Improving cryo-ET reconstructions of ER-associated ribosomes with tomographic reconstruction methods and deep learning

Richard Schoonhoven

a thesis submitted to the Department of Mathematics and the Department of Information and Computing Science at Utrecht University in partial fulfillment of the requirements for the degree of

Masters in Mathematics and Masters in Computing Science

Supervisor:	Dr. Tristan van Leeuwen
Second reader:	Prof. dr. Remco Veltkamp
Daily supervisor:	Dr. Daniël Pelt

02-08-2019

Universiteit Utrecht

Acknowledgments

Most importantly I would like to thank dr. Tristan van Leeuwen and dr. Daniël Pelt whose supervision was crucial for me during the writing of this thesis. Tristan introduced me to the topic, helped me get setup at the CWI and guided me throughout the year. Daan was always in good spirits and never failed to make time for me and help me with my mistakes. He gladly provided advice at each step and I could always drop by unannounced and ask for help. Secondly, I would like to thank prof. dr. Remco Veltkamp for taking me on as his thesis student since it allowed me to find a topic for both degrees which proved difficult at first. Furthermore, I would like to thank prof. dr. Joost Batenburg for having me as an intern in his research group. I also would like to thank prof. dr. Friedrich Förster and his PhD student Gijs van der Schot for providing the topic in the first place and for taking the time on several occasions to help me out with the cryo-ET data. Lastly, I would like to thank everybody in the computational imaging group at the CWI for being very welcoming and kind and for giving me a fun year in the group.

CONTENTS

1 Introduction

Computerized imaging has proved to be a useful tool in biology and biomedical research for decades. It has contributed to a better understanding of the human anatomy and the inner workings of our bodies by enabling researchers and doctors access to the interior of our bodies without invasive action. Clever algorithms and technology has been developed to image biological structures in a wide range of sizes. Consequently, specialized techniques have to be developed which are suitably adapted to the particular length scale and problem definition.

Tomography is the umbrella term that covers the study of imaging by sectioning by using a penetrating wave and it is used in diverse areas such as computed tomography (CT) and magnetic resonance imaging (MRI). The aforementioned methods are the focus of intensive study but are limited in their maximally achievable resolution. For imaging on the subnanometer length scale cryo-electron microscopy (cryo-EM) has been developed. This method of microscopy (not to be confused with its 3D counterpart cryo-ET which we will introduce shortly) allows for the visualization of individual cell organelles and the molecular machines that drive our bodies. This knowledge has yielded insight into pathology and the molecular structures of pathogens.

1.1 A brief overview of cryo-electron tomography and challenges

Cryo-EM was originally invented in the 1930s by Ernst Ruska, which attributed cryo-EM its first Nobel prize in Physics. In 2017, Jacques Dubochet, Joachim Frank and Richard Henderson received the Nobel prize in Chemistry for their contributions to cryo-electron microscopy and it has been an influential and active research area for the past decades [9]. Famously, during the 2016 Zika virus epidemic, researchers managed to image the virus to 3.8 Å resolution using cryo-EM [42] (see Figure 1).

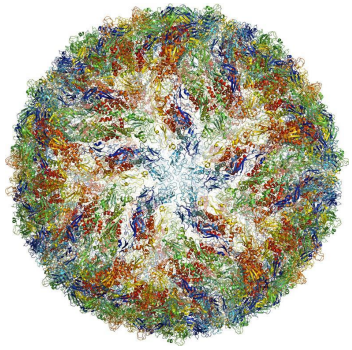


Figure 1: A model of the Zika virus which was created using cryo-EM [19].

Using an electron microscope for biomedical imaging can be challenging since it requires a high vacuum in order to work to avoid electron scattering by gas molecules. The vacuum subsequently dries out the samples and, to compound the problem, the electron beam has such a high energy that it destroys covalent bonds. As a result the imaging process damages the object you aim to image as is illustrated in Figure 2.

On the other hand, a lower electron dose leads to a lower contrast. Therefore, it is not possible to increase the number of electrons up to a dose that creates a suitable contrast in the image.

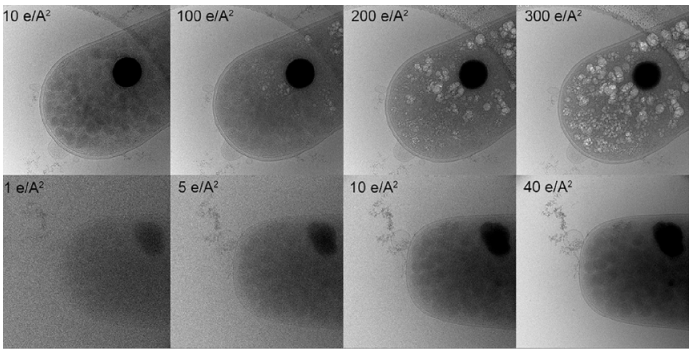


Figure 2: Cell radiation damage caused by various electron doses [18].

Furthermore, the samples need to be cryogenically fixed by a process called vitrification to avoid large changes in the biological tissue during scanning. Vitrification can roughly be split into two processes depending on the sample thickness; plunge freezing and high-pressure freezing. The exact details of these processes are omitted here but an overview can be found in [18]. Whatever process is used, the overarching goal is to freeze the sample faster than the rate of crystallization of water. However, artifacts from the vitrification process can still be present in the sample regardless.

In contrast to electron microscopy, which produces a two-dimensional image of the object, electron cryo-*tomography* (cryo-ET) [10], [24] produces a three dimensional reconstruction in fundamentally the same manner as other tomographic methods. A volume of material is imaged from multiple angles, i.e. we measure the intensity of some incoming beam on a 2D detector after the beam has passed through the material (see Figure 3). We use this *projection data* for every angle to compute a volume that best replicates the original volume. The theory and mathematics of tomographic reconstruction methods are outlined in section 2. In the case of cryo-ET, the beam is made up of electrons and the object is a frozen piece of biological material.

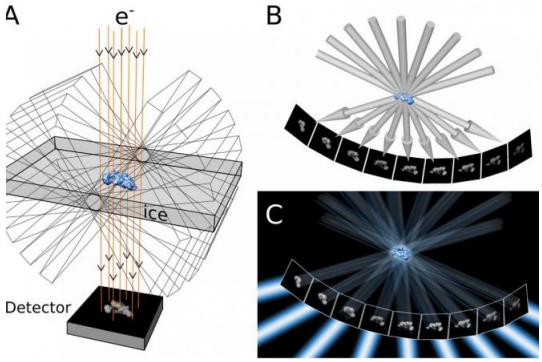


Figure 3: This diagram shows how the sample is rotated to change the incident angle of the electron beam. Each step produces a projection image on the detector corresponding to a different angle [18].

In tomography, we image the same patch several times so the electron dose needs to be distributed over the number of images that we take. Therefore, each projection has a much lower contrast than a single cryo-EM image and the images are extremely noisy. Nevertheless, cryo-ET

can achieve a resolution in the order of ∼10 Å or less which means we can image structures like proteins.

As can be seen in Figure 3, the sample is rotated rather than the beam in a flat holder. Therefore due to physical constraints, we cannot image over the full 180° angular range since the holder will collide with the source. Typically, the projections are taken in a range of ±60° with a 2° separation. This results in so called *missing wedge artifacts* due to the missing information. These artifacts appear as streaking lines in the reconstruction and make the reconstruction process more challenging.

1.2 Specifics of cryo-ET setting in this thesis

In this thesis we aim to build upon a dataset of ER-associated ribosomes created by prof. Friedrich Förster, chair of the Cryo-EM group at Utrecht University. Prof. Förster has published several notable papers on cryo-ET reconstructions of biological structures but most notably has developed a *subtomogram averaging* method [12], [49] in cryo-ET. In subtomogram averaging, the researcher utilizes the advantage of having several (assumed to be) identical particles in a single volume by creating an average of all these identical particles. The details will be discussed in section 3.2 but roughly an algorithm tries to estimate the position and orientation of each particle in the scan an then rotates and averages them to compute the final average.

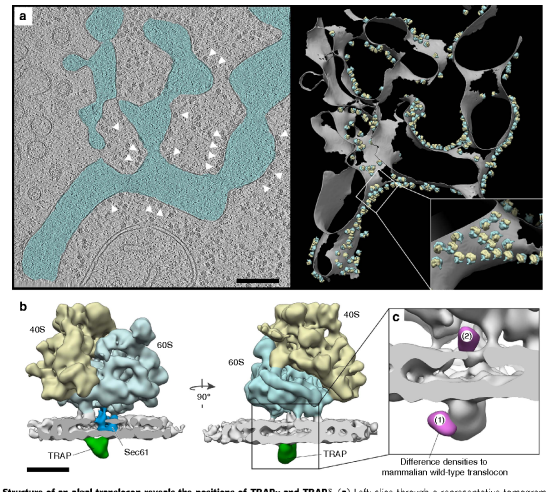
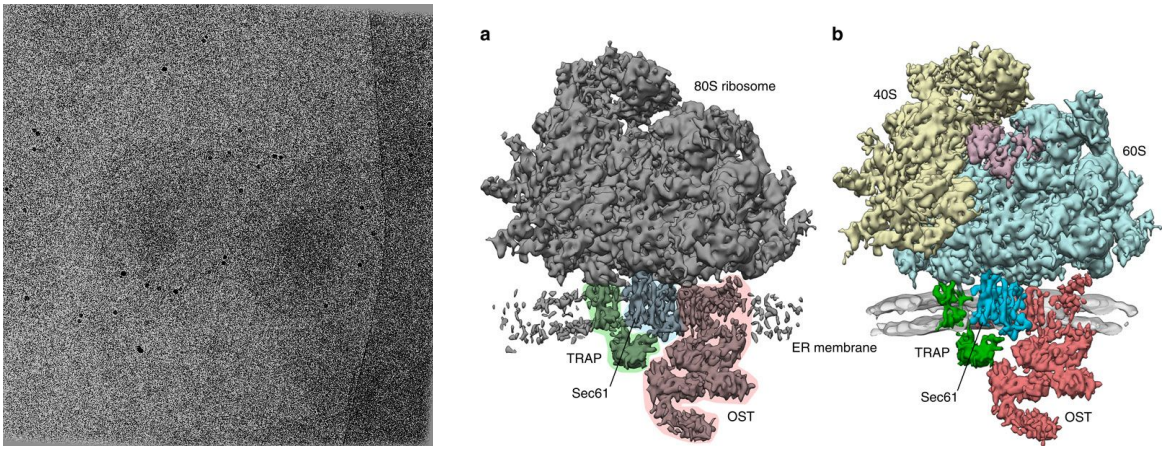


Figure 4: Ribosomes distributed over the membrane of the ER [37].

The *endoplasmic reticulum* (ER) is a cell organelle which produces many of the proteins in our body. The ER is a roughly folded network of membranes which are littered with *ribosomes*. Prof. Förster has published several papers [35], [36], [37] where such a ribosome is imaged using cryo-ET. In Figure 4 we see how a cryo-ET reconstruction of a section of the ER contains several copies of the ribosomes inside the folded membrane.

As mentioned earlier, cryo-ET projections are quite noisy which makes it challenging for researchers to compute a suitably clear reconstruction. One projection image of a tomogram is shown in Figure 5 (a) and one full tomogram can contain up to 200 ribosomes. In total the researchers were able to extract 17000 possible ribosome volumes from the dataset. The sheer

amount of data that needs to be processed to compute the final ribosome reconstruction creates another challenge for researchers to cope with. The ribosome reconstruction that was ultimately obtained by subtomogram averaging (see section 3.2) can be seen in Figure 5 (b).



(a) **(b)**

Figure 5: (a) One of the projections from the dataset, (b) ER-associated ribosome reconstruction with various segments highlighted [35].

1.3 Research goals and contributions

In this thesis we aim to investigate the performance of contemporary tomographic algorithms (see section 2) and mixed-scale dense neural networks (see section 5.4) in cryo-ET. We base our experiments on the aforementioned dataset of ER-associated ribosomes (see Figure 5 (b)) created by prof. Friedrich Förster, chair of the Cryo-EM group at Utrecht University. Specifically we will test FBP, SIRT, SIRTMIN and TV-FISTA on a simulated cryo-ET dataset with extreme noise profiles. Subsequently we will test how the methods interplay with the *subtomogram averaging* method in cryo-ET, which we will outline in section 3.2.

Both the research goals and the approaches we use to study them in this thesis will be two-pronged. The first research goal is *to investigate how the reconstruction quality for various tomographic reconstruction algorithms, including advanced iterative methods, plays out for different cryo-ET scenarios, with and without subtomogram averaging.*

In the experiments concerning this goal we will consider the effects of the limited angular range of the projections that cryo-ET tomograms suffer from in addition to the influence of the extreme noise level. Furthermore, we have developed a new ensemble-reconstruction method that, using the flexibility of the ASTRA toolbox [29], incorporates all the separate scans in identical particle analysis into one reconstruction. We will juxtapose the ensemble method with the standard subtomogram averaging approach.

Secondly, we will attempt *to determine whether a reconstruction of a desirable quality can be obtained with less data by using deep learning, specifically a recently developed architecture called MS-D networks, to denoise the reconstructed volumes.* The aim is to boost the quality of the

individual subtomograms or the result of a smaller average, given that a suitable best average tomogram can be created prior to the deep learning phase. Obviously, the two research questions cannot be fully split; if we can acquire a higher quality reconstruction, we probably need less data to construct an average of a certain quality. However, we expect that neural networks will be well-suited to reduce the data requirements and the tomographic reconstruction methods will be better suited to boost the final quality.

The contributions of this thesis are:

- We study the behaviour of the aforementioned tomographic reconstruction algorithms for several noise profiles that are relevant in the cryo-ET setting and recommend certain advanced iterative methods to use in conjunction with subtomogram averaging. Furthermore, the influence of the missing wedge artifacts seems marginal on the advanced iterative methods, particularly in conjunction with subtomogram averaging.
- We study the applicability of deep learning when it comes to data reduction applications and we show that the quality of a small averaged reconstruction can be boosted by a trained neural network.

1.4 Outline of the thesis

We have split the thesis into two parts where we roughly address each research goal in one part. In the first part we will introduce four tomographic reconstruction methods that we will study in a mathematical framework in section 2. Next, in section 3 we provide details of the algorithms and procedures that we have implemented for the experiments. We also specify how we created a simulated dataset that most of the experiments in this thesis were performed on. Afterwards, we will discuss all the experimental results we obtained for these tomographic algorithms in section 4 which rounds off the first part of the thesis.

Subsequently we will build a mathematical treatise of neural networks from the ground up in section 5 in the second part. For the reader who is familiar with neural networks but unfamiliar with MS-D networks we recommend reading only section 5.4. Lastly, we report on the results of our MS-D network experiments in section 6 and report our findings in section 7. Some recommendations for future work and follow up experiments are mentioned in section 8.

Lastly, in addition to the aforementioned practical goals, this thesis aims to introduce many advanced concepts from tomography and deep learning to master students. All theory will be treated from a mathematical point of view with this audience in mind. Consequently we will not introduce theory concerning elementary calculus, Fourier transformations, linear algebra and matrix calculus. We will introduce the tomographic methods that we will apply in this thesis and we will spend a significant amount of time deriving a suitable mathematical definition for neural networks and we will derive the learning rules. For the reader who is well-versed in these topics, it is possible to skip sections 2 and 5 at their leisure since we have attempted to stick to conventional terminology and notation.

While introducing the necessary preliminary knowledge, we will encounter a variety of advanced mathematical concepts that are out of scope of this thesis. A great deal of such theory will be treated in the Appendix and we will refer to the relevant sections frequently. However, this thesis is not a fully stand-alone compendium of tomography and deep learning theory and we

will refer to further reading material if necessary. Nevertheless, it is a personal goal to properly motivate all the facets of the mathematics so that the underlying considerations are clear to the reader.

CHAPTER 1

TOMOGRAPHIC RECONSTRUCTION METHODS IN CRYO-ET

2 Inverse problems and tomography

In this section we will develop the theory that is required to understand the tomographic reconstruction methods employed in the experiments in section 4. To start, tomographic reconstruction problems fall in the domain of inverse problems. Inverse problems can be considered problems where we want to estimate certain quantities given indirect measurements of these quantities. The field is of great importance due to its wide range of applications in for example biomedical research [3] and geology [5]. In this chapter we will briefly cover the basics of inverse problems such that we understand the difficulties that we may encounter in CT.

Suppose we model our inverse problem as follows

$$Au = f, \quad A : X \to Y \tag{2.1}$$

Here A is an operator (not necessarily linear) and X an Y are Banach spaces. The vector f takes the role of our data or measurement and the vector u is the quantity of interest we are trying to obtain. An inverse problem is *well-posed* if there exists a solution to 2.1, the solution is unique and it is stable. In mathematical terms that means that A is a *homeomorphism* of X onto Y, i.e. A is a bijection and both A and A^{-1} are continuous. If a problem is not well-posed, it is by definition *ill-posed*. Many practical inverse problems are ill-posed.

Example: A canonical inverse problem is to invert the integration of a square integrable function

$$Au := \int_a^x u(t)\, dt = f(x).$$

Assume that X and Y are $\mathcal{L}^2(a, b)$, i.e. they contain functions such that $\int_a^b |g(x)|^2\, dx < \infty$. The problem remains ill-posed since there are functions in $\mathcal{L}^2[a, b]$ whose derivative is not square integrable on $[a, b]$. Hence, there exist functions $f' \in Y$ that do not lie in the range of A so A is not a homeomorphism.

In this thesis we deal with computed tomography (CT) and we will dedicate the rest of this section to describing CT as an inverse problem. Before we can mathematically discuss tomography, we need to formulate imaging problems in mathematical terms. We can consider images from two points of view; we can see them as a discrete set of pixels $U \subset \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_k}$ or as an idealized continuous image $u : \Omega \to \mathbb{R}$, with $\Omega \subset \mathbb{R}^d$ is the d-dimensional domain of the image.

We can transform from the continuous case to the discrete case by overlaying the domain Ω with a regularly spaced grid. We can then for example take the maximum or average value in that pixel as entry for our array. Alternatively, we can define a continuous image from a discrete one by taking u to be the pixel value in the area of the pixel. Note that this makes for a discontinuous function u.

An image as interpreted in the computerized sense can have several channels to define the colours. In this case, the array would have an extra dimension to incorporate for example the RGB values. However, in this thesis we only consider gray-scale images.

Tomography refers to imaging techniques where information about the interior of an object is retrieved using data that is retrieved outside the object without probing it. The most common

example of such techniques is X-ray *computerized tomography* (CT). However, other forms of tomography exist such as *electron tomography (ET)* [10], [24], which is the main focus of this thesis, *thermoacoustic CT* [21] or *Schlieren tomography* [38].

2.1 Tomographic reconstruction algorithms

In our treatise of the mathematics involved in tomography it is not relevant to distinguish between X-ray CT and ET. In both cases, an object of interest is placed between source and a detector. A beam of particles is emitted from the source with intensity I_0, interacts with the material and we measure intensity I_1 at the detector. The scattering and absorption effects are different but these effects are summarized in the attenuation function u.

Without considering the underlying physics, the so called attenuation coefficient $u(x)$ is the quantity we want to measure. Roughly speaking, the attenuation coefficient describes how hard it is for the beam to penetrate the material. In the case of a parallel beam geometry, a CT scan can be regarded as a 2D slice by slice problem. We model the cryo-ET problem as a parallel beam geometry so we limit ourselves to the case where the rays are situated in a particular plane. Other scanning geometries exist such as cone beam but they are outside of the scope of this thesis.

For an arbitrary ray $L \subset \mathbb{R}^2$ the intensity I of the beam at position (x, y) on L is proportional to $u(x, y)$ according to the *Lambert-Beer* law

$$I(x,y) = I_0 \exp\left(- \int_{L|_{(x,y)}} u(x', y')\, dx' dy' \right). \tag{2.2}$$

Suppose we measure I_1 at the detector, then we can measure the value of the path integral

$$p_L = \int_L u(x', y')\, dx' dy' = -\ln\left(\frac{I_1}{I_0}\right).$$

We call p_L the total attenuation of ray L. Assume that the coordinate axes are oriented in the plane of the ray (see Figure 6). Suppose that a set of parallel rays is emitted onto the object at *projection angle* θ. Assume that the central ray passes through the origin of the coordinate system. Note that the detector is oriented perpendicular to the rays.

Let $(x, y) \in \mathbb{R}^2$, then the length of the vector perpendicular to the central ray towards (x, y) is $r = x\cos(\theta) + y\sin(\theta)$. The point (x, y) will be detected at a distance r from where the central ray hits the detector. Therefore, the equation for p_L can be rewritten as

$$p_\theta(r) = \int_{\mathbb{R}^2} u(x,y)\delta(x\cos(\theta) + y\sin(\theta) - r)\, dx\, dy. \tag{2.3}$$

Here $\delta(\cdot)$ is the Dirac delta function which means that we add the attenuation $u(x, y)$ to the total attenuation at distance r when (x, y) is projected onto the detector at that distance.

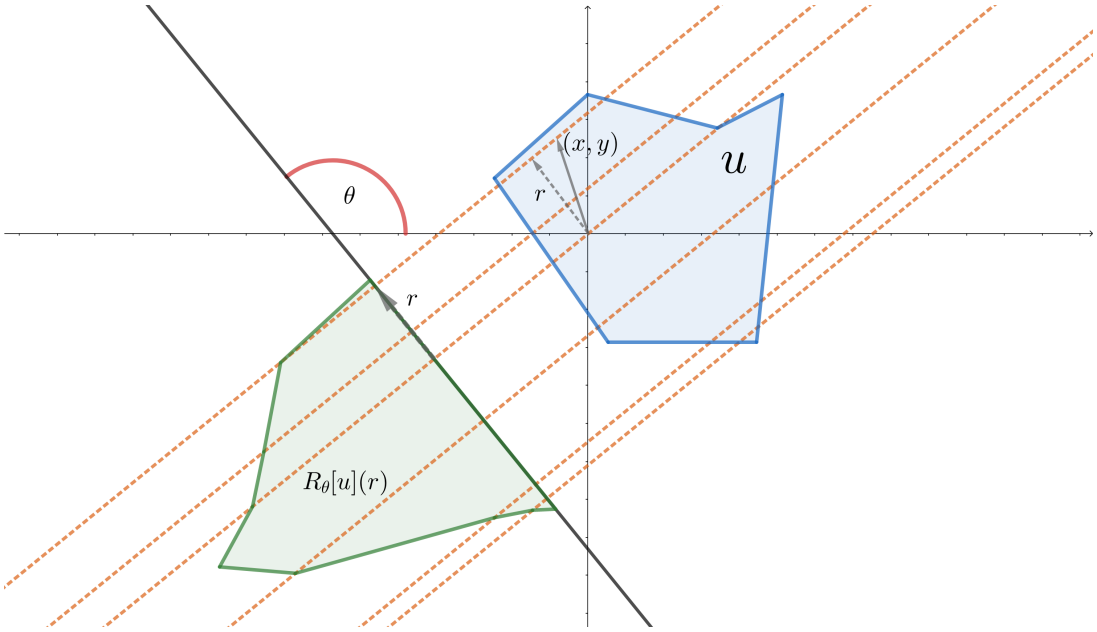


Figure 6: Diagram of polygon $u(x, y)$ and several parallel beams passing through it in the plane of the slice. Here θ is the projection angle and r is the distance at which the point (x, y) will be measured from the center of the detector. $\mathcal{R}_\theta[u]$ is thee Radon transform as defined in equation 2.5.

2.2 Radon transform

Equation 2.3 defines a map between a function on $\mathbb{R}^2$ and the space of straight lines L in $\mathbb{R}^2$. This function is called the *Radon transform*. Let $u : \mathbb{R}^2 \to \mathbb{R}$ be compactly supported and continuous, the Radon transform is defined by the line integral

$$\mathcal{R}u : \mathcal{L}(\mathbb{R}^2) \to \mathbb{R},\ L \mapsto \int_L u(\mathbf{x})\ |d\mathbf{x}|. \tag{2.4}$$

Recall that any point on these straight lines can be parameterized by

$$(x(t), y(t)) = (r\cos(\theta) + t\sin(\theta),\ r\sin(\theta) - t\cos(\theta)).$$

Here r and θ are coordinates such as defined in the previous section. Therefore we can rewrite the Radon transform in these coordinates

$$\mathcal{R}u(r, \theta) := \int_{-\infty}^{\infty} u(r\cos(\theta) + t\sin(\theta),\ r\sin(\theta) - t\cos(\theta))\ dt \tag{2.5}$$

Since the angle θ is fixed for one measurement if we consider this in practice, we introduce the following notation $\mathcal{R}_\theta[u](r) := \mathcal{R}u(r, \theta)$. In practice we obtain measurements $p_\theta(r)$ but to obtain the original function $u(x, y)$ we want to invert the Radon transform. Since the Radon transform expresses points in $\mathbb{R}^2$ as superpositions of sines and cosines, it seems canonical to consider the Fourier transform of the Radon transform. We use the following definitions for the 1D and 2D Fourier transforms

$$\hat{g}(\omega) := \int_{-\infty}^{\infty} g(x)e^{-2\pi i\omega x}\ dx, \quad \hat{h}(\mathbf{w}) := \int_{\mathbb{R}^2} h(\mathbf{x})e^{-2\pi i\mathbf{w}\cdot\mathbf{x}}\ d\mathbf{x}$$

with inversions

$$g(x) := \int_{-\infty}^{\infty} \hat{g}(\omega) e^{2\pi i \omega x} \, d\omega, \quad h(\mathbf{x}) := \int_{\mathbb{R}^2} \hat{h}(\mathbf{w}) e^{2\pi i \mathbf{w}\cdot\mathbf{x}} \, d\mathbf{w}.$$

Theorem 2.1. ***Fourier slice theorem.*** *We have that*

$$\mathcal{F}_{1D}(P_L \circ h(\mathbf{x})) = S_L \circ \mathcal{F}_{2D}(h(\mathbf{x}))$$

where $\mathcal{F}_{1D}$ and $\mathcal{F}_{2D}$ are the 1D and 2D Fourier transforms, P_L is the projection operator that projects h on a line L and S_L is the slice operator that extract a 1D central slice through the origin and parallel to L from a 2D function.

In other words, the Fourier slice theorem states that the 1D Fourier transform of a 2D function projected on a line (creating a 1D function) is equal to the 2D Fourier transform of the 2D function and taking a slice through the origin parallel to the projection line (see Figure 7).

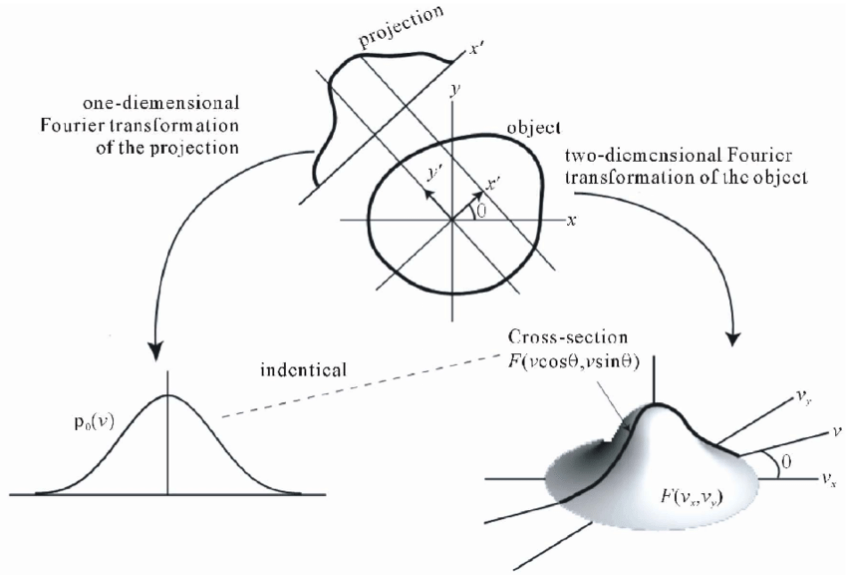


Figure 7: Diagram explaining the Fourier slice theorem from [28]. We have an object in the xy-plane that we call $h(x, y)$ and a projection angle θ. The projection angle defines the perpendicular line L an a different coordinate system x', y'. The left arrow denotes the 1D Fourier transform of the $p_\theta(\nu) := \mathcal{F}_{1D}(P_L \circ h(x, y))$ where ν is the variable in Fourier space. The right arrow denotes the 2D Fourier transform of h. We have that the result of the left operation is equal to the cross section along $\hat{h}(\nu\cos(\theta), \nu\sin(\theta))$ in 2D Fourier space.

In our case, the role of P_L is taken on by the Radon transform and we obtain

$$\widehat{\mathcal{R}}_\theta[u](\omega) = \hat{u}(\omega\cos(\theta), \omega\sin(\theta)).$$

This can be seen by taking both types of Fourier transform. Firstly, the 1D Fourier transform of the Radon transform is

$$\widehat{\mathcal{R}}_\theta[u](\omega) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} u(x(t), y(t)) e^{-2\pi i \omega r} \, dt \, dr.$$

Recall that r was originally defined as $r = x(t)\cos(\theta) + y(t)\sin(\theta)$ so this can be rewritten by going back to the original coordinate system

$$\widehat{\mathcal{R}}_\theta[u](\omega) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} u(x,y)e^{-2\pi i\omega(x\cos(\theta)+y\sin(\theta))}\,dx\,dy.$$

Alternatively, if we take the 2D Fourier transform of u for a radial line at angle θ in Fourier space, we get

$$\hat{u}(\omega\cos(\theta),\omega\sin(\theta)) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} u(x,y)e^{-2\pi i\mathbf{w}\cdot\mathbf{x}}\,d\mathbf{x} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} u(x,y)e^{-2\pi i\omega(x\cos(\theta)+y\sin(\theta))}\,dx\,dy \tag{2.6}$$

and we see that they are indeed equal.

The Radon transform defines the inverse problem for CT. We see that it is a linear and invertible operator but for a perfect reconstruction we require $\mathcal{R}_\theta[u](r)$ to be known for all r and θ. Furthermore, due to discretization and loss of information w.r.t r and θ the reconstructions will not be perfect. Many reconstruction algorithms have been developed to try to combat the lack of knowledge and the effects of noisy data. We will treat several in the following sections.

When reading literature on CT reconstruction with Fourier based methods it can be quite challenging to figure out the details of the method that was used and how it ties into the mathematics. The left- and right-hand sides of the equality of the Fourier slice theorem can be used to categorize two groups of reconstruction methods. We can define the *dual* transform of the Radon transform and reconstruct from the 1D projection data. This is called *filtered back projection*. Alternatively, we can take the 2D Fourier transform of u along a certain radial line and obtain samples in 2D Fourier space as in equation 2.6. Then there exists several (similar) methods to reconstruct our original u from these sample points in Fourier space. These reconstruction methods in the Fourier plane are discussed in Appendix 9.1 since the NFFT originally used to find the ribosomes falls into this category of algorithms. In the next section we will discuss the first type of reconstruction method as it is one of the methods we will be testing.

2.3 Filtered backprojection

Instead of using direct Fourier reconstruction methods, as described in Appendix 9.1, we can use another common reconstruction technique in CT called *Filtered Back Projection*. We can skip using the 2D Fourier transforms and define the dual transform of the Radon transform instead,

$$\mathcal{R}_\theta^*\mathcal{R}_\theta[u](x,y) = \int_{-\infty}^{\infty} \mathcal{R}_\theta[u](r)\;\delta(x\cos(\theta)+y\sin(\theta)-r)\;dr$$

and for the entire function

$$\tilde{u}(x,y) = \mathcal{R}^*\mathcal{R}[u](x,y) = \int_0^\pi \mathcal{R}_\theta^*\mathcal{R}_\theta[u](x,y)\;d\theta = \int_0^\pi\int_{-\infty}^{\infty} \mathcal{R}_\theta[u](r)\delta(x\cos(\theta)+y\sin(\theta)-r)\;dr\;d\theta. \tag{2.7}$$

In image processing the dual transform is often referred to as back projection. If we fix θ and r, we see that the δ function is nonzero for a line of values $\{x, y\}$

$$x = \frac{r - y\sin(\theta)}{\cos(\theta)},\ y \in \mathbb{R} \qquad \text{if } \cos(\theta) \neq 0 \tag{2.8}$$

$$x \in \mathbb{R},\ y = \frac{r}{\sin(\theta)} \qquad \text{if } \cos(\theta) = 0. \tag{2.9}$$

Therefore, we see that the value $\mathcal{R}_\theta[u](r)$ is smeared out along the projection line. Recall that the 1D Fourier transform of the Radon transform is

$$\widehat{\mathcal{R}}_\theta[u](\omega) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} u(x(t), y(t))e^{-2\pi i\omega r}\, dt\, dr.$$

So the inversion is

$$\mathcal{R}_\theta[u](r) = \int_{-\infty}^{\infty} \hat{u}(\omega\cos(\theta), \omega\sin(\theta))\, e^{2\pi i\omega r}\, d\omega.$$

If we combine this with equation 2.7 we see that

$$\tilde{u}(x, y) = \int_0^\pi \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \hat{u}(\omega\cos(\theta), \omega\sin(\theta))\, e^{2\pi i\omega r}\, \delta(x\cos(\theta) + y\sin(\theta) - r)\, d\omega\, dr\, d\theta \tag{2.10}$$

$$= \int_0^\pi \int_{-\infty}^{\infty} \hat{u}(\omega\cos(\theta), \omega\sin(\theta))\, e^{2\pi i\omega(x\cos(\theta)+y\sin(\theta))}\, d\omega\, d\theta. \tag{2.11}$$

By symmetry of the sine and cosine we have $\hat{u}(-\omega\cos(\theta), -\omega\sin(\theta)) = \hat{u}(\omega\cos(\theta + \pi), \omega\sin(\theta + \pi))$. Therefore we get

$$\tilde{u}(x, y) = \int_0^{2\pi} \int_0^{\infty} \frac{1}{\omega}\hat{u}(\omega\cos(\theta), \omega\sin(\theta))\, e^{2\pi i\omega(x\cos(\theta)+y\sin(\theta))}\, \omega d\omega\, d\theta.$$

We have now obtained, by multiplying and dividing by ω, the inverse Fourier transform of $\frac{1}{\omega}\hat{u}(\omega\cos(\theta), \omega\sin(\theta))$ in polar coordinates,

$$\tilde{u}(x, y) = \mathcal{F}^{-1}\left(\frac{1}{\omega}\hat{u}(\omega\cos(\theta), \omega\sin(\theta))\right).$$

The convolution theorem states that

$$\mathcal{F}^{-1}(\hat{f}\cdot\hat{g}) = f * g.$$

Hence we see that back projection gives our original function convolved with a kernel

$$\tilde{u}(x, y) = \mathcal{F}^{-1}\left(\frac{1}{\omega}\hat{u}(\omega\cos(\theta), \omega\sin(\theta))\right) = u(x, y) * \mathcal{F}^{-1}\left(\frac{1}{\omega}\right) = u(x, y) * \frac{1}{r}.$$

The next step is to undo the blurring by the kernel $\frac{1}{r}$. This is the filtering step of filtered back projection. To do so, we multiply by the appropriate kernel in Fourier space

$$\mathcal{R}'_\theta[u](r) = \mathcal{F}^{-1}\left(|\omega|\widehat{\mathcal{R}}_\theta[u](\omega)\right).$$

Back projection then gives us our reconstruction (the absolute value is irrelevant due to the integration domain)

$$\begin{aligned}\tilde{u}_{FBP}(x,y) = \mathcal{R}^*\mathcal{R}'_\theta[u](r) &= \int_0^{2\pi}\int_0^{\infty} \hat{u}(\omega\cos(\theta),\omega\sin(\theta))\, e^{2\pi i\omega(x\cos(\theta)+y\sin(\theta))}\, \omega d\omega\, d\theta \\ &= \mathcal{F}^{-1}(\hat{u}(\omega\cos(\theta),\omega\sin(\theta))) = u(x,y).\end{aligned}$$

Hence we see that FBP analytically gives a perfect reconstruction. The filter $c(r) = \mathcal{F}_{1D}^{-1}(|\omega|)$ is called the *Ramp filter* but it does not exist analytically. In practice, we use some altered filter

$$\tilde{c}(r) = \mathcal{F}_{1D}^{-1}(W(\omega)|\omega|).$$

For example, we can use a function W that approximates the ideal ramp filter in some limit

$$W(\omega) = e^{-\epsilon|\omega|} \quad \Rightarrow \quad \lim_{\epsilon\to 0}\ \mathcal{F}_{1D}^{-1}(e^{-\epsilon|\omega|}|\omega|) = c(r).$$

Another alternative is to use a window function

$$W(\omega) = \text{rect}\left(\frac{\omega}{2\omega_0}\right).$$

Here the rectangle function $\text{rect}(x)$ is 1 if $-1/2 \le x \le 1/2$, i.e. it is 0 if $-\omega_0 \le \omega \le \omega_0$ for some cut-off ω_0. As a last example, the *Hamming ramp filter* is defined by

$$W(\omega) = e^{-\pi\frac{\omega^2}{\omega_0^2}}.$$

Next we will describe a different class of iterative algorithms that will be considered from the discrete viewpoint.

2.4 Iterative schemes and discretization

Thus far we have considered the problem from a continuous viewpoint. To derive the next few algorithms we transition to the practical discrete viewpoint.

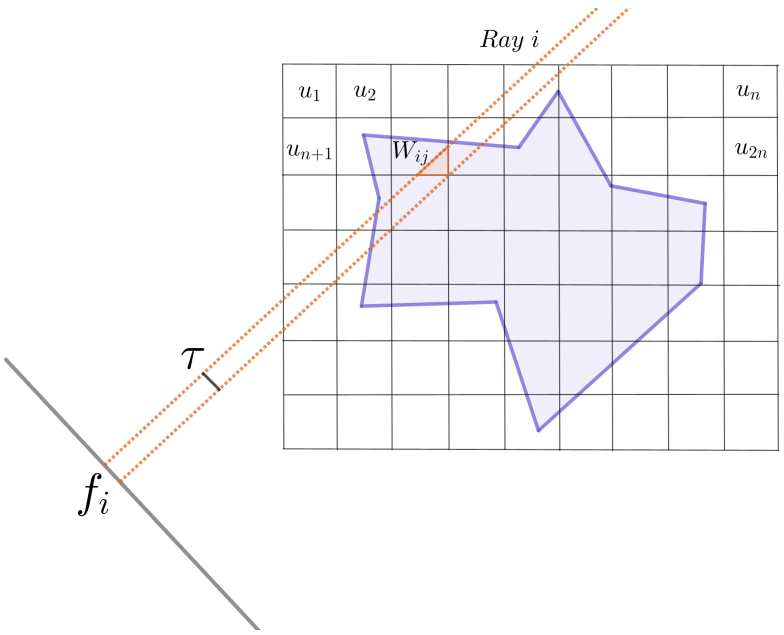


Figure 8: Diagram of discretized slice of some polygon where the attenuation values are modeled in a $n \times m$ grid in a vector $\mathbf{u} \in \mathbb{R}^{nm}$ and W_{ij} denotes the fraction of cell u_j covered by ray i. The width of the ray is τ and f_i denotes the ray sum of the i-th ray.

In this section we will consider the case where $\mathbf{u}$ is a vector that describes a 2-dimensional image pixel-wise and we assume that the value of $\mathbf{u}$ is constant within one pixel. In Figure 8 these pixels values are enumerated as u_j for $j = 1, \ldots, nm = N$ (assume an image of size $m \times n$). Again we take $\mathbf{f}$ to be the measured data and $\mathbf{u} \in \mathbb{R}^{mn}$ to be the vector we want to determine.

We model a ray as a straight beam of width τ. For example, in Figure 8 the i-th ray is illustrated. Let f_i be the *ray-sum* or line integral of the values u_i along ray i. A ray often only partially covers a pixel and we will denote the fractional area covered by ray i in cell j by W_{ij}. Let there be M rays, then the relationship between u_j and f_i is given by

$$\sum_{j=1}^{nm} W_{ij} u_j = f_i, \qquad i = 1, \ldots M. \tag{2.12}$$

Note that the matrix that describes this relationship is sparse for realistic τ since most W_{ij} will be 0. For small M and N we could invert the resulting system of equations but in practice N can be in the millions (and M as well) so this is unrealistic for practical purposes.

2.5 Kaczmarz method and ART

From a linear algebra perspective, for each $i = 1, \ldots, M$ equation 2.12 defines a hyperplane (dimension $nm - 1$). We can find the image $\mathbf{u}$ in the intersection of all such hyperplanes. If we have a well-posed problem, the unique solution is equal to this intersection and is a single point in $\mathbb{R}^{nm}$. This geometric interpretation led to the first algebraic method we will discuss here.

Kaczmarz method [15] involves picking a point on a hyperplane and repeatedly projecting it onto another hyperplane. We start with a random initial guess $\mathbf{u}^{[0]}$ (often simply the zero vector), then we project the point onto the hyperplane defined by the first equation to obtain $\mathbf{u}^{[1]}$ (see Figure 9). Then the second equation etc. After projecting onto the hyperplane for the M-th equation we project back onto the first hyperplane. This process is repeated until the method converges. If a unique solution exists, the method will always converge [15].

We can obtain simple equations for this iterative process as follows (see Figure 9). Denote $\mathbf{w}_i = (\mathbf{e}_i^T W)$ as the i-th row of matrix W. Consider the first equation ($i = 1$) in equation 2.12. We can rewrite it as the dot-product

$$\mathbf{w}_1^T \cdot \mathbf{u} = f_1.$$

Therefore, the hyperplane H_1 defined by the above equation is perpendicular to the row vector $\mathbf{w}_1$. This means that if we project any $\mathbf{x}$ on the hyperplane on $\mathbf{w}_1$, it will have a fixed length. Indeed, we see that this length is

$$\frac{\mathbf{w}_1 \cdot \mathbf{x}}{\|\mathbf{w}_1\|} = \frac{f_1}{\|\mathbf{w}_1\|}.$$

If we project the original initial guess $\mathbf{u}^{[0]}$ on $\mathbf{w}_1$ we get the vector

$$\frac{\mathbf{w}_1 \cdot \mathbf{u}^{[0]}}{\|\mathbf{w}_1\|} \frac{\mathbf{w}_1}{\|\mathbf{w}_1\|}.$$

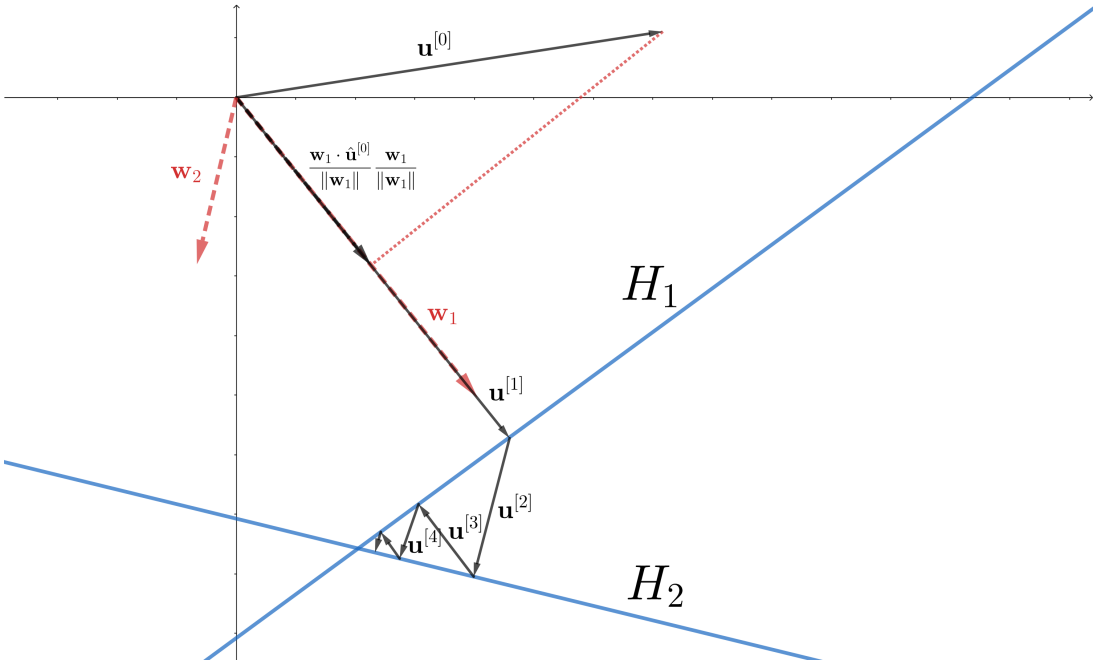


Figure 9: Diagram showing Kaczmarz method/ART update step for two hyperplanes H_1 and H_2 defined by $\sum_{j=1}^{nm} W_{1j}u_j = f_1$ and $\sum_{j=1}^{nm} W_{2j}u_j = f_2$ respectively. The initial guess $\mathbf{u}^{[0]}$ is projected on H_1 to obtain $\mathbf{u}^{[1]}$ etc.

However, this vector does not land on the hyperplane in general. We have that this vector is

$\frac{f_1 \mathbf{w}_1}{\|\mathbf{w}_1\|^2}$ plus (or minus) an extra bit $\mathbf{y}$,

$$\frac{\mathbf{w}_1 \cdot \mathbf{u}^{[0]}}{\|\mathbf{w}_1\|^2}\mathbf{w}_1 = \frac{f_1}{\|\mathbf{w}_1\|^2}\mathbf{w}_1 + \mathbf{y} \quad \Leftrightarrow \quad \mathbf{y} = \frac{1}{\|\mathbf{w}_1\|^2}\left((\mathbf{w}_1 \cdot \mathbf{u}^{[0]})\mathbf{w}_1 - f_1\mathbf{w}_1\right).$$

The vector $\mathbf{y}$ lies in the direction of $\mathbf{w}_1$ and we know that we need to subtract $\mathbf{y}$ from our initial guess to obtain $\mathbf{u}^{[1]}$

$$\mathbf{u}^{[1]} = \mathbf{u}^{[0]} - \frac{(\mathbf{w}_1 \cdot \mathbf{u}^{[0]} - f_1)}{\|\mathbf{w}_1\|^2}\mathbf{w}_1.$$

This generalizes so if we want to project the vector $\mathbf{u}^{[j]}$ onto hyperplane i we get the update rule

$$\mathbf{u}^{[i]} = \mathbf{u}^{[j]} - \frac{(\mathbf{w}_i \cdot \mathbf{u}^{[j]} - f_i)}{\|\mathbf{w}_i\|^2}\mathbf{w}_i. \tag{2.13}$$

As mentioned, we repeatedly project $\mathbf{u}^{[i]}$ onto hyperplane $i+1$ until we obtain $\mathbf{u}^{[M]}$. Then we project $\mathbf{u}^{[M]}$ onto hyperplane 1 and repeat until we obtain $\mathbf{u}^{[2M]}$. Tanabe [44] has shown that if there exists a unique solution $\mathbf{u}^*$ to equation 2.12, then

$$\lim_{k\to\infty} \mathbf{u}^{[kM]} = \mathbf{u}^*.$$

This algorithm is called Kaczmarz method or is referred to as *ART*. Even though Tanabe's result proves convergence of the method, the rate of convergence may still be so slow that the method is impractical. If the angle between the hyperplanes is small, projecting back and forth leads to small gains. In particular, machine precision may lead to false convergence since the steps are too small for a computer to distinguish the new iterate from the old. If the hyperplanes are orthogonalized beforehand, the method convergences in M passes. However, applying e.g. the Gram-Schmidt procedure to such a large system is also computationally expensive.

In general we may have $M > N$ and noisy $\mathbf{f}$. In this case we have the problem that no unique solution exists. When this is the case, the solution will oscillate in the neighbourhood of the intersection of the hyperplanes [39]. A suitable stopping criterion is necessary to extract an approximate solution in this case.

If $M < N$ we have an infinite number of possible solutions, i.e. the intersection of the hyperplanes H has dimension at least 1. If this is the case, any projection of our initial guess onto the intersection space is a feasible solution. ART produces a solution $\tilde{\mathbf{u}}$ such that $\left\|\mathbf{u}^{[0]} - \mathbf{u}^*\right\|$ is minimal. The quality of the solution will in this case strongly depend on the initial guess. In general, ART (and SART and SIRT discussed in section 2.5.1) solves the following minimization problem

$$\mathbf{u}^* := \arg\min_{\mathbf{u}} \left(\frac{1}{2}\|W\mathbf{u} - \mathbf{f}\|_2^2\right).$$

2.5.1 SART and SIRT

The Simultaneous Algebraic Reconstruction Technique (SART) and Simultaneous Iterative Reconstruction Technique (SIRT) are improvements upon ART at the expense of convergence speed. ART suffers from the problem that updating the changes for equation i, alters pixels that were just changed in $i-1$. Hence, updating pixels in ray path i also influences other ray

equations without taking those influences into account. SART and SIRT basically use equation 2.13 to determine the updates but do not alter them immediately.

To arrive at SART and SIRT we can start from the iterative viewpoint with *Landweber's equation.* In general, Landweber's equation reads

$$\mathbf{u}^{[k+1]} = \mathbf{u}^{[k]} - \alpha W^{\dagger}(W\mathbf{u}^{[k]} - \mathbf{f}).$$

Landweber iteration converges tot the solution closest to $\mathbf{u}^{[0]}$ if $0 < \alpha < \frac{2}{\rho(W^{\dagger}W)}$ where $\rho(W^{\dagger}W)$ returns $|\lambda_{max}|$ for the largest eigenvalue λ_{max} of $W^{\dagger}W$ [4]. Here W is the aforementioned $M \times nm$ matrix that represents the data acquisition process.

To define SART and SIRT, define two diagonal normalization matrices S (normalizes weights in pixel j) and V (normalizes weights in ray i):

$$\forall i = 1, \ldots, M, \;\; V_{ii} = \frac{1}{\sum_{k=1}^{mn} W_{ik}}, \qquad \forall j = 1, \ldots, nm, \;\; S_{jj} = \frac{1}{\sum_{k=1}^{M} W_{kj}}.$$

Then we can write down the adapted Landweber's equation

$$\mathbf{u}^{[k+1]} = \mathbf{u}^{[k]} - \alpha S W^{T} V(W\mathbf{u}^{[k]} - \mathbf{f}).$$

We can choose $\alpha = 1$ for the adapted Landweber equation [4]. The matrices S and V contain the inverse of the sum of the columns and rows of W. S and V compensate for the number of rays that hit a certain pixel and the number of pixels that are hit by a ray respectively. For SART, all updates according to equation 2.13 are computed for all rays for every angle. Then each cell is updated with the average value of all updates for that cell combined. Then the next angle is used to update this new solution and so forth.

For SIRT, we compute the updates for all rays *and* all the projection angles before we update each cell with an average update. In other respects it is similar to SART and we continue until some termination condition is met.

The total number of weights W_{ij} needed to describe the system may be computationally expensive to compute. Therefore, ART, SART and SIRT methods may be implemented by using an approximation to equation 2.13. Note that the componentwise version of equation 2.13 is

$$u_k^{[i]} = u_k^{[j]} - \frac{(\mathbf{w}_i \cdot \mathbf{u}^{[j]} - f_i)}{\|\mathbf{w}_i\|^2} W_{ik} = u_k^{[j]} + \Delta u_k^{[i,j]}.$$

In some implementations of the aforementioned algorithms the weights W_{ij} are either 1 if the center of the pixel is in the i-th ray and 0 otherwise [16]. In this case, the norm $\|\mathbf{w}_i\|^2$ is simply the number of pixels N_i whose center lies in the i-th ray. So we get that

$$\Delta u_k^{[i,j]} = \frac{f_i - \mathbf{w}_i \cdot \mathbf{u}^{[j]}}{N_i} \tag{2.14}$$

for pixels with centers in the i-th ray and 0 otherwise. Note that this equation is simply, take the difference with the measured ray-sum and the current estimate and smear it out over all relevant pixels. The approximations for the W_{ij} make the algorithms computationally more efficient but reduce the quality of the solution.

Lastly, since it will feature in this thesis, SIRTMIN is a non-linear version of SIRT where after each update step, all entries of $\hat{\mathbf{u}}^{[k]}$ less than a certain quantity λ are set to λ. Such a constraint is a simple way to incorporate prior knowledge to benefit the reconstruction. In this thesis SIRTMIN will mean SIRT with such a minimum constraint where $\lambda = 0$ which will converge to a solution if a non-negative solution exists [4].

2.6 Variational methods and total variation

The final algorithm we will discuss uses total variation minimization and is called TV-FISTA. Before we can discuss this algorithm we must establish *variational methods*. For our treatise we will generally be considering the continuous case of determining a function $u(\mathbf{x})$ but will alternate between the discrete case occasionally. In general, the goal of variational methods is to determine an unknown function $u(\mathbf{x})$ which best satisfies certain constraints. We formulate the constraints in a so called *energy functional*

$$\mathcal{J}(u) = D(u, f) + \alpha R(u).$$

Here $D(u, f)$ is called the *data fidelity* and the term should model how closely u resembles the data f. This term should be minimal when u resembles f as closely as possible. The second term $R(u)$ is called the *regularization functional* which should model desirable a-priori properties of our solution u. For example, we may desire solutions u with a small norm. The parameter α enables us to scale the influence of the regularization functional on the total energy.

The objective of variational methods is to find

$$u^* := \arg\min(\mathcal{J}(u))$$

over some space of functions that we allow our solutions to live in.

2.6.1 Denoising with variational methods and total variation

In this section we will introduce an energy functional geared at denoising with variational methods. We will also introduce total variation and the ROF model. Firstly, we will consider the canonical example of L_2-regularization.

Example: Transition to the discrete case $u \in \mathbb{R}^{nm}$ and we are provided a noisy image $\mathbf{f} \in \mathbb{R}^{nm}$. We want to construct $\mathbf{u}$ such that it is less noisy yet still resembles the data $\mathbf{f}$. Define the data fidelity term and the regularization term as

$$D(\mathbf{u}, \mathbf{f}) := \|\mathbf{u} - \mathbf{f}\|_2^2 = \sum_i (u_i - f_i)^2, \qquad R(\mathbf{u}) := \|\mathbf{u}\|_2^2 .$$

The above equation for the regularization term computes the pixelwise squared sum in the image. Hence it is expected to be larger for noisy images where the noise may cause the entries to spike in either direction. To obtain our denoised image we find the minimizer

$$\hat{\mathbf{u}} := \arg\min_{\mathbf{u}} \{D(\mathbf{u}, \mathbf{f}) + \alpha R(\mathbf{u})\}, \qquad \hat{\mathbf{u}} = (A^T A + \alpha^2 I)^{-1} A^T \mathbf{f}$$

for some α. This is known as *ridge regression* or L_2-regularization. It is a type of *Tikhonov regularization*. In the more general Tikhonov regularization we would have in this case

$$\mathcal{J}(\mathbf{u}) = \|A\mathbf{u} - \mathbf{f}\|_2^2 + \|\Gamma\mathbf{u}\|_2^2, \qquad \hat{\mathbf{u}} = (A^TA + \Gamma^T\Gamma)^{-1}A^T\mathbf{f}$$

for some suitably chosen Tikhonov matrix Γ. As we can see, the special case of ridge regression sets $\Gamma = \alpha I$.

Example: As another example, let us model this problem in the continuous case if we swap the sums for integrals over the image domain and introduce the gradient

$$\arg\min_{\mathbf{u}} \left[\frac{1}{2} \int_\Omega (u(\mathbf{x}) - f(\mathbf{x}))^2 \, d\mathbf{x} + \frac{\alpha}{2} \int_\Omega |\nabla u(\mathbf{x})|^2 \, d\mathbf{x} \right].$$

We see that strong oscillations in the gradient are punished in the regularization term. This means that noise but also edges in the picture are smoothed out. This seems favorable to punishing large u in general since it is independent of the magnitude of u.

We have now seen two examples of how variational methods can be used to obtain a best fit image under some conditions set in the regularization term. We have defined a regularizer that is independent of the magnitude of u. Next, the goal of TV regularization will be to define one that is independent of the magnitude of ∇u. Before we consider TV regularization, let us consider a few other functionals for the general setup with W some bounded linear operator

$$\mathcal{J}(u) = \frac{1}{2} \|Wu - f\|^2 + \alpha R(u, f).$$

For example, suppose that $\alpha = 1$ and

$$R(u, f) = \frac{1}{2} \int_\Omega \|Du\|^2 \, dx$$

for some linear operator D, e.g. a gradient ∇ operator on the Sobolev space $H^1(\Omega)$ (see Appendix 9.2.1). We see that the optimality condition becomes

$$\mathcal{J}'(u) = W^*(Wu - f) + D^*Du = 0.$$

Here W^* is the (Hermitian) adjoint operator which for bounded linear operators is defined to be such that $\langle Wx, y\rangle = \langle x, W^*y\rangle$. For example, suppose that we are denoising with $W = id$ and $D = \nabla$ as we saw before, then

$$\mathcal{J}'(u) = (u - f) + \nabla^2 u = 0.$$

We see that the regularization term drives u to be smooth and without edges. Consider the more general case for $p > 2$.

$$R(u) = \frac{1}{p} \int_\Omega \|Du\|^p \, dx.$$

The optimality condition becomes

$$\begin{aligned}\mathcal{J}'(u) &= W^*(Wu - f) + \frac{1}{p}\frac{\partial}{\partial u}\|Du\|^p \\ &= W^*(Wu - f) + \frac{1}{p}\frac{\partial}{\partial u}u^*D^*\|Du\|^{p-2}Du \\ &= W^*(Wu - f) + D^*\|Du\|^{p-2}Du = 0.\end{aligned}$$

If $p > 2$ we see that a large gradient will have a large impact on the regularization term and we will therefore always have smoothing effects. Consider the case $p = 1$

$$R(u) = \int_\Omega \|Du\| \; dx.$$

In the case that $\|Du\| \neq 0$, we obtain the optimality condition

$$\begin{aligned}\mathcal{J}'(u) &= W^*(Wu - f) + \frac{\partial}{\partial u}(u^*D^*Du)^{1/2} \\ &= W^*(Wu - f) + \frac{1}{2}D^*\left(\frac{Du}{\|Du\|}\right) = 0.\end{aligned}$$

We see that the regularization term always has a normalized contribution irrespective of the size of the gradient. Suddenly, we have removed the dependence on how "large" the discontinuity is that the gradient measures and only care *that* it exists. Intuitively, we "count" the number of discontinuities without a dependence on the magnitude of the discontinuities.

This observation leads to a well-known more complex variational method called the *Rudin-Osher-Fatemi (ROF) model* [40]. This model uses the total variation function to define its regularization term. We would like to define the total variation function as

$$TV(u) = \int_\Omega |\nabla u| \; dx. \tag{2.15}$$

However, this function makes no sense in a space containing piecewise constant functions or any other discontinuous functions. From the imaging perspective we may want edges in our images. Hence, we will need define a new space of functions and a new total variation function. To do so will require lots of extra mathematics and the process is (partially) outlined in Appendix 9.2.

The considerations outlined in Appendix 9.2 motivates the definition of the *Rudin-Osher-Fatemi* model which we will now restate in terms of the L^1-norm

$$\mathcal{J}(u) := \frac{1}{2}\|u - f\|_2^2 + \alpha\|\nabla u\|_1 \tag{2.16}$$

Here the role of the linear operator D from our earlier considerations is taken on by the gradient operator $D = \nabla$. Practically, when we consider a discrete vector $\mathbf{u}$ as we did previously we can define total variation more simply.

$$\begin{aligned}TV(\mathbf{u}) = &\sum_{i=1}^{m-1}\sum_{j=0}^{n-2}\sqrt{|u_{i+jm+1} - u_{i+jm}|^2 + |u_{i+(j+1)m} - u_{i+jm}|^2} \\ &+ \sum_{j=0}^{n-2}\sqrt{|u_{(j+2)m} - u_{(j+1)m}|^2} + \sum_{i=1}^{m-1}\sqrt{|u_{i+(n-1)m+1} - u_{i+(n-1)m}|^2}.\end{aligned}$$

Alternatively, there is an anisotropic version

$$TV(\mathbf{u}) = \sum_{i=1}^{m-1}\sum_{j=0}^{n-1}(|u_{i+jm+1} - u_{i+jm}|) + \sum_{i=1}^{m}\sum_{j=0}^{n-2}(|u_{i+(j+1)m} - u_{i+jm}|).$$

The equations seem cumbersome but they are simply the row and column wise differences if we transition to the matrix perspective. Define the horizontal difference operator D^h which subtracts row i from $i+1$ and D^v the vertical difference operator which subtracts column j from $j+1$ and we can for example rewrite the isotropic total variation as

$$TV(\mathbf{u}) = \sum_{i,j}\sqrt{(D^h\mathbf{u})_{ij}^2 + (D^v\mathbf{u})_{ij}^2}.$$

Regardless of the complex definition in Appendix 9.2, the above equations are those that are implemented in practice in the discrete case. In addition to defining an energy functional, we need to consider how to solve the minimization problem. The algorithm that we will use to solve the minimization problem is called FISTA and will be outlined in the next section.

2.6.2 FISTA

FISTA comes from *Fast Iterative Shrinkage-Thresholding Algorithm* [2]. We transition back to the discrete practical case where we are trying to solve a minimization problem of the form

$$\min_{\mathbf{x}}\{\|W\mathbf{u} - \mathbf{f}\|_2^2 + \lambda\|\mathbf{u}\|_1\}.$$

The general step in ISTA (not FISTA) is to take a step (for some step size η_k)

$$\mathbf{u}_{k+1} = T_{\lambda\eta_k}\left(\mathbf{u}_k - 2\eta_k W^T(W\mathbf{u}_k - \mathbf{f})\right). \tag{2.17}$$

Here $T_\alpha(\mathbf{u})_i := (|\mathbf{x}_i| - \alpha)^+ \cdot \operatorname{sgn}(\mathbf{x}_i)$ is called the *shrinkage operator*. The shrinkage operator evaluates to the following (for $\alpha > 0$)

$$T_\alpha(\mathbf{u})_i = \begin{cases} \mathbf{u}_i - \alpha & \text{if } \mathbf{u}_i \geq \alpha \\ 0 & \text{if } -\alpha \leq \mathbf{u}_i < \alpha \\ \mathbf{u}_i + \alpha & \text{if } \mathbf{u}_i < -\alpha \end{cases}.$$

We can derive this expression as follows. A typical gradient descent algorithm to minimize some function $g : \mathbb{R}^n \to \mathbb{R}$ is given by

$$\mathbf{u}_{k+1} = \mathbf{u}_k - \eta_k\nabla g(\mathbf{u}_k).$$

Suppose we want to minimize $g + h$ where g is our data fidelity term and h the regularization term. We can express the gradient iteration as a so called *proximal regularization*, the derivation of which can be found in [2]

$$\arg\min_{\mathbf{u}\in\mathbb{R}^n}\left[g(\mathbf{u}_k) + \nabla g(\mathbf{u}_k)^T(\mathbf{u} - \mathbf{u}_k) + \frac{1}{2\eta_k}\|\mathbf{u} - \mathbf{u}_k\|_2^2 + h(\mathbf{u})\right]. \tag{2.18}$$

If we have access to the optimal $\mathbf{u}$ every time, we will have improved convergence over gradient descent. However, we have to solve the minimization problem 2.18 which may be costly. If h is sufficiently "simple" the problem becomes easier. Assume that h is separable, i.e. $h(\mathbf{u}) = \sum_{i=1}^{n} h_i(u_i)$ (here u_i is the i-th component, not the i-th iterate), and the problem reduces to a one-dimensional convex problem for each component u_i. In our case, we chose $h(\mathbf{x}) = \lambda \|\mathbf{u}\|_1$ which is indeed separable. The one-dimensional problem then becomes something of the form

$$\min_{u\in\mathbb{R}} \left[h(u) := \lambda|u| + \frac{1}{2\eta}(u-u_0)^2 \right]$$

for some $u_0 \in \mathbb{R}$. If $u \geq 0$, we get that

$$\frac{d}{du}h(u) = \lambda + \frac{1}{\eta}(u-u_0), \qquad \frac{d^2}{du^2}h(u) = \frac{1}{\eta} > 0.$$

This gives $u = u_0 - \lambda\eta$ as a local minimum but it only exists if $u_0 \geq \lambda\eta$. Similarly, if $u < 0$ we obtain

$$\frac{d}{du}h(u) = -\lambda + \frac{1}{\eta}(u-u_0), \qquad \frac{d^2}{du^2}h(u) = \frac{1}{\eta} > 0.$$

Again, this gives $u = u_0 + \lambda\eta$ as a local minimum but this only exists if $u_0 < -\lambda\eta$. Suppose that $-\lambda\eta \leq u_0 < \lambda\eta$ and note that we can expand $h(u)$

$$h(u) = \frac{u_0^2}{2\eta} + \lambda|u| - \frac{u_0 u}{\eta} + \frac{u^2}{2\eta} \qquad \text{and} \quad h(0) = \frac{u_0^2}{2\eta}.$$

Since $-\lambda\eta \leq u_0 < \lambda\eta$ we have $\left|\frac{u_0 u}{\eta}\right| \leq \lambda|u|$ and so certainly $0 \leq \lambda|u| - \left|\frac{u_0 u}{\eta}\right| \leq \lambda|u| - \frac{u_0 u}{\eta}$. Hence we can conclude that $h(0) < h(u)$ so we obtain that this one-dimensional problem is solved by

$$u \leftarrow T_{\lambda\eta}(u_0) = \begin{cases} x_0 - \lambda\eta & \text{if } u_0 \geq \lambda\eta \\ 0 & \text{if } -\lambda\eta \leq u_0 < \lambda\eta \\ u_0 + \lambda\eta & \text{if } u_0 < -\lambda\eta \end{cases}.$$

Therefore we do not need to solve the minimization problem but we know the solution already in this case. Subsequently the problem in equation 2.18 is solved by

$$\mathbf{u}_{k+1} = T_{\lambda\eta_{k+1}}\left(\mathbf{u}_k - \eta_{k+1}\nabla g(\mathbf{u}_k)\right). \tag{2.19}$$

If we now transition to the case where $g(\mathbf{u}) := \|W\mathbf{u} - \mathbf{f}\|_2^2$ we obtain the original ISTA update step in equation 2.17. If $\eta_k \in \left(0, \frac{1}{\|W^T W\|}\right)$, the method converges [2]. Next, ISTA can be accelerated based on Nesterov's Accelerated Gradient Descent. We omit the details here but if we define

$$\lambda_0 = 1, \quad \lambda_{k+1} = \frac{1+\sqrt{1+4\lambda_k^2}}{2}, \quad \gamma_k = \frac{1-\lambda_k}{\lambda_{k+1}},$$

let $\mathbf{u}_0 = \mathbf{y}_0$ be an arbitrary point, and

$$\mathbf{y}_{k+1} \leftarrow \operatorname*{arg\,min}_{\mathbf{u}\in\mathbb{R}^n} \left[g(\mathbf{u}_k) + \nabla g(\mathbf{u}_k)^T(\mathbf{u}-\mathbf{u}_k) + \frac{1}{2\eta_{k+1}}\|\mathbf{u}_k - \mathbf{u}\|_2^2 + h(\mathbf{u}) \right] \tag{2.20}$$

$$\mathbf{u}_{k+1} \leftarrow (1-\gamma_k)\mathbf{y}_{k+1} + \gamma_k \mathbf{y}_k, \tag{2.21}$$

we have the FISTA update equations. The only parameter that we still need to choose is the stepsize $\eta_k \in \left(0, \frac{1}{\|W^T W\|}\right)$. In chapter 2 of [2] it is proven that if we take $L(\nabla g)$ the smallest Lipschitz constant of ∇g, then step size $\eta = \frac{1}{L(\nabla g)}$ will lead to convergence. For the case where $g(\mathbf{u}) := \|W\mathbf{u} - \mathbf{f}\|_2^2$ we have that the smallest Lipschitz constant of the gradient is

$$L(\nabla g) = 2\sigma_{max}(W^T W)$$

where $\sigma_{max}(\cdot)$ is the largest eigenvalue of the matrix. In summary, the algorithm works as follows:

Algorithm 1 FISTA

procedure FISTA(g, h)
  $L \leftarrow 2\sigma_{max}(W^T W)$
  $\eta \leftarrow \frac{1}{L}$
  $\mathbf{u}_0 \in \mathbb{R}^n$
  $\mathbf{y}_0 \leftarrow \mathbf{u}_0$
  $\lambda_0 \leftarrow 1$
  $k \leftarrow 0$
  while NotTerminated **do**
    $\lambda_{k+1} \leftarrow \frac{1+\sqrt{1+4\lambda_k^2}}{2}$
    $\gamma_k \leftarrow \frac{1-\lambda_k}{\lambda_{k+1}}$
    $\mathbf{y}_{k+1} \leftarrow T_{\lambda\eta}\left(\mathbf{u}_k - \eta\nabla g(\mathbf{u}_k)\right)$
    $\mathbf{u}_{k+1} \leftarrow (1-\gamma_k)\mathbf{y}_{k+1} + \gamma_k \mathbf{y}_k$
    $k \leftarrow k+1$
  return $\mathbf{u}_k$

In this thesis we used an implementation of TV-FISTA which can be found in the **pytvtomo** GitHub repository by D. M. Pelt [30].

3 Methodology

In this section we specify how many procedures of the algorithms and methods that were used in the experiments were implemented and how they are mathematically defined. We do so by providing a mathematical or algorithmic description of the procedure or method. Furthermore, we specify how we created the simulated ribosome dataset that most of the experiments in this thesis were performed on.

3.1 Noise simulation in computed tomography

Before we can perform our tomography experiments on a simulated dataset we need to define noise simulation in our experiments. To get an idea of the influence of noise we need to be able to control the severity of the noise. In cryo-ET, the energy levels of the electrons can be modified but is fixed during a scan. A canonical simple noise model is to take

$$z = x + y$$

where z is the measured signal, $x \sim \text{Poisson}(\lambda)$ models varying electron counts from the mean number of electrons λ passing through the sample and $y \sim \text{Gaussian}(m_e, \sigma_e^2)$ is the electronic noise. Usually, m_e can immediately be determined before a scan by measuring while no electrons are being emitted and can be calibrated to zero. The variance σ_e^2 can be estimated from several of such zero-emission measurements so we can simulate $y \sim \text{Gaussian}(0, \sigma_e^2)$ with σ_e^2 known. However, various attenuation coefficients in the sample can create different Poisson sources. A commonly used model in CT is based on the Lambert-Beer law and assumes that given attenuation $u(s)$, the counts (for some i-th ray L_i) follow a Poisson distribution

$$z_i \sim \text{Poisson}\left(I_0 \exp\left(-\int_{L_i} u(s)\, ds\right)\right). \tag{3.1}$$

Here I_0 is the incident number of electrons which is assumed to be known. For the purposes of practicality we want to discretize this model. We get that the integral can be modeled as a sum of entries of a weight matrix W

$$\int_{L_i} u(s)\, ds = \sum_{j=1}^{n} W_{ij}.$$

Therefore, the discretized model would be

$$z_i \sim \text{Poisson}\left(I_0 \exp\left(-\sum_{j=1}^{n} W_{ij}\right)\right). \tag{3.2}$$

On the sinogram side (the Radon transform of a function is often called a *sinogram*), this means that we can take the noiseless sinogram S as our W. Then we create the Poisson statistic for the noisy sinogram by (for each pixel (k, p) in the sinogram)

$$\lambda_{kp} = I_0 \exp(-S_{kp}). \tag{3.3}$$

Then, we draw from the distribution $\hat{x}_{kp} \sim \text{Poisson}(\lambda_{kp})$. Of course, we need to transition back to the original range of the sinogram. This means that we take

$$x_{pk} = -\ln\left(\frac{\hat{x}_{kp}}{I_0}\right)$$

as our final noisy sinogram. If we consider the divergent behaviour of the logarithm near 0 we can spot a problem for low-dose measurements. In that case, the probability that $\hat{x}_{kp} = 0$ is relatively high which means that the above expression is undefined. If I_0 is sufficiently large, we can simply set $\hat{x}_{kp} = 1$ as in such cases since it hardly matters. If I_0 is small such an intervention may be more problematic. In such cases, *maximum likelihood expectation maximization* algorithms may be more suitable [22] but these algorithms are out of scope for this thesis.

Often, if there are enough projections, the equation involving this ray sum is left out of the equations in the hope that the other projections still contain enough information about the voxels in this ray. If this is not the case, it is undefined how to proceed. Consider the case where a strongly absorbing spherical shell is scanned. We cannot know how to fill the inside of the shell since we do not have information. We could interpolate and fill the entire shell with a uniform material but this is greatly problem dependent. In addition, it can be counter-intuitive to eliminate zero-count equations since they strongly suggest the presence of an object and therefore contain a lot of information.

3.2 Subtomogram averaging

In identical particle analysis, one feature that we may attempt to exploit is that samples often contain several examples of the same object. In biological tissue, one sample may contain dozens if not hundreds of examples of the object of study. This has lead to the idea of *subtomogram averaging* [12], [49] where subvolumes containing the identical copies are extracted from the reconstruction. Then these subvolumes are aligned such that the objects are all oriented similarly and subsequently averaged. This means that the variance in the averaged reconstruction per voxel is reduced. In the case of noise with zero mean we are actually averaging out the noise entirely.

Before we discuss subtomogram averaging for our cryo-ET dataset we will define it formally. Let $S = \{\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \dots, \mathbf{X}_N\}$ be a set of subvolumes with $\mathbf{X}_i \subset \mathbb{R}^3$. Next, for each $\mathbf{X}_i$ we are provided with a set of rotation angles $\alpha_i, \beta_i, \gamma_i \in [0, 2\pi]$. The angles are determined by some algorithm so that all subvolumes are aligned properly. Furthermore, we are provided with a translation vector $\Delta\mathbf{x}_i \in \mathbb{R}^3$ for all $1 \leq i \leq N$ which defines a translation of the origin. We define three rotation matrices to apply in order to define the rotations

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{pmatrix}, \quad R_y(\theta) = \begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{pmatrix},$$
$$R_z(\theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Let $T_{\Delta\mathbf{x}}(\cdot)$ denote the translation operator. We use ZXZ-rotation as convention for this dataset so we define the rotation operator $R_{zxz}[\alpha_i, \beta_i, \gamma_i](\cdot)$ which applies $R_z(\gamma_i)R_x(\beta_i)R_z(\alpha_i)$ to each vector in the set $\mathbf{X}_i$. The result of subtomogram averaging is then (we rotate before we translate)

$$\mathbf{X}^* := \frac{1}{N} \sum_{i=1}^{N} T_{\Delta\mathbf{x}_i} \circ R_{zxz}[\alpha_i, \beta_i, \gamma_i](\mathbf{X}_i). \tag{3.4}$$

When we transition to the discrete case we need to perform interpolation and choose a rotation and translation method. For rotation we have implemented our own version that rotates a regular grid of the correct size and performs linear interpolation with the `RegularGridInterpolator` in `numpy`. For the translation we use the `shift` method from `scipy.ndimage.interpolation` which uses spline interpolation. New voxels that come into the volume and are not set by the interpolator are set to 0.

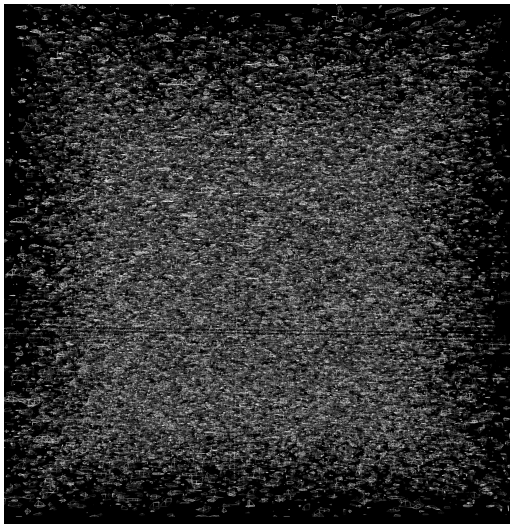


Figure 10: An example subvolume of a possible ER-associated ribosome from the dataset.

For cryo-ET it is in practice it challenging to apply subtomogram averaging because the extremely noisy reconstruction of the sample makes it hard to identify and align the subvolumes (see Figure 10 for an example of a single subvolume). In [35], [36], [37] a complex template matching and cross-correlation maximization algorithm was necessary to obtain best averages which is outlined in Figure 11.

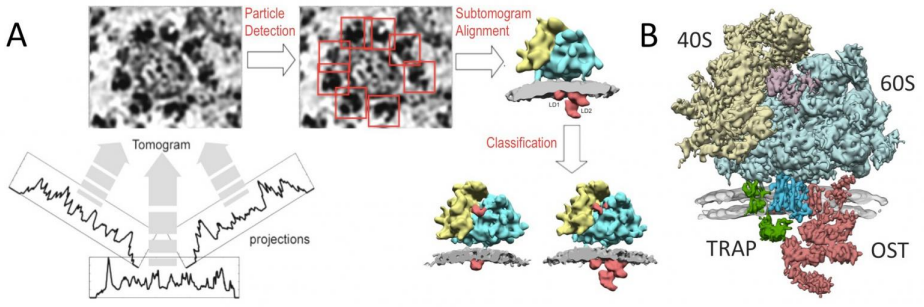


Figure 11: Diagram outlining how prof. Förster uses subtomogram averaging [36]. The projection are reconstructed with an NFFT algorithm. Next, candidate subvolumes are selected by template matching. A cross-correlation maximization algorithm is used to find the shifts and rotations of the subvolumes for the average. The subtomograms are also classified into distinct ribosome classes.

Firstly, the researchers collected the projections using cryo-ET. A non-uniform fast Fourier transform (NFFT) [6] reconstruction algorithm was used to obtain the tomographic reconstructions. Some theory on direct reconstruction methods and NFFTs is included in Appendix 9.1. Next, a template matching algorithm identifies possible particles using a cross-correlation maximization algorithm and an expectation maximization alignment algorithm [7] is used to find the origin shift and rotation angles to align the the subtomograms. A base average is maintained as template for alignment. In this align-and-average step the subtomograms are classified [48] into distinct ribosome classes such as to not mix different types. The algorithm terminates with a best average from the set of subtomograms for that class.

3.3 Ensemble reconstruction method

In addition to the subtomogram averaging method, we will test an alternate method that makes use of several identical particles in a scan at once. In subtomogram averaging, we make several sometimes poor reconstructions and then average them using the estimated rotation angles. Instead, we could use the flexibility of the ASTRA toolbox [29] to use the angles to create one larger reconstruction. Each individual set of projections is then interpreted as a different scan trajectory for the same object. Under the assumption of uniformly sampled rotation angles this should remove the missing wedge artifacts and the projection data will no longer be undersampled.

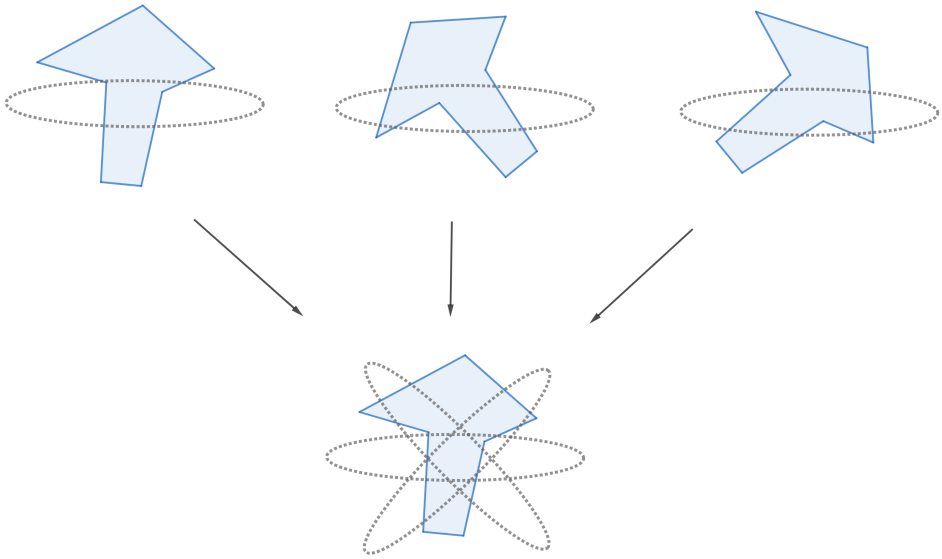


Figure 12: When we image several identical particles in different orientations in a single scan, we can interpret the scanning data as if we scanned the same particle multiple times with different scanning geometries.

Mathematically, let $S = \{\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3, \ldots, \mathbf{P}_N\}$ be a set of sets of projections. In the ASTRA geometry, a projection geometry is spanned by four 3-dimensional vectors. The ray-vector $\mathbf{r}$,

the center vector $\mathbf{c}$, and two vectors that span the detector $\mathbf{u}$ and $\mathbf{v}$. So an element of $\mathbf{P}_i$ is a tuple $(\mathbf{r}_k, \mathbf{c}_k, \mathbf{u}_k, \mathbf{v}_k, \mathbf{p}_k)$ with $\mathbf{r}_k, \mathbf{c}_k, \mathbf{u}_k, \mathbf{v}_k \in \mathbb{R}^3$ and $\mathbf{p}_k \subset \mathbb{R}^2$ the actual projection image.

Like for subtomogram averaging, for each $\mathbf{P}_i$ we are provided with a set of rotation angles $\alpha_i, \beta_i, \gamma_i \in [0, 2\pi]$. In this implementation we have disregarded the translation shifts since they are not present in the simulated dataset. The ASTRA toolbox does facilitate the addition of shifting the scanning geometry translationally as well.

Let φ_{REC} denote the reconstruction algorithm, then we obtain each reconstructed subvolume by

$$\mathbf{X}_i := \varphi_{REC}(\mathbf{P}_i).$$

For the the post-averaging procedure outlined in section 3.2 we would now apply a rotation to the volume $\mathbf{X}_i$, i.e. we define the rotation operator $R_{xyz}[\alpha_i, \beta_i, \gamma_i](\cdot)$ which applies $R_z(\gamma_i)R_y(\beta_i)R_x(\alpha_i)$ to each vector in the set $\mathbf{X}_i$. Note that this is slightly different then in section 3.2 because we perform XYZ-rotation for the simulated dataset. The result of subtomogram/post-averaging is then

$$\mathbf{X}^*_{post} := \frac{1}{N}\sum_{i=1}^{N} R_{xyz}[\alpha_i, \beta_i, \gamma_i] \circ \varphi_{REC}(\mathbf{P}_i)).$$

For the ensemble method we have defined a rotation function on the projection side. The angles $\alpha_i, \beta_i, \gamma_i$ are with respect to some ground truth orientation. To rotate the projection geometry such that it seems as if we scanned the object in the ground truth orientation we need to take the inner product of each of the 4 vectors with $\tilde{R} := R_x(-\gamma_i)R_y(-\beta_i)R_z(-\alpha_i)$. Then the tuple of the four augmented vectors plus the projection image, which is unchanged, $(\tilde{R}\mathbf{r}_k, \tilde{R}\mathbf{c}_k, \tilde{R}\mathbf{u}_k, \tilde{R}\mathbf{v}_k, \mathbf{p}_k)$ is added to a big list.

Let N_θ denote the number of projections in each $\mathbf{P}_i$. With some abuse of notation (it is cumbersome to denote the angles for each rotation operator) we get that the ensemble method is given by

$$\mathbf{X}^*_{ensemble} := \varphi_{REC}\left(\bigcup_{i=1}^{N}\bigcup_{k=1}^{N_\theta}(\tilde{R}\mathbf{r}_{ik}, \tilde{R}\mathbf{c}_{ik}, \tilde{R}\mathbf{u}_{ik}, \tilde{R}\mathbf{v}_{ik}, \mathbf{p}_{ik})\right).$$

Remark: It seems that the "z-angle" γ_i is incorrectly fed into the x-rotation matrix R_x for $\tilde{R}$. However, ASTRA and `numpy` have different conventions on ordering of axes and ordering if the first index in an array is the z-axis etc. If we define the aforementioned matrices as in section 3.2, both post-averaging and the ensemble method should correctly match the transition from ASTRA to `numpy` and vice versa.

3.4 Creating a simulated dataset

In this section we specify how we create the simulated dataset that the tomography experiments were run on. We consider the original reconstructions of the real ribosomes (see Figure 10) to be too noisy to be of use for these experiments. Furthermore, due to the absence of a ground truth for real data, we are not able to reliably determine the quality of the reconstruction. Also, the individual tomogram projection are 3710×3710 so reconstruction is computationally expensive. Furthermore, we would need access to the particle detection algorithm to isolate the (possible)

ribosomes which is computationally expensive as well. Lastly, we would not be able to control the level of noise in the experiments. For these reasons, we have decided to create a simulated dataset that resembles the original problem and perform all our tomography experiments on the simulated dataset.

Firstly, we created a base phantom ribosome for the simulated dataset. In addition to the raw particle data we were also supplied with a best averaged reconstruction obtained from the data for all 17000 particles (see Figure 13).

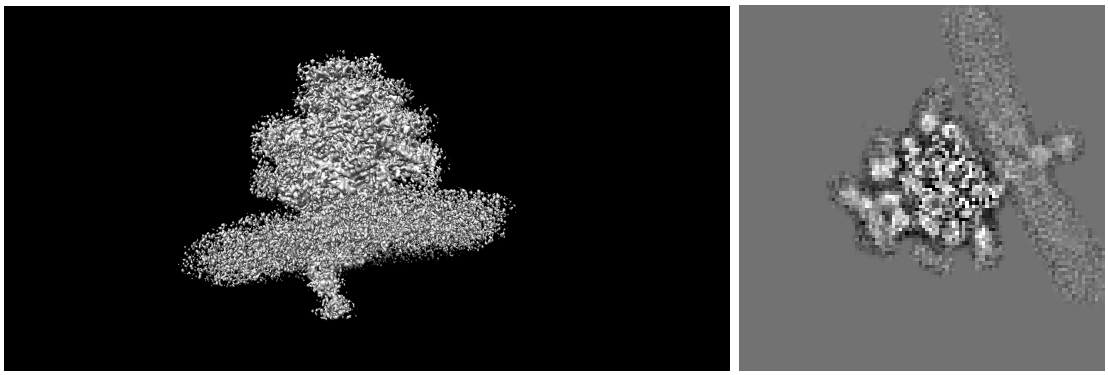


Figure 13: Left-hand side: The best averaged reconstruction from the real dataset obtained by subtomogram averaging over 17000 particles (rendered in UCSF Chimera [34] in surface mode), right-hand side: a middle slice of the best averaged reconstruction rendered in ImageJ.

We chose this reconstruction as a basis to create our phantom. For the phantom we wanted a smoother version of this template that still resembles the original ribosome closely. Firstly, we applied a 3D median filter of size $5 \times 5 \times 5$ followed by another $2 \times 2 \times 2$ median filter. Next, to reduce the graininess we threshold all the values less than 0.135 to 0 and the rest to 1. Lastly, to smooth the image we applied a Gaussian filter with standard deviation 0.75. We end up with the following phantom (see Figure 14).

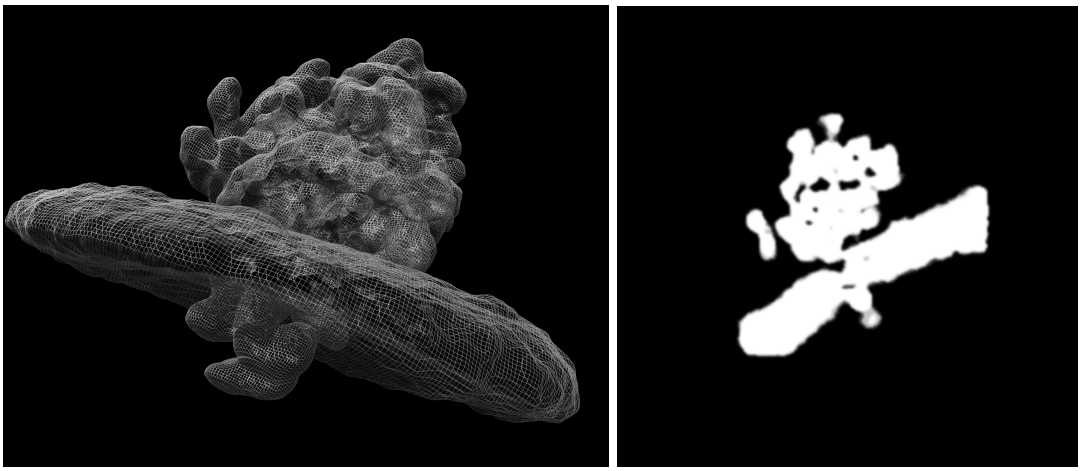


Figure 14: Left-hand side: A 3D rendering of the simulated ribosome base phantom (rendered in UCSF Chimera in mesh mode), right-hand side: a middle slice of the best averaged reconstruction rendered in ImageJ.

This base phantom is used to create our simulated dataset of randomly oriented ribosomes. To determine the orientations we take the following angles to get a uniform distribution of angles in 3D

$$\begin{aligned} &\varphi_x \in U[0, 2\pi] \\ &\varphi_y \in U[0, 2\pi] \\ &t \in U[-1, 1] \quad \Rightarrow \quad \varphi_z = \cos^{-1}(t). \end{aligned}$$

Then we enlarge the volume of the phantom to $320 \times 320 \times 320$ by zero padding. The center of mass of the simulated ribosome remains at the center. We subsequently rotated the ribosome around the x, y and z axis by the angles drawn in the aforementioned manner. We use the same rotation algorithm and interpolation scheme as outline in section 3.2 We created 200 of these randomly oriented ribosome phantoms.

3.4.1 Simulating cryo-ET projections

To simulate cryo-ET we use the ASTRA Toolbox [29] with a slice by slice 2D set up. We set the volume geometry to $(320, 320)$. For the projection geometry we chose a parallel beam set up with a pixel width of 1.0 for the projector and a width of 320 pixels. To mimic the angular range in the original experiments we chose the angular range of the projections to be from $-60°$ to $60°$ with a $2°$ increment. An example slice of a sinogram is provided in Figure 15.

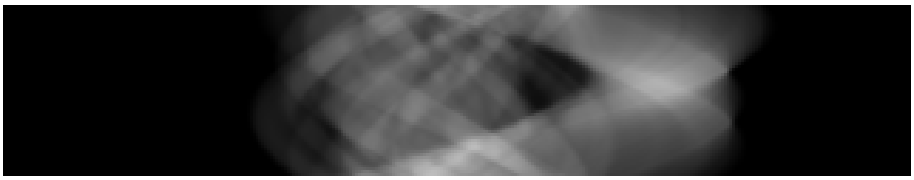


Figure 15: An example slice of a noiseless simulated sinogram created with the ASTRA toolbox [29].

To simulate noise in the experiments we need to add noise to the sinogram. After all, we assume that the object is properly represented in reality but the detector collects noisy projections. In the real dataset there may be lots of other material in the sample interfering with the ribosomes in the scan but we disregard this for now. We decided to create a dataset with various levels of Gaussian noise on the sinograms and and a dataset with Poisson noise.

To create a dataset of sinograms with Gaussian noise we added normally distributed noise $\mathcal{N}(0, \sigma)$ with standard deviations $\sigma \in \{0, 10, 100, 250\}$. These values were chosen such that the 100-Gaussian dataset would be extremely noisy and that for the 250-Gaussian set the signal would be indistinguishable to the human eye. Example slices of the sinograms are given in Figure 16.

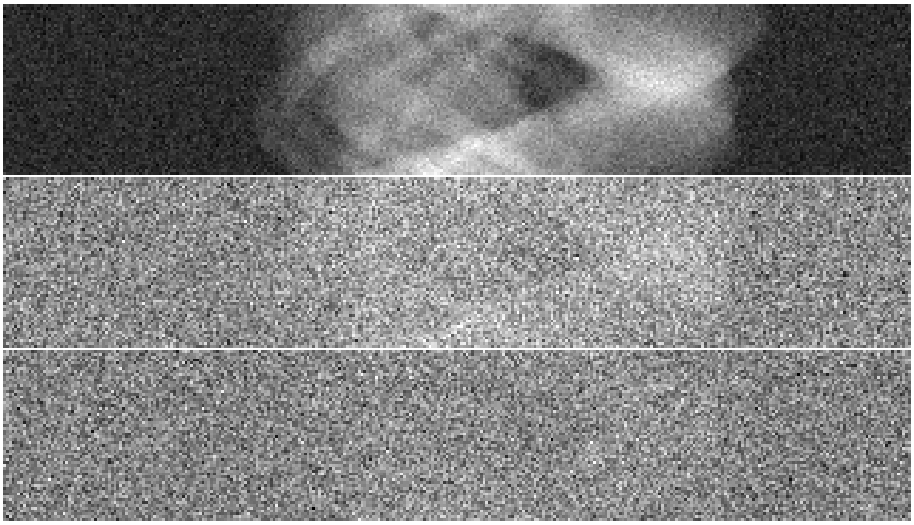


Figure 16: Example slices of simulated sinograms with Gaussian noise with standard deviation $\sigma = 10$ (upper), $\sigma = 100$ (middle) and $\sigma = 250$ (bottom).

To create a dataset of sinograms with Poisson noise we followed the procedure outlined in section 3.1. By choosing a different I_0, we control the noise level on the sinograms. To avoid the divergence properties of the logarithm we set λ to 1 if we happened to draw a 0 from the distribution under which was viable the assumption that the electron dose is sufficiently high that this acceptable. However, since cryo-ET is low-dose it is questionable whether the assumption can be maintained.

The original cryo-ET experiments for the translocon showed that roughly half of the electrons in their cryo-ET experiments were absorbed. To simulate this behaviour we introduced an attenuation coefficient u in our Poisson model

$$\lambda_{kp} = I_0 \exp(-uS_{kp}).$$

We set $u = 0.01$ to obtain an absorption rate of 0.5. We found that if we set $I_0 = 20$ we obtained a realistic noise profile for the least severe case. In addition, we created two additional sets of sinograms with incoming intensities 5.0 and 1.25 respectively. Examples of the sinograms are given in Figure 17.

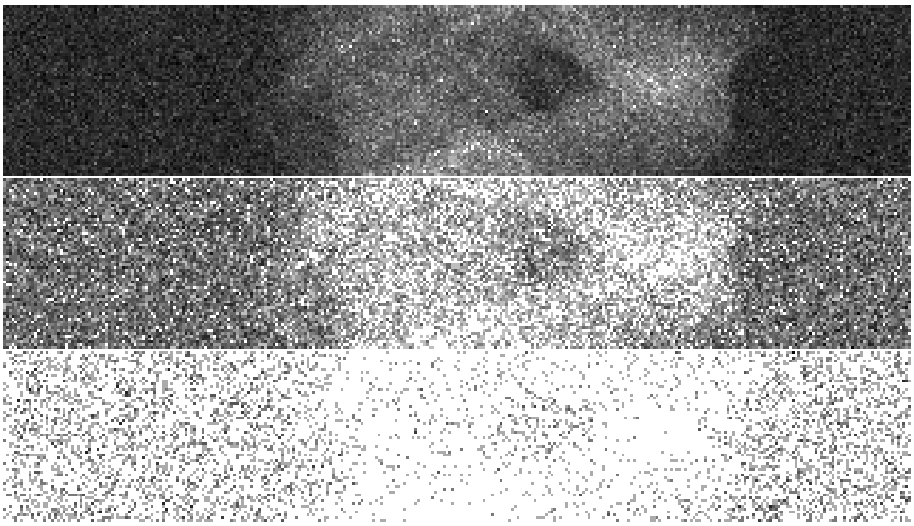


Figure 17: Examples of simulated sinograms with Poisson noise for $I_0 = 20$ (upper), $I_0 = 5.0$ (middle) and $I_0 = 1.25$ (bottom).

The sinograms with $I_0 = 1.25$ display the same pixel values often. This may be indicative that setting $\lambda = 1$ in the case of a zero draw may introduce a disturbance. This is not ideal and maybe different approaches of tackling low-dose Poisson noise models would be better such as a statistical reconstruction algorithm that takes the noise profile into account. Such algorithms are out of the scope of this thesis though.

3.5 Adapted mean-squared error

Unfortunately quantitative analysis is greatly influenced by the exact definition of the error measure and usually a single number cannot capture the nuances of the types of mistakes that an algorithm can produce. We designed the following algorithm to calculate an adapted mean-squared error (MSE) score to best rank performance of our reconstruction algorithms. Suppose we want to compare a ground truth volume $\mathbf{X}^{[gt]}$ to a reconstructed volume $\hat{\mathbf{X}}$. Traditional MSE would be defined as (with N the number of entries in $\mathbf{X}^{[gt]}$ and $\hat{\mathbf{X}}$)

$$MSE(\mathbf{X}^{[gt]}, \hat{\mathbf{X}}) := \frac{1}{N} \sum_{i=1}^{N} (\mathbf{X}_i^{[gt]} - \hat{\mathbf{X}}_i)^2.$$

The MSE is sensitive to scaling of the range of values of its arguments. However, some normalized versions of e.g. NRMSE are scaled by the distance between the maximum and minimum value present in the volume. For high noise reconstructions this scaling would be dominated by the values the noise takes.

The adapted MSE that we implemented for this experiment takes a mask of booleans where the original ground truth image has signal (the background is zero). The values of the noisy reconstruction that are in the mask are averaged to produce a mean signal value. The value in each voxel (also outside the mask) is subsequently divided by this mean to limit the effect of scaling on the MSE score. Next the MSE score is calculated in the traditional fashion where we

compare the original rotated phantom with the reconstruction. The mathematical formulation is

$$AMSE(\mathbf{X}^{[gt]}, \hat{\mathbf{X}}) := \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{\lambda}\mathbf{X}_i^{[gt]} - \hat{\mathbf{X}}_i\right)^2, \qquad \lambda = \frac{\sum_{i=1}^{N} H(\mathbf{X}_i^{[gt]})\hat{\mathbf{X}}_i}{\sum_{i=1}^{N} H(\mathbf{X}_i^{[gt]})}$$

where $H(x)$ is the Heaviside step function which is 1 if $x > 0$ and 0 otherwise. Note that the ground truth volume is non-negative so we do not disregard signal from the ground truth.

4 Tomography Experiments

In this section we will investigate how the reconstruction quality for various tomographic reconstruction algorithms plays out for different cryo-ET scenarios, with and without subtomogram averaging. A strongly regularized method that removes all the noise but also a lot of signal may be best for a single reconstruction. However, we expect that for a sufficiently large amount of averages a method without strong regularization will start to perform better. We expect that for sufficiently large batches the noise will be averaged out but most of the signal will still be present.

In the experiments discussed in this section we will consider the effects of the limited angular range of the projections that cryo-ET tomograms suffer from. We hypothesize that the missing wedge artifacts may largely be dealt with by the alignment and averaging operation. If we assume relatively random orientations for our identical copies in the sample, we will obtain information for many angles in the full range for a large enough batch. For example, since FBP is a linear procedure, it makes no difference whether we reconstruct first and then average all the randomly oriented reconstructed subvolumes or combine all the projections of the subvolumes into one big reconstruction (which has ample data for all angles).

If the orientations are not suitably uniform we may still have missing wedge artifacts. Furthermore, many other reconstruction methods are not linear so the missing wedge artifacts may still be relevant. To investigate the missing wedge problem, we will look at the results of the noiseless reconstructions.

Lastly, we have developed a new ensemble-reconstruction method that, using the flexibility of the ASTRA toolbox [29], incorporates all the separate scans in identical particle analysis into one large reconstruction. We will juxtapose the ensemble method with the standard subtomogram averaging approach. The experiments are carried out on the simulated dataset we constructed in section 3.4. In summary, the outline of the experiments in this section is as follows:

→ **Section 4.1**: *Testing standard FBP, SIRT(MIN) and TV-FISTA*

  ↪ Individual reconstructions for Gaussian and Poisson noise (section 4.1.1).

  ↪ Post-reconstruction averaging for Gaussian and Poisson noise (section 4.1.2).

→ **Figure 35**: Preliminary pipeline

→ **Section 4.2**: *Robustness of FBP, SIRT(MIN) and TV w.r.t. ribosome alignment*

→ **Section 4.3**: *Testing ensemble method*

  ↪ Testing on Gaussian and Poisson datasets.

  ↪ Robustness vs. post-reconstruction averaging.

→ **Figure 45**: Final pipeline

4.1 Traditional reconstruction methods and extreme noise performance

Our first exploratory experiment is to reconstruct each of the ribosome phantoms individually with FBP, SIRT, SIRTMIN and TV-FISTA for both the Gaussian and Poisson datasets. This

will give us insight into the severity of the noise in the reconstructions without subtomogram averaging and will also allow us to see the influence of regularization in the TV-FISTA algorithm for such noisy projections. We will provide both a qualitative and quantitative analysis of the experiments and attempt to deduce strengths and weaknesses for each tomographic reconstruction algorithm. Theory on each of the tomographic reconstruction algorithms can be found in section 2.

We hypothesize that TV-FISTA will perform best since TV is designed to limit the number of connected noise regions in the reconstruction and should therefore suffer least from the severe salt-and-pepper noise. Furthermore, the missing angular range is large so we expect obvious missing-wedge artifacts in the individual reconstructions.

4.1.1 Testing individual reconstructions

We ran SIRT (ASTRA [29] implementation) for 200 iterations, SIRTMIN (ASTRA implementation) for 200 iterations, FBP with RAMLAK filter (ASTRA implementation) and TV-FISTA [30] with various combinations of the regularization parameter and number of iterations. For TV-FISTA, these parameters were first selected based on minimizing the AMSE score on a middle slice. Then the best pair was selected by the author from a set of best pairs if there were several equally performing parameter pairs. To accelerate the computation we ran the reconstruction algorithms on Graphic Programming Units (GPUs) with CUDA version 9.2. The computations were carried out on a server using four NVidia GeForce GTX 1080 GPUs

Firstly, we will consider the noiseless reconstructions. These provide us with an indication of the severity of the missing wedge artifacts in the various reconstructions. For the noiseless TV-FISTA reconstructions we chose 80 iterations with regularization parameter $\lambda = 0.0005$. Figure 19 shows a middle slice of a sample reconstruction for each algorithm. In Figure 18 we have included a middle slice of the ground truth.

Figure 18: Middle slice of a ground truth ribosome phantom.

It seems that FBP and SIRT suffer much more from the missing wedge artifacts than SIRTMIN and TV-FISTA. The latter two do still have some streaking artifacts but not nearly as severe as FBP and SIRT. For TV-FISTA this is to be expected since introducing connected components of the wedge artifacts incurs a penalty in the regularization term.

Next, we consider the Gaussian-10.0 reconstructions. These provide us with an indication of the influence of some noise to the reconstructions. For the TV-FISTA reconstructions we chose 80

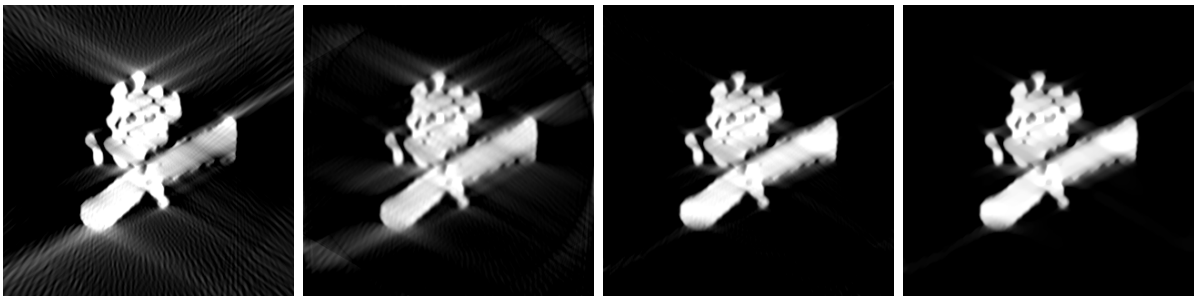


Figure 19: Middle slice of a reconstruction of a sample ribosome phantom without noise. From left to right we have FBP-RAMLAK, SIRT, SIRTMIN and TV-FISTA reconstructions.

iterations with regularization parameter $\lambda = 0.003$. Figure 20 shows a middle slice of a sample reconstruction.

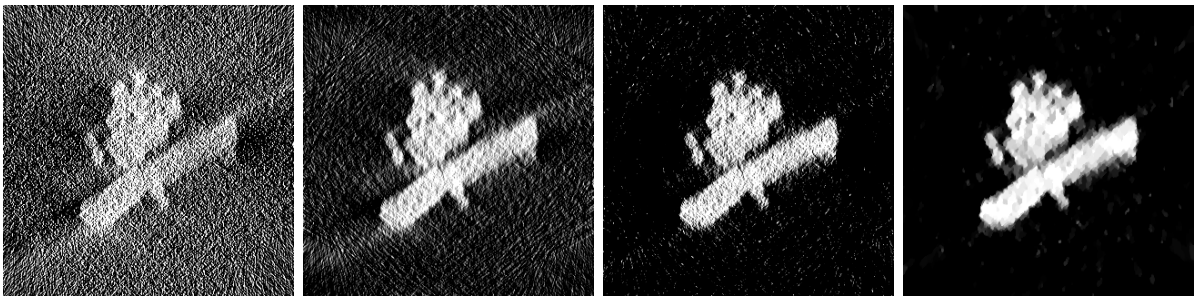


Figure 20: Middle slice of a reconstruction of a sample ribosome phantom with Gaussian noise $\sigma = 10.0$ on the projections. From left to right we have FBP-RAMLAK, SIRT, SIRTMIN and TV-FISTA reconstructions.

It seems that FBP suffers the most from noise. The difference between regular SIRT and SIRTMIN can be seen in the background where SIRTMIN has barely any salt-and-pepper noise or streaking artifacts while these are present in the SIRT reconstruction. Furthermore, we see that TV-FISTA already has the tendency to clump together blobs of the same gray value. This is to be expected with a larger regularization parameter. Due to this, it seems that SIRTMIN may have a slightly more detailed, albeit grainy reconstruction.

Next, we consider the Gaussian-100.0 reconstructions. As mentioned, this noise level is already very high such that the signal in the sinogram is barely distinguishable. This provides us with a relatively close approximation of the noise level in actual cryo-ET although it can be argued that the noise has to be even higher. For the TV-FISTA reconstructions we chose 60 iterations with regularization parameter $\lambda = 0.08$. Figure 21 shows a middle slice of a sample reconstruction.

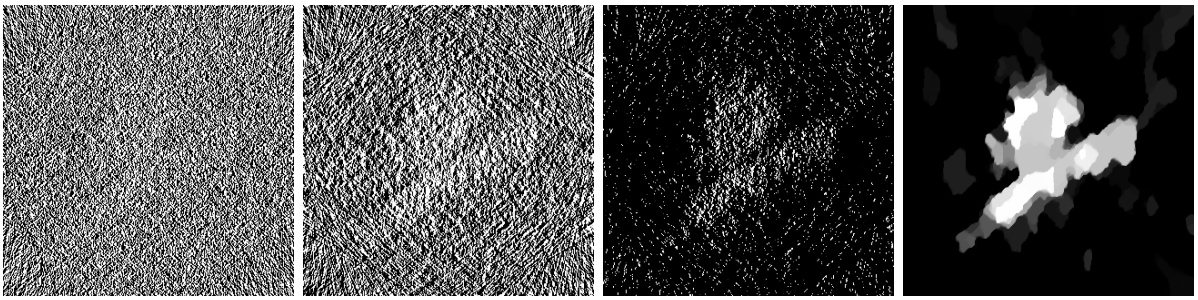


Figure 21: Slice 160/320 of a reconstruction of a sample ribosome phantom with Gaussian noise $\sigma = 100.0$ on the projections. From left to right we have FBP-RAMLAK, SIRT, SIRTMIN and TV-FISTA reconstructions.

In Figure 21 the effects registered for the $\sigma = 10$ case are compounded. FBP and SIRT are practically incapable of recovering the phantom and clumping of the same gray values is more dominant for TV-FISTA. Lastly, we consider the Gaussian-250.0 reconstructions. For the TV-FISTA reconstructions we chose 70 iterations with regularization parameter $\lambda = 0.2$. Figure 22 shows a middle slice of a sample reconstruction.

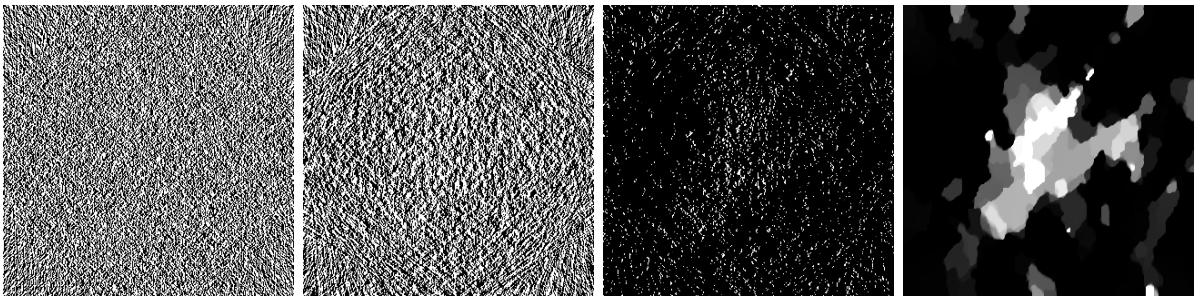


Figure 22: Slice 160/320 of a reconstruction of a sample ribosome phantom with Gaussian noise $\sigma = 250.0$ on the projections. From left to right we have FBP-RAMLAK, SIRT, SIRTMIN and TV-FISTA reconstructions.

Again, both FBP and SIRT are entirely incapable of recovering the phantom at this noise level. Only a vague outline can be spotted for SIRTMIN if you know the orientation of the original object. TV-FISTA is the only method that is able to distinguish any reasonable object but it no longer resembles a ribosome. We would say that TV-FISTA performs best but none of these methods could realistically produce a reconstruction of a ribosome at this noise level. If anything, this illustrates the necessity of using "statistics" in the subtomogram averaging method to recover a suitable reconstruction.

As a final remark we note that the TV-FISTA reconstruction could be used to estimate properties such as the orientation of the ribosome. It seems that it would certainly be easier to estimate the rotation angles from the TV reconstruction than any of the others. For the quantitative analysis the plot of the AMSE scores are shown in Figure 23. The mean of the AMSE over all 200 simulated phantoms is shown with the error bar indicating the standard deviation.

Remark: After analyzing the results we feel that the quantitative analysis can be somewhat misleading. It turns out that e.g. FBP reconstructions contain a lot of graininess in the vacuum outside the phantom. Even after pre-scaling in the AMSE function this remains a factor.

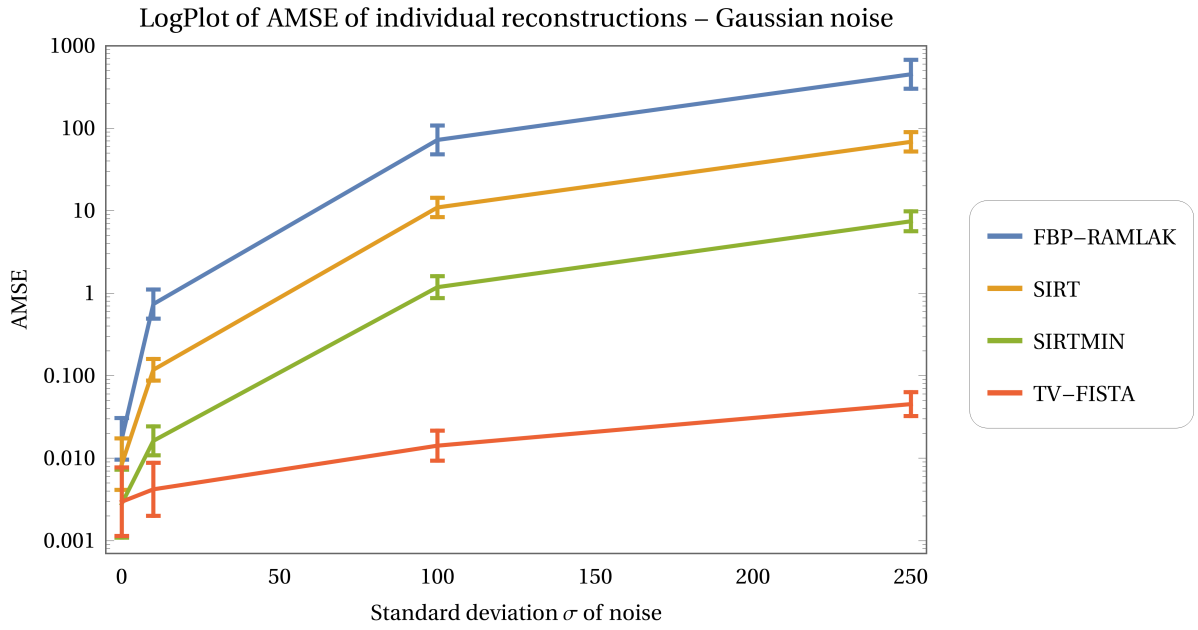


Figure 23: Log plot of AMSE scores of 200 randomly oriented reconstructions with respect to the ground truth for FBP (with RAMLAK filter), SIRT and SIRTMIN (200 iterations) and TV-FISTA for Gaussian noise on the projections with standard deviation $\sigma = 0, 10.0, 100.0, 250.0$.

Therefore, the quantitative measures will be extremely poor since every voxel in the vacuum negatively impacts the score. On the other hand, a method that at least sets the vacuum to 0 mostly (like TV) may have a much better score even if the phantom reconstruction itself is very poor. Therefore, our ultimate assessment is also largely based on interpretation of the pictures as opposed to the quantitative analysis.

Nonetheless, if we compare the data points for $\sigma = 0$ we see that there appears to be a clear ordering from worst to best; FBP, SIRT, SIRTMIN and TV-FISTA which coincides with our qualitative judgment. The poor nature of quantitative analysis is evident when we look at the $\sigma = 10.0$ data points. The AMSE scores greatly increase for FBP, SIRT and SIRTMIN while the TV reconstructions exhibits a much more level increase in error. In figure 20 the differences between SIRT, SIRTMIN and TV-FISTA are not so profound that such a dramatic difference in score seems warranted. As a whole, we maintain our point that that TV-FISTA performs best but none of these methods could realistically produce a reconstruction of a ribosome at the realistic cryo-ET noise levels.

Next, we intend to finalize certain conclusions drawn up in the Gaussian experiment. We will test the quality of the reconstructions for FBP, SIRT, SIRTMIN and TV-FISTA for various levels of Poisson noise. For the sake of brevity, we provide a large overview of all reconstructions in Figure 24.

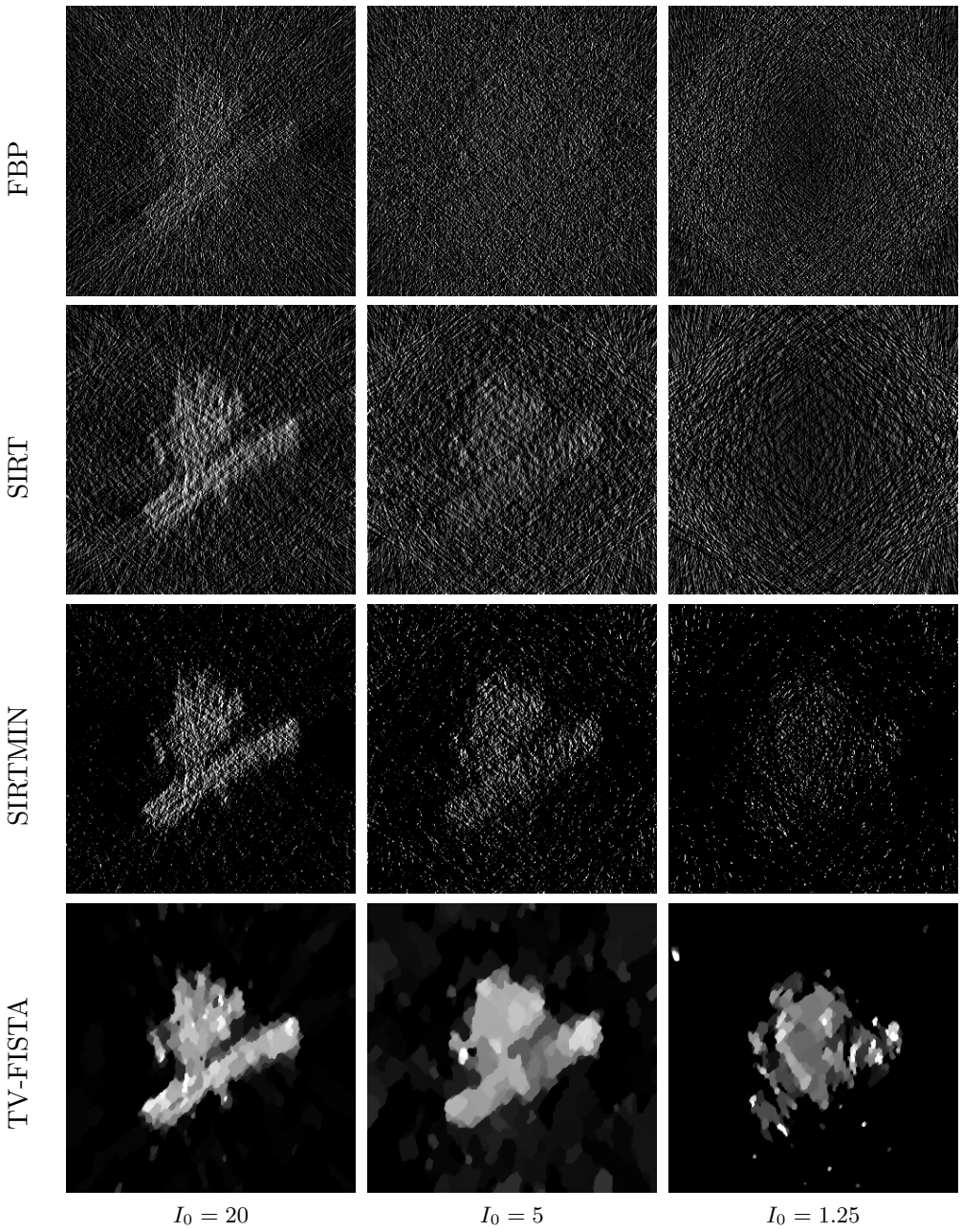


Figure 24: Slice 160/320 of all reconstructions with Poisson noise $I_0 = 20.0, 5.0, 1.25$ on the projections. First row: FBP with RAMLAK filter, second row: SIRT (200 iterations), third row: SIRTMIN (200 iterations) and fourth row: TV-FISTA.

In the previous pictures all display values were clipped to the same values $[0, 1]$. However, many algorithms struggled with non-linear noise so that we could not clip to the same values here. Instead, we used ImageJ to choose the best range of display values and clipped only the negative pixel values.

FBP and SIRT are not capable to extract any signal from the high noise levels as before. A vague perimeter of the object can be seen for the SIRTMIN reconstruction for the highest noise level but only TV-FISTA consistently recovers some object from the data. The AMSE scores for the Poisson experiments are given in Figure 25.

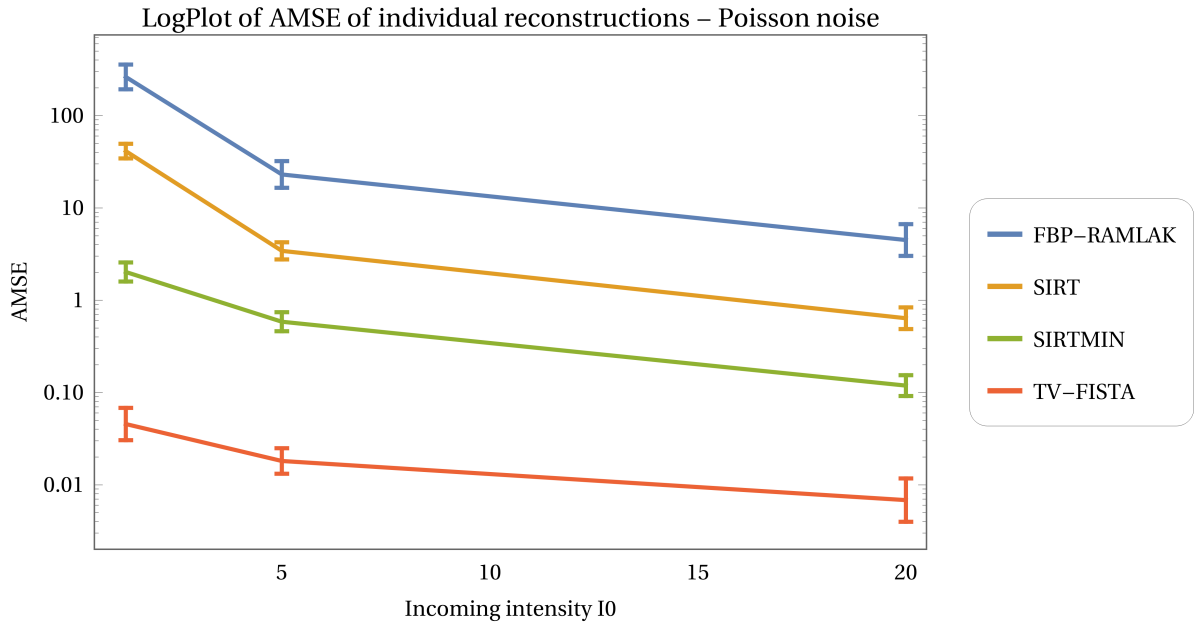


Figure 25: Log plot of AMSE scores of 200 randomly oriented reconstructions with respect to the ground truth for FBP (with RAMLAK filter), SIRT and SIRTMIN (200 iterations) and TV-FISTA for Poisson noise on the projections with incoming intensity $I_0 = 1.25, 5.0, 20.0$.

We see that TV-FISTA and SIRTMIN perform best once again which is in accordance with our earlier observations. It is our opinion that SIRTMIN and TV-FISTA prove superior in dealing with the limited angular range. Furthermore, the quality of the reconstructions for high noise levels seems to be ordered from worst to best as FBP, SIRT, SIRTMIN, TV-FISTA. However, none of the individual reconstructions suitably recover the ribosome features. As mentioned, we think that due to the tendency to remove salt-and-pepper noise and missing-wedge artifacts in TV-FISTA, it seems that the TV reconstructions would be ideally suited to estimate the orientation of the ribosome on.

4.1.2 Testing combined subtomogram averaging approach

Next we will test how the reconstruction algorithms interplay with the subtomogram averaging method. We hypothesize that for sufficiently large sets of averages the noise will be averaged out but most of the signal will still be present for the least intrusive methods such as FBP and SIRT. We hypothesize that certainly TV-FISTA will perform worse due to the influence of the regularization term prior to averaging. We hypothesize that TV regularization will drive the TV-FISTA to create several connected pockets of the same grey value, thereby removing signal which cannot be recovered by averaging.

In this experiment we will use the same dataset as in the previous section but we will average k

of the individual reconstruction for $k \in \{5, 10, 25, 50, 100\}$ to determine which methods obtain the best averages. We use the original angles used to create the dataset so the alignment of the k reconstructions is perfect. In later experiments we will test what happens when the averages are taken with imperfect rotations up to interpolation error. We chose k random reconstructions 10 times for each method and each k. In total we have four reconstruction methods, 4 noise levels and 5 different k which would amount to 80 different example pictures. We will only show a few relevant combinations to illustrate the meaningful results.

Firstly, we consider the noiseless reconstructions to investigate missing wedge problems. Since FBP and SIRT are linear we can already predict that a large average of randomly oriented reconstructions is the same as one big reconstruction. In Figure 27 we have included a middle slice of a sample average of 10. In Figure 26 we have included a middle slice of the ground truth.

Figure 26: Slice 145/320 of the ground truth ribosome phantom.

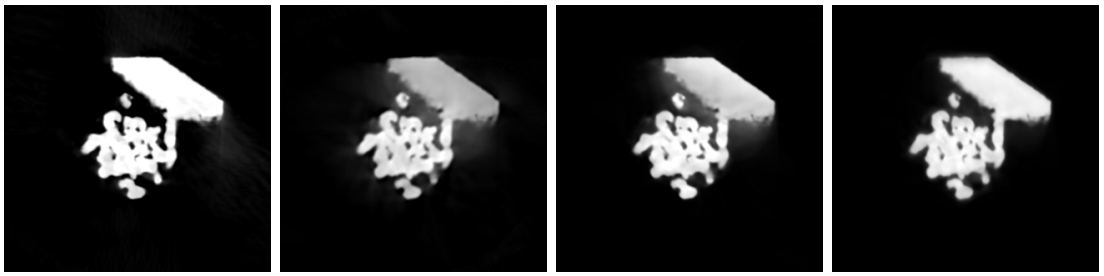


Figure 27: Slice 145/320 of averaged reconstruction with 10 averages and no noise. From left to right we have FBP-RAMLAK, SIRT, SIRTMIN and TV-FISTA reconstructions.

For a batch of 10, the influence of the missing wedge seems to be quite low. We still see some artifacts for FBP and a vague halo for SIRT and SIRTMIN. We see some blurring around the membrane for TV-FISTA. Without displaying the pictures, for $k = 25$ all the methods barely produced any artifacts and for higher averages all the reconstructions were of a very high quality. In Figure 28 we see that for SIRT, SIRTMIN and TV-FISTA the influence of missing wedge has mostly disappeared for 25 averages or more. Interestingly, the plot confirms that FBP eventually is the strongest algorithm since it should average to a perfect reconstruction and there is no regularization. Note that in Figure 28 the mean and standard deviation of the AMSE is taken over the 10 averaged reconstructions.

Secondly, we consider the Gaussian-100 reconstructions to investigate what happens to the high

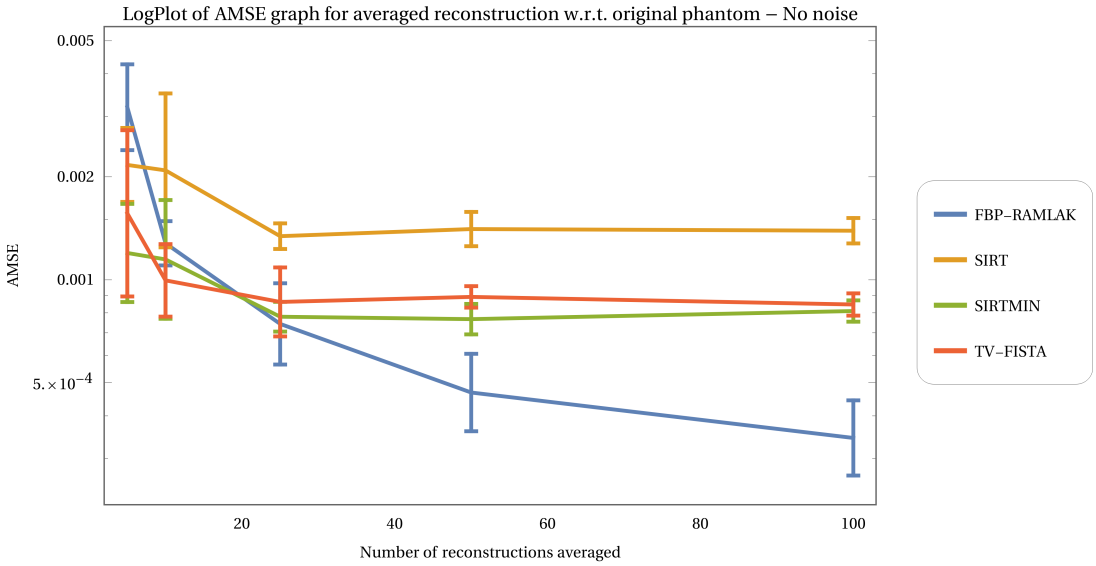


Figure 28: Log plot of AMSE scores 10 batches of k-averaged reconstructions with respect to the ground truth for FBP (with RAMLAK filter), SIRT and SIRTMIN (200 iterations) and TV-FISTA for $k = 5, 10, 25, 50, 100$ without noise.

noise reconstructions. In the first row of Figure 29 we have included a middle slice of a sample average of 10.

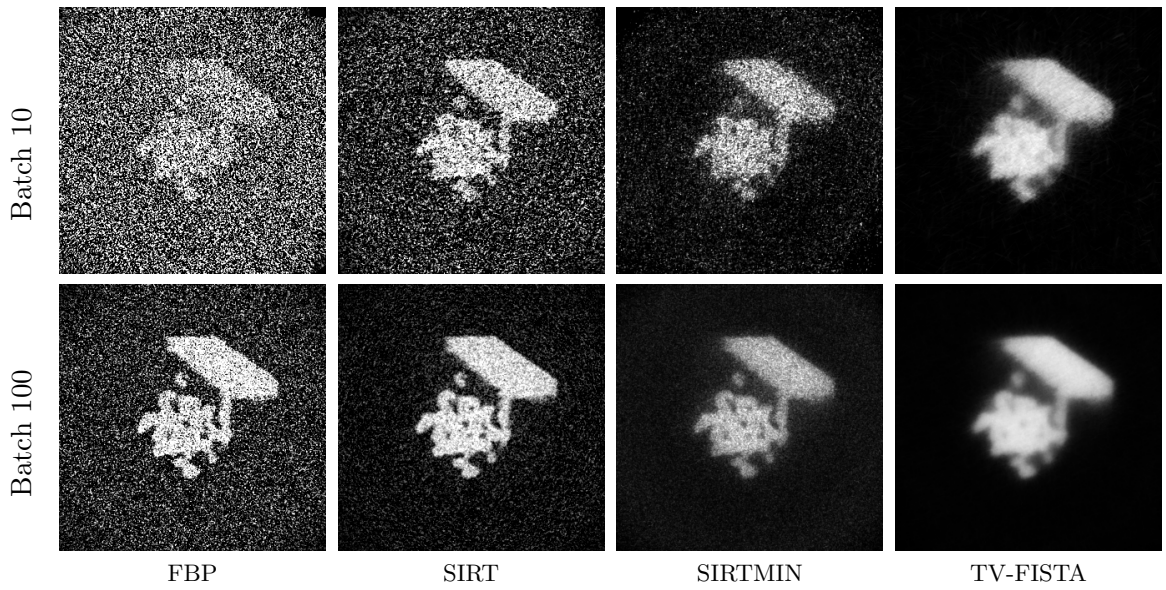


Figure 29: Slice 145/320 of averaged reconstruction with Gaussian noise $\sigma = 100.0$ on the projections. First row 10 averages, second row 100 averages. For both rows, from left to right we have FBP-RAMLAK, SIRT, SIRTMIN and TV-FISTA reconstructions.

For a batch of 10, a large amount of noise remains in the FBP and SIRT reconstructions. However, there are already significant improvements with respect to the single reconstructions. There is not too much of a noise cloud in the SIRTMIN average but we cannot yet extract smaller features from the phantom. The TV-FISTA average has practically no noise but all the small features have been blurred. In the second row of Figure 29 we have included a middle slice of a sample average of 100.

For a batch of 100, we see significant improvements in the averaged reconstructions. FBP, SIRT and SIRTMIN clearly show the phantom and the smaller features are clearly present. Interestingly, the TV-FISTA reconstruction, albeit it entirely noiseless, is still too blurred to make out a lot of the ribosome parts. Arguably, it appears as if the regular SIRT algorithm is better able to preserve the smaller features than the SIRTMIN algorithm. Furthermore, SIRT has a much less prevalent noise cloud than FBP in the vacuum.

These conclusions are partially reflected in the quantitative analysis. In Figure 30 we plotted the AMSE score for the Gaussian noise with $\sigma = 100.0$. We see that the quality of the reconstruction increases with more averages as was noted. According to the AMSE, TV-FISTA is still significantly better than SIRTMIN. Apparently the total absence of noise cloud in the background for TV-FISTA outweighs the slightly more accurate features in SIRTMIN in the AMSE computation. Lastly, in Figure 31 we have included a middle slice for 10 and 100 averages for the $\sigma = 250.0$ case.

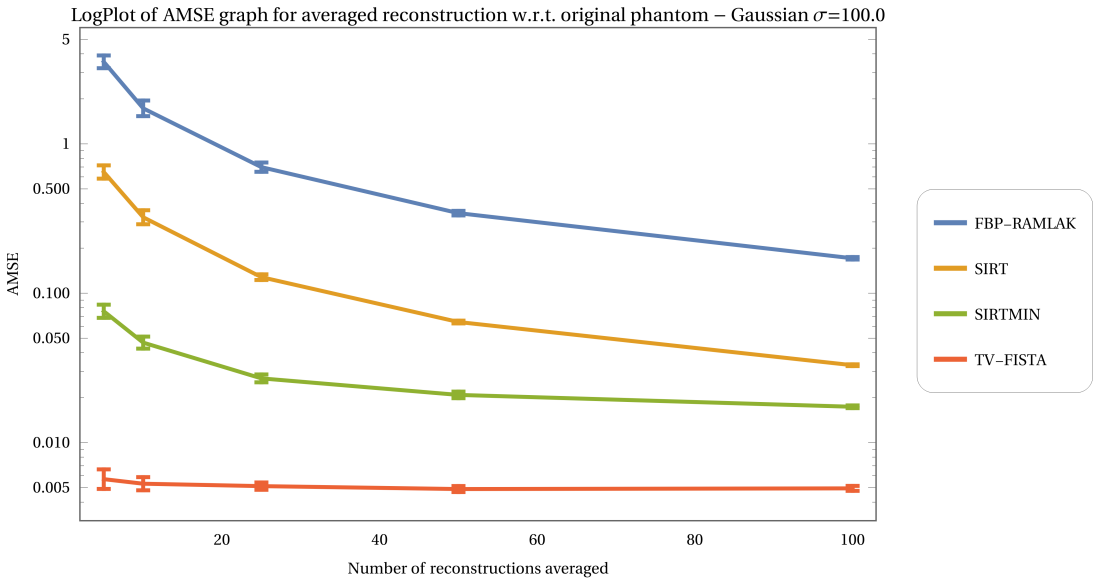


Figure 30: Log plot of AMSE scores 10 batches of k-averaged reconstructions with respect to the ground truth for FBP (with RAMLAK filter), SIRT and SIRT-MIN (200 iterations) and TV-FISTA for $k = 5, 10, 25, 50, 100$ with Gaussian noise $\sigma = 100.0$.

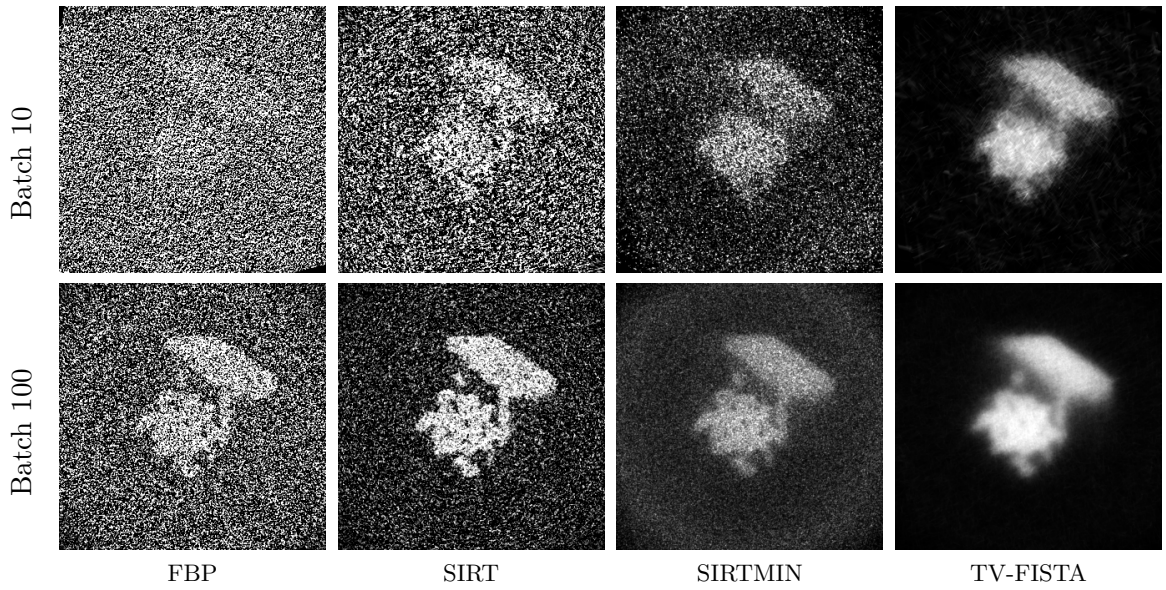


Figure 31: Slice 145/320 of averaged reconstruction with Gaussian noise $\sigma = 250.0$ on the projections. First row 10 averages, second row 100 averages. For both rows, from left to right we have FBP-RAMLAK, SIRT, SIRTMIN and TV-FISTA reconstructions.

In the $\sigma = 250.0$ case the same behaviour is exhibited as in the $\sigma = 100.0$ case. Indeed, the TV-FISTA average remains blurry with respect to SIRT and SIRTMIN as was hypothesized. It is up for debate whether SIRT of SIRTMIN is the better reconstruction here. In Figure 32 we plotted the AMSE score for the Gaussian noise with $\sigma = 250.0$. The conclusions are the same as in the Gaussian 100 case.

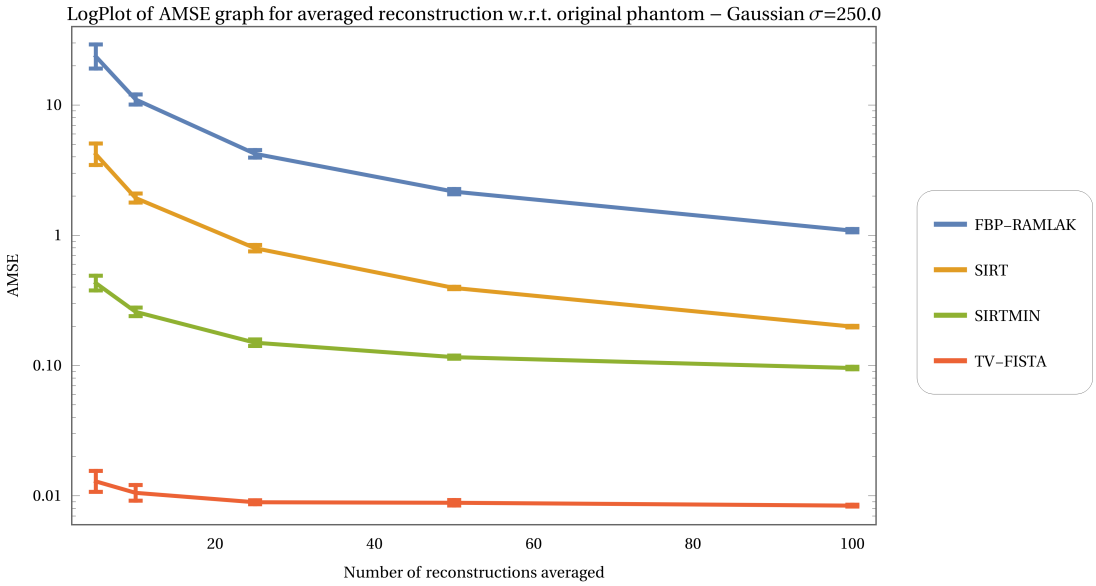


Figure 32: Log pot of AMSE scores 10 batches of k-averaged reconstructions with respect to the ground truth for FBP (with RAMLAK filter), SIRT and SIRTMIN (200 iterations) and TV-FISTA for $k = 5, 10, 25, 50, 100$ with Gaussian noise $\sigma = 250.0$.

These experiments were conducted with Gaussian noise with mean zero. Before we draw final conclusion we will repeat the experiment with Poisson noise for a more realistic simulation. We will only test SIRTMIN and TV-FISTA since these are deemed to be most successful. Since we are checking previous results for a different noise profile, we will only report the images for the hardest noise level $I_0 = 1.25$. In figure 33 we have included a middle slice of a sample average with 10 averages and 100 averages.

For 100 averages we can detect the ribosome object quite well in all reconstructions. It is important to keep in mind that we used the original rotation angles. In contrast to the Gaussian-250 averages, FBP, SIRT and SIRTMIN have a significantly reduced noise cloud. It seems that the smaller features would be harder to extract from the FBP and TV-FISTA reconstructions than for both SIRT methods. In Figure 34 we have plotted the AMSE score for several averages for the intermediate case of $I_0 = 5.0$ (since we do not show those pictures) to illustrate that the Poisson experiments exhibit similar behaviour to the Gaussian experiments.

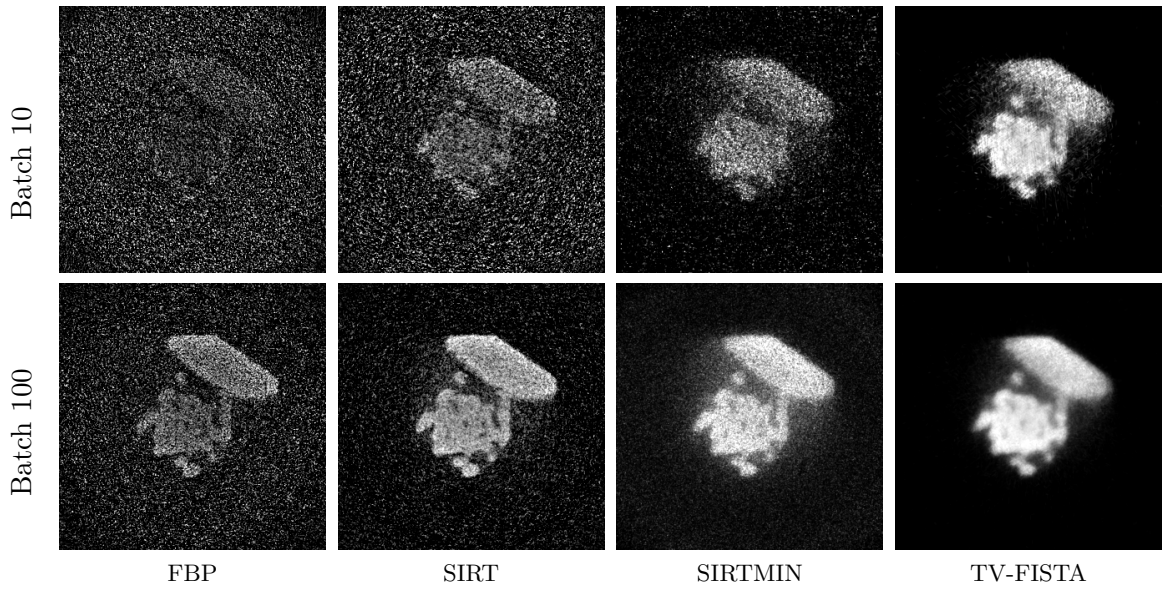


Figure 33: Slice 145/320 of averaged reconstruction with Poisson noise $I_0 = 1.25$ on the projections. First row 10 averages, second row 100 averages. For both rows, from left to right we have FBP-RAMLAK, SIRT, SIRTMIN and TV-FISTA reconstructions.

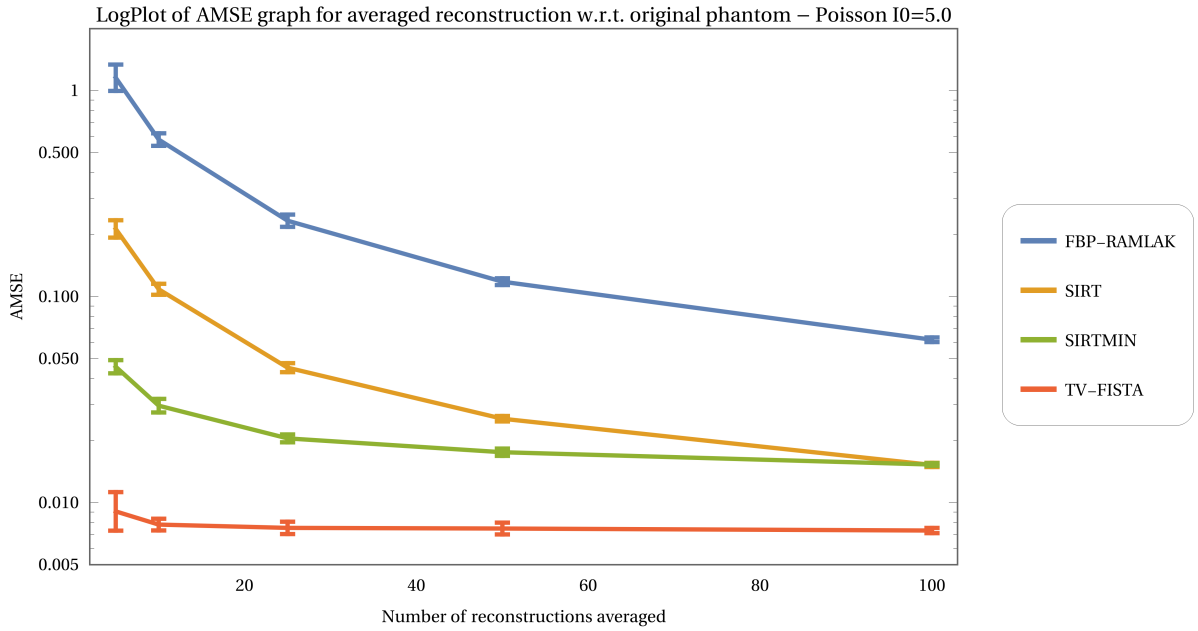


Figure 34: Log plot of AMSE scores 10 batches of k-averaged reconstructions with respect to the ground truth for FBP (with RAMLAK filter), SIRT and SIRTMIN (200 iterations) and TV-FISTA for $k = 5, 10, 25, 50, 100$ with Poisson noise $I_0 = 5.0$.

To conclude, it seems that missing wedge is usually not problematic for averaging methods. For

small averages or single reconstructions, TV is best but it will not recover the smaller features due to blurring effects as a consequence of regularization. Furthermore, we see that for large averages, SIRT or SIRTMIN has sufficiently low noise to be useful but also keeps the features of the ribosome more intact than TV-FISTA.

This conclusion is only partially backed up by quantitative analysis. The AMSE scores would have us conclude that TV-FISTA is the superior algorithm when combined with subtomogram averaging. However, we feel that these quantitative measures probably overemphasize the importance of setting the background to zero and hence penalize methods such as SIRT and SIRTMIN. We therefore stick to our human judgment that SIRT and SIRTMIN perform better for the 100 averages because they recover the finer features of the ribosome.

To decide between these somewhat equal reconstructions we would say that SIRTMIN is slightly better which is backed up by the quantitative analysis. In addition, if we look at the low number of averages and the individual reconstructions, we feel SIRTMIN outperforms traditional SIRT. Therefore, it seems that SIRTMIN is more robust and would perform better in general cases when the number of averages is small. Also we saw that SIRTMIN suffers less from missing wedge artifacts. All in all, we conclude that SIRTMIN is the best performing algorithm out of the four that we tested to perform cryo-ET reconstructions when combined with the subtomogram averaging method.

It stands to reason that a strongly regularized method like TV-FISTA will remove some of the signal for the higher noise reconstructions. Apparently, small features cannot simply be retrieved once they have been lost by regularization. This leads to a blurred version of the original phantom even after averaging. FBP, SIRT and SIRTMIN keep more of the signal hence they are also more noisy for low averages. However, for large batches the noise averages out due to statistics and more signal remains.

We hypothesize that pre-processing with TV-FISTA would allow the algorithm to more accurately align the reconstructions. Subsequently averaging these reconstructions appears to lead to a blurry image though. From these experiments it seems reconstructing again with SIRT or SIRTMIN and then averaging using these improved angles would be a very good approach that combines the strong sides of both algorithms. These considerations have lead to a preliminary reconstruction pipeline which is outlined in Figure 35.

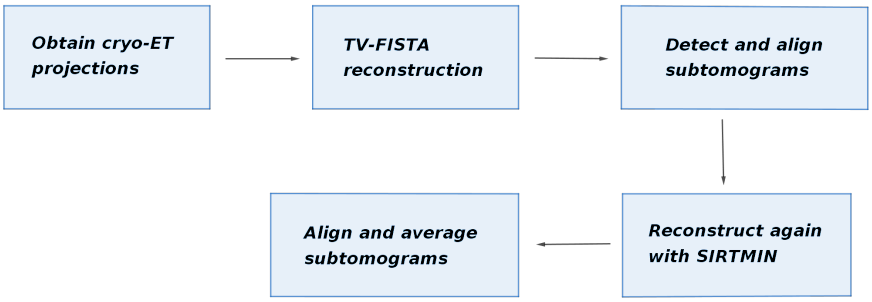


Figure 35: Recommended reconstruction pipeline based on the experiments conducted so far.

4.2 Robustness of reconstruction methods and sensitivity to angular accuracy

We concluded in the previous section that using SIRTMIN as reconstruction algorithm produces the best final reconstruction for larger averages. However, we assumed that the angles were known to great precision. In this section we will explore the effect of misalignment on the quality of the averages.

To test the effect of misalignment we took a set of original angles $(\alpha_1, \alpha_2, \alpha_3)$ and drew from a distribution $\hat{\alpha}_i \sim \mathcal{N}(\alpha_i, \theta)$ for $\theta = 0, 0.03, 0.1, 0.3$. We saw in our previous tests that there appears to be no significantly different results when we use Poisson noise. For convenience we therefore only reobtained the averages for the Gaussian noises. Also we only took k averages for $k = 10, 25, 50$ to reduce computation time. In Figure 36 we show example slices of faulty averages for 10-averages and for no noise.

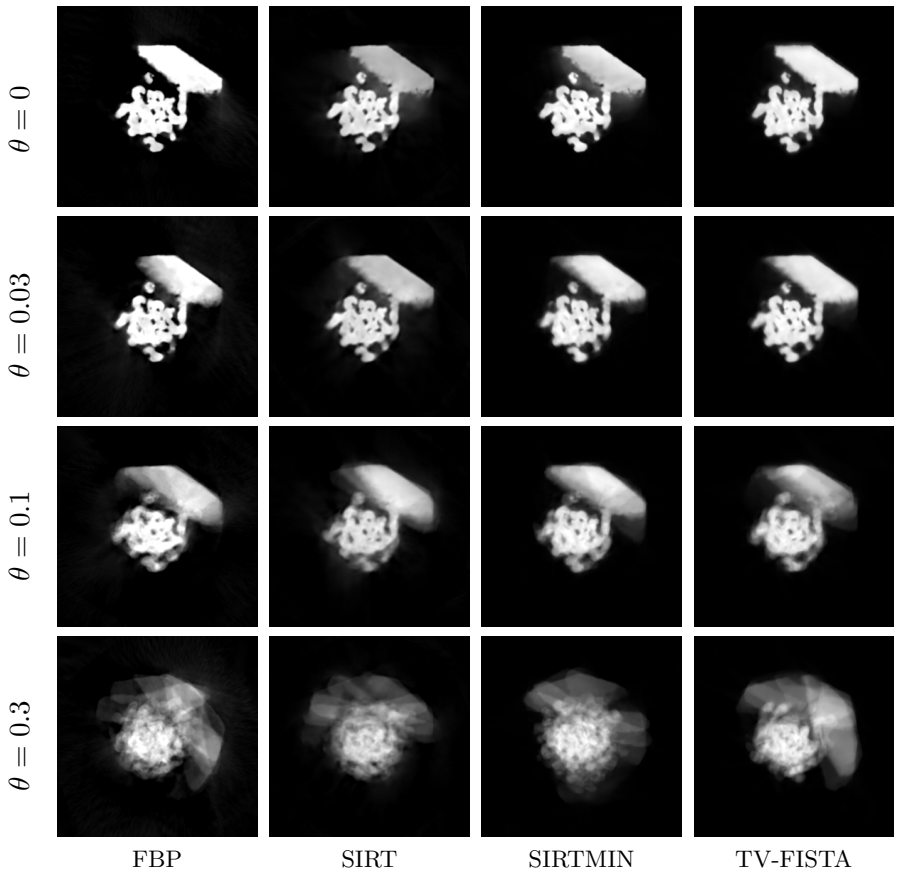


Figure 36: Slice 145/320 of 10-averaged reconstructions without noise with from left to right per row FBP, SIRT, SIRTMIN and TV-FISTA. First row: perfect angles, second row: $\theta = 0.03$, third row: $\theta = 0.1$ and fourth row: $\theta = 0.3$.

We see that the averages get progressively more blurred as is to be expected. For the $\theta = 0.3$ the error is so large that the mean axis is not even the same anymore for 10 random averages. An interesting observation is that although the reconstructions are obviously very poor for the larger errors, the missing wedge artifacts have disappeared regardless. Therefore, we see that subtomogram averaging, even for small averages, mitigates missing wedge problems regardless of the angle accuracy. This is to be expected considering that mean-zero uniformly rotated samples should still provide a uniform sampling of the projection angles in identical particle analysis.

In Figure 37 we show example slices of faulty averages for a sample size of 50 and Gaussian noise for $\sigma = 100.0$. We see the same noise cloud regardless of the angle deviation. This makes sense since the noise is independent with mean zero for each pixel and therefore unrelated to the rotation angles.

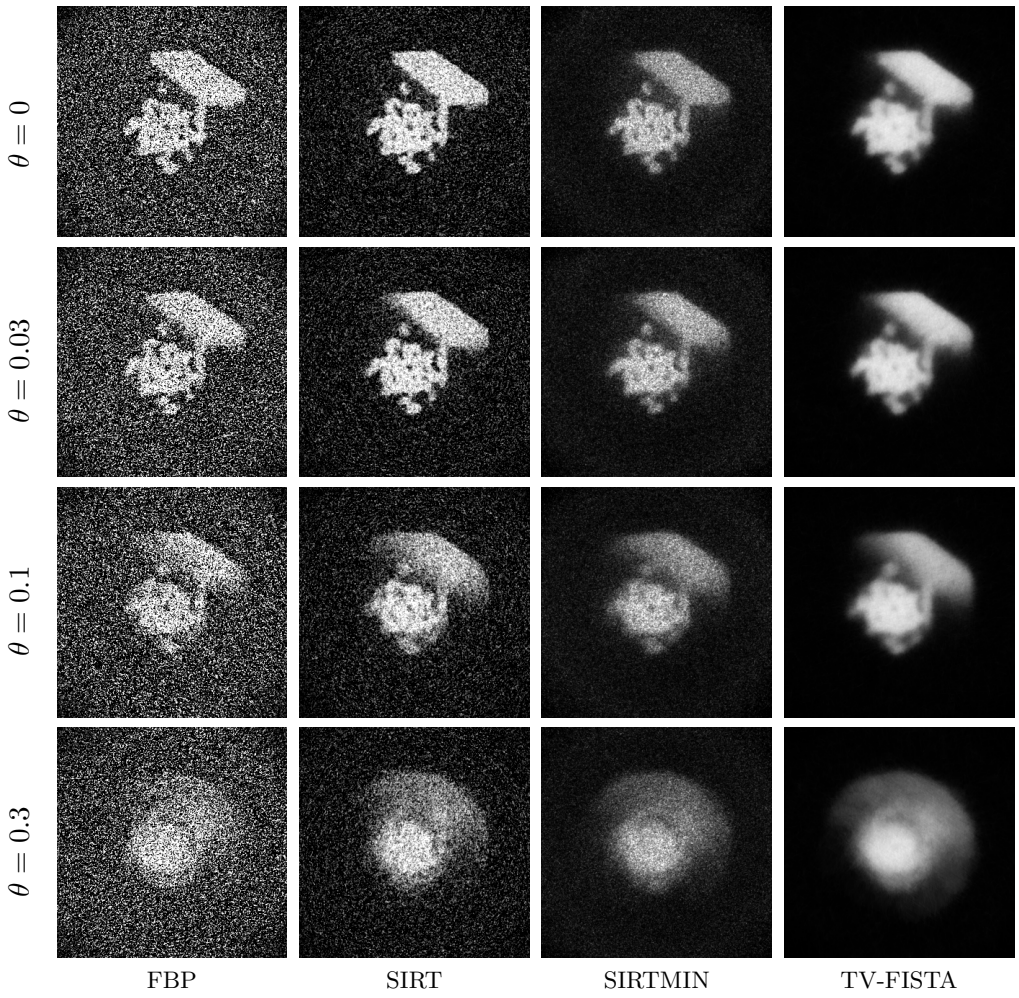


Figure 37: Slice 145/320 of 50-averaged reconstructions with Gaussian noise $\sigma = 100.0$ with from left to right per row FBP, SIRT, SIRTMIN and TV-FISTA. First row: perfect angles, second row: $\theta = 0.03$, third row: $\theta = 0.1$ and fourth row: $\theta = 0.3$.

Without displaying the pictures, a plot of the AMSE scores compared to the ground truth for each θ for Gaussian noise $\sigma = 10.0$ is provided in Figure 38.

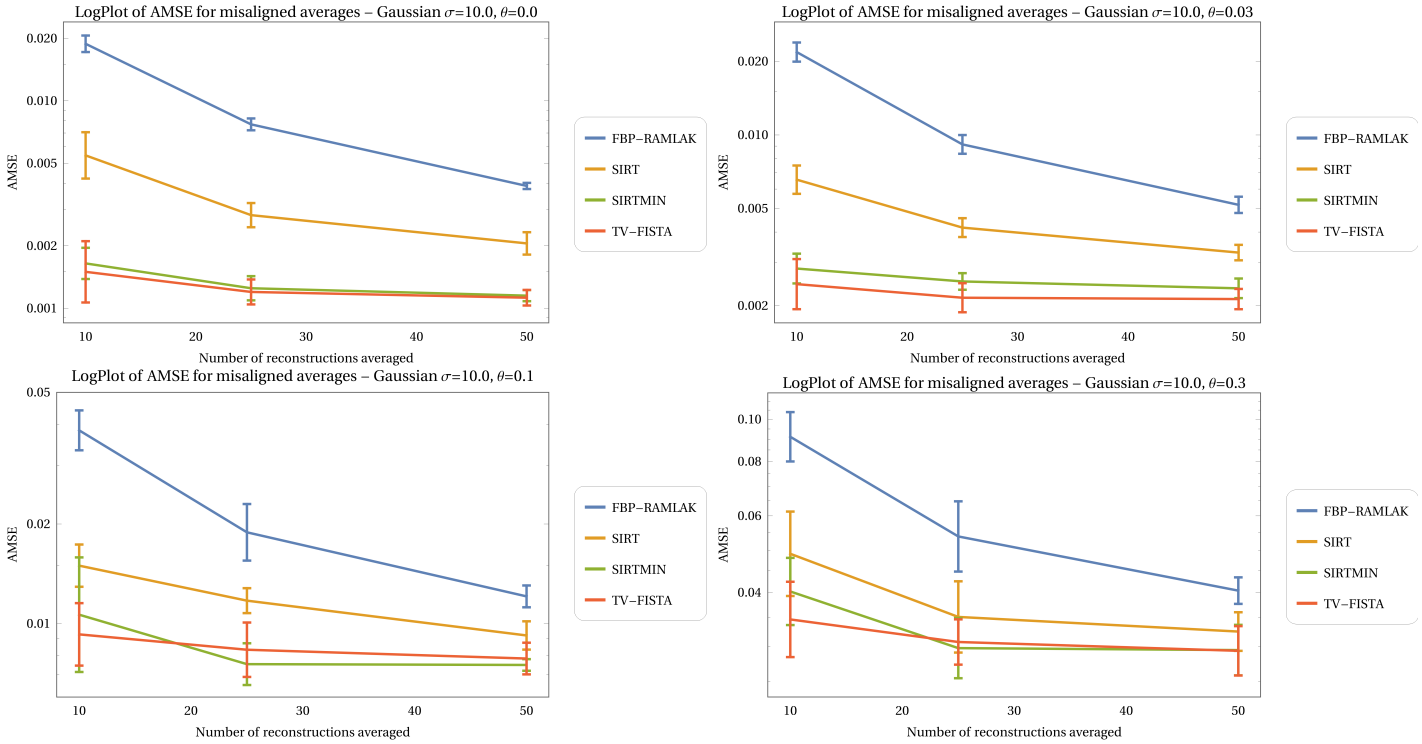


Figure 38: Log plots of AMSE scores 10 batches of k-averaged reconstructions with respect to the ground truth for FBP (with RAMLAK filter), SIRT and SIRTMIN (200 iterations) and TV-FISTA for $k = 10, 25, 50$ with Gaussian noise $\sigma = 10.0$. Upper left is perfect angles, upper right is $\theta = 0.03$, lower left is $\theta = 0.1$ and lower right is $\theta = 0.3$.

The performance deteriorates with the inaccuracy of the angles (as is expected) but we see that the relative performance of the reconstruction methods remains the same. The gaps between the errors for the different methods decreases with θ. This is to be expected since if θ increases, the averaging error starts to dominate. For the Gaussian noise $\sigma = 250.0$ we plotted the AMSE in Figure 39.

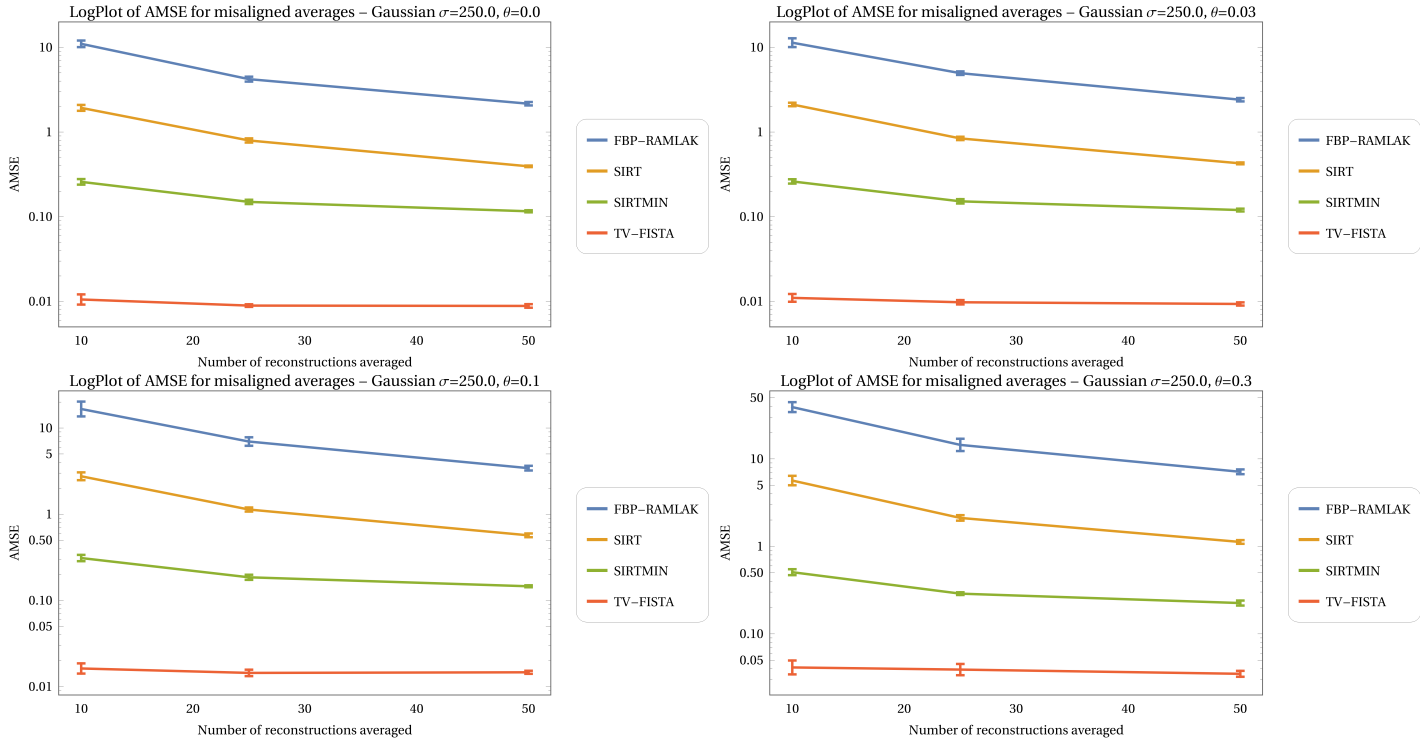


Figure 39: Log plots of AMSE scores 10 batches of k-averaged reconstructions with respect to the ground truth for FBP (with RAMLAK filter), SIRT and SIRTMIN (200 iterations) and TV-FISTA for $k = 10, 25, 50$ with Gaussian noise $\sigma = 250.0$. Upper left is perfect angles, upper right is $\theta = 0.03$, lower left is $\theta = 0.1$ and lower right is $\theta = 0.3$.

The relative inaccuracies stay the same but the performance does not deteriorate nearly as much as for $\sigma = 10.0$. *This suggests that we mainly see the effects of averaging out the noise cloud and not so much the reconstructed object.* Again, the value of quantitative analysis is limited here as this suggests that the AMSE score is dominated by noise.

The quality of the averaged reconstruction decreases with the error in the angles. It is therefore crucial that we can determine these angles up to a reasonable precision. It seems that if we can determine the angles within ± 0.1 radians, the reconstructions will be quite reasonable for 50 averages. If the error is larger the results become significantly blurred. That said, larger sets of averages may mitigate the larger error eventually.

As for the robustness of each separate method, it is hard to conclude anything regarding the increase in error w.r.t. θ. We already mentioned that the reduced gaps as θ increases seems to suggest that we measure the effects of averaging the salt-and-pepper noise. The overall trends of the graphs in Figures 38 and 39 remained the same. There also does not appear to be a discernible difference in the deteriorating images from method to method. We therefore draw no conclusions on the relative robustness of the methods.

4.3 A new ensemble-reconstruction method for subtomogram averaging

From our previous experiments we have concluded that TV-FISTA is best suited to finding individual reconstructions and SIRTMIN performs best in conjunction with the subtomogram

averaging method. However, each reconstruction was still created from a suboptimal set of projections. There were missing wedge problems and the data was undersampled and noisy.

Under the assumption that we can determine the rotated angles of the ribosome quite well, we can wonder what happens when we interchange the order of combining the data. This idea has led to the ensemble reconstruction method outlined in section 3.3. For linear methods such as FBP this should in theory make no difference since we would have equality up to a multiplicative constant(disregarding the translations in subtomogram averaging)

$$\begin{aligned}
\mathbf{X}^*_{ensemble} &= \varphi_{REC}\left(\bigcup_{i=1}^{N}\bigcup_{k=1}^{N_\theta}(\tilde{R}\mathbf{r}_{ik}, \tilde{R}\mathbf{c}_{ik}, \tilde{R}\mathbf{u}_{ik}, \tilde{R}\mathbf{v}_{ik}, \mathbf{p}_{ik})\right) \\
&= \sum_{i=1}^{N}\varphi_{REC}\left(\bigcup_{k=1}^{N_\theta}(\tilde{R}\mathbf{r}_{ik}, \tilde{R}\mathbf{c}_{ik}, \tilde{R}\mathbf{u}_{ik}, \tilde{R}\mathbf{v}_{ik}, \mathbf{p}_{ik})\right) \\
&= \sum_{i=1}^{N} R[\alpha_i, \beta_i, \gamma_i](\mathbf{X}_i) \\
&= N\frac{1}{N}\sum_{i=1}^{N} R[\alpha_i, \beta_i, \gamma_i](\mathbf{X}_i) = N\mathbf{X}^*_{post}.
\end{aligned}$$

However, for non-linear methods like SIRTMIN and TV-FISTA this could make a difference. For TV-FISTA, each poor reconstruction is so noisy that we need a large regularization parameter. Therefore, the resulting reconstruction is quite blurry and we have lost a lot of signal. Averaging afterwards cannot reintroduce knowledge about small features and hence we have a blurry average (this is what we have observed). However, if we do a single reconstruction with more projections we may be able to use a very small regularization parameter since the algorithm has more data to work with. Therefore, the smaller features may not be lost and we will have a better reconstruction.

We compare the results for 5 and 50 averages for the Gaussian-250 and Poisson-1.25 datasets for SIRTMIN since it is deemed the strongest algorithm based on our earlier findings. Furthermore, the currently used library for TV-FISTA [30] does not support reconstructions with the 3D ASTRA geometry that is used for the ensemble method. In Figure 40 we see a middle slice for the SIRTMIN reconstructions for the Gaussian-250 data.

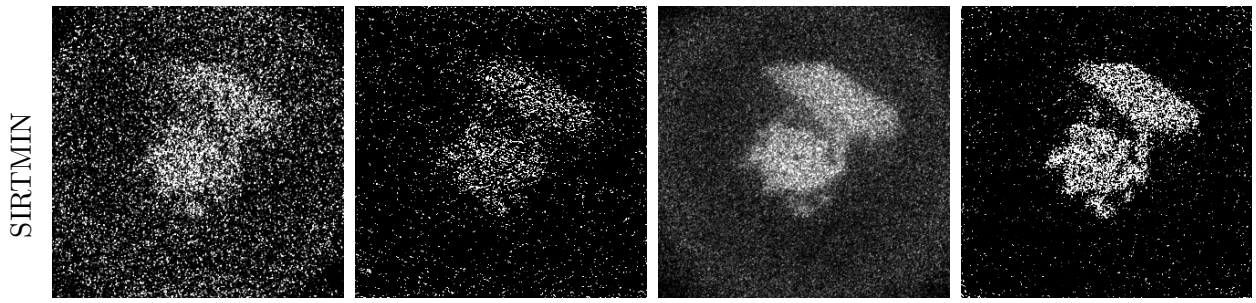


Figure 40: Slice 145/320 of averaged reconstruction with 5 averages; (a) averaging after reconstructing, (b) ensemble method. 50 averages; (c) averaging after reconstructing, (d) ensemble method. The tests were done for with Gaussian noise $\sigma = 250$ on the projections.

In figure 41 we see a middle slice for the SIRTMIN reconstructions for the Poisson-1.25 data. In our opinion, the combined reconstructions are better, certainly the noise cloud has practically disappeared in the combined reconstructions.

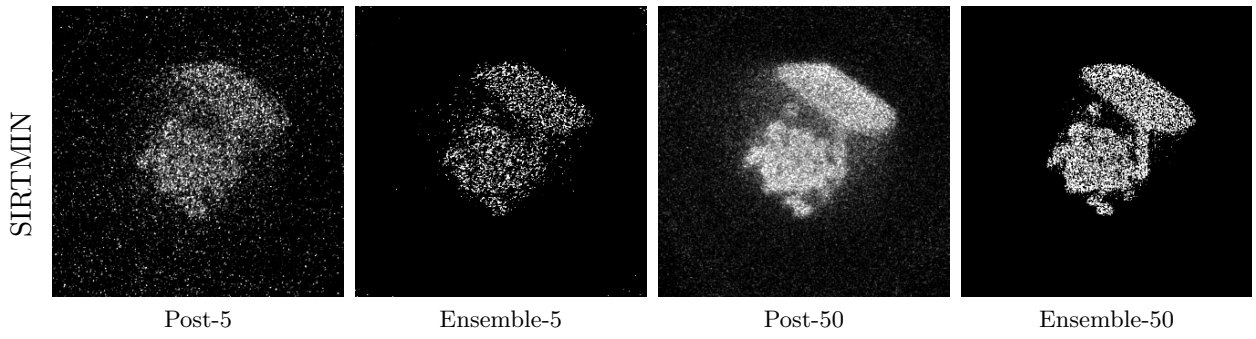


Figure 41: Slice 145/320 of averaged reconstruction with 5 averages; (a) averaging after reconstructing, (b) ensemble method. 50 averages; (c) averaging after reconstructing, (d) ensemble method. The tests were done for with Poisson noise $I_0 = 1.25$ on the projections.

4.3.1 Sensitivity of ensemble-approach to angular accuracy

Although the performance for perfect angles seems to have increased, the new ensemble-method may be more sensitive to faulty angle estimation. We tested the sensitivity to angular estimation as described in section 4.2. To be clear, the faulty rotation angles $(\alpha_x, \alpha_y, \alpha_z)$ are provided as input to the flexible geometry as if they are the correct angles. The algorithm then determines the (incorrect) scanning geometry that the angles define for the projections. We plotted the AMSE scores compared to the ground truth for each noise level in Figure 42.

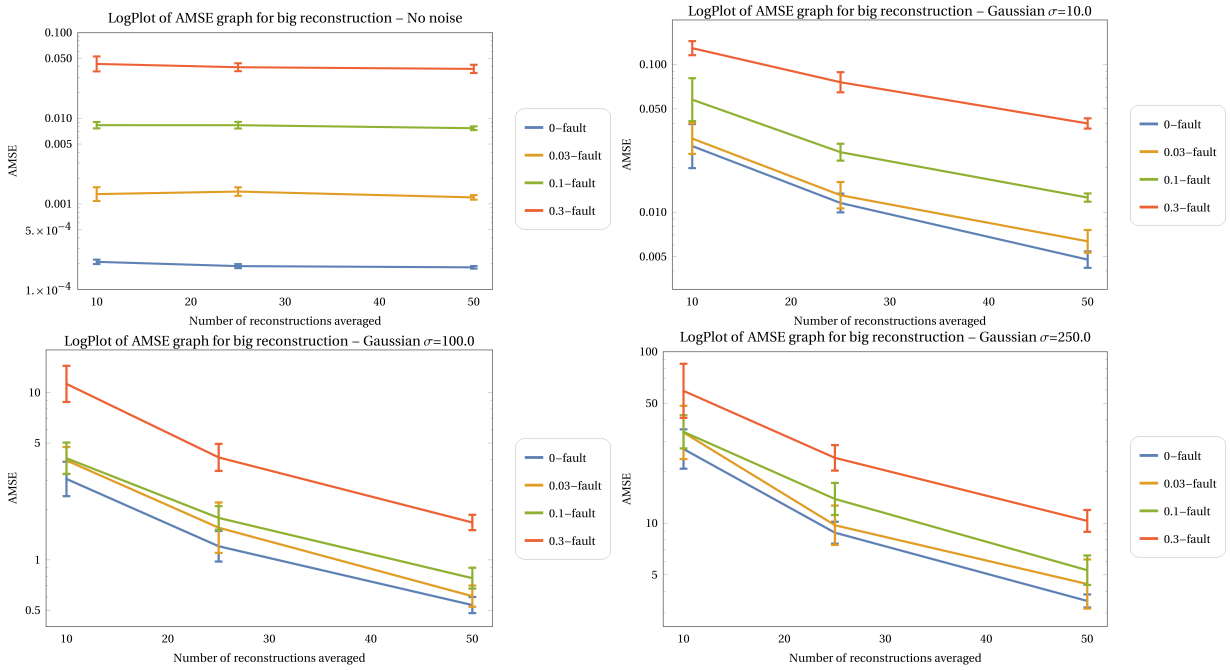


Figure 42: Log plots of AMSE scores; 10 batches of k-averaged reconstructions (for $k = 10, 25, 50$) with respect to the ground truth for SIRTMIN (200 iterations) implemented in the ensemble method. The four plots per figure are for $\theta = 0, 0.03, 0.1, 0.3$. Upper left is noiseless, upper right is Gaussian noise $\sigma = 10.0$, lower left is Gaussian noise $\sigma = 100.0$ and lower right is Gaussian noise $\sigma = 250.0$.

As expected, we see that the least faulty reconstructions obviously perform best and that the error decreases if we take more projections for our big average. Interesting is that for the noiseless reconstructions it seems to make no difference whether we take 10 or 50 sets of projections. Apparently, 10 sets is already sufficient to mitigate missing wedge problems.

It seems that the combined approach works well in the perfect angle case. In Figure 43 we plotted the AMSE scores of the ensemble method and the post-averaging method against θ for each of the noise levels.

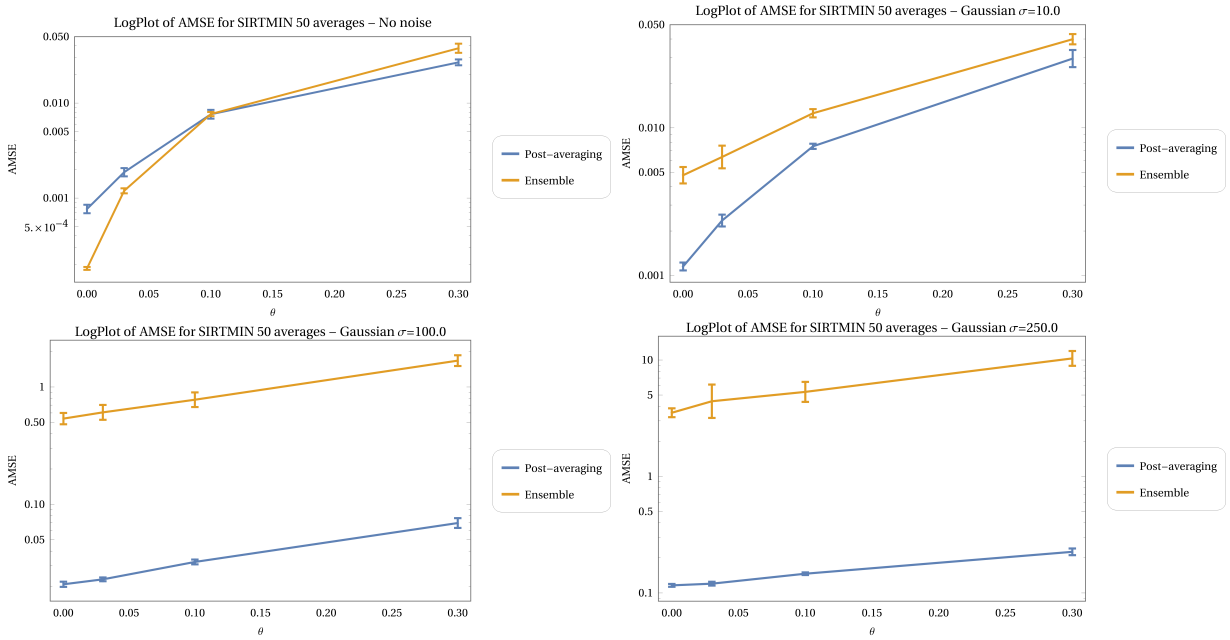


Figure 43: Log plots of AMSE scores; 10 batches of 50-averaged reconstructions with respect to the ground truth for SIRTMIN (200 iterations) comparing post-averaging to the ensemble method for $\theta = 0, 0.03, 0.1, 0.3$. Upper left is noiseless, upper right is Gaussian noise $\sigma = 10.0$, lower left is Gaussian noise $\sigma = 100.0$ and lower right is Gaussian noise $\sigma = 250.0$.

In the noiseless case both methods are similar but for the other noise levels the ensemble method scores significantly worse. The output range of the ensemble method is different than that of the ground truth. The discrepancy is so large that it might be that the AMSE scaling does not properly rescale the range of the ensemble reconstructions. We see in Figure 44 that the ensemble method reconstruction are made up of lots of dots. Perhaps this could explain why the scaling in the AMSE calculation appears to be faulty. Apart from this hypothesis we are at a loss why the quantitative performance is so poor. In any case, these results do not seem indicative of actual performance and we should focus on the images instead.

We saw a few examples for perfect angle reconstructions in Figures 40 and 41. In Figure 44 we show the post-average reconstructions and the combined reconstructions for $\theta = 0.03, 0.1, 0.3$ for Gaussian noise $\sigma = 100.0$ and 50-averages.

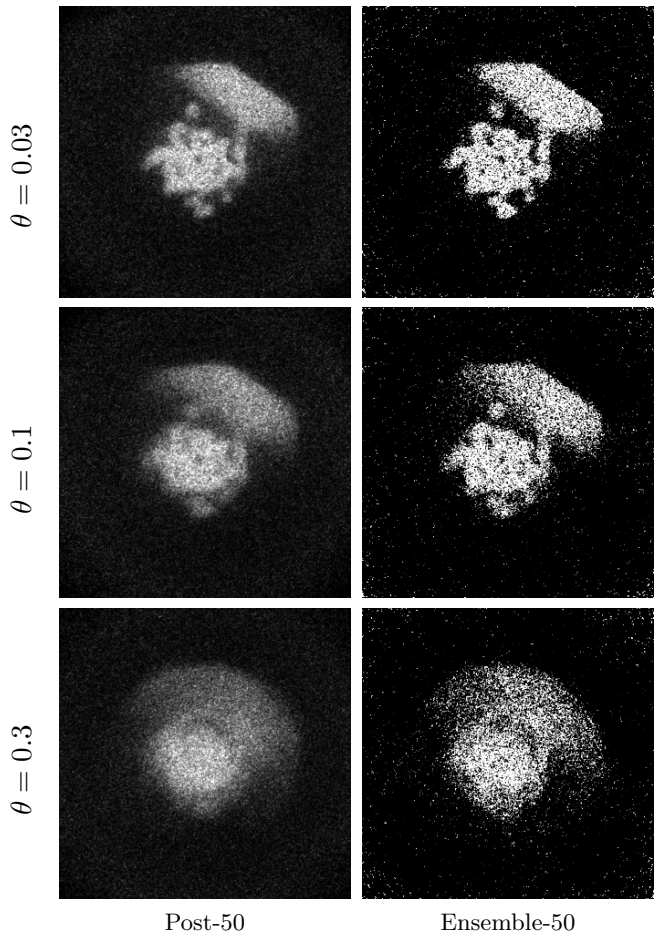


Figure 44: Slice 145/320 of averaged reconstruction with 50 averages and with Gaussian noise $\sigma = 250.0$ on the projections. The left column is the result of averaging after reconstructing and the right column is when we combine the projection data for one large reconstruction.

If we disregard the quantitative analysis the combined reconstruction seems to perform better than averaging after reconstructing if we look at the images in Figures 40, 41 and 44. The noise cloud is almost none existent when we combine 50 sinograms. Furthermore, in our opinion the features are more prevalent in the combined reconstruction as well. Therefore, we conclude that using SIRTMIN in a combined reconstruction is the best approach to perform cryo-ET reconstructions when we have access to several identical particle tomograms.

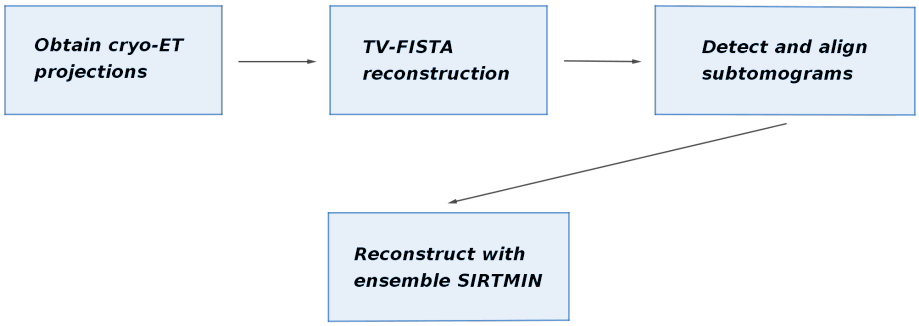


Figure 45: Recommended pipeline for tomographic reconstruction phase.

To conclude, based on these experiments on simulated the data, we recommend the following tomographic reconstruction pipeline 1) create strongly regularized TV-FISTA reconstructions; 2) use the expectation-maximization algorithm to determine the rotation angles for each subtomogram; 3) use the ensemble-reconstruction method with SIRTMIN to create one large reconstruction using the original projections to obtain the final tomogram of the object of study. .

CHAPTER 2

DEEP LEARNING AND DATA REDUCTION

5 Mathematics of deep learning

As mentioned in the introduction, the goals of this thesis are two-pronged. The second goal was to attempt to determine whether a reconstruction of a certain quality can be obtained with less data by using MS-D neural networks to denoise the reconstructed volumes. We think performance can be gained, certainly on the amount of data required, with the use of deep learning to denoise the subtomograms at certain points in the pipeline. This will be the focus of the following chapters in this thesis

In this thesis we will be using a mixed-scale dense convolutional neural network (MS-D network) to tackle the high noise problems in cryo-ET. MS-D networks have shown to improve tomographic reconstructions from limited data and have outperformed standard encoder-decoder networks [33]. Furthermore, MS-D networks have an architecture that is able to achieve accurate results with relatively few parameters and consists of a single set of operations, making it easier to implement, train, and apply in practice, and automatically adapts to different problems [32]. Compared to popular existing architectures for several segmentation problems, the proposed architecture is able to achieve accurate results with fewer parameters, with a reduced risk of overfitting the training data [32].

In this section we will build a mathematical treatise of neural networks from the ground up. For the reader who is familiar with neural networks but unfamiliar with MS-D networks we recommend reading only section 5.4.

5.1 Basics on neural networks

In this thesis we will be using a mixed-scale dense convolutional neural network (MS-D network) to tackle the high noise problems in cryo-ET. Before we consider convolutional neural networks (CNNs) and the different ways of implementing convolutional filters in the architecture, we will briefly cover the basics of feedforward networks and the workings of the perceptron.

The most common neural network is the *feedforward neural network* or *multilayer perceptron* (MLP). As the name suggests the basic unit of the MLP is the *perceptron*. A basic perceptron takes a vector $\mathbf{x} \in \mathbb{R}^n$ as input, takes the inner product with some weight vector $\mathbf{w} \in \mathbb{R}^n$ and adds a bias $b \in \mathbb{R}$. Then the resulting value is fed to a (non-linear) activation function $\sigma : \mathbb{R} \to \mathbb{R}$ to produce the output value

$$f_{perc}(\mathbf{x}) = \sigma(\mathbf{x}^T\mathbf{w} + b) = \sigma\left(b + \sum_i x_i w_i\right).$$

A perceptron can be interpreted as a linear regression followed by a non-linear smoothing or cut-off function. In a neural network the perceptrons are arranged in layers whose outputs are used as input for following layers. Without the activation functions we would in some sense have a complicated linear regression which limits the space of functions that we can approximate. The activation functions allow us to learn a greater domain of functions and certain choices limit undesirable output values.

The model of chained perceptrons in layers is often called a *feedforward network* or *dense network*. The goal of a feedforward network is to approximate a function $f(\mathbf{x}) = \mathbf{y}$ by learning a set of weights W. The network defines a map $F : (\mathbf{x}, W) \mapsto \mathbf{y}$. We can for example have $\mathbf{y} \in \mathbb{Z}_k$

if we are trying to classify into k-categories, $\mathbf{y} \in \mathbb{R}$ for regression problems or $\mathbf{y}$ can be in the same space as $\mathbf{x}$ if we are denoising.

A convenient way to describe a feedforward network is to split it up into its connected layers of neurons. In every layer, starting with the input layer, each neuron outputs a real number in the manner described previously. At the next layer, each neuron collects these real numbers from those neurons it is connected to and proceeds similarly. At the last layer, these outputs determine the prediction the network makes for this particular input.

After computing the output at the final layer, we need some function to define the quality of the output. Such functions are called *loss functions*. These determine the value of the error when comparing the network output to the training value. The most canonical one is the mean squared error (MSE)

$$\mathcal{J}(\mathbf{w}) = \frac{1}{2} \left\| F(\mathbf{w}, \mathbf{x}_i) - \mathbf{y}_i \right\|_2^2$$

for training point $\mathbf{x}_i$ and label $\mathbf{y}_i$. Note that the loss function is a function of the weights. It is subsequently used to alter the weights depending on their contribution to the error in the backward pass.

5.2 Mathematics of back propagation

In this and the following section we will mathematically derive the learning rules for a 1D dense network. This section can safely be skipped if the reader is familiar with the workings of neural networks and back propagation. The basics of error propagation that we derive here will also be applied when deriving harder expressions for the more complicated MS-D network. Some alternate motivation for this type of derivation can be found in the work by Higham and Higham [11].

Suppose for the moment that the input is of the form $\mathbf{x} \in \mathbb{R}^m$ and that we use some activation function σ. Suppose that the first layer has n_1 neurons and suppose that the network is densely connected. Then we may describe the result of applying the first layer to its input by evaluating

$$\sigma(W^{[1]}\mathbf{x} + \mathbf{b}^{[1]})$$

with $W^{[1]} \in \mathbb{R}^{m \times n_1}$ and $b^{[1]} \in \mathbb{R}^{n_1}$. It is not strictly necessary to assume a dense layer since the entries in the weight matrix of the nonexistent edges can be artificially set to zero. Suppose that the third layer has n_2 neurons. Then the result of applying the second layer can be described as

$$\sigma(W^{[2]}\sigma(W^{[1]}\mathbf{x} + \mathbf{b}^{[1]}) + \mathbf{b}^{[2]})$$

with $W^{[2]} \in \mathbb{R}^{n_1 \times n_2}$ and $b^{[2]} \in \mathbb{R}^{n_2}$. Note that the activation functions for the neurons in this layer may be different but we omit this for simplicity. The entire network can be described in this fashion. We can summarize the above with the recurrence relation

$$\begin{aligned} \mathbf{a}^{[0]} &= \mathbf{x} \\ \mathbf{a}^{[i]} &= \sigma(W^{[i]}\mathbf{a}^{[i-1]} + \mathbf{b}^{[i]}) \quad \text{for } i = 1, \ldots, d \end{aligned}$$

with d the *depth* of the network.

Deep learning algorithms involve some kind of optimization method to update the weights. Gradient descent is a standard popular optimization method to use. To fit the training set we need a loss function $\mathcal{J}$. Suppose for notational convenience that all the weights are arranged in some large vector $\omega \in \mathbb{R}^k$. Suppose we take a step $\delta\omega$ for the next iteration, a first-order approximation of the effect on the loss is

$$\mathcal{J}(\omega + \delta\omega) \approx \mathcal{J}(\omega) + \sum_{j=1}^{k} \frac{\partial \mathcal{J}}{\partial \omega_j} \delta\omega_j = \mathcal{J} + \nabla\mathcal{J}(\omega)^T \delta\omega.$$

The expression is minimal when the second term is maximally negative. By Cauchy-Schwarz we know that

$$|\nabla\mathcal{J}(\omega)^T \delta\omega| \leq \left\|\nabla\mathcal{J}(\omega)^T\right\| \|\delta\omega\|.$$

Therefore, the most negative the expression can be is $-\left\|\nabla\mathcal{J}(\omega)^T\right\| \|\delta\omega\|$. Suppose that $\delta\omega = -\nabla\mathcal{J}(\omega)$, then we see that the first order term becomes

$$\nabla\mathcal{J}(\omega)^T \delta\omega = -\|\nabla\mathcal{J}(\omega)\|_2^2 = -\|\nabla\mathcal{J}(\omega)\|_2 \|\delta\omega\|.$$

Because of Cauchy-Schwarz this is the minimum so we conclude that the direction of $\delta\omega$ should be $-\nabla\mathcal{J}(\omega$. Since we have used an approximation and we do not want to have the weights fluctuate to radically after each training point, we will relax this choice somewhat by weighting the size of the step by the *learning rate* λ. So for all training points $\mathbf{x}_i$ we iterate over the weights and update

$$\omega \mapsto \omega - \lambda\nabla\mathcal{J}_{\mathbf{x}_i}(\omega).$$

Often the order of updates is randomized over the training points. A small learning rate means that training can be slow and a large learning rate means that we may overshoot the minimum. One approach is to start with a higher learning rate and reduce it over time.

To derive the weight updates explicitly we will consider a single update step. For this derivation, we use the mean squared error (MSE) as the loss function. Ultimately, we are interested in the following quantities (for each layer l)

$$\frac{\partial\mathcal{J}_{\mathbf{x}_i}(\omega)}{\partial\mathbf{w}_{jk}^{[l]}}, \quad \frac{\partial\mathcal{J}_{\mathbf{x}_i}(\omega)}{\partial\mathbf{b}_{j}^{[l]}}.$$

Here $\mathbf{w}_{jk}^{[l]}$ denotes the weight in layer l that goes from neuron k in the layer $l-1$ to neuron j in the layer l and $\mathbf{b}_j^{[l]}$ is the bias in layer l and neuron j. If we know these quantities we can update

$$\mathbf{w}_{jk}^{[l]} \mapsto \mathbf{w}_{jk}^{[l]} - \lambda\frac{\partial\mathcal{J}_{\mathbf{x}_i}(\omega)}{\partial\mathbf{w}_{jk}^{[l]}} \tag{5.1}$$

$$\mathbf{b}_{j}^{[l]} \mapsto \mathbf{b}_{j}^{[l]} - \lambda\frac{\partial\mathcal{J}_{\mathbf{x}_i}(\omega)}{\partial\mathbf{b}_{j}^{[l]}} \tag{5.2}$$

To do so, we introduce a useful dummy variable $\mathbf{z}^l$. We define $\mathbf{z}_j^l$ to be the input that neuron j at layer l receives, i.e.

$$\mathbf{z}^{[l]} = \mathbf{w}^{[l]}\mathbf{a}^{[l-1]} + \mathbf{b}^{[l]}. \tag{5.3}$$

Lemma 5.1. *For layer $l = 1, \dots, d-1$, we have that*

$$\frac{\partial \mathcal{J}_{\mathbf{x}_i}(\omega)}{\partial \mathbf{z}_j^{[l]}} = \sigma'(\mathbf{z}_j^{[l]}) \sum_{k=1}^{n_{l+1}} \mathbf{w}_{kj}^{[l+1]} \frac{\partial \mathcal{J}_{\mathbf{x}_i}(\omega)}{\partial \mathbf{z}_k^{[l+1]}}. \tag{5.4}$$

Furthermore, in the final layer we have

$$\frac{\partial \mathcal{J}_{\mathbf{x}_i}(\omega)}{\partial \mathbf{z}_j^{[d]}} = \sigma'(\mathbf{z}_j^{[d]})(\mathbf{a}_j^{[d]} - \mathbf{y}_j). \tag{5.5}$$

Here $\mathbf{y}_j$ is the j-th entry of the correct prediction for training point $\mathbf{x}_i$.

Proof. Firstly, we prove equation 5.5. Recall that by definition we have

$$\mathbf{a}^{[d]} = \sigma(\mathbf{z}^{[d]}), \qquad \frac{\partial \mathbf{a}_j^{[d]}}{\partial \mathbf{z}_j^{[d]}} = \sigma'(\mathbf{z}_j^{[d]}).$$

For the MSE loss function we get

$$\frac{\partial \mathcal{J}_{\mathbf{x}_i}(\omega)}{\partial \mathbf{a}_j^{[d]}} = \frac{1}{2} \frac{\partial}{\partial \mathbf{a}_j^{[d]}} \sum_{k=1}^{n_d} (\mathbf{a}_k^{[d]} - \mathbf{y}_k)^2 = \mathbf{a}_j^{[d]} - \mathbf{y}_j.$$

Hence we obtain equation 5.5 by the chain rule

$$\frac{\partial \mathcal{J}_{\mathbf{x}_i}(\omega)}{\partial \mathbf{z}_j^{[d]}} = \frac{\partial \mathcal{J}_{\mathbf{x}_i}(\omega)}{\partial \mathbf{a}_j^{[d]}} \frac{\partial \mathbf{a}_j^{[d]}}{\partial \mathbf{z}_j^{[d]}} = \sigma'(\mathbf{z}_j^{[d]})(\mathbf{a}_j^{[d]} - \mathbf{y}_j).$$

Remark: Actually the multi-dimensional chain rule requires that we sum over all $\mathbf{z}_k^{[d]}$ but all those contributions are immediately zero. We will occasionally omit such sums if the other indices are clearly zero.

Next we will show equation 5.4. Let $l = 1, \dots, d-1$, by using the chain rule we can rewrite

$$\frac{\partial \mathcal{J}_{\mathbf{x}_i}(\omega)}{\partial \mathbf{z}_j^{[l]}} = \sum_{k=1}^{n_{l+1}} \frac{\partial \mathcal{J}_{\mathbf{x}_i}(\omega)}{\partial \mathbf{z}_k^{[l+1]}} \frac{\partial \mathbf{z}_k^{[l+1]}}{\partial \mathbf{z}_j^{[l]}}.$$

Recall that by definition of $\mathbf{z}^{[l]}$ (5.3) we obtain

$$\frac{\partial \mathbf{z}_k^{[l+1]}}{\partial \mathbf{z}_j^{[l]}} = \frac{\partial}{\partial \mathbf{z}_j^{[l]}} \sum_{h=1}^{n_l} \mathbf{w}_{kh}^{[l+1]} \sigma(\mathbf{z}_h^{[l]}) + \mathbf{b}_k^{[l+1]} = \mathbf{w}_{kj}^{[l+1]} \sigma'(\mathbf{z}_j^{[l]})$$

which immediately shows equation 5.4. □

Now that we have calculated expressions for the dependence of the loss function with respect to the input of a certain neuron in a certain layer, we can relate this dependence to the individual

weights and biases. The dependence of the loss function on the bias follows from the chain rule and the fact that

$$\frac{\partial \mathbf{z}_j^{[l]}}{\partial \mathbf{b}_j^{[l]}} = 1 \quad \Rightarrow \quad \frac{\partial \mathcal{J}_{\mathbf{x}_i}(\omega)}{\partial \mathbf{b}_j^l} = \frac{\partial \mathcal{J}_{\mathbf{x}_i}(\omega)}{\partial \mathbf{z}_j^l} \frac{\partial \mathbf{z}_j^{[l]}}{\partial \mathbf{b}_j^{[l]}} = \frac{\partial \mathcal{J}_{\mathbf{x}_i}(\omega)}{\partial \mathbf{z}_j^l}.$$

This expression is already known by Lemma 5.1 for each l. For the weight dependence, recall that by equation 5.3 we see that

$$\frac{\partial \mathbf{z}_j^{[l]}}{\partial \mathbf{w}_{jk}^{[l]}} = \mathbf{a}_k^{[l-1]}.$$

Hence we obtain that

$$\frac{\partial \mathcal{J}_{\mathbf{x}_i}(\omega)}{\partial \mathbf{w}_{jk}^l} = \frac{\partial \mathcal{J}_{\mathbf{x}_i}(\omega)}{\partial \mathbf{z}_j^l} \frac{\partial \mathbf{z}_j^{[l]}}{\partial \mathbf{w}_{jk}^{[l]}} = \frac{\partial \mathcal{J}_{\mathbf{x}_i}(\omega)}{\partial \mathbf{z}_j^l} \mathbf{a}_k^{[l-1]}.$$

Again the implied sum is left out since $\frac{\partial \mathbf{z}_s^{[l]}}{\partial \mathbf{w}_{jk}^{[l]}} = 0$ if we do not have $j = s$. We now have a clear derivation of how we can determine the individual contribution to the loss function per neuron. This contribution is sometimes viewed as the error but it is more correct to consider it as the sensitivity of that neuron to the loss function.

For this thesis I implemented a neural network library in C++ using the update rules derived in this section. In the forward pass the algorithm can save the relevant **z**-values and **a**-values such that the gradients can quickly be computed. The process of calculating the gradients starting from the rear and moving towards the beginning iteratively is called *back propagation* since we propagate the dependencies backwards through the network.

5.3 Convolutions and dimensionality reduction

The most canonical layer in neural networks is the densely connected layer we have just discussed. In image processing it is not obvious how to interpret images for 2D dense layers. A first attempt might be to assign an input neuron to each pixel. This would lead to an infeasible number of neurons and parameters. For this reason, the introduction of convolutional layers in neural networks has caused a breakthrough in performance in image processing. The feature extraction problem is solved by learning a relatively small set of weights that define a set of convolutional filters.

In a convolutional neural network (CNN) the network learns the values of the weights in the filter during training. Usually the size of the filter is fixed beforehand. For example, for a 3×3 kernel,

$$\mathbf{h} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix},$$

there are 9 weights to be learned. Consider 1D convolutions for a moment and let $\mathbf{h} \in \mathbb{R}^3$ instead. Consider a convolutional layer with input $\mathbf{x} \in \mathbb{R}^n$ and outputs $\mathbf{y} \in \mathbb{R}^m$. Suppose that there is only one filter for the moment. Then the input and output are related by

$$\sigma(\mathbf{h} * \mathbf{x} + \mathbf{b}) = \mathbf{y}$$

where $*$ denotes the convolution. In a typical dense layer they would be related by

$$\sigma(W\mathbf{x} + \mathbf{b}) = \mathbf{y}$$

where W is the weight matrix. This means that we have to learn nm weights. If we return to the convolutional layer, we see that input and output are related as

$$\begin{aligned}\sigma(h_1x_1 + h_2x_2 + h_3x_3 + b_1) &= y_1\\ \sigma(h_1x_2 + h_2x_3 + h_3x_4 + b_2) &= y_2\\ &\vdots\end{aligned}$$

So the weight matrix looks like

$$\begin{pmatrix} \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \iddots \\ \dots & h_1 & h_2 & h_3 & 0 & 0 & \dots \\ \dots & 0 & h_1 & h_2 & h_3 & 0 & \dots \\ \dots & 0 & 0 & h_1 & h_2 & h_3 & \dots \\ \iddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

Not only is the matrix sparse, we only have 3 weights to learn. In essence, we are using *several copies of the same neuron* with same weights over and over. This means that the network only needs to learn a specific function once instead of learning it over and over again. Learning how to extract an edge is useful for the entire image and only needs to be learned once and there is less room for error. Convolutional layers therefore greatly reduce the number of parameters that need to be learned which is necessary if we consider the number of pixels most high resolution images have.

Convolutional neural networks link several convolutional layers in succession. Mathematically, we define real-valued images as $\mathbf{x} \in \mathbb{R}^{m\times n\times c}$ with m rows, n columns and c channels. We denote the output image of layer l, channel c by $\mathbf{a}^{[l,c]}$. If the channel is irrelevant, it is left out of the brackets. For each layer l, the learned filter is applied to the input image $\mathbf{a}^{[l-1]} \in \mathbb{R}^{m_{l-1}\times n_{l-1}\times c_{l-1}}$, which is the output of the previous layer, and outputs a *feature map* $\mathbf{a}^{[l]} \in \mathbb{R}^{m_l\times n_l\times c_l}$. The final output image is the network output and the input image $\mathbf{x}$ is the input for the first layer. It is common to convolve each channel of the input image with a separately learned filter, then sum the resulting images, add the bias and apply the activation function pixelwise. Note that the output dimensions of the convolutional layer need not be the same as the input dimensions.

To extract high-level abstract features from a raw image, it is theorized that deep learning works since it tackles the representation issue by learning simpler representations in the shallow layers and combining them to form more complex representation in the deeper layers [26]. Traditional encoder-decoder architectures include downscaling and upscaling operations to capture features at different scales. The first part of the network, the encoder section, downscales the feature maps while the second part, the decoder section, upscales the images.

5.4 Mixed-scale dense convolutional neural networks

Traditional encoder-decoder networks can be relatively deep since they need to encompass several downscaling and upscaling layers. The increased depth can make training difficult and gradients

can become too small or too large [13]. Arguably we do not need to impose the restriction that an image computed by a certain filter can only be accessed by the layers directly adjacent to it. Often the number of parameters needed to fit such networks still runs in the millions which can increase the training time [20] and the chance of overfitting [43]. Proponents of alternative architectures argue that the restrictive nature of the traditional network needs to be compensated by a larger amount of filters and therefore more parameters. The issues can be combated with additional options such as dropout layers (arbitrarily disregard a percentage of the nodes during training) [43] or batch normalization layers [13].

The additional options have improved the performance of these neural networks [23] but they arguably make for a less user-friendly network. In addition to the depth and channel widths of the network the user has to decide on the number, size and placement of the pooling and dropout layers or additional parameters that the extra options require.

An alternative architcture called the *mixed-scale dense network* has been proposed to combat several such issues [32]. The narrative is that we keep the effective convolutional layers and make barely any other restrictions on the network. In summary, all input and output images are the same size and a filter at a certain layer has all images from all previous layers as input.

Note that all images are of equal size, hence any output image produced in any channel of a layer can be used in any other layer. The network is connected such that all previously computed output images are available for computing the next layer. Note that every channel of every layer also has a bias parameter that needs to be learned. All together the computation becomes

$$\mathbf{a}^{[l,c]} = \sigma(g_{lc}(\mathbf{a}_0, \ldots, \mathbf{a}_{l-1}) + \mathbf{b}^{[l,c]}) \tag{5.6}$$

$$g_{lc}(\mathbf{a}_0, \ldots, \mathbf{a}_{l-1}) = \sum_{i=0}^{l-1} \sum_{j=0}^{c-1} \mathbf{w}^{[l,c,i,j]} * \mathbf{a}^{[i,j]} \tag{5.7}$$

where $\mathbf{w}^{[l,c,i,j]}$ denotes the convolution kernel in layer l and channel c which takes ouput image $\mathbf{a}^{[i,j]}$ as input. We model the architecture of the MS-D network in this thesis along the lines of [32]. The kernel size was fixed at 3×3. The final layers was a 1×1 convolution which is simply attributing a weight to all the previous feature maps. We set the number of channels for each layer to 1. Since all computed images are accessible to all following layers, the necessity for channels has been removed. The depth d of the network defines the number of hidden layers in the network and will be taken to be $d = 100$ or $d = 200$.

One issue that we have not touched on so far is how to capture large scale features, a property of which is implemented by downscaling/pooling in traditional encoder-decoder networks. To introduce such a property without placing restrictions on which images can be inputs for which layers, the authors propose to use dilated convolutions.

Let $\mathbf{h}$ be a regular convolutional filter $\mathbf{h} \in \mathbb{R}^{k_1 \times k_2}$. A *dilated* convolution "blows up" the filter by equi-spacing the filter weights at positions that are some $s \in \mathbb{Z}_{>0}$ away from the center (in both directions). The other entries are 0. Suppose we have 3×3 kernel,

$$\mathbf{h} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}.$$

The following would be 2 and 3 dilated convolutions respectively

$$\mathbf{h}_2 = \begin{bmatrix} h_{11} & 0 & h_{12} & 0 & h_{13} \\ 0 & 0 & 0 & 0 & 0 \\ h_{21} & 0 & h_{22} & 0 & h_{23} \\ 0 & 0 & 0 & 0 & 0 \\ h_{31} & 0 & h_{32} & 0 & h_{33} \end{bmatrix}, \quad \mathbf{h}_3 = \begin{bmatrix} h_{11} & 0 & 0 & h_{12} & 0 & 0 & h_{13} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ h_{21} & 0 & 0 & h_{22} & 0 & 0 & h_{23} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ h_{31} & 0 & 0 & h_{32} & 0 & 0 & h_{33} \end{bmatrix}.$$

An example architecture is displayed in Figure 46.

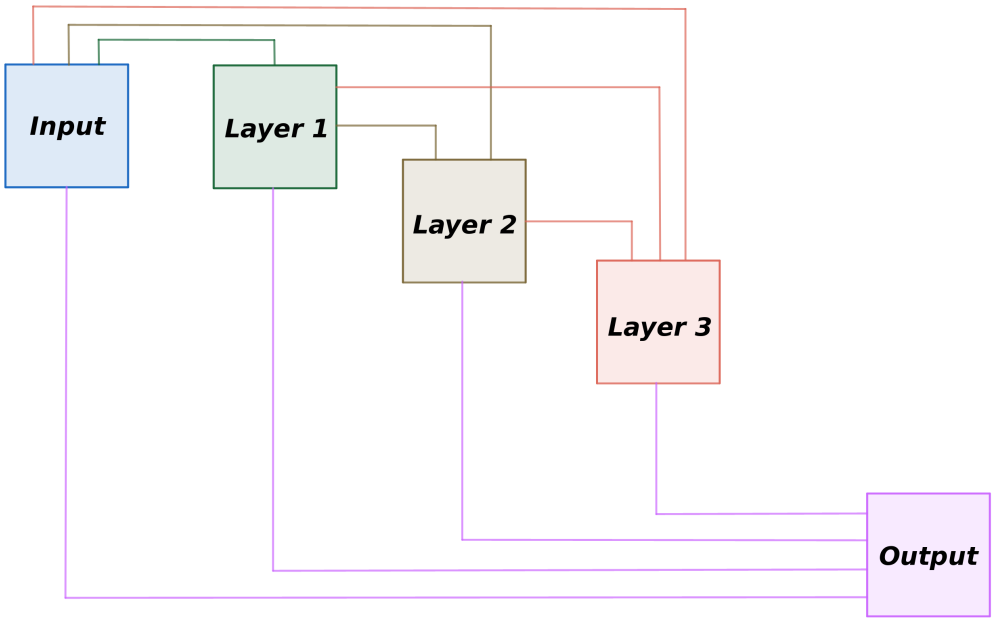


Figure 46: An example diagram of a MS-D network with 4 layers and a channel width of 1. The lines indicate which feature maps are input for that particular layer and the colours indicate that these inputs may be convolved with a kernel with a different dilation size. Furthermore, the final convolutions (in purple) are 1×1 while the rest is typically 3×3.

For this thesis we also implemented a C++ library for MS-D networks. We used the same type of derivation as described in section 5.2. The derivation is cumbersome due to the higher dimensionality of the problem. Therefore, we have provided the derivation of the update equations in appendix 9.3 for the interested reader. In the library we implemented stochastic gradient descent optimization and ADAM [17]. The details of the ADAM optimizer can be found in appendix 9.3.2.

6 Deep learning experiments

In this thesis we roughly reduced the research goals to; 1) improving the quality of the cryo-ET reconstructions by looking into different tomographic reconstruction algorithms and 2) reducing the amount of data needed to obtain a reconstruction of a certain quality. Thus far we have extensively tested combinations of algorithms and subtomogram averaging - and the ordering of the steps in the pipeline - where we based our conclusions on the quality of the final reconstruction.

In this section we aim to tackle the second research problem by training a variety of neural networks, specifically MS-D networks as outlined in section 5.4, to denoise individual reconstructions after they have been reconstructed from the projections. The goal is to show that if a suitable best average ribosome can be obtained, further scans can be boosted in quality by a trained neural net which should reduce the number of scans needed to obtain a ribosome reconstruction of a certain quality.

In addition, it may be that an average of MS-D network processed reconstructions will also have a better quality. Even though we implemented our own neural network during the construction of this pipeline, it proved more advantageous to use an external library for these experiments developed by dr. D. M. Pelt [31]. This library allows for GPU computations and therefore greatly reduced the computation time.

6.1 Post-processing with MS-D networks

In this section we will train 8 MS-D networks on different sets of training data and ground truth volumes. We did some initial tests and concluded that since MS-D networks are able to adapt to the problem at hand, the same hyperparameters could be used for each experiment. The best results were obtained with a depth $d = 200$ and equally distributed dilation sizes $d_i \in [1, 5]$. Furthermore, we used slabs, a set of consecutive slices as input, instead of single slices and the initialization of the trainable parameters can be found in [32].

We did not train a 3D MS-D network due to the computational load of a 3D network. Instead we trained a 2D network where 5 consecutive slices are entered as one training point, i.e. there are 5 input channels for the network. We trained on slices 90 up to 210 (out of 320) to avoid a large part of the dataset to be vacuum where the ribosome is not present. This means that the first training point would be a slab of slices 90 to 94, the next 91 to 95 and the last training point is 209 to 213.

In our simulations we made 200 randomly oriented example ribosomes for each noise level and each reconstruction algorithm. We split this set up into 160 training ribosomes, 30 validation ribosomes and 10 testing ribosomes. This gives us a total of 19200 training slabs and 3600 validation slabs per run. When we apply the MS-D network to our test ribosomes we apply it to all the slices since there appears to be enough vacuum present in the training set for it to denoise the vacuum properly. In addition, we normalized all the input slices and augmented the dataset by flipping and rotating the images.

In Table 6.1 we have outlined the setup for each of the 8 experiments. To accelerate the computation we ran the training procedure on Graphic Programming Units (GPUs) using the MS-D network library written by dr. D. M. Pelt with CUDA version 9.2. The networks were

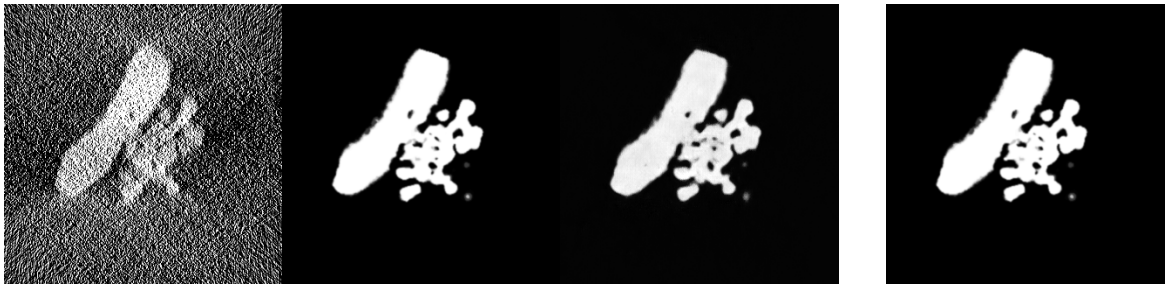
Identifier	Input reconstructions	Target reconstructions
fbp-10g-gt	FBP recs for Gaussian noise $\sigma = 10.0$	Phantom ground truths
fbp-100g-gt	FBP recs for Gaussian noise $\sigma = 100.0$	Phantom ground truths
fbp-100g-av	FBP recs for Gaussian noise $\sigma = 100.0$	Best SIRTMIN-100g average
fbp-250g-gt	FBP recs for Gaussian noise $\sigma = 250.0$	Phantom ground truths
fbp-250g-av	FBP recs for Gaussian noise $\sigma = 250.0$	Best SIRTMIN-250g average
SIRTMIN-100g-gt	SIRTMIN recs for Gaussian noise $\sigma = 100.0$	Phantom ground truths
SIRTMIN-100g-av	SIRTMIN recs for Gaussian noise $\sigma = 100.0$	Best SIRTMIN-100g average
SIRTMIN-5p-gt	SIRTMIN recs for Poisson noise $I_0 = 5.0$	Phantom ground truths

Table 2.1: Table of experiments containing the combinations of training data and target labels that the MS-D networks were trained on.

trained on a server using four NVidia GeForce GTX 1080 GPUs and we used the ADAM optimizer.

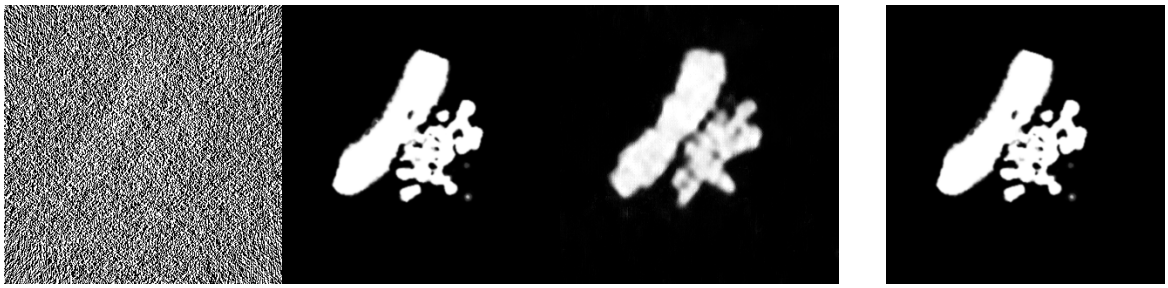
The loss function was the MSE between the network output and the target. We ran for a maximum of 70 training iterations or if there had not been an improvement for 10 iterations. We ran validation after every batch of 1800 training slabs. At each validation step, we check the stopping criterion and the network parameters with the lowest validation error are saved during training.

We subsequently applied the trained networks to the test images to obtain a set of denoised ribosomes. An example slice for each experiment before and after denoising by the MS-D network is given in Figures 47 to 54. The examples shown are from the test set meaning that the network has never seen these images before.



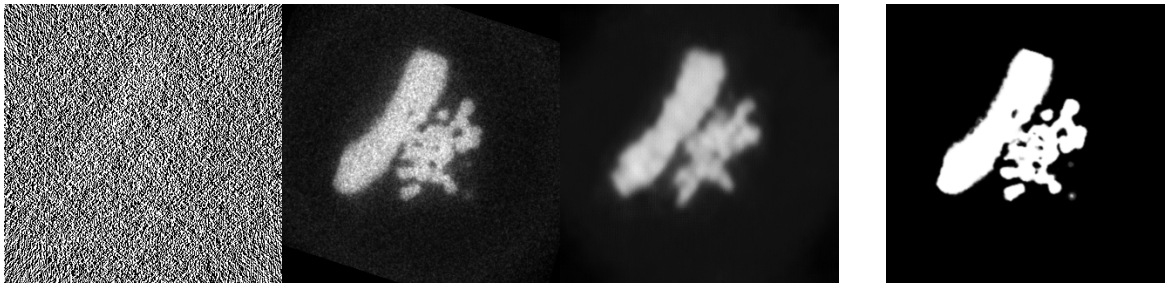
(a) **(b)** **(c)** **(d)**

Figure 47: **FBP-10g-gt** Middle slice for several reconstructions. (a) Input FBP reconstruction slice for Gaussian noise $\sigma = 10.0$. (b) Target ground truth image. (c) network output slice. (d) original ground truth slice for comparison.



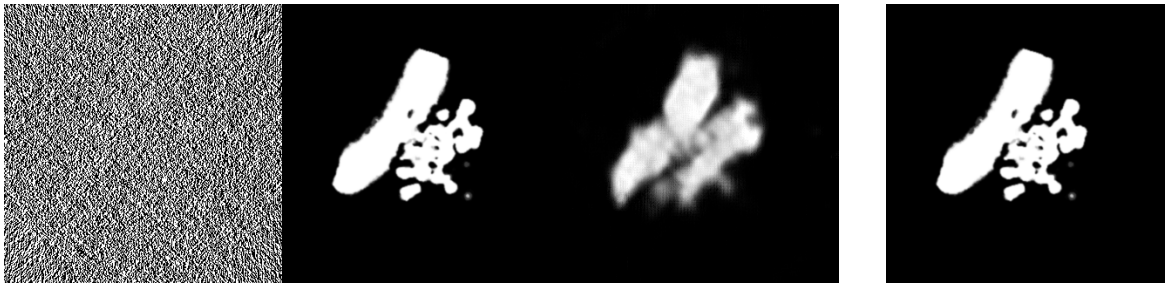
(a) **(b)** **(c)** **(d)**

Figure 48: **FBP-100g-gt** Middle slice for several reconstructions. (a) Input FBP reconstruction slice for Gaussian noise $\sigma = 100.0$. (b) Target ground truth image. (c) network output slice. (d) original ground truth slice for comparison.



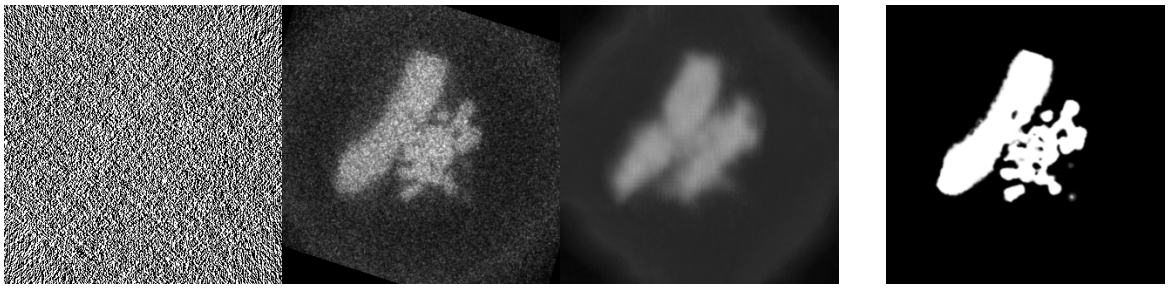
(a) **(b)** **(c)** **(d)**

Figure 49: **FBP-100g-av** Middle slice for several reconstructions. (a) Input FBP reconstruction slice for Gaussian noise $\sigma = 100.0$. (b) Target best SIRTMIN average for this noise level. (c) network output slice. (d) original ground truth slice for comparison.



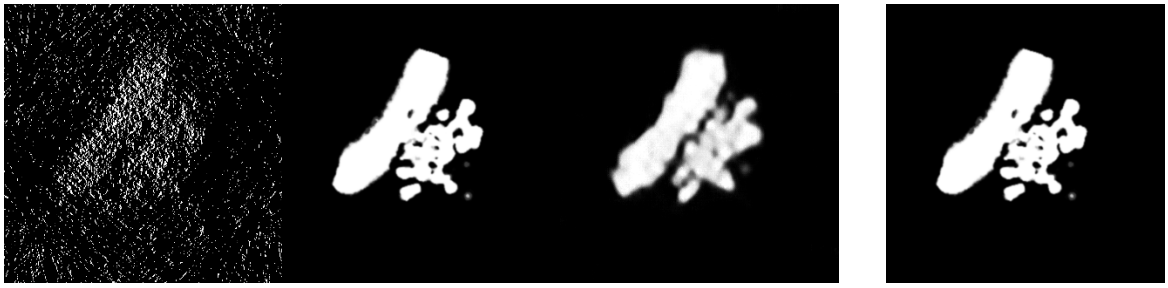
(a) **(b)** **(c)** **(d)**

Figure 50: **FBP-250g-gt** Middle slice for several reconstructions. (a) Input FBP reconstruction slice for Gaussian noise $\sigma = 250.0$. (b) Target ground truth image. (c) network output slice. (d) original ground truth slice for comparison.



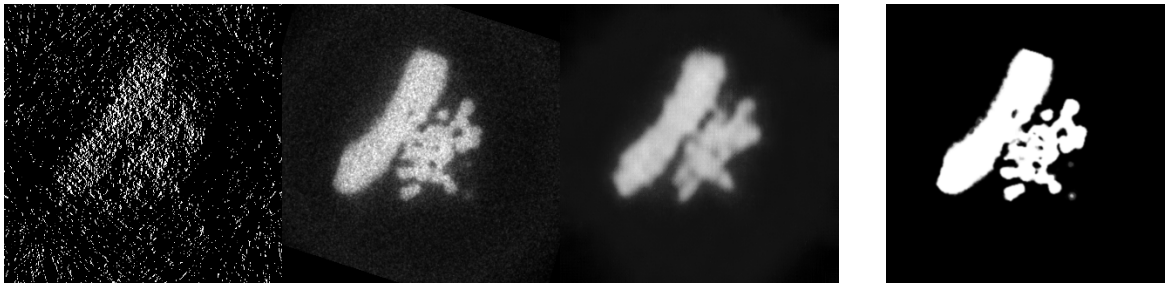
(a) **(b)** **(c)** **(d)**

Figure 51: **FBP-250g-av** Middle slice for several reconstructions. (a) Input FBP reconstruction slice for Gaussian noise $\sigma = 250.0$. (b) Target best SIRTMIN average for this noise level. (c) network output slice. (d) original ground truth slice for comparison.



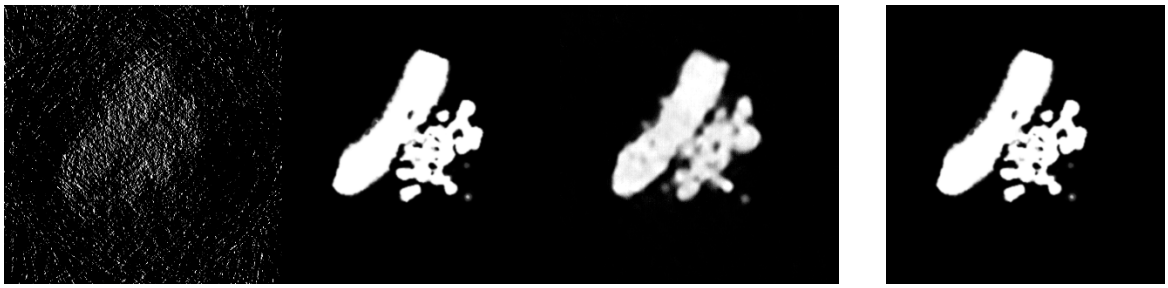
(a) **(b)** **(c)** **(d)**

Figure 52: **SIRTMIN-100g-gt** Middle slice for several reconstructions. (a) Input SIRTMIN reconstruction slice for Gaussian noise $\sigma = 100.0$. (b) Target ground truth image. (c) network output slice. (d) original ground truth slice for comparison.



(a) **(b)** **(c)** **(d)**

Figure 53: **SIRTMIN-100g-av** Middle slice for several reconstructions. (a) Input FBP reconstruction slice for Gaussian noise $\sigma = 100.0$. (b) Target best SIRTMIN average for this noise level. (c) network output slice. (d) original ground truth slice for comparison.



(a) (b) (c) (d)

Figure 54: SIRTMIN-5p-gt Middle slice for several reconstructions. (a) Input SIRTMIN reconstruction slice for Poisson noise $I_0 = 5.0$. (b) Target ground truth image. (c) network output slice. (d) original ground truth slice for comparison.

The processed *FBP-10g-gt* images are nearly perfect. Even though this is the simplest noise level the noise cloud would be considered to be quite severe for regular tomography problems. More promising is that we see that the *FBP-100g-gt* and *FBP-100g-av* images in Figures 48 and 49 are also properly denoised as they are more indicative of Cryo-ET noise levels. Training with a best average creates a more blurry image which might be because the target is somewhat blurry and grainy itself. In particular, we see a halo around the object since the averages used as targets still contain some noise cloud.

Even though the images are devoid of a noise cloud, we see that the post-processed objects have artifacts that are not present in the original ribosome. This is worrying since we do not want the MS-D network to "invent" ribosome components or omit important components. This behaviour is amplified in the highest noise cases *FBP-250g-gt* and *FBP-250g-av*. In these cases, the network output does resemble a ribosome roughly but the image is more severely blurred and not a reliable ribosome. However, we should keep in mind that the network input, *FBP-250g-gt* (a) and *FBP-250g-av* (a), seems to the human eye pure static noise without signal.

Another useful conclusion is that in our opinion the network outputs the same quality reconstruction when it is given an FBP reconstruction compared to a SIRTMIN reconstruction. We saw that the original SIRTMIN reconstructions are better in our earlier experiments. However, FBP is computationally a lot faster than SIRTMIN so this suggests that we could speedup the pipeline by using FBP reconstructions instead. Furthermore, the SIRTMIN results illustrate the versatility of the MS-D network since it performs well on the much more sparse SIRTMIN inputs with an entirely different noise profile.

Lastly, we checked the performance on the Poisson noise reconstructions. We conclude that the MS-D network does not perform worse with Poisson noise. Arguably it performs better but it is not possible to compare the severity of the noise between the Gaussian en Poisson inputs. In Table 6.1 we present the AMSE scores for the original reconstructions and the post-processed volumes. The difference in MSE is so large that we also provide the multiplicative factor decrease in MSE.

In all cases we see a decrease in AMSE of several orders of magnitude. We see that training to ground truth images is a lot better than training with averages as targets. Also the net-

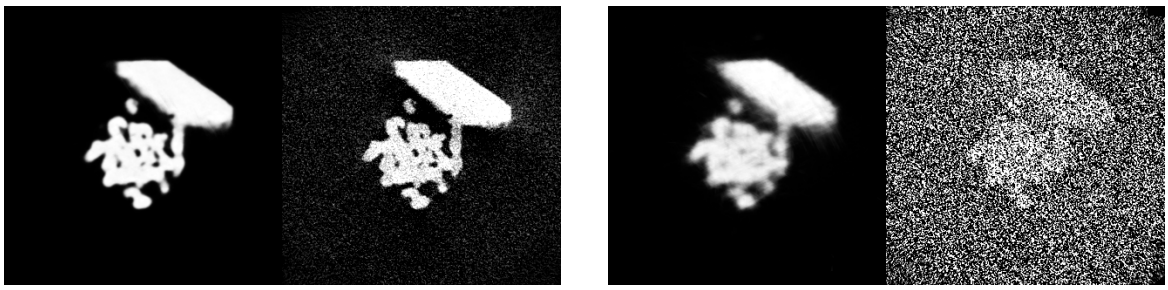
Identifier	Original AMSE	Network AMSE	Factor decrease in AMSE
fbp-10g-gt	$(7.41 \pm 3.01) \cdot 10^{-1}$	$(6.43 \pm 4.22) \cdot 10^{-4}$	1152
fbp-100g-gt	$(7.24 \pm 2.91) \cdot 10^{1}$	$(4.83 \pm 1.97) \cdot 10^{-3}$	14989
fbp-100g-av	$(7.24 \pm 2.91) \cdot 10^{1}$	$(1.75 \pm 0.30) \cdot 10^{-2}$	4124
fbp-250g-gt	$(4.52 \pm 1.83) \cdot 10^{2}$	$(1.05 \pm 0.48) \cdot 10^{-2}$	43168
fbp-250g-av	$(4.52 \pm 1.83) \cdot 10^{2}$	$(1.03 \pm 0.15) \cdot 10^{-1}$	4387
SIRTMIN-100g-gt	$(1.19 \pm 0.36) \cdot 10^{0}$	$(4.49 \pm 1.50) \cdot 10^{-3}$	265
SIRTMIN-100g-av	$(1.19 \pm 0.36) \cdot 10^{0}$	$(1.77 \pm 0.18) \cdot 10^{-2}$	67
SIRTMIN-5p-gt	$(5.87 \pm 1.39) \cdot 10^{-1}$	$(3.26 \pm 1.10) \cdot 10^{-3}$	180

Table 2.2: Table of the AMSE scores of the MS-D network reconstructions for each experiment with respect to the ground truth phantom. We also provide the multiplicative factor decrease in AMSE score between the original and MS-D network processed volumes. The first 7 are Gaussian noise experiments and the last experiment is on Poisson noise with $I_0 = 5.0$. *gt* or *av* designates if the networks were trained with the original ground truth or the best average as targets.

work improves a lot more on the FBP reconstructions than on the SIRTMIN reconstructions. However, this is likely due to the fact that the SIRTMIN reconstructions are a lot better to start with. We can however generally conclude that the numerical results support our conclusions that post-processing with the MS-D networks significantly improves the quality of the reconstructions.

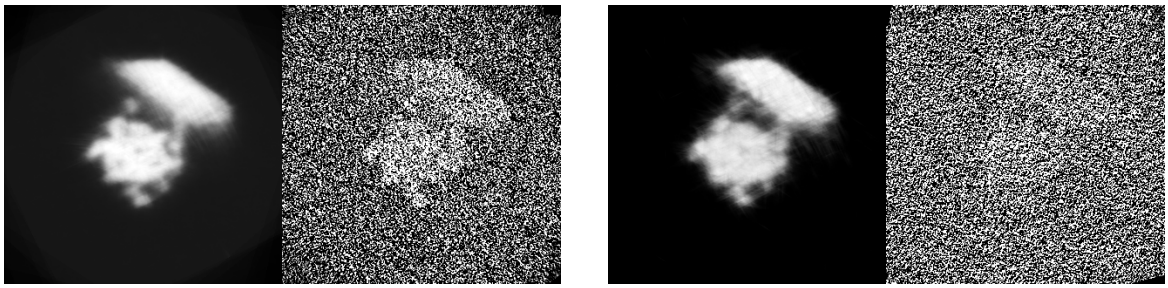
6.2 Subtomogram averaging and MS-D network outputs

The results of the MS-D network post-processing are promising on individual reconstructions. However, we mentioned possible problems with the network "inventing" its own ribosome features and the blurriness of the network output. Therefore, in this section we will average the MS-D network outputs of the test set with the original angles and compare them to similar size averages without MS-D network post-processing.



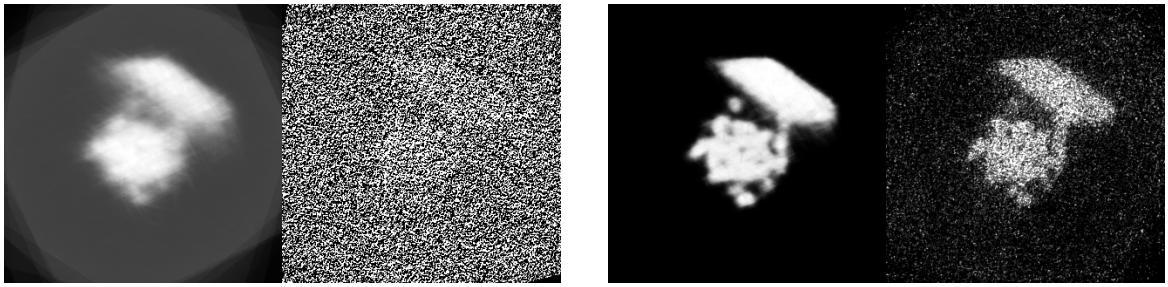
(a) **(b)** **(c)** **(d)**

Figure 55: **FBP-10g-gt** (a) vs 10-average of FBP Gaussian $\sigma = 10.0$ (b) and **FBP-100g-gt** (c) vs 10-average of FBP Gaussian $\sigma = 100.0$ (d).



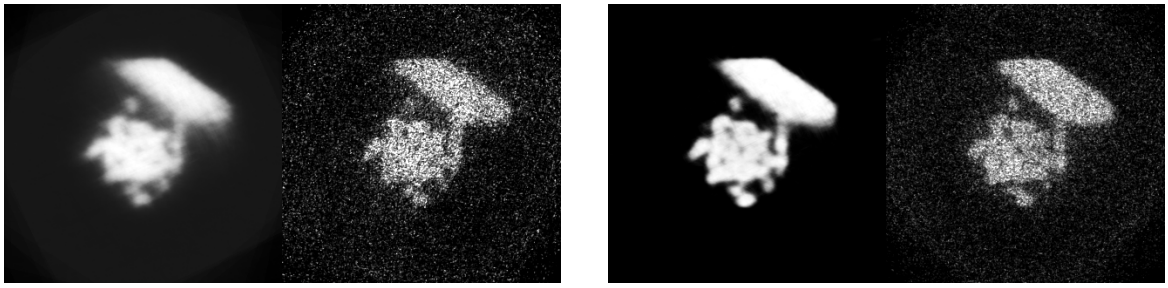
(a) **(b)** **(c)** **(d)**

Figure 56: **FBP-100g-av** (a) vs 10-average of FBP Gaussian $\sigma = 100.0$ (b) and **FBP-250g-gt** (c) vs 10-average of FBP Gaussian $\sigma = 250.0$ (d).



(a) **(b)** **(c)** **(d)**

Figure 57: **FBP-250g-av** (a) vs 10-average of FBP Gaussian $\sigma = 250.0$ (b) and **SIRTMIN-100g-gt** (c) vs 10-average of SIRTMIN Gaussian $\sigma = 100.0$ (d).



(a) **(b)** **(c)** **(d)**

Figure 58: **SIRTMIN-100g-av** (a) vs 10-average of SIRTMIN Gaussian $\sigma = 100.0$ (b) and **SIRTMIN-5p-gt** (c) vs 10-average of SIRTMIN Poisson $I_0 = 5.0$ (d).

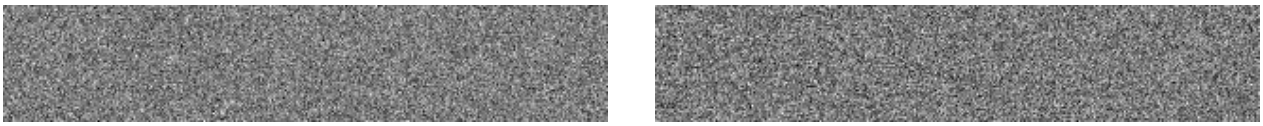
In our opinion the MS-D network averages are better across the board although they become increasingly blurry as the difficulty of the experiment increases. This has always been an issue in subtomogram averaging and we saw in earlier experiments that these issues can be mitigated by

taking a larger average. More importantly, none of the MS-D network averages seem to exhibit foreign or invented ribosome components which seems to suggest that artifacts are averaged out as we hoped.

6.3 Overfitting of MS-D networks

In this section we want to check that the MS-D networks recover the ribosome from the noise cloud without overfitting on the data. To test this we decided to look into the network output when the input contains no signal but only noise. If we can discern a ribosome oriented in a certain direction or some other bias even though we only provided noise as input, it would indicate that the network had learned to output a certain ribosome regardless of the input.

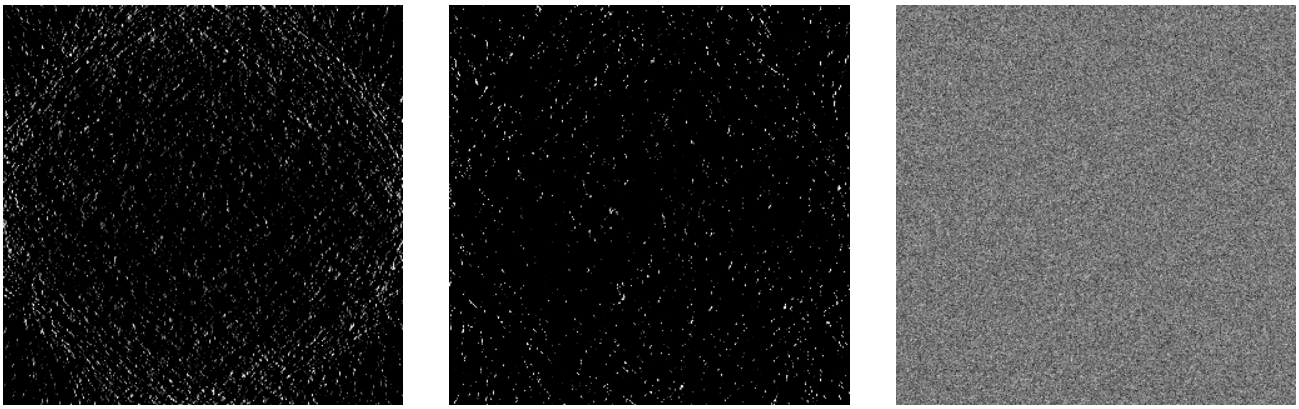
Firstly, we created two sinograms where each pixel is a random number drawn from a normal distribution $\mathcal{N}(100, 100)$ and $\mathcal{N}(0, 100)$ respectively. This produced the sinograms displayed in Figure 59.



(a) **(b)**

Figure 59: One projection of the sinogram produced by drawing each pixel from a normal distribution (a) $\mathcal{N}(100, 100)$ and (b) $\mathcal{N}(0, 100)$.

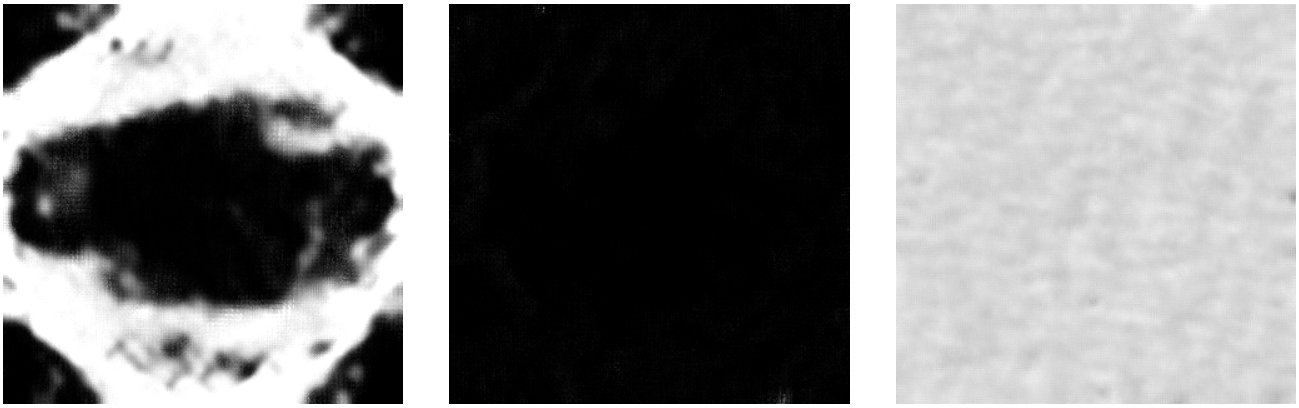
We decided to test the *SIRTMIN-100g-gt* network for overfitting so we created the SIRTMIN reconstructions of the pure noise sinograms (Figure 60 (a) and Figure 60 (b)). We also created a cube where each voxel was drawn from $\mathcal{N}(1, 1)$ (Figure 60 (c)).



(a) **(b)** **(c)**

Figure 60: (a) One slice of the SIRTMIN reconstruction produced from the noisy sinogram in Figure 59 (a), (b) one slice of the SIRTMIN reconstruction produced from the noisy sinogram in Figure 59 (b) and (c) one slice of the noisy cube.

Apart from the missing-wedge halo in Figure 60 (a), we do not see any signal as expected. We applied the *SIRTMIN-100g-gt* trained MS-D network to each input and obtained the outputs displayed in Figure 61. The images are displayed in the same range as the output range used in 52.



(a) **(b)** **(c)**

Figure 61: (a) One slice of the MS-D network output produced from the noisy reconstruction in Figure 60 (a), (b) one slice of the MS-D network output produced from the noisy reconstruction in Figure 60 (b) and (c) one slice of the MS-D network output produced from the noisy cube in Figure 60 (c).

Images 61 (b) and (c) do not exhibit signs of overfitting in our opinion. Figure 61 (a) seems to suggest that the MS-D network attempts to interpret the missing-wedge halo as signal. However, there does not appear to be a ribosome present in the network output. Furthermore, we may have expected the network to overtrain on the missing-wedge halo but this does not seem to be the case since there is no discernible pattern present in Figures 61 (b) and (c). This would indicate that the network is not overfitting and the results are genuine.

6.4 Performance of MS-D networks on cryo-ET data

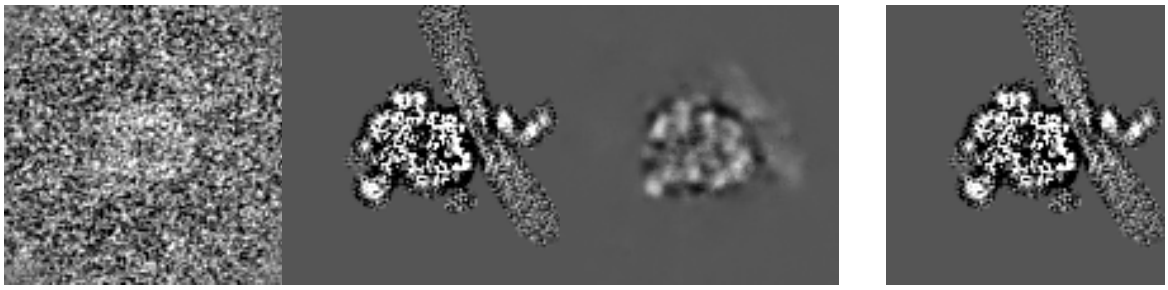
In this section we have attempted to replicate some of the strong results from the previous section on the actual cryo-EM dataset. We had access to 1000 particles from the total 17000 in the dataset. Furthermore, we were provided with an XML-file containing the translational shifts and rotation angles found by the alignment algorithm to orient each particle to the base orientation. In the experiments performed on the simulated data, the ribosomes were only rotated and we did not perform any translational shifts. On the real data we also perform a translational shift of the origin as outlined in section 3.2.

Firstly, we will investigate whether a trained MS-D network can reduce the amount of samples necessary for a certain quality reconstruction obtained by subtomogram averaging. For this experiment we created 1000 averages of 100 samples each out of $\binom{1000}{100} \approx 10^{139}$ as a training dataset. 850 ribosomes were used for training, 50 for validation and 100 were used as a test set.

Next, we trained two networks; **100av-to-gt** with the 17000-average best ground truth as

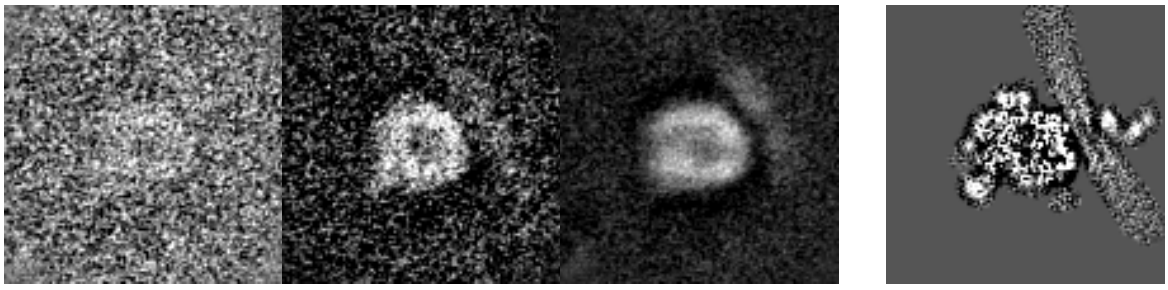
target and **100av-to-1000av** with the average of all 1000 particles in our slice of the data as target.

To reduce computation time we used a depth of $d = 100$ and equally distributed dilation sizes $d_i \in [1, 5]$. Furthermore, we once again used slabs instead of single slices and the initialization of the trainable parameters is once again as in [32]. We used batches of 10 training slabs for this experiment. The loss function was the MSE between the network output and the target. We ran for a maximum of 100 training iterations or if there had not been an improvement for 15 iterations. We ran validation after 5500 training slabs, i.e. after 550 passes of batches of 10. The results of applying the trained networks to an example ribosome from the test set are displayed in Figures 62 and 63.



(a) **(b)** **(c)** **(d)**

Figure 62: **100av-to-gt** Slice 45/110 for several reconstructions. (a) Input 100-average of cryo-ET particles from the dataset. (b) Target ground truth of 17000 particles. (c) network output slice. (d) original ground truth of 17000 particles for comparison.



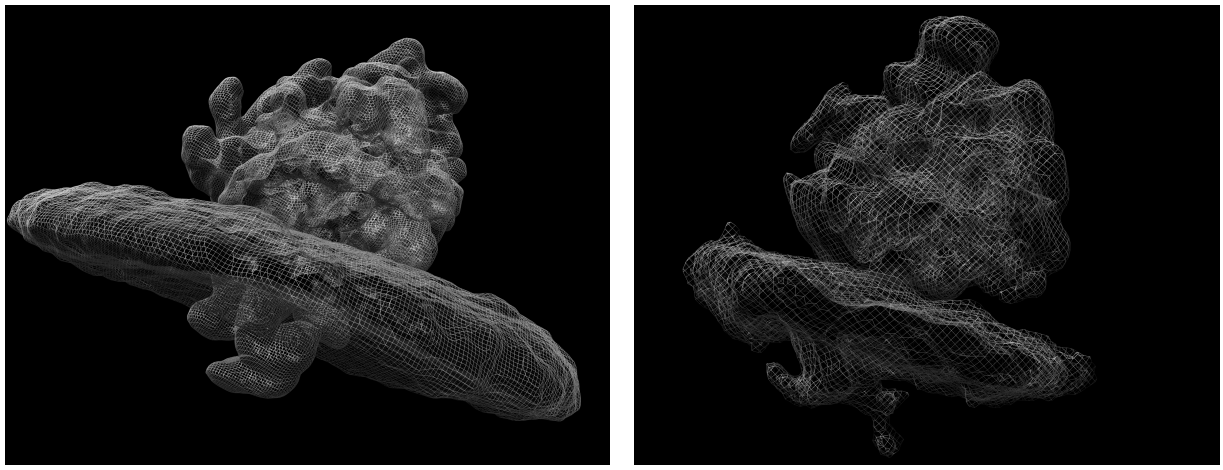
(a) **(b)** **(c)** **(d)**

Figure 63: **100av-to-1000av** Slice 45/110 for several reconstructions. (a) Input 100-average of cryo-ET particles from the dataset. (b) Target ground truth of 1000 particles. (c) network output slice. (d) original ground truth of 17000 particles for comparison.

In Figure 63 we see that the network has learned to significantly reduce the noise cloud. However, the network output does not resemble the ground truth image displayed in Figure 63 (d) when it comes to the more detailed features. Furthermore, an average of all network outputs on the test set remained blurred. Therefore, it seems that training a network on an average of only

1000 which does not contain finer feature itself may lead to blurring effects and does not lead to sharpening of the image.

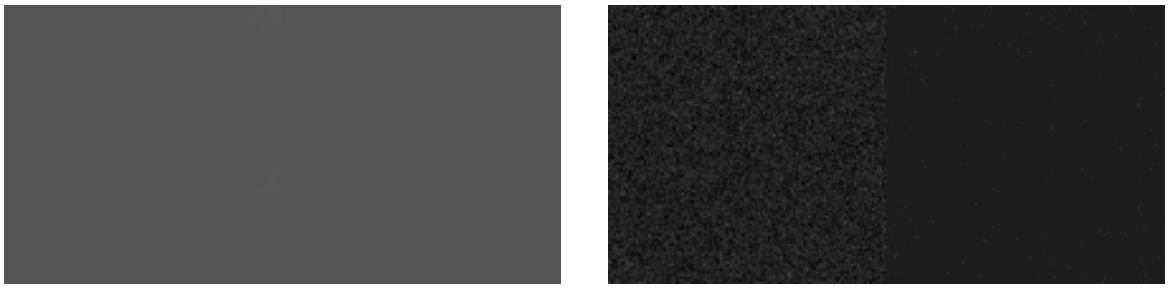
In Figure 62 we see that the network has learned to remove the salt-and-pepper noise. Furthermore, in our opinion the network has learned to sharpen the image so that we can make out more features and details of the ribosome. However, the network output seems to differ from the ground truth in a number of places. Therefore, we created an average of all the network outputs on the test set which we rendered in UCSF Chimera in Figure 64 (b). The ground truth rendering is displayed in Figure 64 (a).



(a) **(b)**

Figure 64: Comparison of (a) cleaned-up 17000 ground truth average to (b) average of **100av-to-gt** network outputs (rendered in UCSF Chimera in mesh mode).

In our opinion the network has learned to extract more ribosome features but the object does not appear to be a proper copy of the ground truth. Most notably, the section below the membrane is shaped significantly differently. This might suggest that the network, in addition to denoising and sharpening the images, also creates non-existent ribosome parts to lower the MSE. This is unwanted behaviour which might be mitigated with a different loss function. Some recommendations for future experiments with adapted loss functions are given in section 8. Next, we presented the trained networks with false pure noise input in the proper range and the results are displayed in Figure 65.



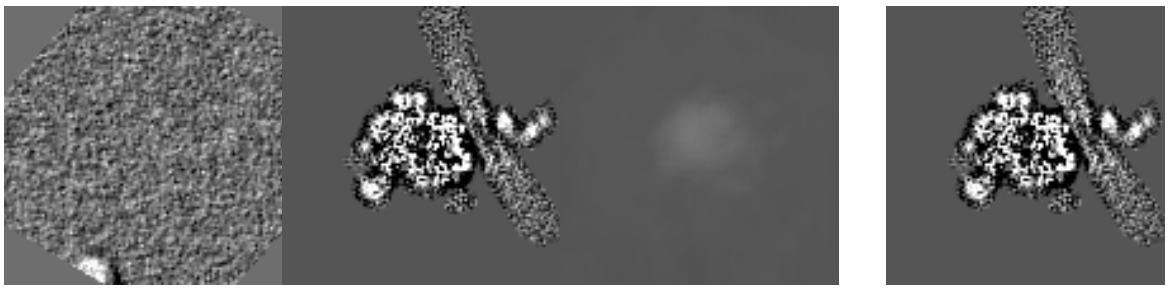
(a) **(b)** **(c)** **(d)**

Figure 65: (a) Network output of **100av-to-gt** for $\mathcal{N}(1,1)$ cube. (b) Network output of **100av-to-gt** for pure noise SIRTMIN reconstruction of $\mathcal{N}(0,10)$ sinogram. (c) Network output of **100av-to-1000av** for $\mathcal{N}(1,1)$ cube. (d) Network output of **100av-to-1000av** for pure noise SIRTMIN reconstruction of $\mathcal{N}(0,10)$ sinogram.

We tried a $\mathcal{N}(1,1)$-cube and a SIRTMIN reconstruction of a $\mathcal{N}(0,10)$ sinogram. In none of the four images in Figure 65 is there a clear example of overfitting. We do see that the **100av-to-1000av** network has not learned to get rid of the salt-and-pepper noise cloud entirely in Figure 65 (c).

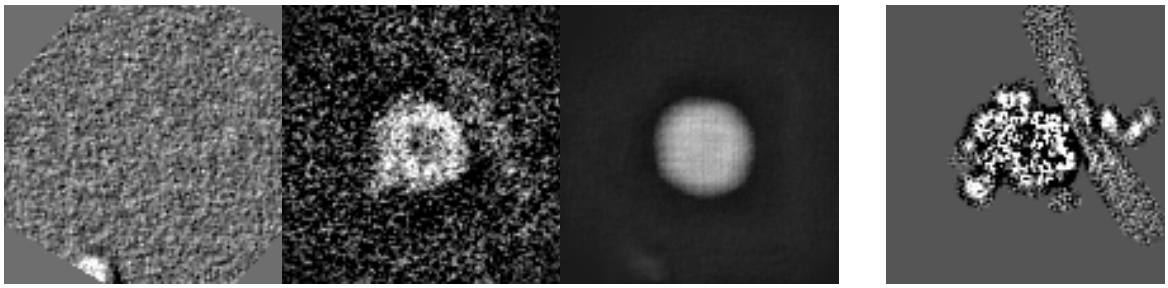
As a second experiment, we have trained two MS-D networks with the same settings as before to denoise individual ribosome particles. However, for this experiment we used a batch size of 100 instead of 10. One network was trained with 1000-average as target and one network was trained with the ground truth as target. The input particles were rotated so as to match the orientation of the target averages with the angles provided by the alignment algorithm specified in section 3.2. The 1000 particles were split in 800 training particles, 100 validation particles and 100 test particles. We ran validation after 5500 training slabs, i.e. after 55 passes of batches of 100. The results of applying the trained network to an example ribosome from the test set can be seen in Figures 66 and 67.

Remark: We have elected to show an example input where a gold fiducial marker is present to illustrate that the inputs can contain unusual artifacts. Furthermore, we think the example illustrates how hard it is to detect the ribosome in contrast to the gold marker.



(a) **(b)** **(c)** **(d)**

Figure 66: **single-to-gt** Slice 45/110 for several reconstructions. (a) Input particle 976 of cryo-ET dataset. (b) Target ground truth of 17000 particles. (c) network output slice. (d) original ground truth of 17000 particles for comparison.



(a) (b) (c) (d)

Figure 67: **single-to-1000av** Slice 45/110 for several reconstructions. (a) Input particle 976 of cryo-ET dataset. (b) Target ground truth of 1000 particles. (c) network output slice. (d) original ground truth of 17000 particles for comparison.

The results of the experiments were largely unsuccessful. The networks did learn to denoise the input images but the ribosomes were barely distinguishable in the network outputs. In Figure 67 (c) we see that the network seems to have learned to output a circular artifact rather than sharpen the images. Furthermore, the averages of the test set were some circular artifact oriented at the origin and not worth displaying.

We hypothesize that the relatively poor results of this experiment are due to a certain inaccuracy in the orientation estimation of the subtomogram. When we rotated sample particles from the dataset to the ground truth orientation we often found a significant deviation. Even though we consider it impressive that such orientation estimates could be obtained on this dataset in the first place, it seems that the error in the orientation for the single reconstructions is too large for a 2D network to denoise.

That said, our previous experiment suggests that it might be possible to reduce the data requirements for a certain quality average using 2D MS-D networks. Furthermore, the first experiment might suggest that if the orientation is sufficiently precise, the network could also denoise the single reconstructions but this will have to be studied in further experiments.

7 Conclusion

In this thesis we investigated how FBP, SIRT, SIRTMIN and TV-FISTA interplay with the subtomogram averaging method. We created a simulated dataset of ribosomes based on actual cryo-ET data with added Gaussian and Poisson noise with various levels of severity.

Firstly, we tested the quality of the individual reconstructions on the simulated phantoms. It seems that FBP and SIRT suffer much more from the missing wedge artifacts than SIRTMIN and TV-FISTA. It is our opinion that SIRTMIN and TV-FISTA prove superior in dealing with the limited angular range. Furthermore, the quality of the reconstructions for high noise levels seems to be ordered from worst to best as FBP, SIRT, SIRTMIN, TV-FISTA. However, none of the individual reconstructions suitably recover the ribosome features at realistic cryo-ET noise profiles.

We saw that TV-FISTA has a tendency to clump together blobs of the same gray value for

higher noise levels. This is to be expected due to the necessity of having a large regularization parameter in these cases. As mentioned, we think that due to the tendency to remove salt-and-pepper noise and missing-wedge artifacts in TV-FISTA, it seems that the TV reconstructions would be ideally suited to estimate the orientation of the ribosome on.

In the next experiment we investigated which methods obtain the best (subtomogram) averages for a range of batch sizes $k \in \{5, 10, 25, 50, 100\}$. It seems that missing wedge artifacts are not problematic for averaging methods for uniformly sampled rotation angles. The quantitative analysis suggests that FBP is eventually the strongest algorithm for dealing with the limited angular range.

For all noise levels, we see significant improvements in the averaged reconstructions. FBP, SIRT and SIRTMIN clearly show the phantom and the smaller features are clearly present. Interestingly, the TV-FISTA reconstruction, albeit it largely noiseless, is still too blurred to make out a lot of the finer ribosome parts.

The quantitative analysis would also have us conclude that TV-FISTA is the superior algorithm when combined with subtomogram averaging. However, we feel that these quantitative measures may overemphasize the importance of setting the vacuum to zero and hence penalize methods such as SIRT compared to TV-FISTA. Ultimately, it is our opinion that SIRTMIN is the strongest algorithm. In the experiments large averages for SIRT and SIRTMIN were similar but for lower averages SIRTMIN performs better and it was less sensitive to missing wedge artifacts.

Additionally we explored the effect of misalignment on the quality of the averages. Naturally we observed that the averages get progressively more blurred as angular accuracy dropped. Regardless of angular accuracy, subtomogram averaging, even for small averages, mitigates missing wedge problems for uniformly sampled rotation angles.

As for the robustness of each separate method we could not conclusively determine that one algorithm was more robust than the other since the quantitative results seemed to suggest that we mainly measured the results of averaging out salt-and-pepper noise. We did not observe a discernible difference in the deteriorating images from method to method.

After the initial experiments, we remarked that each reconstruction was still created from a set of projections taken from a limited angular range. We created an ensemble reconstruction method and used the flexibility of the ASTRA toolbox to create a set of flexible scanning geometries which allows us to create one large reconstruction, thereby interchanging the order of combining the subtomograms and reconstructing from the projections.

We tested the ensemble method for SIRTMIN and obtained promising results. Unsurprisingly, we observed that the least noisy reconstructions perform best and that the error decreases if we take more projections for our ensemble average. It seemed that 10 sets of projections was already sufficient to mitigate missing wedge problems. Overall we found that using SIRTMIN in a combined reconstruction is the best approach to perform cryo-ET reconstructions when we have access to several identical particle tomograms.

In general it seems that the tomographic reconstruction pipeline that would perform best is to 1) create strongly regularized TV-FISTA reconstructions initially; 2) use the expectation-maximization algorithm to determine the rotation angles for each subtomogram; 3) use the

ensemble-reconstruction method with SIRTMIN to create one reconstruction using the original projections to obtain the final tomogram of the object of study.

As an alternate goal we aimed to improve the quality of cryo-ET reconstructions and reduce the amount of data required to obtain it using post-reconstruction denoising with MS-D networks. We trained 8 different MS-D networks on different sets of training and target data. A preliminary conclusion from the initial tests was that the networks perform better when the volume slices were provided in adjacent slabs.

In all cases we observed a profound decrease in salt-and-pepper noise. The networks were able to recover the phantoms from data that at first-glance seemed devoid of signal. These conclusions were maintained in the quantitative analysis where an AMSE drop of several orders of magnitude was observed. No evidence of overfitting was found when the network was presented with false input data.

In addition, we found that the network improved FBP reconstructions more than SIRTMIN reconstructions. We hypothesize that this is due to fact that SIRTMIN initially outputs reconstructions which resemble the ground truth more closely. In our opinion the network outputs the same quality reconstruction when it is given an FBP reconstruction compared to a SIRTMIN reconstruction. FBP is computationally a lot faster than SIRTMIN so this suggests that we could speedup the pipeline by using FBP reconstructions instead. Furthermore, the comparable results suggest that the MS-D networks are suitably versatile regardless of the tomographic reconstruction algorithm that constructs the initial volumes.

Next, we saw no evidence that indicates that an MS-D network performs worse with Poisson noise as opposed to Gaussian noise. An initial concern was that the network might invent its own features. However, after averaging several post-processed volumes we found no evidence to support that the averaged MS-D network outputs might contain false ribosome components on the simulated dataset.

In our experiments on the real cryo-ET dataset we found that 2D MS-D networks are capable of removing salt-and-pepper noise and sharpen the details on 100-batch averages when trained with the 17000-average ground truth as target. Training with a 1000-batch average as target resulted in blurring effects. However, the network trained on the 17000-average seemed to have learned to invent non-existent ribosome parts in some cases. This might be mitigated by using an adapted loss function which will be discussed briefly in section 8.

In addition we performed an experiment where 2D MS-D networks were trained to denoise single particle reconstructions. We deem the experiments to be unsuccessful since neither the single network output nor the test set average resembled a discernible ribosome. After some tests we concluded that the angles estimated by the alignment algorithm were probably of insufficient precision. The 2D MS-D network architecture is ill-suited to learn 3D rotations as well as denoising. However, since we obtained promising results when the orientations of the 100-averages and the ground truth aligned, we hypothesize that this experiment will be more successful if the angles can be estimated more accurately.

Generally we conclude that substantial gains can be made to the quality of subtomogram averages by post-processing with trained MS-D networks when a higher quality average is available. If a suitable best average ribosome can be obtained, further scans can be boosted in quality by

a trained neural net which should reduce the data requirements for a ribosome reconstruction of a certain quality. In addition, it may be that an average of MS-D network processed reconstructions will will ultimately reveal more ribosome structure. However, such a conclusion is outside of the scope of this thesis since it would require a more thorough test on the full dataset and it would require re-aligning the subtomograms.

8 Recommendations for future work

Finally we want to comment on possible follow up experiments. We have not investigated what happens to the missing wedge artifacts when the ribosomes have a limited set of orientations. We understand from prof. Förster that this is certainly a realistic scenario so such experiments would be useful to better understand the importance of missing wedge artifacts. Furthermore, we have been careful to compartmentalize the separate experiments of this thesis and we have therefore not touched on the alignment algorithm. We hypothesize that TV reconstructions will allow for a more accurate estimate of the orientation than the current NFFT reconstructions. Furthermore, we have not run our TV-FISTA and ensemble-method pipeline on the actual cryo-ET data since it would require reobtaining all the orientations and re-running the alignment algorithm. We did not have the time to implement and run this algorithm and get it to work for real data. Furthermore, the currently used TV-FISTA library [30] does not support the required 3D ASTRA geometry for the ensemble method. For future work, this support could be implemented and tests could be run with a much lower regularization parameter for TV-FISTA.

Next, it should be briefly mentioned that we attempted to align the subtomograms using a GOMEA evolutionary algorithms [45], [46], [47]. However, the method failed and we have therefore not discussed it in this thesis. Perhaps a better framework could get such hybrid evolutionary algorithms to work for 3D alignment.

Alternatively, since the MS-D network results showed promise in some cases, perhaps denoising with MS-D networks before alignment may lead to performance gains. Perhaps a comparison of SIRTMIN or FBP in conjunction with MS-D denoising can be made to TV-FISTA alignment on the basis of total running time. Furthermore, due to the computational load of 3D networks we have limited ourselves to 2D MS-D networks with slabs. If a sufficiently fast 3D network architecture can be developed we could utilize deep learning methods for more than denoising. Perhaps alignment can be performed by a 3D network from the viewpoint of deformable registration. Some recent results where neural networks were used for deformable registration have been promising [1], [14].

Lastly, we encountered issues with trained networks arguably inventing ribosome features in the experiments performed in section 6.4. We recommend that an adapted loss function be used for further experiments which incurs a special penalty if the network output deviates strongly from the ground truth where there is strong signal. This might be realized by a weighted sum of two MSE functions where one is applied to thresholded versions of the network output and ground truth.

BIBLIOGRAPHY

[1] Guha Balakrishnan, Amy Zhao, Mert R Sabuncu, John Guttag, and Adrian V Dalca. Voxelmorph: a learning framework for deformable medical image registration. *IEEE transactions on medical imaging*, 2019.

[2] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Img. Sci.*, 2(1):183–202, March 2009.

[3] M Bertero and Michele Piana. *Inverse problems in biomedical imaging: Modeling and methods of solution*, pages 1–33. 03 2007.

[4] Charles L Byrne. Iterative algorithms in inverse problems. 4 2006.

[5] Maria Cameron, Sergey Fomel, and James Sethian. Inverse problem in seismic imaging. *PAMM*, 7:1024803 – 1024804, 12 2007.

[6] Yuxiang Chen and Friedrich F˙ Iterative reconstruction of cryo-electron tomograms using nonuniform fast fourier transforms. *Journal of Structural Biology*, 185(3):309 – 316, 2014.

[7] Yuxiang Chen, Stefan Pfeffer, Thomas Hrabe, Jan Michael Schuller, and Friedrich F˙ Fast and accurate reference-free alignment of subtomograms. *Journal of Structural Biology*, 182(3):235 – 245, 2013.

[8] Hendrik Meinert Dirks. *Variational Methods for Joint Motion Estimation and Image Reconstruction*. PhD thesis, Westfälische Wilhelms-Universität Münster, 2015.

[9] J Dubochet. Cryo-em—the first thirty years. *Journal of microscopy*, 245(3):221–224, 2012.

[10] Friedrich F˙ Recent advances in electron tomography. *Journal of Structural Biology*, 197(2):71 – 72, 2017. Electron Tomography.

[11] Catherine F. Higham and Desmond J. Higham. Deep learning: An introduction for applied mathematicians. *CoRR*, abs/1801.05894, 2018.

[12] Thomas Hrabe, Yuxiang Chen, Stefan Pfeffer, Luis Kuhn Cuellar, Ann-Victoria Mangold, and Friedrich F˙Pytom: A python-based toolbox for localization of macromolecules in cryo-

electron tomograms and subtomogram analysis. *Journal of Structural Biology*, 178(2):177 – 188, 2012. Special Issue: Electron Tomography.

[13] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[14] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015.

[15] S. Kaczmarz. Angenäherte Auflösung von Systemen linearer Gleichungen. *Bulletin International de l'Académie Polonaise des Sciences et des Lettres*, 35:355–357, 1937.

[16] Avinash C. Kak and Malcolm Slaney. *Principles of computerized tomographic imaging*. Society for Industrial and Applied Mathematics, 2001.

[17] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[18] Roman I. Koning, Abraham J. Koster, and Thomas H. Sharp. Advances in cryo-electron tomography for biology and medicine. *Annals of Anatomy - Anatomischer Anzeiger*, 217:82 – 96, 2018.

[19] Katrina Kramer. *The birth of the cool*, 2017 (accessed May 6, 2019).

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[21] Richard A. Kruger, Kenyon K. Kopecky, Alex M. Aisen, Daniel R. Reinecke, Gabe A. Kruger, and William Lester Kiser. Thermoacoustic ct with radio waves: a medical imaging paradigm. *Radiology*, 211 1:275–8, 1999.

[22] Kenneth Lange, Richard Carson, et al. Em reconstruction algorithms for emission and transmission tomography. *J Comput Assist Tomogr*, 8(2):306–16, 1984.

[23] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[24] Vladan Lucic, Alexander Rigort, and Wolfgang Baumeister. Cryo-electron tomography: The challenge of doing structural biology in situ. *The Journal of cell biology*, 202:407–19, 08 2013.

[25] Russell M. Mersereau and A.V. Oppenheim. Digital reconstruction of multidimensional signal from their projection. 62:1319 – 1338, 11 1974.

[26] Kevin P. Murphy. *Machine learning : a probabilistic perspective*. MIT Press, Cambridge, Mass. [u.a.], 2013.

[27] Frank Natterer. Fourier reconstruction in tomography. 47:343–353, 09 1985.

[28] Mahsa Noori Asl and Alireza Sadremomtaz. Analytical image reconstruction methods in emission tomography. *Journal of Biomedical Science and Engineering*, 06:100–107, 01 2013.

[29] Willem Jan Palenstijn, Jeroen Bédorf, Jan Sijbers, and K. Joost Batenburg. A distributed astra toolbox. *Advanced Structural and Chemical Imaging*, 2(1):19, Dec 2016.

[30] Daniël Pelt. Python library for Total Variation minimization in tomography, August 2015. `https://github.com/dmpelt/pytvtomo/blob/master/tvtomo/FISTA.py`.

[31] Daniël Pelt. Python library for mixed-scale dense convolutional neural networks, July 2019. `https://github.com/dmpelt/msdnet`.

[32] Daniël M. Pelt and James A. Sethian. A mixed-scale dense convolutional neural network for image analysis. *Proceedings of the National Academy of Sciences*, 115(2):254–259, 2018.

[33] Daniël Pelt, Joost Batenburg, and James Sethian. Improving tomographic reconstruction from limited data using mixed-scale dense convolutional neural networks. *Journal of Imaging*, 4(11):128–128, October 2018.

[34] E.F. Pettersen, T.D. Goddard, Conrad Huang, Greg Couch, D.M. Greenblatt, and Elaine Meng. Ucsf chimera—a visualization system for exploratory research and analysis. *J Comput Chem*, 25, 01 2004.

[35] Stefan Pfeffer, Laura Burbaum, Pia Unverdorben, Markus Pech, Yuxiang Chen, Richard Zimmermann, Roland Beckmann, and Friedrich Förster. Structure of the native sec61 protein-conducting channel. In *Nature communications*, 2015.

[36] Stefan Pfeffer, Johanna Dudek, Marko Gogala, Stefan Schorr, Johannes Linxweiler, Sven Lang, Thomas Becker, Roland Beckmann, Richard Zimmermann, and Friedrich Förster. Structure of the mammalian oligosaccharyl-transferase complex in the native er protein translocon. *Nature communications*, 5:3072, 2014.

[37] Stefan Pfeffer, Johanna Dudek, Miroslava Schaffer, Bobby G Ng, Sahradha Albert, Jürgen M Plitzko, Wolfgang Baumeister, Richard Zimmermann, Hudson H. Freeze, Benjamin D Engel, and Friedrich Förster. Dissecting the molecular organization of the translocon-associated protein complex. In *Nature communications*, 2017.

[38] T. A. Pitts, J. F. Greenleaf, J. L. Jian-Yu Lu, and R. R. Kinnick. Tomographic schlieren imaging for measurement of beam pressure and intensity. In *1994 Proceedings of IEEE Ultrasonics Symposium*, volume 3, pages 1665–1668 vol.3, Oct 1994.

[39] Azriel Rosenfeld and Avinash C. Kak. *Digital Picture Processing: Volume 1.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2 edition, 1982.

[40] Leonid I. Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Phys. D*, 60(1-4):259–268, November 1992.

[41] Yousef Saad. *Iterative methods for sparse linear systems*, volume 82. siam, 2003.

[42] Devika Sirohi, Zhenguo Chen, Lei Sun, Thomas Klose, Theodore C. Pierson, Michael G. Rossmann, and Richard J. Kuhn. The 3.8 å resolution cryo-em structure of zika virus. *Science*, 352(6284):467–470, 2016.

[43] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[44] K. TANABE. Projection method for solving a singular system of linear equations and its applications. *Numerische Mathematik*, 17:203–214, 1971.

[45] Dirk Thierens. The linkage tree genetic algorithm. In Robert Schaefer, Carlos Cotta, Joanna Kołodziej, and Günter Rudolph, editors, *Parallel Problem Solving from Nature, PPSN XI*, pages 264–273, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[46] Dirk Thierens and Peter AN Bosman. Optimal mixing evolutionary algorithms. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 617–624. ACM, 2011.

[47] Dirk Thierens and Peter A.N. Bosman. Hierarchical problem solving with the linkage tree genetic algorithm. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, GECCO '13, pages 877–884, New York, NY, USA, 2013. ACM.

[48] Pia Unverdorben, Florian Beck, Paweł Śledź, Andreas Schweitzer, Günter Pfeifer, Jürgen M. Plitzko, Wolfgang Baumeister, and Friedrich Förster. Deep classification of a large cryo-em dataset defines the conformational landscape of the 26s proteasome. *Proceedings of the National Academy of Sciences*, 111(15):5544–5549, 2014.

[49] W. Wan and J.A.G. Briggs. Chapter thirteen - cryo-electron tomography and subtomogram averaging. In R.A. Crowther, editor, *The Resolution Revolution: Recent Advances In cryoEM*, volume 579 of *Methods in Enzymology*, pages 329 – 367. Academic Press, 2016.

[50] Wikipedia. *Sobolev inequality*, 2018 (accessed November 16, 2018).

CHAPTER 3

APPENDIX

9 Appendix

9.1 Reconstruction algorithms in the Fourier plane

Recall that the Radon transform was defined as

$$\mathcal{R}_\theta[f](r) := \int_{-\infty}^{\infty} f(r\cos(\theta) + t\sin(\theta),\ r\sin(\theta) - t\cos(\theta))\ dt.$$

If we Fourier transform each angular projection $\mathcal{R}_\theta[f](r)$ (for all r and all θ) we end up with the Fourier transform of f using the Fourier slice theorem. We can then use the inverse Fourier transform to obtain f. Note that this is a proper inverse and analytically this is a perfect reconstruction. However, if we only sample at a finite set of points r_i and angles θ_i, we will not be able to fully construct $\hat{f}(x, y)$.

Moreover, regular spaced points r_i correspond with regularly spaced angles $\omega_i = \omega r_i$ in Fourier space. Therefore, the points on a line $\omega(\cos(\theta), \sin(\theta))$ in Fourier space are only sampled at $\omega_i(\cos(\theta), \sin(\theta))$ (so equidistant on the radial line). Hence, even if we measured all angles θ perfectly we would have samples on circles centered at the origin with radii ω_i (in Fourier space). The finite angles make it so that we only have points at particular angles on these circles.

If we had samples on a regular rectangular grid in Fourier space, we could use algorithms such as the fast Fourier Transform (FFT) to reconstruct f. This is not the case since the density of the samples decreases as we move radially outward. One approach is to sample at angles θ such that we create a grid of concentric squares of samples [27] rather than concentric circles and then use a (discrete) FFT to transform back. Such methods fall in the category of *(direct) Fourier Reconstruction* methods.

9.1.1 Sampling on a polar raster

Suppose we bandlimit f to some constant b, i.e. assume that $\hat{f}(\mathbf{w})$ is negligible for $|\mathbf{w}| \geq b$. A more formal definition is that the Fourier transform of a function needs to be zero outside a

finitely bounded region in Fourier space. Note that if a function contains a discontinuity (for example an edge in an image) then it is already not bandlimited. If a function is bandlimited by b, it can be represented by samples by ([25] eq. 3.4)

$$\hat{f}(x,y) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \hat{f}\left(\frac{m\pi}{b}, \frac{n\pi}{b}\right) \frac{\sin(\frac{b}{\pi}(x - \frac{m\pi}{b}))\sin(\frac{b}{\pi}(y - \frac{n\pi}{b}))}{\frac{b^2}{\pi^2}(x - \frac{m\pi}{b})(y - \frac{n\pi}{b})} \tag{9.1}$$

Furthermore, note that a bandlimit of b guarantees that $\hat{f}$ is contained in a circle of radius $\frac{\pi}{\sqrt{2}}b$. Hence, we should limit our measurements on the radial lines to be equispaced within this circle.

As mentioned, we would like to have samples on a regular rectangular grid if we want to quickly invert the Fourier transform with FFT. This can be achieved by estimating the missing DFT points by interpolation.

With *zeroth order interpolation* the missing DFT grid point is assigned the value of the nearest sample point. With *first order interpolation* the assigned value is the weighted average of the four nearest sample points

$$\hat{f}(c) = \frac{\frac{1}{d_1}\hat{f}(p_1) + \frac{1}{d_2}\hat{f}(p_2) + \frac{1}{d_3}\hat{f}(p_3) + \frac{1}{d_4}\hat{f}(p_4)}{\frac{1}{d_1} + \frac{1}{d_2} + \frac{1}{d_3} + \frac{1}{d_4}}.$$

Here p_i represents the nearest points in Fourier space and the samples are weighted inversely with the Cartesian distance d_i between the sample point and the missing grid point. Unfortunately, these interpolation methods only approximate the original Fourier transform and therefore introduce noise and artifacts in the reconstruction.

To summarize, we obtain samples of $\hat{f}$ in Fourier space by using the 1D transform on the projections along a line. Then, we create samples of $\hat{f}$ on a square grid by using some form of interpolation. Next, use a 2D inverse Fourier transform to obtain samples of f. Lastly, we set f to be as in equation 9.1.

9.1.2 Sampling on a square raster

Instead of sampling similarly for each angle, we can change the sampling according to each angle of projection. For the polar raster, we spaced the samples equally on a circle of radius $\pi b/\sqrt{2}$. Assume that we take N samples for each slice. Consider the square area within which f is not bandlimited. If $\theta = 0$, we get equally spaced samples along the ω_x axis with spacing b/N. However, at $\theta \neq 0$, the distance over which we sample is more than b. Hence, we will get a larger spacing if we still only obtain N samples. Instead, suppose that we sample

$$N_\theta = \frac{b}{\max\{|\cos(\theta)|, |\sin(\theta)|\}}$$

points for each θ. This will put each sample on a concentric square grid. The advantage is that by definition the bandlimit b is defined as a square area. So instead of sampling on a circle that contains it, we sample on exactly that square. For a big polar raster, approximately 35% of samples may fall outside the bandlimit region.

Furthermore, we only need to use 1D interpolation since the samples are on the same vertical and horizontal lines as the Cartesian grid assumed for FFT. By placing the interpolation positions at such a rectangular grid we should decrease the interpolation error [25].

9.1.3 Nonuniform Fast Fourier Transform (NFFT or NUFFT)

In many cases, the input samples are not uniformly spaced and cannot easily be interpolated to a regular grid without many artifacts. This can happen when we samples are not obtained over the entire 180 degree range. However, equispaced data is a requirement for using FFT algorithms. In cases where the data is not uniformly distributed, it is more useful to use a nonuniform fast Fourier transform (often called non-equispaced fast Fourier transform NFFT or nonuniform fast Fourier transform NUFFT).

The term NFFT or NUFFT has a slightly different interpretation in many cases. Generally it comes down some sort of discrete Fourier transform as we will describe shortly. Often difference lie in whether there is some interpolation and whether there is some assumed step size along a certain axis. We will give a general nonuniform discrete Fourier transform (NDFT) equation here.

Let $x_i \in \mathbb{R}$ and $\mathbf{x}_i \in \mathbb{R}^2$ with $i = 0, \dots N-1$ be the sample points where we measure g and h respectively (1D and 2D case). In that case the DFTs are defined as

$$\hat{g}_m := \sum_{j=0}^{N-1} g(x_j) e^{-2\pi i \omega_m x_j}, \quad \hat{h}(\mathbf{w}_m) := \sum_{j=0}^{N-1} h(\mathbf{x}_j) e^{-2\pi i \mathbf{w}_m \cdot \mathbf{x}_j}.$$

9.1.4 Iterative Nonuniform fast Fourier transform Reconstruction method (INFR)

In this section we will briefly discuss the (iterative) NFFT that was used to create the original reconstructions as described in [6]. As mentioned in the previous section, we start with an NUFFT expression of the form

$$\hat{h}(\mathbf{w}_m) := \sum_{\mathbf{x} \in I_N} h(\mathbf{x}) e^{-2\pi i \mathbf{x} \cdot \mathbf{w}_m}, \quad 1 \leq m \leq M. \tag{9.2}$$

Here the set I_N was defined by $N \in \mathbb{Z}_{>0}$, d is the dimension and $I_N := \{\mathbf{x} \in \mathbb{Z}^d \mid -\frac{N}{2} \leq x_i \leq \frac{N}{2},\ 1 \leq i \leq d\}$. In other words, I_N defines an equispaced grid with bandwidth N for all dimensions. Note that we have no restrictions on $\mathbf{w}_m$.

Define the $M \times |I_N|$ matrix A to be $A := e^{-2\pi i \mathbf{x} \cdot \mathbf{w}_m}$ where m indicates the row number. Define the vector $\mathbf{h} = (h(\mathbf{x}_1), h(\mathbf{x}_2), \dots)$ that contains each $h(\mathbf{x}_j)$ for $\mathbf{x}_j \in I_N$. Then the matrix equation

$$\hat{\mathbf{h}} = A\mathbf{h}$$

gives the entries $\hat{h}(\mathbf{w}_m)$. Now consider the tomographic setup where we have M observations (projection values) in Fourier space which we collect in a vector $\mathbf{b}$. Then the matrix equation becomes

$$A\mathbf{h} = \mathbf{b}. \tag{9.3}$$

Here each $\mathbf{w}_m$ is chosen depending on the sampling geometry that determines $\mathbf{b}$. Solving equation 9.3 can be modeled as an optimization problem

$$\mathbf{h} = \arg\min_{\mathbf{h}'} \left\| \mathbf{b} - A\mathbf{h}' \right\|.$$

In [6] it is shown that for a suitable density compensation matrix W, we can transform the problem into solving

$$A^* W A\mathbf{h} = A^* W \mathbf{b}. \tag{9.4}$$

Here $A^* := \hat{f}(\mathbf{w}_m) e^{2\pi i \mathbf{x}\cdot\mathbf{w}_m}$ is the conjugate transpose of A. Equation 9.4 is subsequently solved by conjugate gradient methods on the normal equations [41].

9.2 Defining total variation

Let U, V be Banach or Hilbert spaces. A Hilbert space is a vector space over the real or complex numbers with an inner product such that the norm defined by the inner product turns the space into a complete metric space (all Cauchy sequences converge to a limit in the space). A Banach space is a complete vector space over the real and complex numbers with a norm. The difference between the two is subtle but a Hilbert space is a Banach space with norm induced by an inner product.

Let $K : U \to V$ and $f \in V$, consider the inverse problem

$$Ku = f.$$

In this continuous setting, we define the above problem to be *well-posed* if

- $Ku = f$ has a *unique* solution $u \in U$ for all $f \in V$ (existence and uniqueness).
- If $f_n \to f$ is a converging sequence in V, then $\{u_n\}$ defined by $Ku_n = f_n$ is a converging sequence in U with $u_n \to u$ (stability). In other words, K^{-1} is continuous if K^{-1} exists.

Tomography suffers from existence and uniqueness problems. One way to tackle such ill-posed problems is to define a variational framework with a suitable regularizer.

Let $u : \Omega_2 \to \mathbb{R}$ and $f : \Omega_1 \to \mathbb{R}$ with $\Omega_i \subseteq \mathbb{R}^d$ ($d = 2, 3$ usually). We consider the general formulation of the energy functional for imaging problems

$$\mathcal{J}(u) = \frac{1}{2} \int_{\Omega_1} \|[Ku](\mathbf{x}) - f(\mathbf{x})\|^2 \, d\mathbf{x} + \alpha \int_{\Omega_2} F(\mathbf{y}, u, \nabla u) \, d\mathbf{y} \quad \to \quad \min_{u \in U} \tag{9.5}$$

for some function F. F should be defined such that the integral is large for "unreasonable" images and small for good images. There are several choices to be made here, e.g. do we want smooth images or strong edges? Furthermore, the question arises what space of functions U we allow u to live in. We would like to define U such that the signal is easily separated from the noise or that the noise is not even in the space.

9.2.1 Sobolev spaces

Firstly, consider the *Lebesgue spaces*

$$L^p(\Omega) := \left\{ u : \Omega \to \mathbb{R} \;\middle|\; \int_\Omega \|u(\mathbf{x})\|^p \, d\mathbf{x} < \infty \right\}.$$

$$L^\infty(\Omega) := \left\{ u : \Omega \to \mathbb{R} \;\middle|\; \text{ess.sup}_{\mathbf{x}\in\Omega}(\|u(\mathbf{x})\|) < \infty \right\}.$$

The *essential supremum* is basically a generalization of the maximum for measurable functions. We define the L^P- and L^∞-norms to be

$$\|u\|_{L^P(\Omega)} := \left(\int_\Omega \|u(\mathbf{x})\|^p \; d\mathbf{x} \right)^{1/p}, \quad \|u\|_{L^\infty(\Omega)} := \text{ess.sup}_{x\in\Omega}(\|u(\mathbf{x})\|)$$

respectively.

The *dual space* of the Lebesgue space $L^p(\Omega)$ is $L^q(\Omega)$ with $\frac{1}{p}+\frac{1}{q}=1$. Curiously, $L^1(\Omega)^\vee = L^\infty(\Omega)$ but $L^\infty(\Omega)^\vee \neq L^1(\Omega)$. Lebesgue spaces contain all reasonable images but also all noisy images. For example, a normally distributed function $g(\mathbf{x}) \sim \mathcal{N}(0,\sigma^2)$ has bounded L^2-norm

$$\|g\|_{L^2(\Omega)} = \left(\int_{\mathbb{R}} \|g(\mathbf{x})\|^2 \; d\mathbf{x} \right)^{1/2} = \sigma.$$

Therefore, noise is included in the Lebesgue space and we need to define a smaller space.

We define *Sobolev spaces* as

$$W^{k,p} := \{u \in L^p(\Omega) \mid D^\alpha u \in L^p(\Omega), |\alpha| \leq k\}.$$

Here $k \in \mathbb{N}_0$ and $1 \leq p < \infty$. It is convention to write $W^{k,2}(\Omega) = H^k(\Omega)$.

Firstly, $D^\alpha u$ is the *weak derivative of order* α of u. Suppose that u is locally integrable on Ω, u is said to be *weakly differentiable* w.r.t. to the i-th index $\mathbf{x}_i$ if there exists a locally integrable function g_i on Ω such that

$$\int_\Omega u\partial_i\varphi \; d\mathbf{x} = -\int_\Omega g_i\varphi \; d\mathbf{x}, \quad \text{for all } \varphi \in \mathcal{C}_c^\infty(\Omega) \tag{9.6}$$

The space of functions $\mathcal{C}_c^\infty(\Omega)$ is the space of smooth functions with compact support on Ω. The function g_i is the i-th weak derivative of u and we denote it by $\partial_i u$. This means that the "integration by parts" formula holds for all $\varphi \in \mathcal{C}_c^\infty(\Omega)$ since the term $[u\varphi]_{\partial\Omega} = 0$ on the boundary.

This seems like a complicated definition but for a continuously differentiable function the weak derivative agrees with the pointwise derivative. Intuitively, the weak derivative looks differentiable everywhere except on "sets of measure 0". In other words, if a function is discontinuous on places where we cannot fix it by changing the function on a measure 0 set then it cannot have a weak derivative.

For example, the absolute value function $f(x) = |x|$ is weakly differentiable. Any function $g(x)$ which is 1 for $x > 0$ and -1 for $x < 0$ is a weak derivative (we can choose any value at $x = 0$). Take caution since many functions that look that they may be weakly differentiable may not be. The following lemma proves this for a common example.

Lemma 9.1. *The function*

$$f(x) = \begin{cases} -1 & x \leq 0 \\ 1 & x > 0 \end{cases}$$

is not weakly differentiable.

Proof. Let $\varphi \in \mathcal{C}_c^\infty(\mathbb{R})$, the left-hand side of equation 9.6 is

$$\begin{aligned} \int_{\mathbb{R}} f(x)\varphi'(x)\,dx &= \int_0^\infty f(x)\varphi'(x)\,dx + \int_{-\infty}^0 f(x)\varphi'(x)\,dx \\ &= \int_0^\infty \varphi'(x)\,dx - \int_{-\infty}^0 \varphi'(x)\,dx \end{aligned}$$

But φ has compact support and is smooth so this means that

$$\int_{\mathbb{R}} f(x)\varphi'(x)\,dx = -2\varphi(0).$$

Now take a sequence of smooth functions $\varphi_n \in \mathcal{C}_c^\infty(\mathbb{R})$ which have $\varphi_n(0) = 1$ but for $x \neq 0$, $\lim_{n\to\infty} \varphi_n(x) = 0$. So they converge to spike function at 0. If g is a weak derivative we have the equality for all n

$$2\varphi_n(0) = \int_{\mathbb{R}} g(x)\varphi_n(x)\,dx.$$

But in the limit this means that

$$2 = \lim_{n\to\infty} \int_{\mathbb{R}} g(x)\varphi_n(x)\,dx = 0$$

which is a contradiction. □

Now let $\alpha \in \mathbb{N}_0^n$ be a multi-index. The function u has *weak derivative of order* α, $\partial^\alpha u$, if

$$\int_\Omega u(\partial^\alpha \varphi)\,d\mathbf{x} = (-1)^{|\alpha|} \int_\Omega (\partial^\alpha u)\varphi\,d\mathbf{x}, \quad \text{for all } \varphi \in \mathcal{C}_c^\infty(\Omega). \tag{9.7}$$

Back to Sobolev spaces, the norm in the Sobolev space is defined as

$$\|u\|_{W^{k,p}}^p = \sum_{|\alpha|\leq k} \|D^\alpha u\|_{L^p}^p .$$

Consider $W^{1,p}$, this is the space of functions where u and the weak derivatives of u of order 1 are p-integrable. We saw earlier that there was a discontinuous functions without a weak derivative and which is therefore not in the Sobolev space. In fact, we have that if $u \in W^{1,p}(\mathbb{R}^n)$, then u is Hölder continuous [50] which implies continuity. In fact, the following lemma holds.

Lemma 9.2. *Let $S \subset \Omega \subset \mathbb{R}^d$ be a bounded subdomain with $\mathcal{C}^1$-boundary. Then the step-function*

$$f(x) = \begin{cases} 1 & x \in S \\ 0 & \textit{otherwise} \end{cases}$$

is not in $W^{1,p}(\Omega)$ for $p > 1$.

Proof. Proof omitted. □

We do not want to go into further details on Sobolev spaces but several inequalities proving the above statements van be found here [50]. The point is that we see that the Sobolev space is to restrictive since discontinuous functions which may resemble edges in an image are not included.

9.2.2 Space of bounded variations

We saw in the previous sections that Lebesgue spaces allow noisy functions as possible solutions and Sobolev spaces are too restrictive to allow for all types of images we may encounter. The *space of bounded (total) variation* is constructed such that it contains exactly what an observer would call an image.

Firstly, we will define the space of bounded variation and the Rudin-Osher-Fatemi model. The ROF model is defined as the minimization problem

$$\frac{1}{2}\int_\Omega (u-f)^2 \, d\mathbf{x} + \alpha |u|_{TV(\Omega)} \quad \rightarrow \quad \min_u$$

where we define the *total variation* of u to be

$$|u|_{TV(\Omega)} := \sup_{g \in \mathcal{C}_0^\infty(\Omega,\mathbb{R}^d),\, \|g\|_\infty \leq 1} \int_\Omega u \cdot \nabla g \, d\mathbf{x}.$$

Subsequently, the space of functions of bounded variation is defined as

$$BV(\Omega) := \{u \in L^1(\Omega) \mid |u|_{TV(\Omega)} < \infty\}$$

In this case, the norm $\|g\|_\infty$ is defined as

$$\|g\|_\infty = \text{ess.sup}_{x\in\Omega} \sqrt{g_1(x)^2 + \cdots + g_d(x)^2}.$$

The norm on BV is then given by ($|u|_{TV(\Omega)}$ is only a semi-norm)

$$\|u\|_{BV} = \|u\|_{L^1} + |u|_{TV(\Omega)}.$$

This is a very complicated definition but if $u \in W^{1,1}(\Omega)$ then the total variation simplifies to

$$|u|_{TV(\Omega)} = \int_\Omega |\nabla u| \, d\mathbf{x}$$

which coincides with the definition we initially wanted to impose.

Theorem 9.3. *Suppose $\Omega \subset \mathbb{R}^n$ and $1 \leq p \leq \frac{n}{n-1}$. Then there exists a continuous embedding $BV(\Omega) \hookrightarrow L^p(\Omega)$. For $p < \frac{n}{n-1}$ the embedding is compact.*

The theorem shows that for we certainly have an embedding into $L^2(\Omega)$, reinforcing the idea that BV is inbetween Sobolev space and Lebesgue space. There is much more theory on this interpretation of total variation in [8]. However, it is out of scope of this thesis to treat more. One important result is the Fundamental theorem of optimization which shows that there exists a global minimum to functionals such as the ROF functional.

9.3 Additional mathematics on deep learning

9.3.1 Deriving update equations for MS-D networks

In this section we repeat the procedure in section 5.2 for MS-D networks. The equations in 5.6 can be rewritten in the terminology used in section 5.2. Since we are dealing with higher dimensional data, the notation can be quite tricky. However, we are ultimately still interested in the sensitivity of the loss function $\mathcal{J}_{\mathbf{x}_i}(\mathbf{\Theta})$ to some scalar weight w. We take the input images to have n rows and m columns.

Remark: Throughout this section the equations may seem daunting but please keep in mind that we are merely expanding the previous method to higher dimensional data and a more complex architecture.

Let $\mathbf{W}^{[l,j,k,h]} \in \mathbb{R}^{f\times f}$ be the kernel weight matrix of the convolution performed at layer l, channel j and input image from layer k, channel h. Denote the input image as $\mathbf{A}^{[k,h]}$. Let $\mathbf{B}^{[l,j]}$ denote the bias matrix for the convolution at layer l, channel j. Note that this matrix has the same value $b^{[l,j]}$ at each entry.

In the notation of section 5.2 we have

$$\mathbf{Z}^{[l,j]} = \sum_{k=1}^{l-1}\sum_{h=1}^{w} \mathbf{W}^{[l,j,k,h]} \circledast \mathbf{A}^{[k,h]} + \mathbf{B}^{[l,j]} \tag{9.8}$$

$$\mathbf{A}^{[l,j]} = \sigma(\mathbf{Z}^{[l,j]}). \tag{9.9}$$

Let subscripts $\mathbf{Z}^{[l,j]}_{x,y}$ denote the entry of the matrix $\mathbf{Z}^{[l,j]}$ in row x and column y. Consider the entry (p,q) of the kernel matrix. When performing the convolution, dilations and reflective padding make it difficult to determine with which entry of $\mathbf{A}^{[k,h]}$ the kernel weight should be multiplied. In my C++ code, a big tensor mapping (x,y,p,q) to the correct indices for each convolution is pre-generated. For now, we will assume that the function $\varphi_{l,j,k,h} : \mathbb{Z}^4 \to \mathbb{Z}^2$ performs this mapping.

Furthermore, in the C++ code another tensor tracks which weights were involved when we calculated $\mathbf{Z}^{[l,j]}_{x,y}$. With abuse of notation we will denote this map by $\varphi^{-1}_{l,j,k,h}$ even though this mapping is not invertible in the mathematical sense.

Remark: It is not strictly necessary to track these indices in a data structure since this is quite memory expensive. They can also be re-computed dynamically so this is a design choice between speed and memory.

We will disregard the subscripts in $\varphi_{l,j,k,h}$ for readability but the reader should remember that the correct mapping for this convolution is implied. With these notations we have that

$$\mathbf{Z}^{[l,j]}_{x,y} = \sum_{k=1}^{l-1}\sum_{h=1}^{w} \left[\mathbf{W}^{[l,j,k,h]} \circledast \mathbf{A}^{[k,h]}\right]_{x,y} + \mathbf{B}^{[l,j]}_{x,y} \tag{9.10}$$

$$= \sum_{k=1}^{l-1}\sum_{h=1}^{w} \left(\sum_{p=1}^{f}\sum_{q=1}^{f} \mathbf{W}^{[l,j,k,h]}_{p,q} \mathbf{A}^{[k,h]}_{\varphi(x,y,p,q)}\right) + \mathbf{B}^{[l,j]}_{x,y}. \tag{9.11}$$

Therefore we immediately see that

$$\frac{\partial \mathbf{Z}_{x,y}^{[l,j]}}{\partial \mathbf{W}_{p,q}^{[l,j,k,h]}} = \mathbf{A}_{\varphi(x,y,p,q)}^{[k,h]} \tag{9.12}$$

$$\frac{\partial \mathbf{Z}_{x,y}^{[l,j]}}{\partial \mathbf{B}_{x,y}^{[l,j]}} = 1. \tag{9.13}$$

The chain rule implies that

$$\frac{\partial \mathcal{J}_{\mathbf{x}_i}(\mathbf{\Theta})}{\partial \mathbf{B}_{x,y}^{[l,j]}} = \sum_{p=1}^{m}\sum_{q=1}^{n} \frac{\partial \mathcal{J}_{\mathbf{x}_i}(\mathbf{\Theta})}{\partial \mathbf{Z}_{p,q}^{[l,j]}} \frac{\partial \mathbf{Z}_{p,q}^{[l,j]}}{\partial \mathbf{B}_{x,y}^{[l,j]}} = \frac{\partial \mathcal{J}_{\mathbf{x}_i}(\mathbf{\Theta})}{\partial \mathbf{Z}_{x,y}^{[l,j]}}. \tag{9.14}$$

This can be seen by using the chain rule for functions $f : \mathbb{R}^m \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}^m$ and using the isomorphism of vector spaces $\mathbb{R}^{m\times n} \cong \mathbb{R}^{mn}$. The derivation becomes easier if we recall that the matrix $\mathbf{B}^{[l,j]}$ contains the same value at each pixel entry. Therefore, we can express it as

$$\mathbf{B}^{[l,j]} = b^{[l,j]}\mathbf{J}_{mn}$$

where $\mathbf{J}_{mn}$ is a matrix of only ones. Therefore, we can calculate the sensitivity of the loss function with respect to $b^{[l,j]}$ using the chain rule once more

$$\frac{\partial \mathcal{J}_{\mathbf{x}_i}(\mathbf{\Theta})}{\partial b^{[l,j]}} = \sum_{p=1}^{m}\sum_{q=1}^{n} \frac{\partial \mathcal{J}_{\mathbf{x}_i}(\mathbf{\Theta})}{\partial \mathbf{B}_{p,q}^{[l,j]}} \frac{\partial \mathbf{B}_{p,q}^{[l,j]}}{\partial b^{[l,j]}} = \sum_{p=1}^{m}\sum_{q=1}^{n} \frac{\partial \mathcal{J}_{\mathbf{x}_i}(\mathbf{\Theta})}{\partial \mathbf{Z}_{p,q}^{[l,j]}}. \tag{9.15}$$

Similarly, the chain rule implies

$$\frac{\partial \mathcal{J}_{\mathbf{x}_i}(\mathbf{\Theta})}{\partial \mathbf{W}_{p,q}^{[l,j,k,h]}} = \sum_{x=1}^{m}\sum_{y=1}^{n} \frac{\partial \mathcal{J}_{\mathbf{x}_i}(\mathbf{\Theta})}{\partial \mathbf{Z}_{x,y}^{[l,j]}} \frac{\partial \mathbf{Z}_{x,y}^{[l,j]}}{\partial \mathbf{W}_{p,q}^{[l,j,k,h]}} = \sum_{x=1}^{m}\sum_{y=1}^{n} \frac{\partial \mathcal{J}_{\mathbf{x}_i}(\mathbf{\Theta})}{\partial \mathbf{Z}_{x,y}^{[l,j]}} \mathbf{A}_{\varphi(x,y,p,q)}^{[k,h]}. \tag{9.16}$$

In order to fully determine what the sensitivity of all the weights to the loss function we need to compute $\frac{\partial \mathcal{J}_{\mathbf{x}_i}(\mathbf{\Theta})}{\partial \mathbf{Z}_{x,y}^{[l,j]}}$. To do so, we first consider the final layer $l = d$. We can express this partial derivative with respect to the output as

$$\frac{\partial \mathcal{J}_{\mathbf{x}_i}(\mathbf{\Theta})}{\partial \mathbf{Z}_{x,y}^{[d,j]}} = \frac{\partial \mathcal{J}_{\mathbf{x}_i}(\mathbf{\Theta})}{\partial \mathbf{A}_{x,y}^{[d,j]}} \frac{\partial \mathbf{A}_{x,y}^{[d,j]}}{\partial \mathbf{Z}_{x,y}^{[d,j]}}.$$

Both of these derivatives are easily calculated (with MSE loss $\mathcal{J}$).

$$\frac{\partial \mathcal{J}_{\mathbf{x}_i}(\mathbf{\Theta})}{\partial \mathbf{A}_{x,y}^{[d,j]}} = \mathbf{A}_{x,y}^{[d,j]} - \mathbf{Y}_{x,y}^{j}$$

$$\frac{\partial \mathbf{A}_{x,y}^{[d,j]}}{\partial \mathbf{Z}_{x,y}^{[d,j]}} = \sigma'\left(\mathbf{Z}_{x,y}^{[d,j]}\right).$$

Here $\mathbf{Y}^{[j]}$ is the true image for channel j. If we combine this we see

$$\frac{\partial \mathcal{J}_{\mathbf{x}_i}(\mathbf{\Theta})}{\partial \mathbf{Z}_{x,y}^{[d,j]}} = \sigma'\left(\mathbf{Z}_{x,y}^{[d,j]}\right)(\mathbf{A}_{x,y}^{[d,j]} - \mathbf{Y}_{x,y}^{j}). \tag{9.17}$$

For the other layers $l = 1, \ldots, d-1$ we need to think which other neurons are influenced by the input at $[l, j]$. Consider layer l and channel j, in section 5.2 we only needed to sum over the input values of layer $l+1$ since there were no channels and the layers were sequentially connected. This is sufficient since there the effect of $\mathbf{z}^{[l]}$ on $\mathcal{J}$ was fully contained in $\mathbf{z}^{[l+1]}$. Here, the input $\mathbf{Z}^{[l,j]}$ influences all other layers $l+1, \ldots, d$ and all their channels. So we obtain

$$\frac{\partial \mathcal{J}_{\mathbf{x}_i}(\boldsymbol{\Theta})}{\partial \mathbf{Z}_{x,y}^{[l,j]}} = \sum_{k=l+1}^{d} \sum_{h=1}^{w} \sum_{p=1}^{m} \sum_{q=1}^{n} \frac{\partial \mathcal{J}_{\mathbf{x}_i}(\boldsymbol{\Theta})}{\partial \mathbf{Z}_{p,q}^{[k,h]}} \frac{\partial \mathbf{Z}_{p,q}^{[k,h]}}{\partial \mathbf{Z}_{x,y}^{[l,j]}}. \tag{9.18}$$

The last partial derivative we need to evaluate to get all our results is $\frac{\partial \mathbf{Z}_{p,q}^{[k,h]}}{\partial \mathbf{Z}_{x,y}^{[l,j]}}$. To do so, recall that we have

$$\begin{aligned}
\mathbf{Z}^{[k,h]} &= \sum_{t=1}^{k-1} \sum_{s=1}^{w} \mathbf{W}^{[k,h,t,s]} \circledast \sigma\left(\mathbf{Z}^{[t,s]}\right) + \mathbf{B}^{[k,h]} \\
\mathbf{Z}_{p,q}^{[k,h]} &= \sum_{t=1}^{k-1} \sum_{s=1}^{w} \left[\sum_{i=1}^{f} \sum_{j=1}^{f} \mathbf{W}_{i,j}^{[k,h,t,s]} \sigma\left(\mathbf{Z}_{\varphi(p,q,i,j)}^{[t,s]}\right) \right] + \mathbf{B}_{p,q}^{[k,h]}.
\end{aligned}$$

This complicated expression simplifies when we take the derivative with respect to $\mathbf{Z}_{x,y}^{[l,j]}$

$$\frac{\partial \mathbf{Z}_{p,q}^{[k,h]}}{\partial \mathbf{Z}_{x,y}^{[l,j]}} = \sum_{\eta \in \varphi^{-1}(x,y)} \mathbf{W}_{\varphi^{-1}(x,y)}^{[k,h,l,j]} \sigma\left(\mathbf{Z}_{x,y}^{[l,j]}\right)$$

Therefore we end up with

$$\frac{\partial \mathcal{J}_{\mathbf{x}_i}(\boldsymbol{\Theta})}{\partial \mathbf{Z}_{x,y}^{[l,j]}} = \sum_{k=l+1}^{d} \sum_{h=1}^{w} \sum_{p=1}^{m} \sum_{q=1}^{n} \frac{\partial \mathcal{J}_{\mathbf{x}_i}(\boldsymbol{\Theta})}{\partial \mathbf{Z}_{p,q}^{[k,h]}} \sum_{\eta \in \varphi^{-1}(x,y)} \mathbf{W}_{\varphi^{-1}(x,y)}^{[k,h,l,j]} \sigma\left(\mathbf{Z}_{x,y}^{[l,j]}\right). \tag{9.19}$$

This looks complicated but it merely expresses the partial derivative of an earlier layer with respect to those further on. The analysis is cumbersome since there are many convolutions tied together and the same kernel weights influences many pixels. However, if the data structures currently represented by φ and φ^{-1} are properly maintained all of these operations are relatively straightforward to implement.

To recap, equation 9.17 allows us to find the sensitivity of the inputs for the final layer w.r.t. the loss function. Equation 9.19 then allows us to calculate all other such partial derivatives in a backward pass over the layers. Equations 9.16 and 9.15 then let us determine the relevant partial derivatives for the weights and biases. We can then do the regular update step with a certain learning parameter λ.

9.3.2 Adaptive moment estimation (ADAM)

When we propagate the error through the network in the backward pass, updates in the deeper layers (in reverse order) will often become arbitrarily small. To combat this, we could increase

the learning rate λ. However, this would lead to excessively large updates in the layers closest to the output layer. It seems therefore sensible to use different learning rates for the different layers and channels. When training a neural network, we have to choose an algorithm that sets these learning rates and adapts them over time. These algorithms go by the name of *optimizers* in deep learning.

In this section we will discuss *adaptive moment estimation* or ADAM [17]. It iteratively determines individual learning rates from estimates of the first and second order moments (mean and uncentered variance) of the gradients. During training, we want to minimize the loss function $\mathcal{J}(\boldsymbol{\Theta})$. However, during each iteration we consider a random training point $\mathbf{x}_i$ so the value of $\mathcal{J}(\boldsymbol{\Theta})$ is stochastic. It is therefore more prudent to minimize the expected value of the loss $\mathbb{E}[\mathcal{J}(\boldsymbol{\Theta})]$ with respect to its parameters $\boldsymbol{\Theta}$.

We denote the realizations of the loss function at each time step/iteration as $\mathcal{J}_{\mathbf{x}_i}(\boldsymbol{\Theta})$. Recall that previously we determined that we should update some j-th weight $\mathbf{w}_j$

$$\mathbf{w}_j \mapsto \mathbf{w}_j - \lambda \frac{\partial \mathcal{J}_{\mathbf{x}_i}(\boldsymbol{\Theta})}{\partial \mathbf{w}_j}$$

as we would do with stochastic gradient descent. Now we will iteratively update λ. Moreover, considering the now stochastic nature of the loss function, we will update by an estimate of the first moment of the gradient (normalized by the second moment) instead of simply the gradient as computed for that training point. We will denote the moving average of the first order moment of the gradient as

$$m_{\mathbf{x}_i} := \mathbb{E}_{\mathbf{x}_0,\ldots,\mathbf{x}_i}\left[\nabla \mathcal{J}_{\mathbf{x}_i}(\boldsymbol{\Theta})\right].$$

Furthermore, we will denote the moving average of the second order moment of the gradient as

$$v_{\mathbf{x}_i} := \mathbb{E}_{\mathbf{x}_0,\ldots,\mathbf{x}_i}\left[(\nabla \mathcal{J}_{\mathbf{x}_i}(\boldsymbol{\Theta}))^2\right].$$

Here the square means the element wise square of the gradient. Both the first-order and second-order moments are weighted by two scaling factors $\beta_1, \beta_2 < 1$ at each update step. Generally, default parameter settings are $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We will denote the gradient after the forward pass for training point $\mathbf{x}_i$ as $g_{\mathbf{x}_i}$. At the start of training both moving averages are set to zero. Then we perform the following updates

$$m_{\mathbf{x}_i} \mapsto \beta_1 m_{\mathbf{x}_{i-1}} + (1-\beta_1) g_{\mathbf{x}_i}, \qquad v_{\mathbf{x}_i} \mapsto \beta_2 v_{\mathbf{x}_{i-1}} + (1-\beta_2) g^2_{\mathbf{x}_i}.$$

Since the estimates are initialized at 0, and $1-\beta_1$ and $1-\beta_2$ are close to 0, these moving averages will initially be very close to 0. In [17] an expression is derived for the discrepancy between v_t and the real second moment

$$\mathbb{E}[v_{\mathbf{x}_i}] = \mathbb{E}[\left[(\nabla \mathcal{J}_{\mathbf{x}_i}(\boldsymbol{\Theta}))^2\right]] \cdot (1-\beta_2^i).$$

Therefore, after updating $v_{\mathbf{x}_i}$ we divide by the appropriate term to correct for the initialization bias (similarly for the first-order moment). The learning rate λ still influences the magnitude of the steps taken when updating the parameters. However, ADAM increments with the normalized first-order moment $\widehat{m_{\mathbf{x}_i}}/\sqrt{\widehat{v_{\mathbf{x}_i}}}$. If this quantity is small, the step size will also be small. This seems advantageous since this happens when the variance is large, i.e. we have a lot of

uncertainty regarding the proper direction of $\widehat{m_{\mathbf{x}_i}}$. To conclude, Adam performs the following set of updates until some convergence criterion is met:

$i \mapsto i+1$	
$g_{\mathbf{x}_i} \mapsto \nabla \mathcal{J}_{\mathbf{x}_i}(\boldsymbol{\Theta})$	Set the gradient
$m_{\mathbf{x}_i} \mapsto \beta_1 m_{\mathbf{x}_{i-1}} + (1-\beta_1) g_{\mathbf{x}_i}$	Update first-order moment
$v_{\mathbf{x}_i} \mapsto \beta_2 v_{\mathbf{x}_{i-1}} + (1-\beta_2) g^2_{\mathbf{x}_i}$	Update second-order moment
$\widehat{m}_{\mathbf{x}_i} \mapsto \frac{m_{\mathbf{x}_i}}{1-\beta_1^i}$	Correction first-order moment
$\widehat{v}_{\mathbf{x}_i} \mapsto \frac{v_{\mathbf{x}_i}}{1-\beta_2^i}$	Correction second-order moment
$\boldsymbol{\Theta}^{[i]} \mapsto \boldsymbol{\Theta}^{[i-1]} - \lambda \frac{\widehat{m}_{\mathbf{x}_i}}{\sqrt{\widehat{v}_{\mathbf{x}_i}} + \epsilon}$	Update parameters.

The parameter ϵ is simply there to avoid crashes in the case of zero variance. The authors recommend setting $\epsilon = 10^{-8}$ and $\lambda = 0.001$ [17].