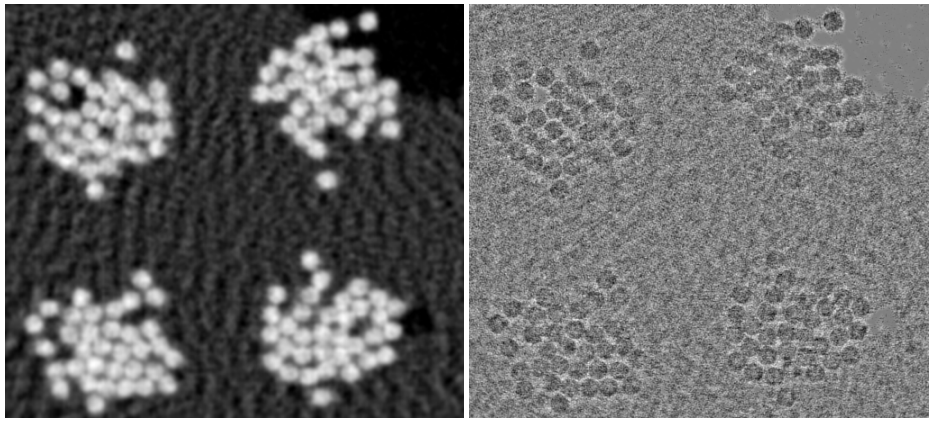
As can be seen in figure 21 the raising of C to 10^4 leads to a reduction of high frequency noise in the image. It seems to have little effect on the details themselves, except for maybe softening the edge towards the zero-regions.

Further increasing C to 10^5, seen in figure 22, further reduces the high frequency noise in the image. Now details have started to soften and the image appears blurred. The difference image shows quite clearly that the edges in the image have smoothed out.



(a) $C = 10^4$ **(b)** Difference from $C = 5$

Figure 21: The effect of increasing C to 10^4 to make it take more of an effect.



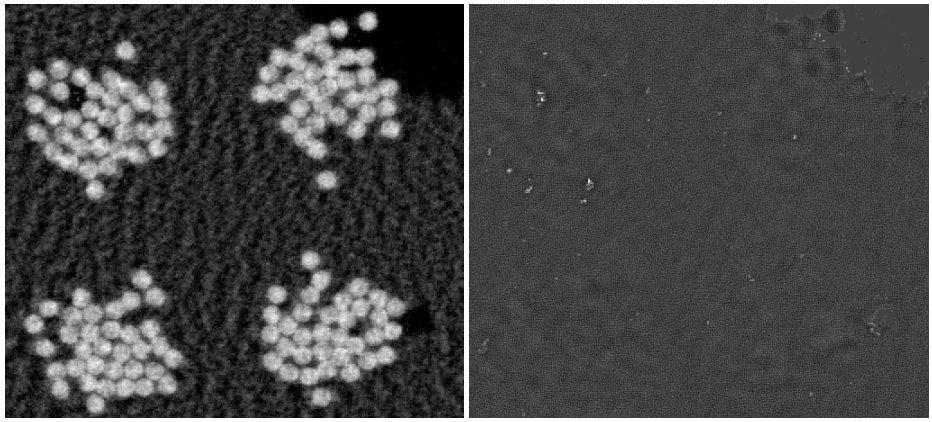
(a) $C = 10^5$ **(b)** Difference from $C = 10^4$

Figure 22: The effect of further increasing C to 10^5.

9.3.4 L_1 using DWT transform with Haar wavelet

Only the detail part of the transform was used for the L_1-regulation. Using the approximation in the regulation led to artefacts in the image. Using multilevel DWT deeper than 2 levels resulted in significant blocking effects in the image. Only level 1 one and 2 DWT are shown here. As with the DCT, DWT proved sensitive to the starting position.

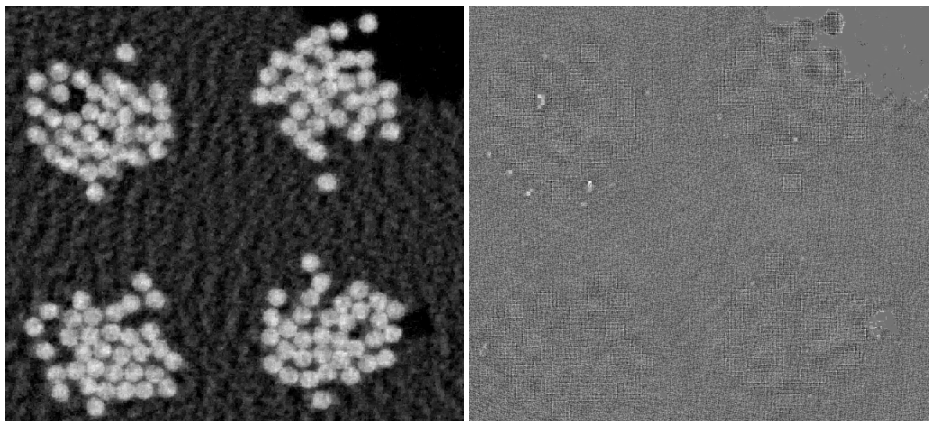
Raising C resulted in somewhat clearer edges to begin with, but quickly caused pixelation noise, i.e. distinct edges between pixels that should not be there. $C = 10^4$ was found to be around the sweet spot.



(a) DWT level 1 **(b)** Difference from SIRT

Figure 23: Regulation using level 1 DWT with $C = 10^4$

As seen in figure 23 applying the DWT once with $C = 10^4$ shows a slight emphasis on the hairs compared to SIRT, but does not improve the edge resolution. Using the DWT also introduces artefacts compared to SIRT that can not be related to the desired outcome. In figure24 the result of a level 2 DWT show most of the same features as the level 1 regulation. The artefacts from level 1 are still there, but less distinct. Instead there has been introduced additional unwanted edges around the hairs corresponding to the level 2 detail matrices.



(a) DWT level 2 **(b)** Difference from SIRT

Figure 24: Regulation using level 2 DWT with $C = 10^4$.

9.3.5 Combining DCT and DWT transforms

Iterating between DCT and DWT, using the output of one as the input of the other, proved to combine the defects of both without really doing much to improve the result. Using the DCT-transform on the approximation matrix from the DWT also gave poor results. The DWT dominated the regulation, and increasing C led to the pixelation noise dominating without getting much of the effect from the DCT.

The most successful way to combine the two transformed turned out to be running them individually and then taking the average of the two. But with noise added to the projections the benefit of this approach dissipated.

9.4 Data throughput reduction

9.4.1 Conventional image compression

To give a reference for what kind of compression one would want for data throughput reduction, the result of compressing the projections from data-set

1 and 2 have been calculated. These algorithms are not necessarily usable with lots of data in a data-stream, but they present some reference point to base one self on.

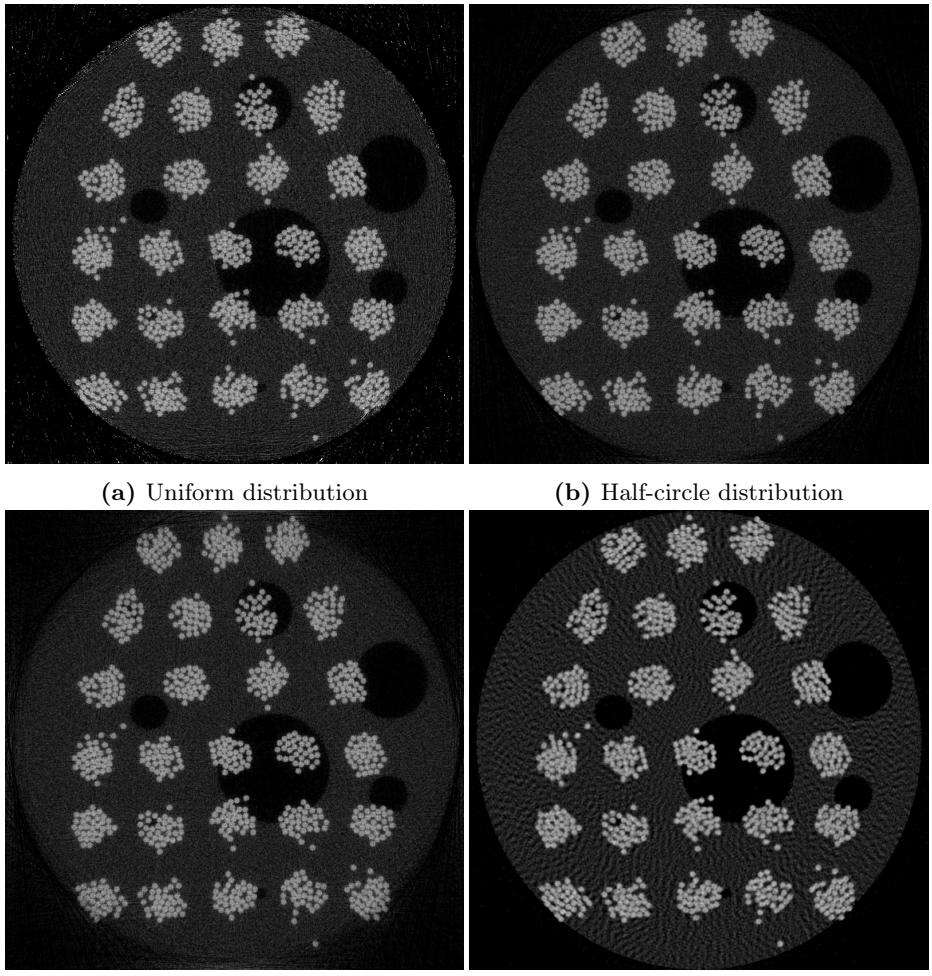
For these reference values PNG was used for lossless compression and JPEG for lossy compression. The original projections are 2000×2000 16 bit pixels and thus 8 MB of size. PNG only reduced this to about 7 MB. This is not great, but quite logical as noise is hard to compress lossless. JPEG reduced this to around 300kB, or around 3.8 % of the original size.

9.4.2 Random point distributions

In this section the results of using three different random point distributions are showed. The distributions used is a uniform distribution, the half-circle distribution described in (24) and the square of the half-circle distribution. This random distributions were applied to a sinogram of the three-phase sample using 3142 projections. To avoid the empty space at the edges giving unwanted favour to the center-heavy distribution, a truncated version of the sample image was used. The sinogram had noise added giving an average SNR of 51. The distributions were made with the numpy.random.choice() function. The reconstructions were done with with L-BFGS-B for the normal reconstructions and with OWL-QN for the L_1-regulated reconstruction.

From figure 25 it is apparant that the half-circle distribution gives quite a flat distribution. It is less affected by noise towards the edges than the uniform distribution. The squared distribution loses quite a bit of definition towards the edges, and is clearly to emphasised on the center. Compared to the reconstruction using 120 conventional projections the random point reconstructions trade the lower frequency artefacts for more more high frequency noise.

In figure 26 a closer comparison of the details are performed. The center of rotation is located above the right edge. From the uniform to the half-circle distribution there is a noticeable improvement. The difference between the half-circle and the squared half-circle distributions is less marked. The hairs towards the top seem generally more defined in the latter, but the hole in the top-left bundle seems less defined. The hairs in the bottom left corner also appear less defined. The zero areas are more defined using the conventional projections, as the point distributions have high frequency noise bleeding into them.

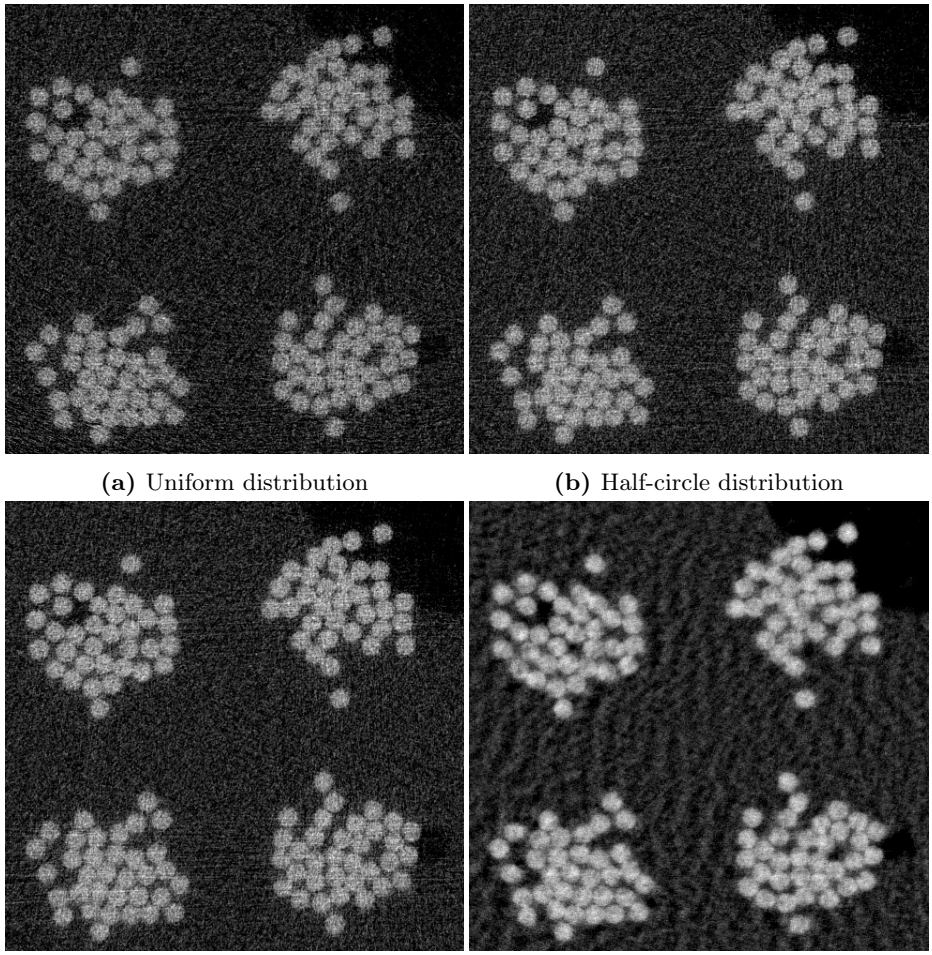


(a) Uniform distribution **(b)** Half-circle distribution

(c) Squared half-circle distribution **(d)** Conventional projections

Figure 25: Comparison of the whole reconstructions with random points. The number of points are equivalent of 120 projections, 3.8 % of the full 3142 projection series.

In figure 27 the effect of using the OWL-QN algorithm with L_1-regulation in the DCT domain and $C = 10^4$ can be seen. The random point distribution seems to give good conditions for the compressive sensing. Compared to the L-BFGS-B



(a) Uniform distribution

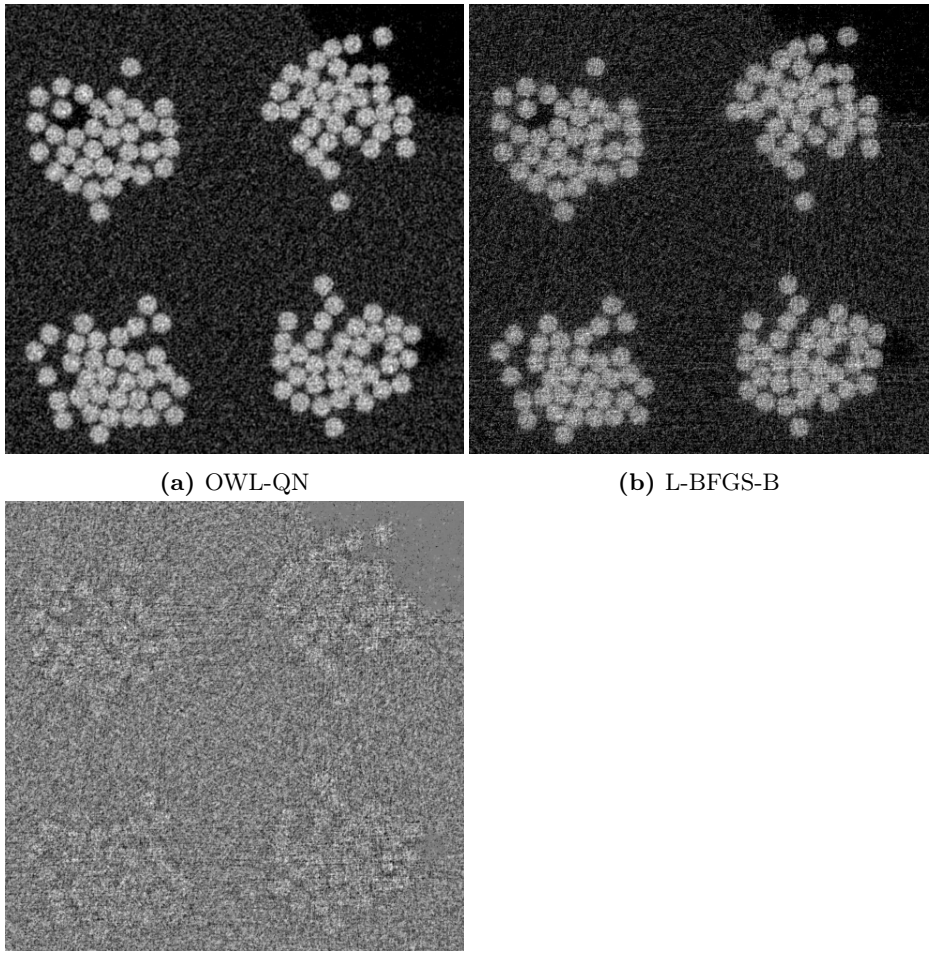
(b) Half-circle distribution

(c) Squared half-circle distribution

(d) Conventional projections

Figure 26: Comparison of details with random points. The number of points are equivalent of 120 projections, 3.8 % of the full 3142 projection series.

reconstruction both the hairs and the zero-regions are better defined. Most of the streaks have also been removed. There seems to be a very slight emphasis of noise in a frequency-range slightly below the highest frequencies.



(a) OWL-QN

(b) L-BFGS-B

(c) Differential image

Figure 27: Comparison of the normal L-BFGS-B implementation and OWL-QN using DCT. The algorithms are used with the half-circle distribution using 3.8 % of the pixels from the full series

9.4.3 Feasibility of keyframing

Too explore the possibility of applying keyframing the difference between the projections in the two real data sets have been compared. Both these data-sets are originally taken with a full set of 3142 projections, and the difference between the projections should therefore be minimal.

The noise-level was approximated by observing that the difference between the two frames were mostly noise. Thus the standard deviation of the noise was approximated as the standard deviation of this differential image. This value set as the noise level, any changes under this limit is assumed to be indistinguishable from the noise. Zeroing all the values below this threshold showed that this was a reasonable estimate, as the remaining data was concentrated around the features in the projections.

Further using this value for the noise-level the signal to noise ratio (SNR) is estimated to be the average signal strength divided by this noise-level. The average signal strength is calculated as the average over the projection.

The max signal used as a reference is estimated from the difference between the maximum and minimum value in the projections and is meant to represent the range actually containing signal, not the full range of the detector.

The first set has an approximated average SNR of 73. Here the top 300 rows of unused pixels have been discarded. This set is less symmetrical than the first, and has a significant part of the sample moving out and in of frame as a consequence of this. There is also a bit more empty space on the sides on the top part of the projection, making maybe around 20 % unused space.

For the second set the approximated average SNR is 53. To better represent the change in the actual object itself the original width of the images have been reduced from 2000 pixels to 1000 pixels around the center. This leaves about 10 to 15 percent of the image as empty space. It also means the ideal number of projections halves from 3142 to 1571.

In figure 28 the root-mean-square between projections with a given number of total projections is illustrated. The lowest data point equals a total of 10 projections. It shows that even when the jumps between the projections are quite big the total intensity change over the projection is quite low. For the full projection series it is around 1 % of the total signal. It keeps low down

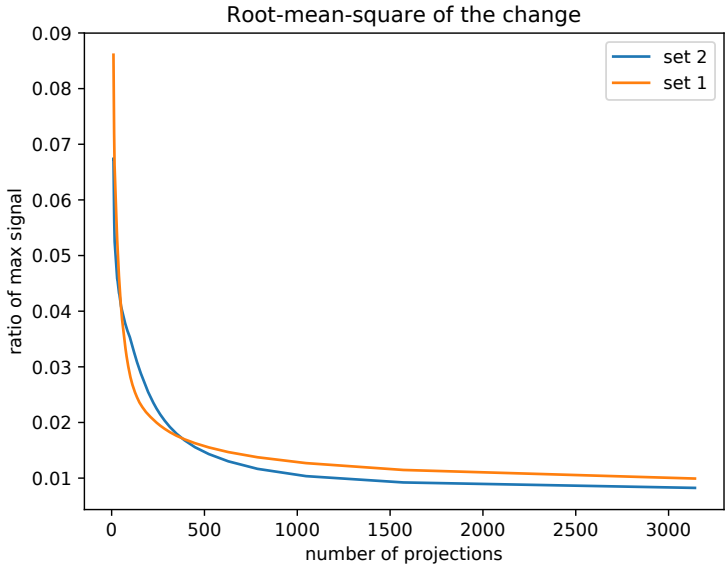


Figure 28

to around 500 projection, where it starts to raise quickly, but even at 10 total projections the change is below 9 %.

Adding the information in the information contained in figure29 a clearer picture is painted. For the full projection series both sets shows around 70 % of the data being insignificant, which is to be expected from the way the noise level is calculated. The fact that this quite slowly and smoothly decreases with a reduced number of projections shows that this was probably a good estimate for the noise level. The amount of data below this noise limit keeps above 50 % down to about 300 projections for data-set-1 and 600 projections for data-set 2. At a 10 projection series the values are down to what is probably the non-changing empty space.

Combining these two graphs one can see that down to about a sixth of a complete projection series the change from projection to projection is relative small, and the relevant change is contained in less than half of the data-points

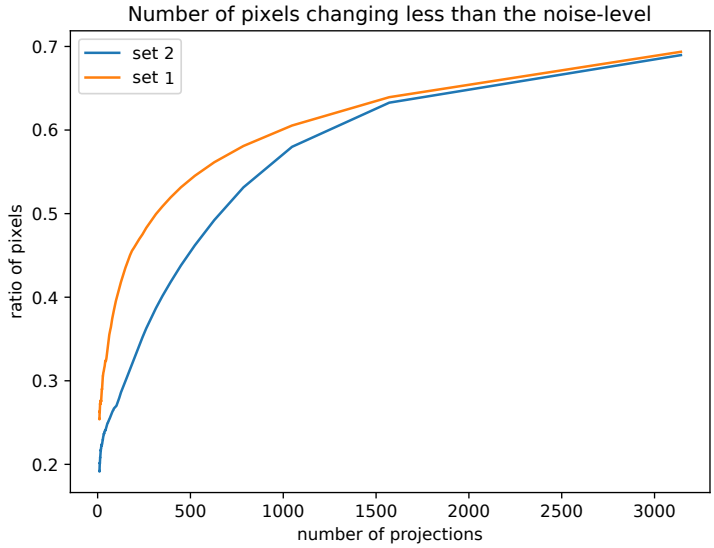


Figure 29: The ratio of pixels changing less than the approximated noise-level

As seen in figure 30 the amount of pixels not changing at all is almost non-existing for data-set 2. For data-set 1 it is actually surprisingly high, the larger empty areas should not be enough to explain this difference. An uneven background intensity observed in the projections of data-set 1 give a hint that this might be caused by some regions being over-exposed. The amount is anyway much less than the pixels under the noise level, which means an effective way to deal with this noise is needed to implement keyframing.

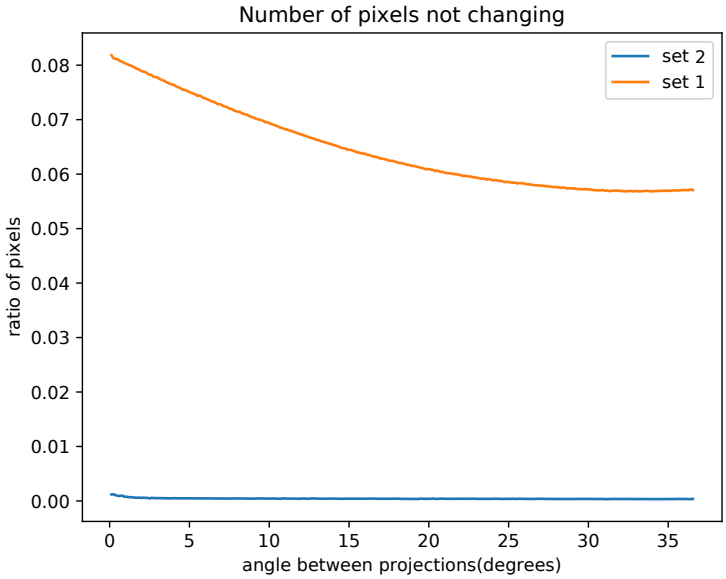
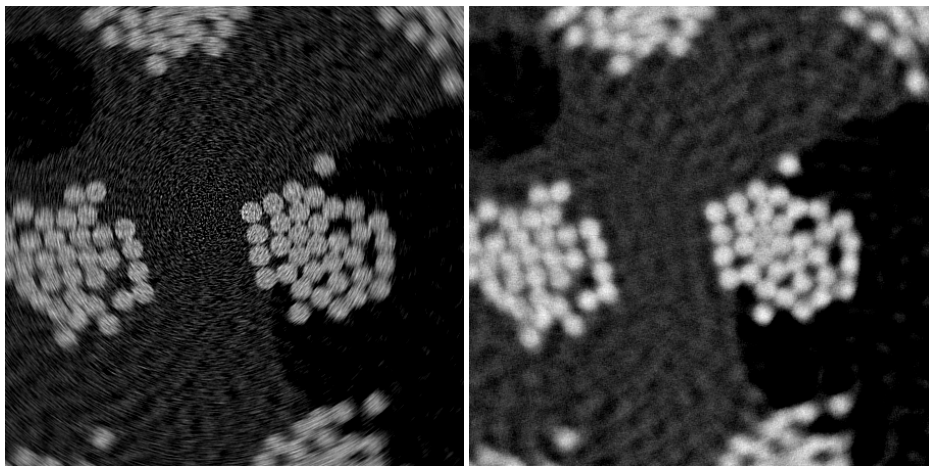


Figure 30: The ratio of pixels not changing at all from projection to projection

9.5 Applying corrected weighting for continuous scanning

Comparison between L-BFGS-B reconstruction using weighting super sampled to 3142 projections and a SIRT reconstruction using normal weighting and 2000 iterations. The projections have an added noise giving a SNR of 71.

At the center, as seen in figure 31, the advantages of the continuous weighting is obvious. It gives a sharper reconstruction and dampens the low frequency artefacts. As you move away from the center the blurring effect comes across as obvious motion blur, where with the normal weighting it just appears blurred. When moving to the edge, as seen in figure 32, the streaks from the motion blur become a dominant feature. This makes the corrected weighting appear worse than the normal weighting at first glance. The corrected weighting still keeps a lot of the artefacts present in the normal weighting, although with less intensity. The advantage of the continuous weighting is that the source of the

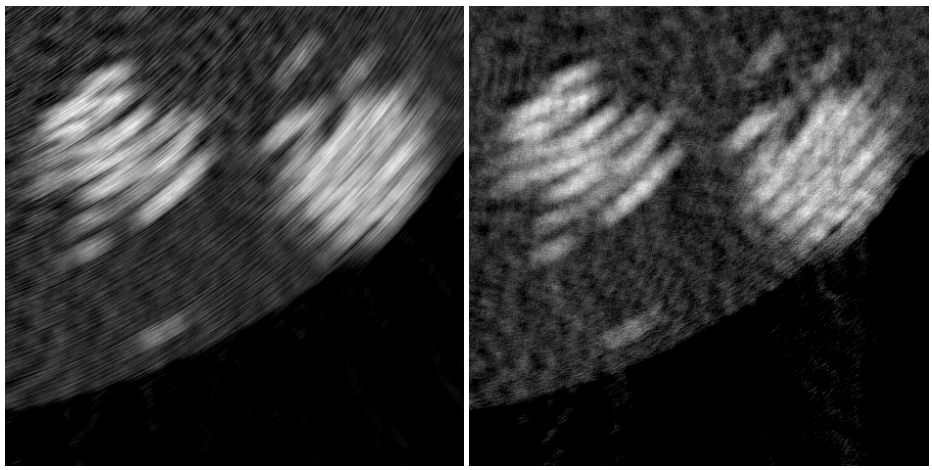


(a) Corrected weighting **(b)** Normal weighting

Figure 31: Difference between continuous adapted weighting and normal weighting in the center

defect is obvious, and could possibly be corrected for more easily.

Attempts to use the corrected weighting with the compressive sensing algorithms, as was the original intention, gave garbage results for some unknown reason that there was no time to test.



(a) Corrected weighting **(b)** Normal weighting

Figure 32: Difference between continuous adapted weighting and normal weighting at the edge

10 Additional comments

It is interesting to observe that the ART-based algorithms struggle to even get close to the minimum they are seeking when using a low number of projections. It would be natural to think that a low number of projections would give a high dimensional freedom, and thus make it easy for the algorithms to get close to a measurable minimum. This appears to not be true, even for ideal systems without noise. It could be specific to the implementation used, but there was not found any clear error to blame. This is a problem for compressive sensing, as it results in a L_1-regulation with low fidelity.

10.1 Other avenues not explored

During the testing done for this paper there were some avenues that were not explored due to lack of time and resources. Some of these are briefly described here.

10.1.1 Inter-slice correspondence

Inter-slice correspondence is another domain were information about the sample can be put to use. This exploits the fact that different slices along the height of the detector, which in conventional algorithms are independent (or semi-independent for the cone-beam), are connected in some way through the properties of the sample. The simplest way to do this is to extend the transformations used for the L_1-regulation to all three dimensions in compressive sensing. This is a logical extension that sadly wasn't tested due to lack of time and problems with the algorithms. This connectedness could possibly also be implemented with some kind of continuity-condition similar to those applied in some segmentation algorithms.

10.1.2 Uneven loss-function

Taking what we learned from the random point distributions, and that the algorithms have a hard time reducing the loss function for low numbers of projections, it would seem that given different pixels different weights in the loss function could maybe have some merit. Basically saying that we care more about how the central pixels change than the edge pixels. This could also have an adverse effect, but would be interesting to test.

For compressive sensing an uneven loss-function could also be useful. For the DCT the components at certain frequencies could be prioritized and for the DWT a multilevel implementation could possible get rid of, or at least limit, the gridding effect by giving deeper levels less weight. This would however need an algorithm that could handle uneven weights in the L_1-regulation, which is not necessarily trivial

10.1.3 Compressive sensing on reduced sinogram

Using compressive sensing on the sinogram itself is usually not a productive endeavour. The regular nature of the data points causes severe artefacts. But using a random point distribution the data would be more appropriate for such an endeavour. This would likely produce worse results than just using normal reconstruction, as the best way to represent the sinogram in a sparse way is

probably through the radon-transform. It would non the less be an interesting avenue to explore.

11 Conclusion

It seems L-BFGS-B with a SIRT gradient could easily be used as a faster converging equivalent of a SIRT reconstruction.

Putting the reference points on a Fourier-grid for compressive sensing had poor results when using L_1-regulation in real space and the DCT. This is probably because these transforms create a less sparse representation than the Fourier-grid on its own. The Fourier-grid might have potential in compressive sensing if a suitable transform for L_1-regulation is found..

Doing L_1-regulation in real space with OWL-QN seems to work quite well, giving decent results with a sparse sample.

Doing L_1-regulation using the DWT with the Haar wavelet through OWL-QN gave underwhelming results. There was a slight improvement in the differentiation between different segments, but a lot of artefacts where introduced into the reconstruction.

It appears that choosing a suitable value for C makes L_1-regulation in the DCT-domain efficient at removing high frequency noise, but in this implementation it has offered little in improved edge definition or reduction of other frequency defects in the image. To improve on these results I would start by getting a proper implementation of a combination of L-BFGS-B and OWL-QN so a better search directions can be found. Exploring other gradient-approximations than the SIRT-gradient could also give better results.

Using a random point distribution to reduce data throughput reduction exchanges low frequency errors in the reconstruction for high frequency noise, which is easier to deal with afterwards. Using a half-circle shaped distribution proved better than a uniform-distribution. Doing a DCT based L_1-regulation further improved upon these results, showing great promise.

Keyframing a projection series seem to be promising for projection series down to about a sixth of a full projection series. Below this it is uncertain if the change will be small enough from projection to projection to justify keyframing. This is

given that a good strategy for dealing with the noise is found. Dithering would be a natural thing to try here.

Initial tests with corrected weighting for continuous scanning show promise for it being usable with some kind of regulation penalizing the motion blur.

References

[1] Eberhard Haug & Werner Nakel (2004). The elementary process of bremsstrahlung. River Edge NJ: World Scientific. p. Scientific lecture notes in physics, vol. 73. ISBN 981-238-578-9.

[2] S.D. Poisson, Probabilité des jugements en matière criminelle et en matière civile, précédées des règles générales du calcul des probabilitiés (Paris, France: Bachelier, 1837), page 206.

[3] Casella, George; Berger, Roger L. (2001). Statistical Inference (2nd ed.). Duxbury. ISBN 0-534-24312-6.

[4] Dirac, Paul (1958), The Principles of Quantum Mechanics (4th ed.) §15 The δ function, p. 58, Oxford at the Clarendon Press, ISBN 978-0-19-852011-5.

[5] Abir, Muhammad Imran Khan, "Iterative CT reconstruction from few projections for the nondestructive post irradiation examination of nuclear fuel assemblies" (2015). Doctoral Dissertations. 2400. `http://scholarsmine.mst.edu/doctoral_dissertations/2400`

[6] Turoňová, B. , Simultaneous Algebraic Reconstruction Technique for Electron Tomography using OpenCL. Master's thesis, Saarland University, 2011. `https://graphics.cg.uni-saarland.de/fileadmin/cguds/papers/2011/turonova_mt2011/MasterThesis.pdf`

[7] Kak, A. C. and Slaney, M. Principles of Computerized Tomographic Imaging. IEEE Press, 1988.

[8] Gordon, R. , Bender, R. , Herman, G. T. . Algebraic Reconstruction Techniques (ART) for three-dimensional electron microscopy and X-ray photography. Journal of Theoretical Biology, 29(3):471 – 481, 1970.

[9] Mueller, K. , Fast and accurate three-dimensional reconstrucion from conebeam projection data using algebraic methods. Master's thesis, School of The Ohio State University, 1998.

[10] Fletcher, Roger (1987), Practical methods of optimization (2nd ed.), New York: John Wiley & Sons, ISBN 978-0-471-91547-8

[11] Pryor, Jr., A., Yang, Y., Rana, A., Gallagher-Jones, M., Zhou, J., Lo, Y.H., Melinte, G., Chiu, W., Rodriguez, J.A., Miao, J.: GENFIRE: A generalized Fourier iterative reconstruction algorithm for high-resolution 3D imaging. Sci. Rep. 7, (2017) free-access version: `https://arxiv.org/abs/1706.04309`

[12] Yang, Y., Chen, C.-C., Scott, M., Ophus, C., Xu, R., *Pryor, Jr.*, A., Wu, L., Sun, F, Theis, W., Zhou, J., et al.: Deciphering chemical order/disorder and material properties at the single-atom level. Nature 542 (7639), 75–79 (2017)

[13] Smith, Steven W. (1999). "Chapter 8: The Discrete Fourier Transform". The Scientist and Engineer's Guide to Digital Signal Processing (Second ed.). San Diego, Calif.: California Technical Publishing. ISBN 0-9660176-3-3.

[14] Brigham, E. Oran (1988). The fast Fourier transform and its applications. Englewood Cliffs, N.J.: Prentice Hall. ISBN 0-13-307505-2

[15] W. van Aarle, W. J. Palenstijn, J. Cant, E. Janssens, F. Bleichrodt, A. Dabravolski, J. De Beenhouwer, K. J. Batenburg, and J. Sijbers, "Fast and Flexible X-ray Tomography Using the ASTRA Toolbox", Optics Express, 24(22), 25129-25147, (2016), `http://dx.doi.org/10.1364/OE.24.025129`

[16] W. van Aarle, W. J. Palenstijn, J. De Beenhouwer, T. Altantzis, S. Bals, K. J. Batenburg, and J. Sijbers, "The ASTRA Toolbox: A platform for advanced algorithm development in electron tomography", Ultramicroscopy, 157, 35–47, (2015), `http://dx.doi.org/10.1016/j.ultramic.2015.05.002`

[17] W. J. Palenstijn, K J. Batenburg, and J. Sijbers, "Performance improvements for iterative electron tomography reconstruction using graphics processing units (GPUs)", Journal of Structural Biology, vol. 176, issue 2, pp. 250-253, 2011, `http://dx.doi.org/10.1016/j.jsb.2011.07.017`

[18] Gürsoy D, De Carlo F, Xiao X, and Jacobsen C. Tomopy: a framework for the analysis of synchrotron tomographic data. Journal of Synchrotron Radiation, 21(5):1188–1193, 2014.

[19] Okazaki, N. (2018). libLBFGS: L-BFGS library written in C. [online] Chokkan.org. Available at: http://www.chokkan.org/software/liblbfgs/ [Accessed 23 Jun. 2018].

[20] Pelt D, Gürsoy D, Palenstijn WJ, Sijbers J, De Carlo F, and Batenburg KJ. Integration of tomopy and the astra toolbox for advanced processing and reconstruction of tomographic synchrotron data. Journal of Synchrotron Radiation, 23(3):842–849, 2016.

[21] Pryor, Jr. GENFIRE-Python 1.1.11 (2017), Avaliable at `https://github.com/genfire-em/GENFIRE-Python`

[22] Stéfan van der Walt, S. Chris Colbert and Gaël Varoquaux. The NumPy Array: A Structure for Efficient Numerical Computation, Computing in Science & Engineering, 13, 22-30 (2011), DOI:10.1109/MCSE.2011.37

[23] Python Software Foundation. Python Language Reference, version 3.6. Available at `http://www.python.org`

[24] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu and the scikit-image contributors. scikit-image: Image processing in Python, PeerJ 2:e453 (2014)

[25] John D. Hunter. Matplotlib: A 2D Graphics Environment, Computing in Science & Engineering, 9, 90-95 (2007), DOI:10.1109/MCSE.2007.55

[26] Hans Bornefalk and Mats Danielsson. Photon-counting spectral computed tomography using silicon strip detectors: a feasibility study. Physics in Medicine & Biology, 13, 7 (2010) `http://stacks.iop.org/0031-9155/55/i=7/a=014`

A List of python packages with version numbers

In this project python 3.6.3 together with the following packages and versions are used

Reconstructions:

- astra-toolbox 1.8.3
- tomopy 1.1.2
- genfire 1.1.11
- pyLBFGS 0.2.0.8
- scipy 1.0.1

Calculations:

- numpy 1.13.3
- scikit-image 0.13.1
- scikit-learn 0.19.1

Data import and management:

- pillow 5.1.0
- tables 3.4.3

Plus matplotlib 2.2.2 for data visualization.