Image-based Control and Automation of High-speed X-ray Imaging Experiments

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Tomáš Faragó

aus Banská Bystrica

Tag der mündlichen Prüfung: 30.1.2017

Erster Gutachter: Prof. Dr.-Ing. Rüdiger Dillmann

Zweiter Gutachter: Prof. Dr. Tilo Baumbach

Ich versichere hiermit wahrheitsgemäß, die Arbeit bis auf die angegebenen Hilfen selbständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und als kenntlich gemacht zu haben, was aus Arbeiten anderer und eigenen Veröffentlichungen unverändert oder mit Änderungen entnommen wurde.

Karlsruhe, den 25.11.2016

Tomáš Faragó

Abstract

X-ray imaging experiments shed light on internal material structures. The success of an experiment depends on the properly selected experimental conditions, mechanics and the behavior of the sample or process under study. Up to now, there is no autonomous data acquisition scheme which would enable us to conduct a broad range of X-ray imaging experiments driven by image-based feedback. This thesis aims to close this gap by solving problems related to the selection of experimental parameters, fast data processing and automatic feedback to the experiment based on image metrics applied to the processed data.

In order to determine the best initial experimental conditions, we study the X-ray image formation principles and develop a framework for their simulation. It enables us to conduct a broad range of X-ray imaging experiments by taking into account many physical principles of the full light path from the X-ray source to the detector. Moreover, we focus on various sample geometry models and motion, which allows simulations of experiments such as 4D time-resolved tomography.

We further develop an autonomous data acquisition scheme which is able to fine-tune the initial conditions and control the experiment based on fast image analysis. We focus on high-speed experiments which require significant data processing speed, especially when the control is based on compute-intensive algorithms. We employ a highly parallelized framework to implement an efficient 3D reconstruction algorithm whose output is plugged into various image metrics which provide information about the acquired data. Such metrics are connected to a decision-making scheme which controls the data acquisition hardware in a closed loop.

We demonstrate the simulation framework accuracy by comparing virtual and real grating interferometry experiments. We also look into the impact of imaging conditions on the accuracy of the filtered back projection algorithm and how it can guide the optimization of experimental conditions. We also show how simulation together with ground truth can help to choose data processing parameters for motion estimation by a *high-speed* experiment.

We demonstrate the autonomous data acquisition system on an *in-situ* tomographic experiment, where it optimizes the camera frame rate based on tomographic reconstruction. We also use our system to conduct a *high-throughput* tomography experiment, where it scans many similar biological samples, finds the tomographic rotation axis for every sample and reconstructs a full 3D volume on-the-fly for quality assurance. Furthermore, we conduct an *in-situ* laminography experiment studying crack formation in a material. Our system performs the data acquisition and reconstructs a central slice of the sample to check its alignment and data quality.

Our work enables selection of the optimal initial experimental conditions based on high-fidelity simulations, their fine-tuning during a real experiment and its automatic control based on fast data analysis. Such a data acquisition scheme enables novel *high-speed* and *in-situ* experiments which cannot be controlled by a human operator due to high data rates.

Zusammenfassung

Moderne Röntgenbildgebung gibt Aufschluss über die innere Struktur von Objekten aus den verschiedensten Materialien. Der Erfolg solcher Messungen hängt dabei entscheidend von einer geeigneten Wahl der Aufnahmebedingungen ab, von der mechanischen Instrumentierung und von den Eigenschaften der Probe oder des untersuchten Prozesses selbst. Bisher gibt es kein bekanntes Verfahren für autonome Datenakquise, welches auch für sehr verschiedene Röntgenbildgebungsexperimenten die Steuerung über bildbasiertes Feedback erlaubt. Die vorliegende Arbeit setzt sich als Ziel, diese Lücke zu schließen, indem gezielt die hierbei auftretenden Probleme angegangen und gelöst werden: die Auswahl der experimentellen Startparameter, eine schnelle Verarbeitung der aufgenommenen Daten und ein automatisches Feedback zur Korrektur der laufenden Messprozedur.

Um die am besten geeigneten experimentellen Bedingungen zu bestimmen, gehen wir von den Grundlagen der Bildentstehung aus und entwickeln ein Framework für dessen Simulation. Dieses ermöglicht uns eine große Bandbreite an virtuellen Röntgenbildgebungsexperimenten durchzuführen, wobei die entscheidenden physikalischen Prozesse auf dem Weg der Röntgenstrahlung von der Quelle bis zum Detektor berücksichtigt werden. Darüber hinaus betrachten wir verschiedene Probenformen und -bewegungen, was uns die Simulation von Experimenten wie etwa 4D (zeitaufgelöster) Tomographie ermöglicht.

Außerdem entwickeln wir eine autonome Prozedur für die Datenakquise, welches die Startbedingungen des Versuchs dann während der schon laufenden Messung auf Basis schneller Bildanalyse das nachjustiert und auch andere Parameter des Experiments steuern kann. Besonderes Augenmerk legen wir hier auf Hochgeschwindigkeitsexperimente, welche hohen Anforderungen an die Geschwindigkeit der Datenverarbeitung stellen, vor allem wenn die Steuerung auf rechenintensiven Algorithmen wie etwa für die tomographische 3D Rekonstruktion der Probe basiert. Um hierzu einen effizienten Algorithmus zu implementieren, verwenden wir ein hochgradig parallelisiertes Framework. Dessen Ausgabe kann dann zur Berechnung verschiedener Bildmetriken verwendet werden, um quantitative Information über die aufgenommenen Daten zu erhalten. Diese bilden die Grundlage zur Entscheidungsfindung in einem geschlossenen Regelkreis, in dem die Hardware für die Datenakquise betrieben wird.

Die Genauigkeit des entwickelten Simulationsframeworks zeigen wir, indem wir virtuelle und reale Experimente vergleichen, die auf Gitterinterferometrie basieren und damit spezielle optische Elemente für die Kontrastbildung einsetzen. Außerdem untersuchen wir im Detail den Einfluss der Bildgebungsbedingungen auf die Genauigkeit des implementierten Algorithmus für gefilterte Rückprojektion, und inwiefern unter dessen Berücksichtigung eine Optimierung der experimentellen Bedingungen möglich ist.

Wir demonstrieren die Fähigkeiten des von uns entwickelten Systems zur autonomen Datenakquise anhand eines in-situ Tomographieexperiments, bei dem es

basierend auf 3D-Rekonstruktion die Framerate der Kamera optimiert und damit sicherstellt, dass die aufgezeichneten Datensätze ohne Artefakte rekonstruiert werden können. Außerdem nutzen wir unser System, um ein Tomographieexperiment mit hohem Probendurchsatz durchzuführen, bei dem viele ähnliche biologische Proben gescannt werde: Für jede davon wird automatisch die tomographische Rotationsachse bestimmt und schließlich zur Sicherstellung der Qualität schon während der Messung ein komplettes 3D Volumen rekonstruiert. Darüber hinaus führen wir ein in-situ Laminographieexperiment durch, welches die Rissbildung in einer Materialprobe untersucht. Hierbei führt unser System die Datenakquise durch und rekonstruiert einen zentral gelegenen Querschnitt durch die Probe, um dessen korrekte Ausrichtung und die Qualität der Daten sicherzustellen.

Unsere Arbeit ermöglicht - basierend auf hochgenauen Simulationen - die Wahl der am besten geeigneten Startbedingungen eines Experiments, deren Feinabstimmung während eines realen Experiments und schließlich dessen automatische Steuerung basierend auf schneller Analyse der gerade aufgezeichneten Daten. Ein solches Vorgehen bei der Datenakquise ermöglicht neuartige *in-vivo* und *in-situ* Hochgeschwindigkeitsexperimente, die bedingt durch die hohen Datenraten nicht mehr von einer menschlichen Bedienperson gehandhabt werden könnten.

Acknowledgements

I would like to thank Prof. Tilo Baumbach, who provided me the opportunity to work in a lively interdisciplinary group, which I have enjoyed a lot. I am grateful for all his support, fruitful discussions and his enthusiasm, which has always been a source of my motivation. I would also like to thank Prof. Rüdiger Dillmann for supervising this work and for always finding time for me when it was needed. I thank Petr Mikulík, because he is the reason why I found my way to the field of X-ray imaging. I would like to thank many of my colleagues, with whom I had countless interesting discussions and a lot of fun, especially Daniel Hänschke, who was always there for me when I needed help with all aspects of my work and beat me in squash all the time. I thank Ruth Heine, Julian Moosmann, Angelica Cecilia, Elias Hamann, Marcus Zuber, Sabine Engelhardt, Thomas van de Kamp, Alexey Ershov, Dmitry Karpov, Martin Köhl, Ralf Hofmann, Wolfgang Mexner, David Haas, Roman Shkarin and Tomy dos Santos Rolo. I greatly appreciate all the help from Matthias Vogelgesang, productive and fun discussions with him. I would also like to thank his colleagues Andrei Shkarin, Timo Dritschler and Suren Chilingaryan. I want to thank my family for their endless support and of course Zuzka for bearing with me the whole time.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The tremendous increase of data processing speed and storage capacity has made computational technology essential for economy and society. Computer science is a driving force for new research and development concepts in manifold science areas.

Digitization of imaging revolutionized communication in our every day's life, from television to smart phone cameras and many other areas. Moreover, *fast* digital image recording and *online* image analysis enable image-based feedback which can be used by new control strategies to drive for example complex technological processes, autonomous cars [1] or humanoid robots [2].

Since the discovery of X-rays [3], X-ray imaging diagnostics have found broad applications in medicine, non-destructive testing, homeland security and many other fields. At first, X-ray images were recorded on X-ray films. With the onset of the digital era, the films could be digitalized and later it became possible to detect digital images directly by using pixelized line and array detectors. Direct digital image recording dramatically improved the possibility to use advanced imaging methods routinely, for example computed 3D tomography [4], which allows us to reconstruct the structure of a material, a device or an organism in three dimensions.

Advanced X-ray pixel array detector technology in combination with improved X-ray sources, such as microfocus X-ray tubes and especially the new generations of synchrotron storage rings [5] pushed limits of X-ray imaging possibilities for scientific applications of a very broad user community. The high photon flux density of modern storage rings together with fast cameras enables us to record hundreds of thousands of frames per second and even *time-resolved* 3D imaging experiments became feasible [6, 7, 8, 9]. This opens new opportunities for *high-throughput* imaging of large sample series and for *in-situ* imaging of technological or biological processes.

Such advanced imaging methods enabled by complex instrumentation solutions require careful preparation and precise data acquisition. Users have to select the optimal combination of experimental conditions and data processing algorithms and parameters for their particular scientific questions. By experimental conditions we mean both the imaging conditions and the conditions of the studied sample or a process, e.g. ambient temperature, liquid speed, etc. The overall parameter space is

large and it might become difficult to disentangle the dependencies between the experimental conditions and the data processing parameters, especially when it comes to measurements which involve new imaging methods or samples with completely unknown structures.

To ensure the success of an experiment and to conduct it efficiently, new strategies for automation and image-based control of the whole data acquisition and analysis pipeline are required. The pipeline includes sample loading and alignment, image acquisition and reconstruction, analysis and decision making about the course of an experiment. The decisions may be used to adjust the experimental conditions and the data processing parameters. For example, when a sample moves outside of the field of view (FOV), we need to react to these changes by re-positioning the sample or the detector. However, image-based control needs fast data processing, especially when it requires 3D or even 4D reconstruction of the sample structure.

Even though *high-speed* experiments are already possible and massively parallel architectures, like modern graphics processing units (GPUs), enable significant computational speedup [10, 11], a system which would automatically drive an X-ray imaging experiment based on *online* analysis of the acquired data is still missing at synchrotrons. Moreover, a simulation tool which would be able to conduct *in-silico* (virtual) experiments by considering many important aspects of the complete image formation process from an X-ray source to the camera readout electronics including sample dynamics is missing as well. Such simulations are important for the development and benchmarking of data processing algorithms and for the determination of the optimal initial combination of experimental conditions and data processing algorithms.

The purpose of this work is therefore to provide software infrastructure for preparation, automation and image-based control of high-speed synchrotron X-ray imaging experiments. Such autonomous data acquisition scheme is depicted in Figure 1.1. First, an *X-ray imaging experiment* is simulated. The simulation models a sample or a process, its imaging with some experimental conditions and the synthetic data are inserted into the data processing pipeline for data analysis. The analyzed sample or process is compared with the simulation input, and based on how close the result is, the virtual experimental and data processing parameters may be adjusted. The optimal parameters are then used by the real experiment. Since the simulation works only with approximations, the system sends the real image stream to the data processing pipeline and based on various image metrics fine-tunes the experimental and data processing parameters (e.g. increases beam intensity when SNR is too low or slows down the studied process if there is motion blur). The analysis might even require 3D reconstruction of the sample structure and 4D spatio-temporal reconstruction of a process, so the data processing must be very efficient and provide results with *low latency*, which is important for *high-speed* experiments. In this context, latency is the time required to process one data item, e.g. an X-ray projection.

To provide such a system for synchrotron X-ray imaging experiments, we will develop a framework for conducting a broad range of *in-silico* X-ray imaging exper-

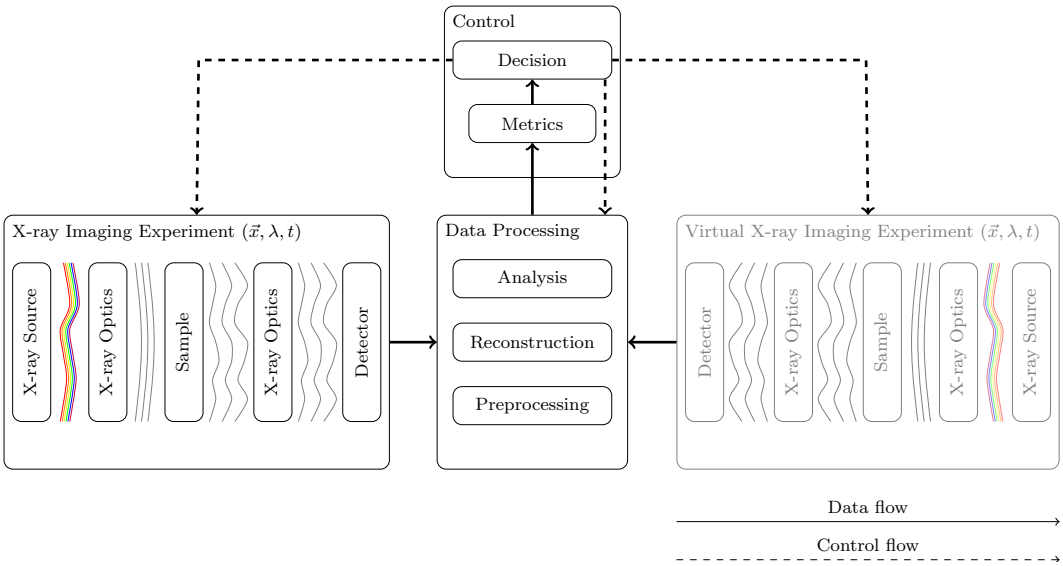


Figure 1.1: Autonomous data acquisition scheme for X-ray imaging experiments. Initial experimental conditions are selected based on simulation and refined during the real experiment based on online data analysis.

iments. We will focus on high fidelity of the simulated data and on the fact that *high-speed* experiment simulations might produce thousands of images, thus we will pay particular attention to the efficient implementation of the framework.

Our next concern will be the automation and image-based control of real high-speed experiments, especially the 4D ones because of their high computational demands. We will investigate how to quickly reconstruct the 3D sample structure, so that we can apply suitable image metrics to assess the data quality *online* (e.g. to determine if the sample has moved) and use them to automatically adjust experimental conditions. We will use GPU computing for the efficient implementation of a 3D reconstruction algorithm and we will develop a system for autonomous data acquisition thanks to image-based control. Moreover, we will integrate the reconstruction algorithm into our system to even enable control based on 3D reconstruction.

In the remainder of this chapter we will describe the problems which need to be investigated and solved in order to develop the described system. These problems will give rise to a handful of research questions and a set of goals which will define our contribution to the X-ray imaging field.

1.1 Problem Statement and Research Questions

Synchrotron X-ray imaging is an *interdisciplinary* field with complex image formation, instrumentation and data analysis tasks. Before we conduct an actual experiment we need to estimate which conditions are best suited for our case and which data processing algorithms do we need to extract the desired information about a

sample or a process.

Analysis of X-ray imaging experiments typically involves data processing pipelines composed of many specialized algorithms, e.g. advanced projection normalization [12], phase retrieval [13, 14], 3D reconstruction [15, 16], motion estimation [17] and segmentation [18]. The selection of these algorithms and their parameters strongly depends on the experimental conditions. For instance, if we place the sample further away from a detector, the Huygens–Fresnel principle applies and the sample edges are pronounced due to Fresnel diffraction and free-space propagation. This can be leveraged by motion estimation algorithms but makes thresholding-based segmentation more difficult because the signal spike at the edge might cause the threshold to be estimated incorrectly.

In general, the dependence of the data analysis accuracy on the experimental conditions can be very complex and if we want to maximize beam time efficiency by spending less beam time on finding the optimal experimental conditions for a set of data processing algorithms, we need to ask:

> Question 1: *How to find the optimal combination of experimental conditions and data processing algorithms before the experiment starts?*

During an experiment we need prompt feedback about its progress, which enables us to quickly reveal common problems, like mechanical failure, undesired sample motion, system vibrations, beam fluctuations and many more. Unfortunately, simple preview of the acquired data is often not sufficient to extract the desired information and we need to apply some processing steps, e.g. perform a 3D reconstruction of the sample structure. We will focus on the 3D reconstruction here, because it is a common task for various 3D imaging methods and it is computationally very demanding, what until now prevented it from being used to drive an experiment. Thus, our next question is:

> Question 2: *How to speed up 3D reconstruction so that it can be used to drive an experiment?*

The 3D reconstruction of the sample structure requires precise information about the alignment of the sample with respect to the detector. Even a slight error in this information can lead to very deteriorated reconstruction quality. If the alignment information is incorrect or missing, we need to be able to retrieve it from the acquired data. This raises the following question:

> Question 3: *How to determine the sample alignment information from the acquired data?*

Up to now, we picked specific issues which need to be solved if we want to enable automated synchrotron X-ray imaging experiments. Now it is time to bring all the building blocks together, from fast access to camera images, to image reconstruction, analysis and to automatic decision making and feedback to the experiment.

The prerequisite for making automatic decisions are data analysis algorithms which provide us with answers about the course of our experiment, from simple ones, e.g. the sample moved or not, to more complex ones, e.g. the current speed of a process under study. However, we do not know which analysis algorithms will be used by future experiments, so we need a general approach and a solution with reasonable compromise between the reaction latency and flexibility. From the performance point of view, the best solution would be to integrate an analysis tool directly into the control system. From the flexibility point of view, we would need an interface to a versatile analysis *toolbox* capable of applying various algorithms to different problems. Thus, we need to ask:

> Question 4: *How to enable image-based control and automation of a broad range of synchrotron X-ray imaging experiments while keeping the response time of the system low?*

1.2 Objectives and Contribution

This work does not try to solve issues related to one particular experiment but provides a solution applicable to a broad range of experiments. Our first objective is therefore to enable preparation of experiments based on high-fidelity simulations. Our second objective is to develop a data acquisition scheme which enables experiment automation and control based on *online* image analysis.

The answers to the raised questions and specified objectives require deep understanding and investigation of many aspects of the synchrotron X-ray imaging field from physical mechanisms behind image formation, to 3D reconstruction of the sample structure, its parallelization possibilities and extensive knowledge about instrumentation. Based on these prerequisites, we will be able to push the X-ray imaging field one step forward by our contributions:

1. We will develop an X-ray imaging simulation framework covering the main components of the complete image formation process from an X-ray source to the camera read out electronics. We will take into account many important physical mechanisms, so that the framework can produce virtual data with high fidelity. This is very important for preparing real experiments and benchmarking data analysis pipelines with their numerous parameters. We will include various approaches for the creation of sample shape and in particular sample motion which broadens the applicability of the framework to high-speed 4D experiments. With the computational cost of such experiments in mind, we will choose reasonable compromises between the physical accuracy and computational speed of the simulations and provide an efficient implementation of all compute-intensive parts by using OpenCL [19].

2. We will select a technique which enables fast 3D reconstruction of tomographic and laminographic data sets. It will be particularly well suited for low latency reconstruction required for automating high-speed 4D experiments. We will investigate various possibilities for decreasing the amount of copied projection data and provide its efficient parallel implementation in OpenCL.

3. We will investigate various image metrics sensitive to artifacts in the 3D reconstruction related to wrong information about sample alignment.

4. We will implement the 3D reconstruction algorithm within a highly parallel data processing framework [20], which is also capable of performing various image analysis tasks. The framework is integrated into a beam line control system for low latency data acquisition [21]. We will further integrate the reconstruction metrics into the control system, extend the control system by a decision making scheme and thus enable fully automatic image-based control of synchrotron experiments.

1.3 Outline

In this thesis, we will first introduce X-ray image formation principles and common imaging methods with focus on synchrotrons in Chapter 2. We will also review the existing tools for the simulation of X-ray imaging and for experiment control and briefly describe parallel computing as well as OpenCL.

In Chapter 3, we will design and develop *syris*, a framework for conducting virtual X-ray imaging experiments. We will demonstrate the framework's capabilities on three examples:

- a high-speed radiography which shows how high-fidelity simulations can be used to find suitable image processing parameters,
- a tomography experiment showing how we can reduce the amount of acquired projections and at the same time keep high reconstruction accuracy and
- a grating interferometry experiment which sheds light on the contrast formation process by special optics.

Chapter 4 investigates algorithms suitable for low latency 3D reconstruction. Once we find a suitable algorithm, we provide its efficient parallel implementation by using OpenCL. We then further optimize it by reducing the amount of processed projection data and leveraging multiple compute devices.

In Chapter 5 We will use a beam line control system with fast access to camera images and to a *high-performance* computing framework to implement a data acquisition scheme which will enable image-based control and automation of *high-speed* X-ray imaging experiments. Moreover, we will integrate the 3D reconstruction algorithm within the *high-performance* computing framework, so that it can be used to provide 3D image-based feedback to experiments. We will further investigate various image metrics for the determination of the sample alignment from the acquired data and integrate them into our system.

In Chapter 6 we demonstrate our system by performing four experiments:

- a tomographic experiment studying a liquid foam which changes its structure during its formation process. We will optimize the data acquisition speed based on *online* tomographic reconstruction to obtain artifact-free reconstruction and at the same time high signal to noise ratio (SNR),
- a *high-throughput* tomographic experiment with fast automatic 3D sample structure reconstruction for quality inspection,
- an *in-situ* laminographic experiment with online laminographic reconstruction to check sample alignment and data quality and position the region of interest and
- a grating interferometry experiment to validate our simulation framework *syris* from Chapter 3 by comparing real and simulated image data.

In Chapter 7, we summarize the achieved results and our contributions to the X-ray imaging field. We also discuss the possible extensions and future directions of the developed system.

Chapter 2

Preliminaries and Related Work

X-ray imaging is a complex interdisciplinary field which includes many physical mechanisms and requires a lot of instrumentation. In this chapter we will introduce X-ray imaging principles necessary to develop the simulation framework in Chapter 3 and the 3D material structure reconstruction algorithm in Chapter 4. We will also describe X-ray imaging methods which we will automate in Chapter 5.

Further, we will review the existing X-ray imaging simulation tools and automation possibilities. We will discuss their limitations which we will need to overcome to enable autonomous X-ray imaging experiments. We will also shortly review the history of parallel computing, describe the massively parallel architecture of GPUs and the Open Computing Language (OpenCL) standard for programming parallel architectures, which we will exploit later for the implementation of various algorithms.

2.1 X-ray Imaging

X-rays are electromagnetic waves with very short wavelengths ranging from 0.01 nm to 10 nm and their importance stems from the fact that they can penetrate materials opaque to the visible light, like the human body.

X-ray imaging methods use an X-ray beam generated by a source which is further adjusted for a particular sample in terms of its shape and spectral properties. It is modified by a sample, propagates to the detector where it is detected and recorded, see Figure 2.1.

An X-ray beam is a complex wavefield which has an amplitude and a phase and it can be described by the complex function

$$u(\vec{x}, z) = \sqrt{I(\vec{x}, z)} e^{j\varphi(\vec{x}, z)}, \tag{2.1}$$

where $\vec{x}$ is the 2D spatial coordinate perpendicular to the beam propagation direction z, $\vec{x} = (x, y)$, x is the horizontal coordinate and y the vertical one. $\sqrt{I(\vec{x}, z)}$ is the amplitude of the wavefield and $\varphi(\vec{x}, z)$ is its phase at $(\vec{x}, z)$. The intensity is related

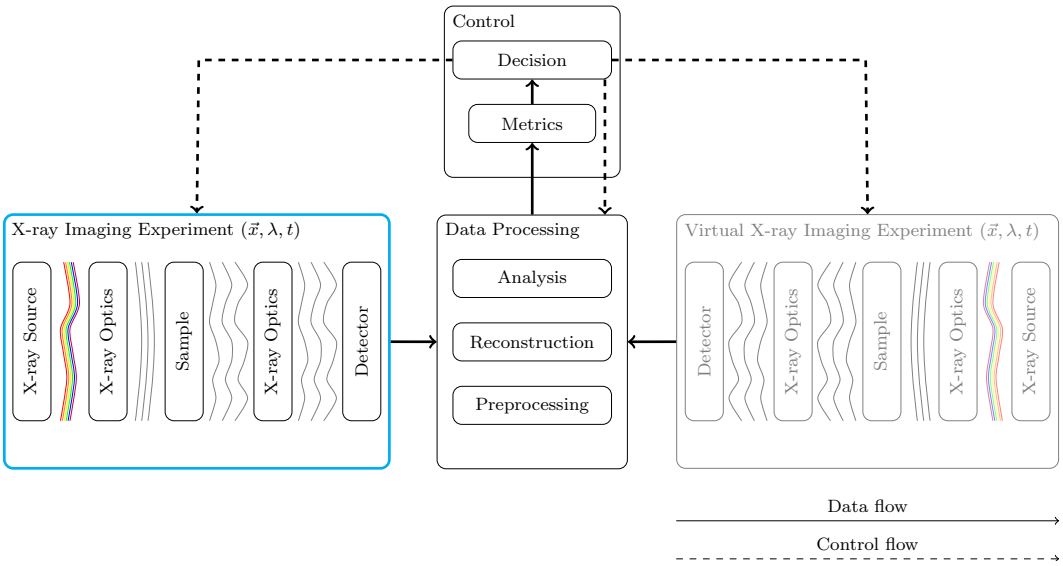


Figure 2.1: X-ray image formation light path highlighted in cyan.

to the wavefield by

$$I(\vec{x}, z) = |u(\vec{x}, z)|^2. \tag{2.2}$$

2.1.1 X-ray Sources

An X-ray source generates a wavefield which travels along the optical axis z to the first object in the light path. We will describe the basic principles of two types of sources commonly used for X-ray imaging, *X-ray tubes* and various *synchrotron sources*.

X-ray Tubes

Laboratory setups use X-ray tubes as the source of the X-rays. An X-ray tube consists of a cathode, which emits electrons, a high-power voltage generator, which accelerates them towards an anode. An important X-ray tube characteristic is the energy spectrum of the generated X-rays. It is given by the *bremsstrahlung* effect and the X-ray emission properties of the anode's target material. Bremsstrahlung occurs when the electrons from the cathode are decelerated by the atomic nuclei of the anode's target material and it generates a smooth spectrum of X-ray photons.

When the electron beam excites the anode material it might eject electrons from inner shells around the nuclei. This creates electron holes which can be filled with electrons from outer shells and the energy difference between the lower-energy shells and the higher-energy shells may be released in the form of X-rays. This effect creates material-specific peaks in the X-ray energy spectrum of the tube and may be leveraged to create X-ray setups specialized for imaging at specific X-ray energies.

Focal spot size of the electron beam on the anode is another important property of an X-ray tube for imaging and it influences the achievable resolution and coherence properties.

Synchrotron Sources

Synchrotron radiation is emitted by radially accelerating charged particles moving along a curved trajectory at relativistic speeds [5]. At synchrotron facilities, charged particles, e.g. electrons, travel along a circular trajectory in a so-called storage ring with a diameter up to several hundred meters. Curvature of the storage ring is realized by *bending magnets*, which use strong magnetic field to deflect the electron beam from a straight trajectory. As the electrons are deflected, they are accelerated and produce wide electromagnetic radiation spectrum, including X-rays (see Figure 2.2). Thus, bending magnets themselves are one possible synchrotron source of X-ray radiation.

However, storage rings are not only made of bending magnets. They include also straight sections which give us the possibility to place there another types of X-ray sources. These are magnetic periodic structures which force the electron beam to oscillate and emit radiation with a broad energy spectrum, including X-rays. These sources can be *wigglers* and *undulators*.

Roughly speaking, the period and magnetic field strength in a wiggler are set in such a way that the radiation from the oscillating electrons does not interfere and wigglers can be thought of as series of bending magnets. Their radiation spectrum is smooth and broad and the intensity scales with the number of periodic structures in the wiggler.

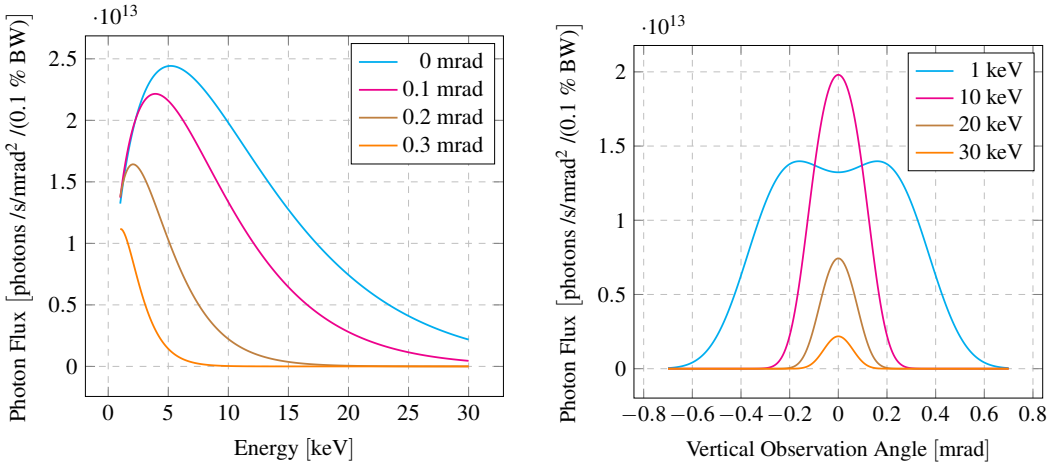
On the other hand, undulators are constructed in such a way that the radiation generated by the oscillating electrons can constructively interfere, which makes the energy spectrum to be structured and the intensity within the source harmonics scales with the square of the number of periods.

Synchrotron radiation has several properties suitable for X-ray imaging. Thanks to the narrow radiation cone it has high *brilliance*, i.e. the photon *flux density* is high even at large distances from the source. This enables acquisition rates of hundreds of thousands of frames per seconds and *high-speed* and *high-resolution* experiments. Moreover, the large propagation distance increases the spatial coherence of the wavefield in the object plane, which enables us to exploit interference for imaging. If the source is a point source and the distance from it to an object is much greater than the lateral object size, we may treat a monochromatic wavefield in the object plane as a *plane wave* traveling along the optical axis z towards the object and write it as

$$u(\vec{x}, z) = \sqrt{I} e^{jkz}, \tag{2.3}$$

where k is the *wavenumber* $k = 2\pi/\lambda$ and λ is the X-ray wavelength. For simplicity, we put polychromaticity and coherence effects off until Section 2.1.4.

Another interesting property of synchrotrons is that electrons in the storage ring travel in bunches, thus the radiation has a *pulsed structure* and one pulse can be



(a) Fixed observation angles

(b) Fixed energies

Figure 2.2: Bending magnet energy spectra for various X-ray photon energies and vertical observation angles. The storage ring energy is 2.5 GeV, magnetic field is 1.5 T and electric current is 200 mA.

tuned to have duration of several ps, which can be used for stroboscopic imaging of very fast processes.

2.1.2 X-ray and Matter Interaction

An X-ray beam can be *absorbed* and *scattered* in matter. These interactions can be described by the *complex refractive index* (see Figure 2.3), which is a 3D function of the object structure

$$n(\vec{x}, z) = 1 - \delta(\vec{x}, z) + j\beta(\vec{x}, z). \tag{2.4}$$

The real part $\delta(\vec{x}, z)$ describes the refraction and scattering and $\beta(\vec{x}, z)$ describes absorption. In the X-ray regime, both of these quantities are related to the electron density of the material. Moreover, the *linear attenuation coefficient* $\mu(\vec{x}, z)$ is related to $\beta(\vec{x}, z)$ by

$$\mu(\vec{x}, z) = \frac{4\pi}{\lambda}\beta(\vec{x}, z), \tag{2.5}$$

If we assume an object to be sufficiently thin we can use the *projection approximation* and describe the wavefield propagation through matter by the *transmission function* $T(\vec{x})$. It considers the *attenuation* and the *phase shift* of the wavefield which propagates through an object by integrating the object's complex refractive index along the mean beam propagation direction. If the incident wavefield is a plane wave traveling along z as in (2.3), the transmission function can be written as

$$T(\vec{x}) = e^{-B(\vec{x})+j\phi(\vec{x})}, \tag{2.6}$$

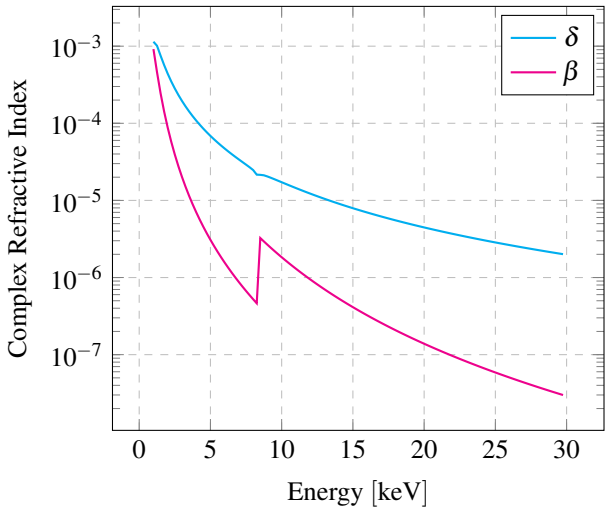


Figure 2.3: Complex refractive index of nickel with its absorption edge around 8 keV.

where

$$\begin{aligned} B(\vec{x}) &= k \int \beta\,(\vec{x}, z)\,\mathrm{d}z \\ \phi(\vec{x}) &= k \int [1 - \delta(\vec{x}, z)]\,\mathrm{d}z. \end{aligned} \tag{2.7}$$

The wavefield created by an X-ray source can in general interact with many *objects* along z and propagates between them. Objects include X-ray optics, the studied sample and the detector. The general *light path* is depicted in Figure 2.4 and we will use the following notation to describe it. The X-ray source is at position z_0 and the i^{th} object in the light path is in position z_i, $i > 0$. The detector is the last object and its position is z_{N+1}. The distance between object i and $i+1$ is Δz_i and we say that object $i+1$ is *downstream* from object i and conversely, object i is *upstream* from object $i+1$.

An object has an *entrance plane* right in front of it and an *exit plane* right behind it. Within the projection approximation, both of these planes are at the same distance z_i from the source. The wavefield in the i^{th} object's entrance plane is $u_{i-1}(\vec{x}, z_i)$, the wavefield in its exit plane is $u_i(\vec{x}, z_i)$ and the wavefield propagation through it is described by the transmission function

$$u_i(\vec{x}, z_i) = u_{i-1}(\vec{x}, z_i) \cdot T_i(\vec{x}). \tag{2.8}$$

2.1.3 Free-space Propagation of a Wavefield

The wavefield can free-space propagate to some Δz from the exit plane of an object. Thus, the wavefield propagated from the exit plane of object i to the entrance plane of the next object $i+1$ is $u_i(\vec{x}, z_i + \Delta z_i)$.

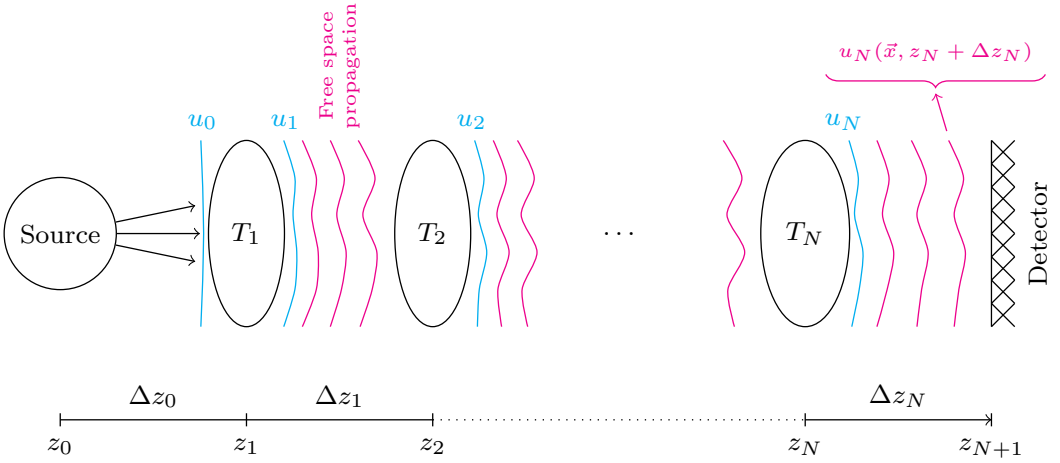


Figure 2.4: X-ray image formation light path. Wavefield generated by a *source* interacts with many objects in terms of their transmission functions T_i, propagates between them and reaches the *detector*, which records its intensity. Figure from [22].

The propagation in air can be usually well approximated by free-space propagation in vacuum and described by the Huygens–Fresnel principle, where every point of a wavefield in the exit plane z_i is a source of a spherical wave. The wavefield in the plane $z_i + \Delta z$ is the superposition of all the spherical waves from the plane z_i. We define $\vec{\eta}$ to be 2D spatial coordinates, like $\vec{x}$, with the only difference that $\vec{\eta}$ is located at a distance z_i, whereas $\vec{x}$ is located downstream at $z_i + \Delta z$. This will hold also later in the text when we discuss two planes at different distances. Assuming that $\Delta z \gg \lambda$, the wavefield propagation from the plane $z = 0$ to the plane Δz can be written as [23]

$$u(\vec{x}, \Delta z) = \frac{1}{j\lambda} \int u(\vec{\eta}, 0) \frac{e^{jkr}}{r} \mathrm{d}\vec{\eta}, \tag{2.9}$$

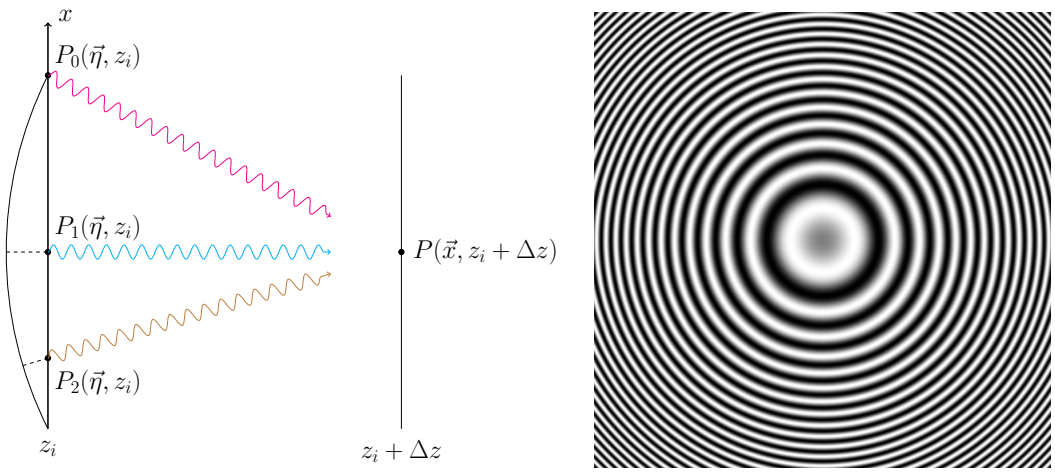
where $r = \sqrt{(\vec{\eta} - \vec{x})^2 + (\Delta z)^2}$. Propagation can be interpreted as a 2D convolution with the *propagator* kernel

$$P(\vec{x}, \Delta z) = \frac{e^{jkr}}{j\lambda r}. \tag{2.10}$$

Such propagator is depicted in Figure 2.5. The propagated wavefield can be written recursively for any object in the light path as

$$\begin{aligned} u_0(\vec{x}, z_1) &= \sqrt{I_0} e^{jk\Delta z_0}, \\ u_i(\vec{x}, z_i + \Delta z_i) &= [u_{i-1}(\vec{x}, z_i) \cdot T_i(\vec{x})] * P(\vec{x}, \Delta z_i), \end{aligned} \tag{2.11}$$

where I_0 is the intensity of the plane wave emitted by a source.



(a) (b)

Figure 2.5: A 2D light propagation scheme in (a). Although waves from all three points $P_0(\vec{\eta}, z_i)$, $P_1(\vec{\eta}, z_i)$ and $P_2(\vec{\eta}, z_i)$ contribute to point $P(\vec{x}, z_i + \Delta z)$, they arrive with different phase, which means that some of them interfere constructively and others destructively. This strongly affects the final wavefield $u_i(\vec{x}, z_i + \Delta z)$. The spherical cap on the left indicates the positions for which the points would arrive with equal phases. In (b) is the real part of a 2D real space propagator (2.10) with wavelength $\lambda = 1\,\text{nm}$, distance $\Delta z = 10\,\text{cm}$ and field of view $100\,\mu\text{m} \times 100\,\mu\text{m}$. It is a cut perpendicular to the optical axis through the 3D space at z_i.

2.1.4 Partial Coherence

Until now we assumed a source emitting plane monochromatic waves. However, common X-ray sources are laterally extended and emit a radiation field with a broad energy spectrum. If we assume that the emission of photons with different energies occurs incoherently, the resulting partial degree of coherence results from limiting the spectrum bandwidth $\Delta\lambda$ and propagating the wavefield to large distances, as we mentioned in Section 2.1.1.

Polychromaticity

An X-ray beam generated by a source can have a broad energy spectrum. On the one hand, this spectrum provides us with sufficient flux density so that we can conduct high-speed experiments, but on the other it reduces resolution. Since the propagated intensity pattern depends on the wavelength, it is always a little different and the broader the spectrum is, the more the patterns vary. If we take into account the different energies by incoherently summing their contributions, the final image will be blurred because of the slightly different propagated patterns. If we express the intensity dependence on the wavelength as $I_i\left(\vec{x}, z_i + \Delta z_i, \lambda\right)$, we can write the

superimposed intensity as

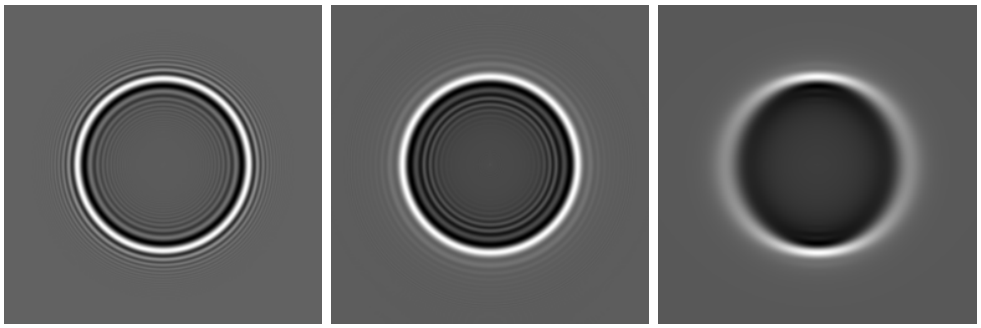
$$I_i(\vec{x}, z_i + \Delta z_i) = \int I_i\left(\vec{x}, z_i + \Delta z_i, \lambda\right) \mathrm{d}\lambda. \tag{2.12}$$

Spatial Coherence

If the detector is the last (N + 1)$^{\text{st}}$ element in the light path, $z_1 \gg \Delta z_N$ and $\Delta z_N \gg \sum_{i=1}^{N-1} \Delta z_i$, i.e. the distance from the source to the first object is sufficiently larger than the distance from the last object to the detector, moreover if this is much larger than the sum of the distances between the other objects, we can employ the van Cittert–Zerinke theorem [24]. This means that we can account for the extended source by blurring the image with the rescaled normalized intensity distribution of the source $S(\vec{x}z_1/\Delta z_N)$. Based on the assumed source, objects and detector distances above, the rescaling is given by the geometrical projection of the source distribution on the imaging plane through the first object. The observed intensity profile is then

$$I_N(\vec{x}, z_{N+1}) = I_N(\vec{x}, z_{N+1}) * S\left(\frac{z_1}{\Delta z_N}\vec{x}\right). \tag{2.13}$$

As we can see in (2.13), the projected source distribution becomes smaller with increasing z_1 which accounts for larger coherence. Moreover, the coherence of synchrotron sources is typically much larger vertically than horizontally, as depicted with some other image formation details in Figure 2.6.



(a) (b) (c)

Figure 2.6: Simulated intensity patterns from propagation of a wavefield modified by a sphere with various X-ray image formation effects taken into account. (a) perfectly coherent radiation, (b) polychromatic radiation (10 - 30 keV), (c) partial spatial coherence due to an extended source (vertically ten times larger coherence than horizontally) applied on (b). Figure from [22].

2.1.5 X-ray Detectors

X-ray detectors record the intensity of a wavefield given by the square of the absolute value of its amplitude. In the digital era, X-ray films have been replaced with 2D

pixel array detectors which give us digital representation of the incident intensity. Detectors can be *direct* or *indirect*. Direct detectors convert the X-ray beam into electric charge and they use a semiconductor material electrically coupled to the read out electronics and they can achieve pixel sizes in the order of 10 µm.

Flat-panel systems are an example of indirect detectors. They use fluorescent screens coupled to TFT- or CMOS-based read out electronics and have pixel size in the order of 100 µm.

Another versatile indirect detector setup uses a *scintillating* material which converts the X-ray beam into visible light magnified by an optical system and then reaches a CCD- or CMOS-based camera sensor able to detect it. The versatility of such systems stems from the fact that their parts are loosely coupled, i.e. we can exchange the scintillator, optical system and cameras to match specific experimental requirements. Pixel size of such a system is in the order of 1 µm.

Every detector system has several characteristics which impact the quality of the recorded digital image. *Spatial resolution* is limited by the fact that the detected charge may spread across several surrounding pixels which results in the blurring of the recorded image. It can be described by the Point Spread Function (PSF), which describes the spreading of one point in the detected image.

Another important property of a detector is *dynamic range* which describes the resolution of the detected signal on the brightness level. It tells us how much *contrast* in terms of brightness level change is needed between two sample structures for the detector to be able to resolve the two structures.

Linearity [25] is a measure of how much the detected counts deviate from the perfect linear mapping between the number of detected photons and the detector counts.

SNR is an important measure of the amount of noise in the detected signal. The detection of photons by a camera sensor is subject to various noise sources [25]. Number of electrons emitted in a pixel varies over time with signal dependent Poisson distribution and variance $\sigma_e^2(\vec{x})$. This noise is often referred to as *shot noise*. Furthermore, we take into account the signal independent normally distributed *electronics noise* with variance σ_d^2 and the uniformly distributed *quantization noise* [26] with variance σ_q^2, which is caused by analog to digital conversion.

The camera signal recorded during an exposure time Δt, $C(\vec{x}, \Delta t)$ may be modeled by applying the shot and electronics noise to the detected signal, amplifying the noisy signal by the *overall system gain* K and taking into account the quantization noise. The total variance of the recorded signal can thus be written as [25]

$$\sigma^2(\vec{x}) = K^2 \left(\sigma_d^2 + \sigma_e^2(\vec{x})\right) + \sigma_q^2. \tag{2.14}$$

2.1.6 X-ray Imaging Methods

The aim of X-ray imaging methods is to visualize the sample structure from the acquired projections in 2D (*radiography*), 3D (tomography [15] or laminography [16]), or even 4D (time-resolved tomography [9]).

Moreover, these methods can be combined with various contrast formation mechanisms. For example, *absorption contrast* sensitive to the local variation of $\beta(\vec{x}, z)$ may be too weak for imaging biological tissue and we need to look for alternatives. E.g., we may want to exploit variations in $\delta(\vec{x}, z)$ for contrast creation. However, since it cannot be measured directly we need to find ways how to create *phase contrast* indirectly. For example, this can be realized by letting the wavefield to propagate in free-space [13, 14, 27] or by using special phase-sensitive optical elements [28, 29]. Furthermore, one can use other contrasts, like *fluorescence* [30, 31] and *diffraction* [32, 33].

Radiography

Radiography is an imaging method which gives us integrated information about the sample structure superimposed along the beam propagation direction. The aim is to retrieve $B(\vec{x})$ or $\phi(\vec{x})$ from (2.7). Since $\phi(\vec{x})$ is lost during the detection process, we need to measure it indirectly by employing some phase-sensitive technique.

If the wavefield in the object entrance plane is monochromatic, we use the projection approximation and we are interested in the intensity in the object's exit plane, we may neglect the phase changes caused by the object and use the Beer–Lambert law [24] to retrieve $B(\vec{x})$, directly present in the recorded intensity pattern. For simplicity, we will not use object indexing in the following, because we are interested only in the intensity changes caused by one object, which is

$$I(\vec{x}) = I_0(\vec{x}) e^{-\int \mu(\vec{x}, z)\, \mathrm{d}z}, \tag{2.15}$$

where $I_0(\vec{x})$ denotes the intensity in the entrance plane and $I(\vec{x})$ the intensity in the exit plane.

If we record the intensity with and without the sample, we can compute the *absorbance* of the sample which removes the beam profile (see Figure 2.7b) as

$$\int \mu\left(\vec{x}, z\right)\, \mathrm{d}z = \ln\left(\frac{I_0(\vec{x})}{I(\vec{x})}\right). \tag{2.16}$$

Moreover, let's take into account the dark current recorded by the detector which is the signal without the incident beam $I_d(\vec{x})$ (see Figure 2.7a). The recorded intensity is then $I(\vec{x}) = I_s(\vec{x}) + I_d(\vec{x})$, where $I_s(\vec{x})$ is the signal without the dark current. If we acquire $I_d(\vec{x})$, we can compute absorbance (see Figure 2.7d) as

$$\int \mu\left(\vec{x}, z\right)\, \mathrm{d}z = \ln\left(\frac{I_0(\vec{x}) - I_d(\vec{x})}{I(\vec{x}) - I_d(\vec{x})}\right) \tag{2.17}$$

Tomography

Tomography is a 3D imaging technique [34] which enables us to reconstruct not only the projected $B(\vec{x})$ or $\phi(\vec{x})$, but the 3D complex refractive index components

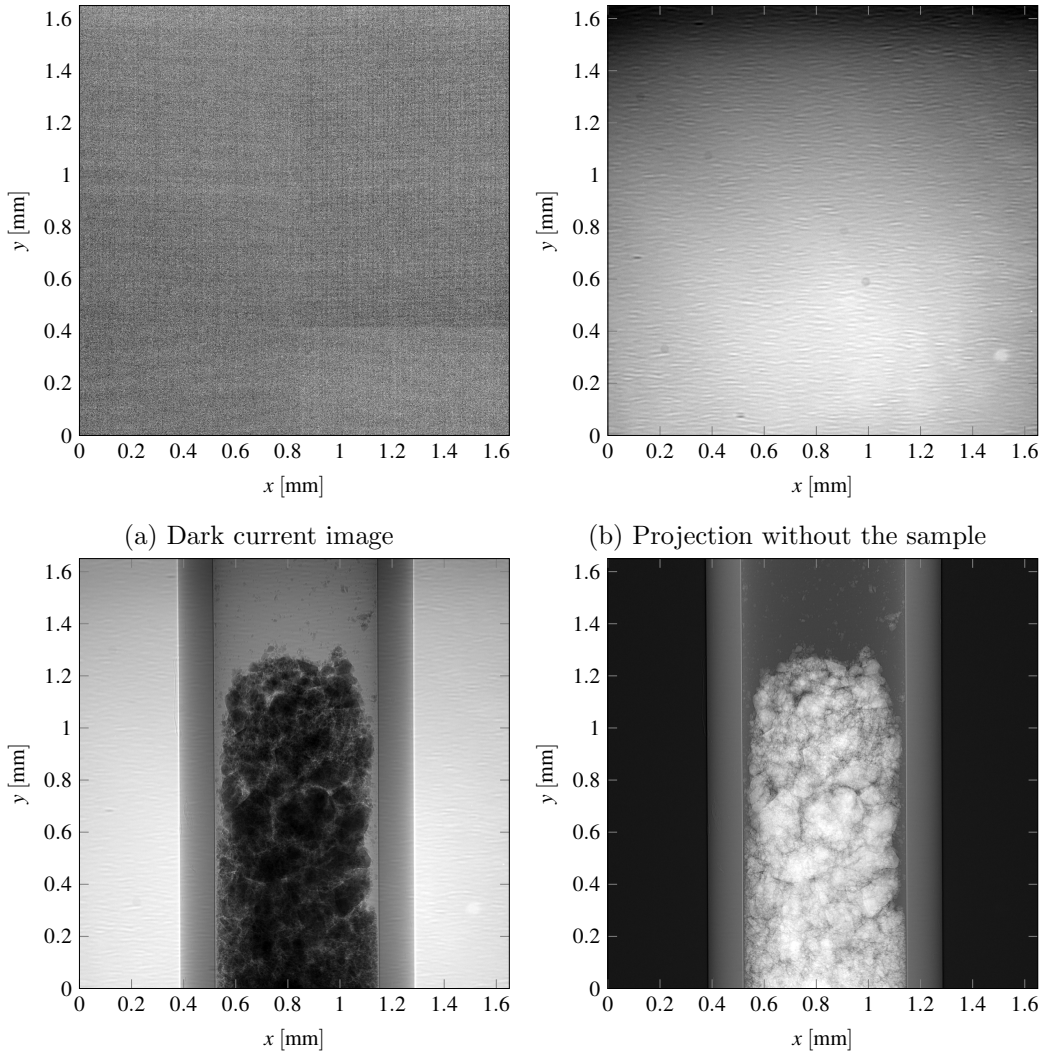
$\beta(\vec{x}, z)$ and $\delta(\vec{x}, z)$. It requires sample rotation around an axis perpendicular to z. A projection is acquired for every rotation angle ϕ between 0° and 180°, as depicted in Figure 2.8.

The 3D sample structure can be reconstructed from such a set of projections by exploiting the Fourier slice theorem [26] or algebraically [35, 36, 37, 38]. The time complexity of reconstruction algorithms is high, which makes it difficult to obtain a 3D volume shortly after the data acquisition. Various parallelization possibilities have been proposed to speed up the calculations, from computational clusters[39, 40] to GPU computing [41, 42, 10, 11, 43]. Various reconstruction approaches will be discussed in more detail in Chapter 4.

Laminography

Tomography applied on laterally extended samples may lead to reduced reconstruction quality because the projected thickness at some angles ϕ can be so large that all X-rays are absorbed in the sample, which means that the beam might not penetrate the sample, resulting in no signal on the detector. Such missing areas in projections require special treatment [44]. We can avoid this problem by tilting the rotation axis by some angle θ which reduces the projected thickness, as depicted in Figure 2.9. Projections of such a tilted sample are recorded around 360°. This is called *laminography* [45] and we can use generalized tomographic reconstruction algorithms to obtain the 3D volume.

The tilted rotation axis causes the sample to take up more detector rows during rotation than by tomography, which means that we need to process larger portions of projections to reconstruct the 3D volume. Even though efficient implementations of the laminographic reconstruction were proposed in [46, 47], their performance is still inferior to the tomographic case.



(a) Dark current image

(b) Projection without the sample

(c) Projection with the sample

(d) Absorbance

Figure 2.7: Absorbance calculation from a dark current image $I_d(\vec{x})$ in (a), a projection without the sample $I_0(\vec{x})$ in (b) and a projection with the sample $I(\vec{x})$ in (c). Absorbance computed by (2.17) is depicted in (d).

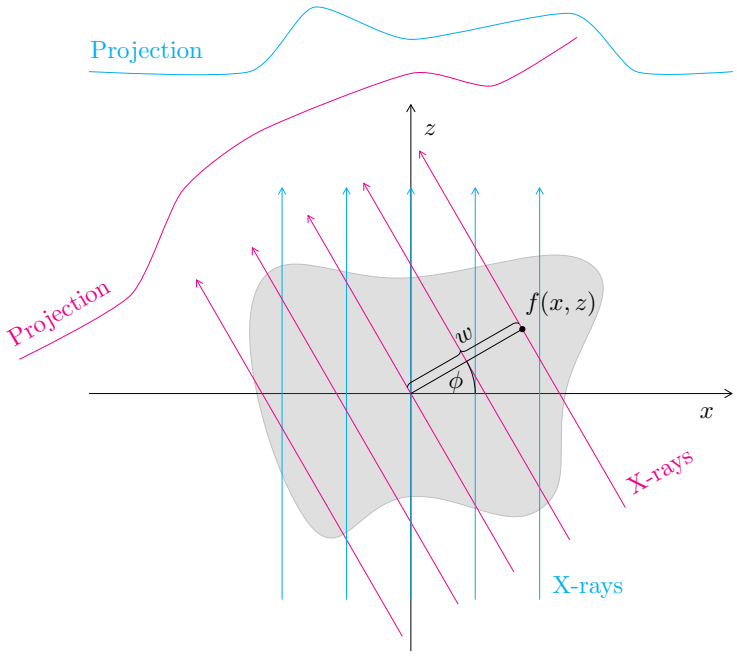


Figure 2.8: Data acquisition by parallel beam tomography, view of an object slice cut through the xz plane which is parallel with the X-ray beam. The sample is rotated around the y-axis and X-rays at different angles ϕ project the slice to a 1D *projection* on the detector row. 2D projections from all detector rows over the angle range $[0, \pi)$ form a tomographic data set.

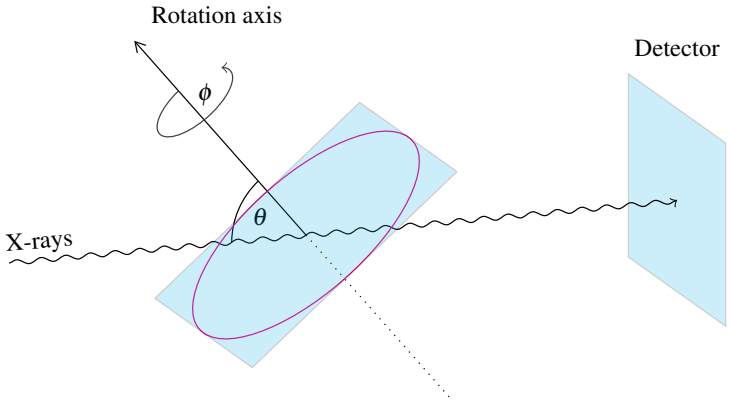
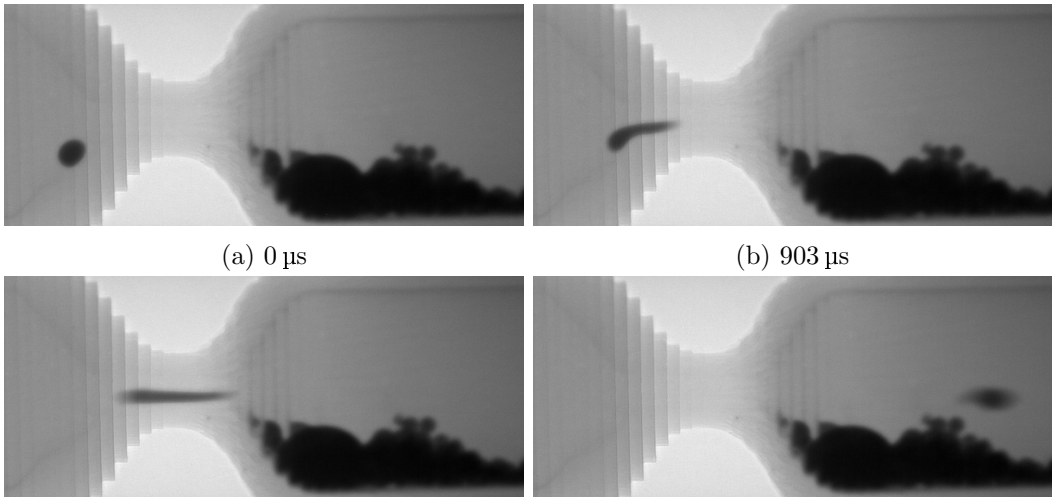


Figure 2.9: Laminography setup. The rotation axis is tilted by θ which reduces the projected thickness of laterally extended samples. The sample is rotated around the tomographic angle ϕ which gives us a set of projections like by tomography. Tomography may be seen as a special case of laminography with $\theta = 90°$.

2.2 Available Software for X-ray Imaging Simulation

As we have seen in Section 2.1, X-ray imaging experiments require a lot of instruments from X-ray sources to detector systems. Moreover, laboratories and synchrotron beam lines are often optimized for specific application, e.g. *high-speed* imaging of processes or *high-resolution* imaging of materials. Such end-stations are very expensive and need to be thoroughly planned before construction. Thus, simulation of the X-ray image formation including various specialized X-ray optics is essential to ensure that the end-station will perfectly serve its purpose.

For this reason, many specialized simulation programs which use ray tracing methods to simulate X-ray sources and optics have been developed over the past decades [48, 49, 50, 51, 52, 53]. There are also packages which focus on the computation of X-ray projections of various phantoms [54, 55] which can be used to study the behavior of various reconstruction and analysis algorithms. Tools for simulating the mechanical [56] and fluid [57] dynamics of various imaged processes are also existing. Light detection process has been studied and simulated for example in [58].



(a) 0 µs (b) 903 µs

(c) 1354 µs (d) 2708 µs

Figure 2.10: Projections of a capillary with changing diameter and an iodine droplet traveling through it at various times. The droplet moves slowly in the region where the capillary is wide in (a) and as it comes to the thinner part of the capillary it accelerates and elongates in (b). The droplet shape is lost completely in (c) and as it arrives to the wider capillary part it slows down and forms a droplet again in (d).

Even though there exist simulators for all parts of the image formation process, an integral framework which would enable simulations of a broad range of X-ray imaging experiments including X-ray sources, interaction with matter, light propagation, detection process and *sample dynamics* is still missing. Such a framework would enable us to prepare our experiments prior to the actual measurement, help us choose suitable data processing algorithms based on the expected imaging condi-

tions and train the algorithms to provide the highest possible analysis accuracy. For example, when we want to perform a *high-speed in-situ* experiment and analyze the motion of our sample, we can apply some optical flow algorithm. However, these algorithms are sensitive to noise, motion blur and sample shape changes, like the droplet in Figure 2.10. Thus, it would be very beneficial to know how the algorithms perform based on various data quality. Such *categorization* would help us choose the suitable algorithm for our particular measurement. Another example is a 3D X-ray imaging experiment of a thick sample which causes the lack of signal in the recorded projections at some rotation angles, like in the limited angle tomography [44] or laminography [45]. Such limitations require novel reconstruction approaches which can be tested and evaluated on simulated data with respect to the *ground truth*, i.e. the known 3D sample structure.

2.3 Available Software for Automation of X-ray Imaging Experiments

The complexity of laboratory and beam line setups requires flexible and versatile control possibilities. Facilities around the world use different control systems either developed *in-house*, or they adopt existing products to fit their particular needs.

SPEC is a commercial control system originally developed for X-ray diffraction but it was later extended for X-ray imaging experiments. It provides modules for the control of beam line devices, a scripting language which can be used to program data acquisition pipelines, and a Command Line Interface (CLI). TANGO [59] is a distributed device control system developed at the European Synchrotron Radiation Facility (ESRF) and SOLEIL[1] synchrotron. It is based on Common Object Request Broker Architecture (CORBA) and it is independent of a concrete programming language. TANGO is the cornerstone of *MxCuBE* [60], control system develped at ESRF for the control of the macromolecular crystallography experiments and it includes routines for automatic data collection and analysis required by such experiments. Another control system based on TANGO is Sardana[2], originally developed at ALBA[3]. It provides data acquisition routines and dynamic generation of CLIs and Graphical User Interfaces (GUIs). Experimental Physics and Industrial Control System [61] (EPICS) is a distributed process control system and it is extensively used at the Advanced Photon Source[4] (APS) to control the accelerator itself and many beam lines.

All the control systems mentioned above are able to control devices and most of them provide data acquisition routines, e.g. performing a tomographic scan. At the TOmographic Microscopy and Coherent rAdiology experimenTs[5] (TOMCAT)

[1] `www.synchrotron-soleil.fr/`
[2] `www.sardana-controls.org`
[3] `www.albasynchrotron.es`
[4] `www1.aps.anl.gov`
[5] `www.psi.ch/sls/tomcat`

beam line of the Swiss Light Source (SLS), EPICS was extended by a set of Python scripts to conduct *high-throughput* tomographic experiments [62]. Sample exchange, alignment and data acquisition can be performed automatically with throughput 4.4 samples per hour for scanning times 7 minutes per sample.

All the mentioned systems can be used to control experiments and automate data acquisition, but the automation is either not based on image analysis, which makes it difficult to deal with sample positioning and quality assurance of the acquired data, or the image-based feedback is too slow for controlling *high-speed* experiments. Moreover, the automation solutions are tailored for specific experiment types and not flexible enough to provide a general autonomous data acquisition scheme for a broad range of experiments.

2.4 Parallel Computing

The tendency of increasing clock rates of CPUs had to stop in the last decade due to technological limitations. This forced the industry to come up with alternative solutions which would prevent processor performance stagnation. The answer were multi-core processor designs. If we want to enable autonomous *high-speed* experiments, we need to make sure our algorithms are implemented very efficiently, i.e. they enable high data throughput with low latency. This can be achieved by leveraging their parallelization possibilities discussed in the respective chapters. For this we need to introduce parallel computing and define basic terminology.

Parallel computing is a broad term which spans from multi-core processors to tens of thousands of processors either in a close proximity (computer cluster) or distributed over a large area (grid computing). The amount of parallelism can be described by Flynn's taxonomy [63]:

- Single Instruction Single Data (SISD): one processing unit executes exactly one instruction on one data item at a time.

- Single Instruction Multiple Data (SIMD): one processing unit performs single instruction on multiple data items in parallel, this is e.g. the SSE instruction set designed by Intel.

- Multiple Instructions Single Data (MISD): multiple instructions operate on one data item, this is an uncommon architecture.

- Multiple Instructions Multiple Data (MIMD): multiple processing units are completely independent and execute different instructions on different data. Processing units can either share their memory space which requires explicit synchronization or they can work with their own dedicated memory spaces and exchange messages to communicate results.

2.4.1 Multi-core Architectures

Modern CPUs use parallelism on different levels, from *instrucion-level* parallelism to *task-level* parallelism.

Instruction pipelining is a technique which splits the basic instruction cycle into a series of steps, called a pipeline. Instead of processing every instruction sequentially, the processor overlaps the execution of different steps of multiple instructions.

Superscalar architecture dispatches instructions to multiple execution units within one processor, like the Arithmetic Logic Unit (ALU). This way, all the execution units can work on their dedicated instruction pipelines, thus the processor uses its resources much more efficiently.

Instruction pipelining and superscalar processing still fall into the SISD category of the Flynn's taxonomy. SIMD was made possible by instruction sets like MMX, 3DNow!, SSE and AVX. They enable vector processing, e.g. an addition operation is executed simultaneously on multiple operands.

Simultaneous Multithreading (SMT) is a task-level parallelism technique and it allows the processor to switch between two threads while one thread waits for its sub-task to complete (e.g. an I/O operation).

The ultimate step towards MIMD was to place multiple cores on one chip. Every core can exploit all the parallelization techniques explained so far and on the top of that a thread which is being executed on a core can use its resources exclusively, thus the threads between cores are independent of each other.

2.4.2 GPU Architecture

CPUs are designed to perform general tasks, thus their design is complex. On the other hand, GPUs are specialized co-processors, originally designed to perform graphics tasks, which allows much simpler design of the execution units. This means that more of them can be placed onto a single chip, which offers tremendous amount of data parallelism (SIMD). GPUs are typically divided into multiple compute units which contain many individual processing elements (cores). E.g. the NVIDIA® GeForce GTX Titan X contains 3584 processing elements.

General-purpose computing on GPUs (GPGPU) was difficult to realize in the beginning because programmers had to write their code in a way which would fit the graphics pipeline. Despite that, GPU computing became popular for implementations of highly-parallelizable algorithms and vendors eventually extended their architectures to support full programability. NVIDIA® released its parallel computing platform and programming model called Compute Unified Device Architecture (CUDA). Its vendor-independent counterpart, OpenCL supports parallel programming of diverse processors, including CPUs, GPUs or FPGAs. However, its programming model seems to target the architecture of modern GPUs and is very similar to the one of CUDA.

2.4.3 OpenCL Programming

Since we are interested to support broad range of experiments at various facilities, we don't want to bound the users to a specific vendor. On the other hand, we do want to use the power of modern GPUs, which is why we select OpenCL to implement our parallel algorithms in this thesis.

OpenCL defines a *host*, which interacts with the environment external to the OpenCL program. A single host can contain multiple *compute devices*. A compute device consists of *compute units* and they are composed of *processing elements* [19].

OpenCL programs consist of two parts, a *host program*, which executes on the host, and one or more *kernels*. A kernel is a function written in OpenCL C language, compiled with the OpenCL compiler and executed on a compute device. To start kernel execution, the host program issues a special OpenCL command which creates an integer index space and executes an instance of the kernel for every point in this space.

An instance of an executing kernel is called a *work item*. Work items are organized into *work groups*, see Figure 2.11. Work items within a work group execute concurrently on processing elements of a single compute unit. The index space can have 1, 2 or 3 dimensions. Every work group has its unique ID within the index space and so does every work item. These IDs are known by every work item, thus they can base their calculations on them, e.g. every work item can load a specific pixel of a 2D image. Kernel execution by work items falls into the SIMD category. However, kernels support code-branching (e.g. an `If` statement) and in this case they can be categorized as Single Program Multiple Data (SPMD) to reflect the fact that code branching may cause the execution of different instructions by different work items.

OpenCL commands, like copying memory between the host and the device or executing a kernel are queued in so-called *command queues*, which enable task-level parallelism. Commands in these queues may be executed out-of-order and OpenCL provides synchronization primitives to control the execution order. The execution of multiple commands at once is beneficial for example to simultaneously execute the kernel code and transfer data to or from the host.

Memory Model

OpenCL distinguishes multiple memory spaces with various advantages which should be carefully considered when we design a kernel. This will be very important in Chapter 4, where we will develop a kernel for laminographic back projection. The memory spaces are:

- Host memory: visible only to the host program and inaccessible from a kernel.
- Global memory: off-chip memory visible to all work items from all work groups.
- Constant memory: as global memory but read-only, which may be used to

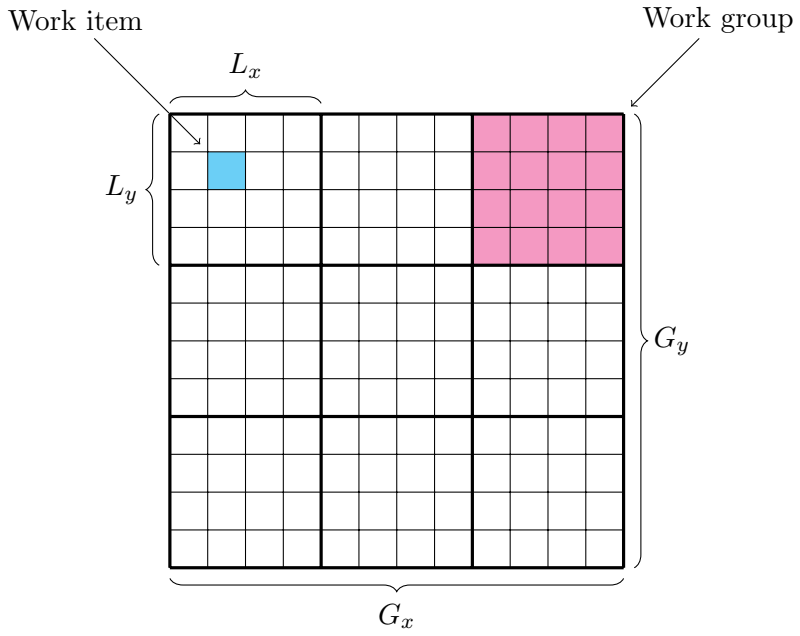


Figure 2.11: 2D index space division in OpenCL. (L_x, L_y) are the work group dimensions and (G_x, G_y) are the global dimensions.

implement special access patterns and caching policies to reduce the required memory bandwidth [64].

- Local memory: visible only to work items within a work group. This memory is often cached on-chip which can increase the effective bandwidth compared to the global memory. It is typically used by algorithms with highly localized memory access patterns.
- Private memory: visible only to a work item. This memory is often located on-chip, thus has very fast access times. However, if a kernel uses this memory space too much, parts of it may be stored in the off-chip global memory, called *register spilling*.

Performance Considerations

On the one hand, OpenCL provides reasonable abstractions to support a wide range of compute devices. On the other hand, architectural differences have great impact on the overall program performance and we may need to implement the same program multiple times to exploit the most of a specific architecture. For example, the memory spaces listed above may be cached differently, work group sizes may need to be chosen differently between devices to fit the layout of their compute units, we should hide the latency of data fetching from global memory by sufficient amount of arithmetic operations and many, many more. A good overview is provided in [65].

We will use various performance optimization strategies to develop a 3D sample structure reconstruction algorithm in Chapter 4.

2.5 Summary

In this chapter, we introduced X-ray imaging principles and methods which are the cornerstone of various X-ray imaging experiments. The imaging principles will be important in Chapter 3 for the development of our simulation framework *syris*.

We further reviewed the existing X-ray imaging simulation and automation software and pointed out limitations which need to be solved to enable autonomous experiments.

We also briefly discussed parallel computing, multi-core and GPU architectures and described the basics of OpenCL, which we will use to implement a 3D sample structure reconstruction algorithm in Chapter 4.

Chapter 3

Simulation Framework for X-ray Imaging Experiments

In this section we address research Question 1: *How to find the optimal combination of experimental conditions and data processing algorithms before the experiment starts?*

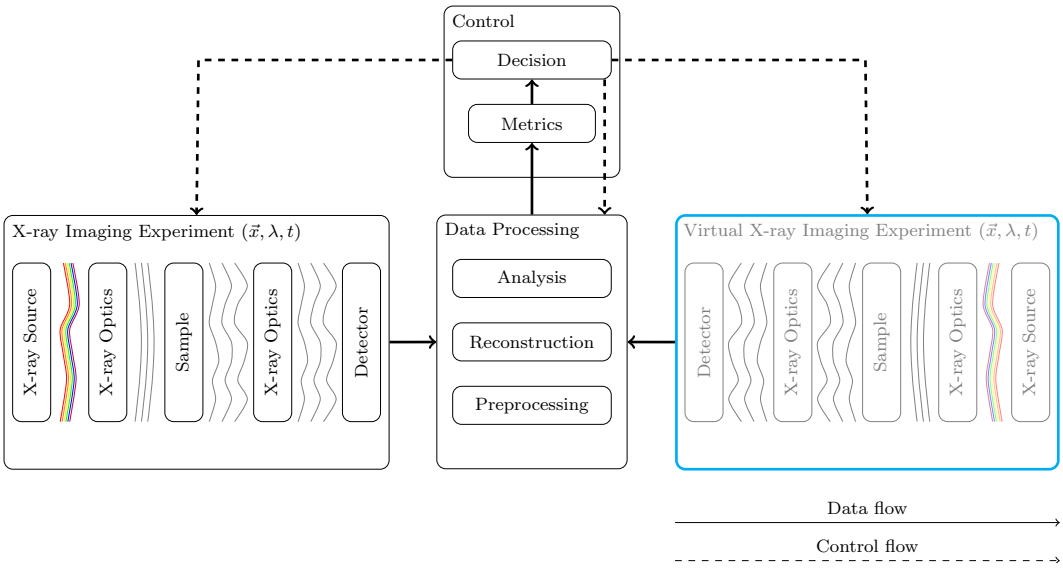


Figure 3.1: Our focus in this chapter, *virtual X-ray imaging experiments* highlighted in cyan.

Thus, our concern here are *virtual X-ray imaging experiments*, highlighted in Figure 3.1. We will design and implement a framework called *syris* [22] for conducting a broad range of *in-silico* X-ray imaging experiments based on suitable approximations of the image formation mechanisms described in Chapter 2. We will include sample motion and we will pay special attention to numerical issues related with discretization. Compute- and memory-intensive algorithms will be highly parallelized

by using GPU computing in order to enable fast simulations of high-speed experiments which produce tremendous amounts of data, e.g. time-resolved tomography.

We will demonstrate the capabilities of *syris* on three examples:

- In Section 3.2.2, we will conduct virtual high-speed radiography which shows how high-fidelity simulations can be used to find suitable image processing parameters.
- We will simulate a tomography experiment in Section 3.2.1 to show that simulation can be used to select suitable data acquisition strategies. In this particular example, we show how one can reduce the amount of acquired projections and at the same time keep high reconstruction accuracy.
- Grating interferometry experiment in Section 3.2.3 sheds light on the contrast formation process by special optics. We will investigate various approximations of the image formation process on simulated data and discuss their validity limits.

3.1 Implementation

syris has a clear application programming interface (API) written in Python[1] programming language, which makes it is easy to extend and change particular aspects of the image formation process in order to fit specific simulation needs. However, *syris* is usable off-the-shelf because it ships with implementations of all the necessary modules to conduct 4D experiments.

We will first describe the design of *syris* and then some important implementation details, like object shape creation algorithms, motion and image formation details, especially the sampling requirements for free-space propagation. We will also describe optimization and parallelization possibilities and their implementation.

3.1.1 Design

A simplified class diagram of *syris* is shown in Figure 3.2. The most important class is the `OpticalElement` which describes anything capable of creating or interacting with a wavefield at a given time. It can either be an `XraySource` which creates the initial wavefield $u_0(\vec{x}, z_1)$, a spectral beam `Filter` or an object with a particular shape modeled by the `Body` class providing $T_i(\vec{x})$ from (2.6).

`Body` uses `Material` class for obtaining the complex refractive index. Its purpose is then to compute the projected thickness, apply refractive index and provide $T_i(\vec{x})$ for different wavelengths. This particular implementation takes the projected thickness on input, so the users can provide their own, regardless how they calculated them. `MovableBody` is a base class for objects which can compute the time-dependent transmission function. The actual implementations of this class in *syris*

[1] http://www.python.org

are `Metaball` and `Mesh`, first of which is suitable for creating blob-like objects and the second for any shape creation based on a polygonal mesh. We will describe both in more detail in Section 3.1.2. `MovableBody` uses `Trajectory` class for motion description and its specifics are discussed in Section 3.1.3.

`Detector` is used to detect the wavefield at the imaging plane. Our implementation uses indirect detectors described in Section 3.1.6. The class uses subclasses `Scintillator` for converting X-rays to visible light photons, then `Lens` which magnifies the image and finally a `Camera`, which converts the light to detector counts.

Virtual experiment is conducted by the `Experiment` class, which takes care of the individual image formation steps and yields image sequences with all the specified physical aspects taken into account, including motion blur.

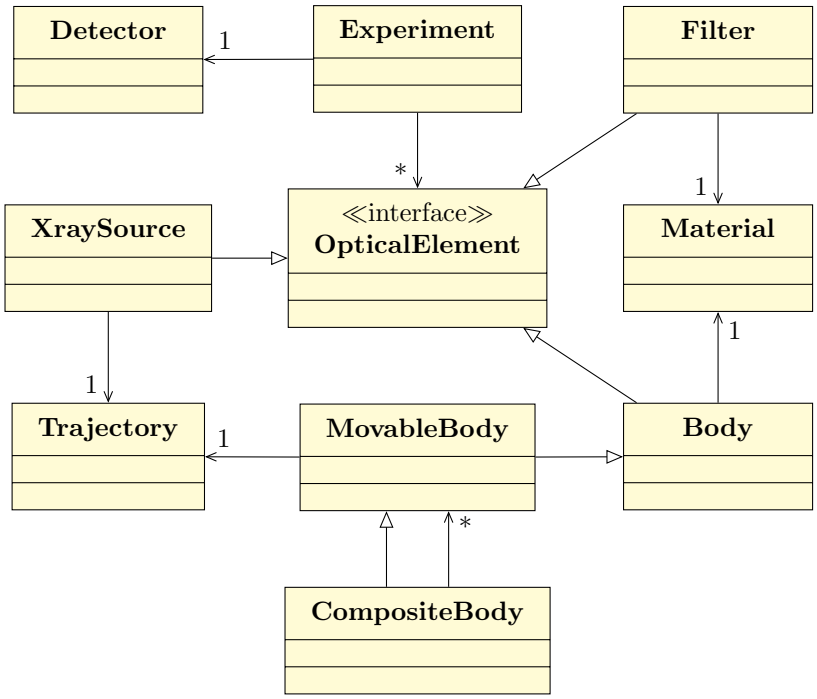
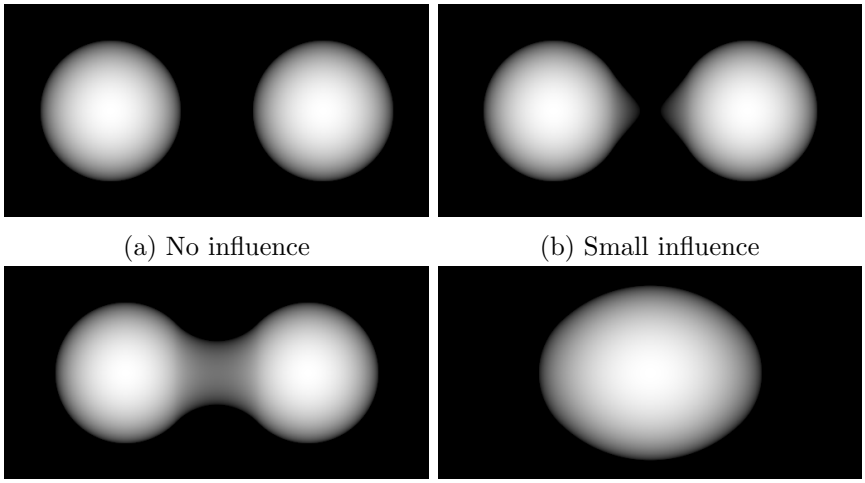


Figure 3.2: Simplified UML class diagram of *syris*. An `OpticalElement` is an abstract class for wave field manipulation. `XraySource` is a time-dependent source of X-rays, `Filter` is a spectral beam filter, `Body` represents objects with shapes and complex refractive indices obtained from `Material`. `MovableBody` adds motion to objects by using a `Trajectory`. `CompositeBody` encompasses multiple bodies for complex motion description. `Detector` implements light detection process. `Experiment` executes a virtual experiment and yields image sequences.

3.1.2 Object shapes

Object shape can be very complex, e.g. a composite material, biological tissue or even a small animal. Geometric primitives are not sufficient to model the tremendous



(a) No influence (b) Small influence

(c) Contact (d) Merged to one

Figure 3.3: Projections of two metaballs with different distances between them. The falloff function of one metaball in (a) doesn't the influence the fallof function of the other, the metaballs in (b) are closer and their falloff functions overlap by a small amount, which changes the final shape. The metaballs in (c) are so close that their falloff functions create one blob, which is even more pronounced in (d).

range of samples used in X-ray imaging experiments, which is why we need a more flexible approach for their modeling.

That is why we employ two object shape models. The first one is suitable for modeling liquid-like samples with smooth transitions and the second uses triangular meshes, which allow us to describe arbitrary shapes.

Metaballs

Objects with smooth transitions, like liquid blobs or organic materials can be conveniently modeled by metaballs [66], which is why we include them in *syris*. They are used to describe isosurfaces as combinations of falloff functions [67] resulting in organic looking blobs, see Figure 3.3. We will use metaballs with a finite falloff function. If a metaball's radius is r and we define its *influence region* to be $2r$, we can define the falloff function to be

$$f(d) = \begin{cases} \frac{\left(4r^2 - d^2\right)^2}{9r^4}, & |d| \leq R \\ 0, & |d| > 2r, \end{cases} \tag{3.1}$$

where $d = \sqrt{(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2}$ and (x_0, y_0, z_0) is the metaball origin. The isosurface at each point is then given by the sum of the individual metaballs $F(x, y, z) = \sum_i f_i(x, y, z)$. If the final value is greater or equal to one, point (x, y, z) is *inside* an object, *outside* otherwise.

A naive computation of the isosurface intersections for M metaballs and a volume of size N^3 voxels is described by Algorithm 3.1. Its advantage is that it is numerically stable with minor surface imperfections due to the rounding error. The problem is however the high computational cost $\mathcal{O}\left(MN^3\right)$.

Algorithm 3.1: Naive metaball isosurface intersections

Input: A list of metaballs M
Output: 2D set of ordered intersections $I(x, y)$

```
1  inside = −1
2  for x ← 0 to N − 1 do
3      for y ← 0 to N − 1 do
4          I(x, y) ← ∅
5          for z ← 0 to N − 1 do
6              s = 0
7              foreach m ∈ M do
8                  d = √((x − x_m0)² + (y − y_m0)² + (z − z_m0)²)
9                  if d < 2r_m then
10                     s += f(d)
11             if s ≥ 1 then
12                 if inside = 0 then
13                     I(x, y) ← I(x, y) ∪ {z}
14                 inside = 1
15             else
16                 if inside = 1 then
17                     I(x, y) ← I(x, y) ∪ {z}
18                 inside = 0
```

If we take into account that the influence region of a metaball is limited to $2r$ we don't have to evaluate the isosurface at every z position. Instead, we can split the z axis to intervals based on the influence regions. Any time a metaball starts or ends its influence, its coefficients are added or subtracted from $F(x, y, z)$, respectively [66]. We then need to find the roots of the quartic at every interval which give us the intersection points. This is shown in Algorithm 3.2, where we make use of two lists, L, sorted by the start of the influence of metaballs in pixel (x, y) and R, sorted by the end of influences. *HEAD* is a non-destructive read of the first list item, *POP* is its destructive counterpart. A metaball B starts its influence along z at B_{start} and ends at B_{end}. A metaball is either added to F or removed. *EVALUATE* then solves the quartic and provides the computed intersections, if any.

Since the quartic can be solved algebraically its time complexity is $\mathcal{O}(1)$. We employ the heapsort [68] algorithm for metaball sorting which time complexity is

$\mathcal{O}(n\log(n))$. Since there are maximum $2M+1$ intervals for M metaballs, the time complexity of Algorithm 3.2 is $\mathcal{O}\left(N^2 M\log(M)\right)$, which is an order of magnitude better than Algorithm 3.1 with respect to N. This approach however requires that only a small amount of metaballs influence one voxel because we need to store and sort the influencing metaballs per voxel, which drastically increases the amount of memory we need for L and R. However, this is usually satisfied because the level of detail of the resulting isosurface is given by the radius of the metaballs, thus there are typically many small metaballs spread across the volume, i.e. we can neglect many of them in one particular voxel.

Algorithm 3.2: Interval-based metaball isosurface intersections

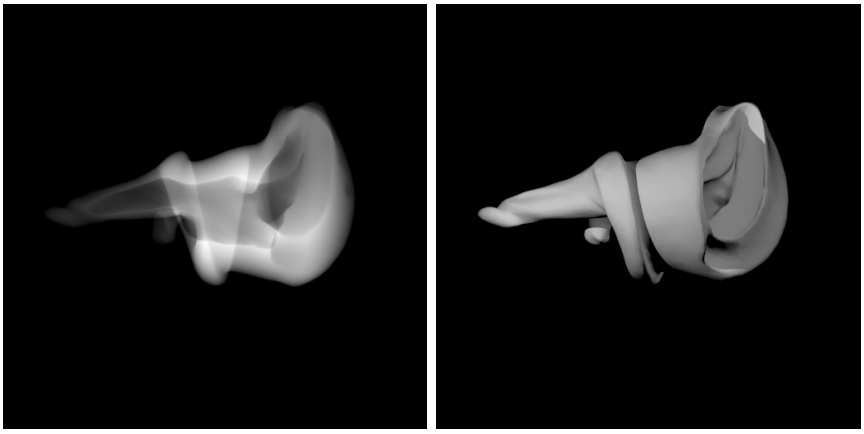
Input: A list of metaballs M
Output: 2D set of ordered intersections $I(x,y)$

```
1  for x ← 0 to N − 1 do
2      for y ← 0 to N − 1 do
3          I(x, y) ← ∅
4          F = 0
5          L ← SORTSTART(M)
6          R ← SORTEND(M)
7          while right ≠ ∅ do
8              if HEAD(L)_start < HEAD(R)_end then
9                  F = F + POP(L)
10             else
11                 F = F − POP(R)
12             I(x, y) ← I(x, y) ∪ {EVALUATE(F)}
```

Meshes

Triangular meshes can be used to create arbitrary shapes, such as the biological screw found in the hip joint of a beetle in Figure 3.4. As in this example, we can make a real tomography of a particularly interesting sample, segment it, convert to surface mesh, use it as an input for *syris*, conduct virtual experiments with various conditions and optimize the data analysis algorithms to achieve more precise reconstruction. An example of such an optimization is given in Section 3.2.1. Such simulation-based measurement and data analysis optimization is particularly interesting for long-term experiments which study many samples of the same kind or various conditions applied on a particular sample.

`Mesh` class requires a triangular mesh as its input. We then use the Möller–Trumbore [70] algorithm to determine the ray–triangle intersection. To speed up the computation by leaving out some triangles which are outside a pixel, we additionally prepare the mesh by sorting it with respect to the x coordinate and compute the



(a) Projected thickness (b) 3D rendering

Figure 3.4: (a) projected thickness of a mesh obtained by segmenting a real tomogram of a biological screw [69] and for illustration in (b) 3D mesh rendering.

largest distance w along the x-axis between the three triangle vertexes from all triangles. For an x coordinate and such a distance w, we may skip all triangles which end before x or start after $x + w$, as described in Algorithm 3.3. where the *FINDLEFTMOST*(T) routine finds the index to the triangle list T for which a triangle $T(i)$ doesn't end before x.

Once we have the intersections $I(x, y)$ we can convert them to the 2D projected thickness by subtracting pairs of intersections, regardless of which algorithm was used to compute them, the one for metaballs (Algorithm 3.2) or meshes (Algorithm 3.3). The intersections are then also used for computing 3D volumes which serve as the *ground truth* for object shapes. This is done by filling in the respective voxels between intersections instead merely subtracting them.

Algorithm 3.3: Ray–mesh intersection

Input: A list of triangles T
Output: 2D set of ordered intersections $I(x, y)$

```
1  SORTX(T)
2  for x ← 0 to N − 1 do
3      for y ← 0 to N − 1 do
4          I(x, y) ← ∅
5          i = FINDLEFTMOST(T)
6          while i < SIZE(T) and T(i)_right <= x + w do
7              p = COMPUTEINTERSECTION(T(i))
8              if p > −1 then
9                  I(x, y) ← I(x, y) ∪ {p}
10         SORT(I(x, y))
```

3.1.3 Motion

The position of a `MovableBody` can be specified either manually or automatically by assigning a `Trajectory` to it, implemented as a B-spline [71] $\vec{B}(t)$ with time parameter t. Parametrization by time enables us to place the body at any point of the spline at any time, i.e. we can create arbitrary velocity profiles. Moreover, `MovableBody` has a defined direction vector. The pose at a time t is then given by placing the body at $\vec{B}(t)$ and aligning its direction vector with the spline derivative $\vec{B}'(t)$. For instance, a metaball moving along the x-axis with constant velocity $5\,\mu\mathrm{m\,s}^{-1}$ can be created like this:

```
# Create a linear trajectory
x = np.linspace(0, 1, 128)
y = z = np.zeros_like(x)
# Create spline control points
control_points = zip(x, y, z) * q.mm
# Trajectory with a constant velocity
trajectory = Trajectory(control_points,
                        velocity=5 * q.um / q.s)
# MetaBall with 20 um radius
body = MetaBall(trajectory, 20 * q.um)
```

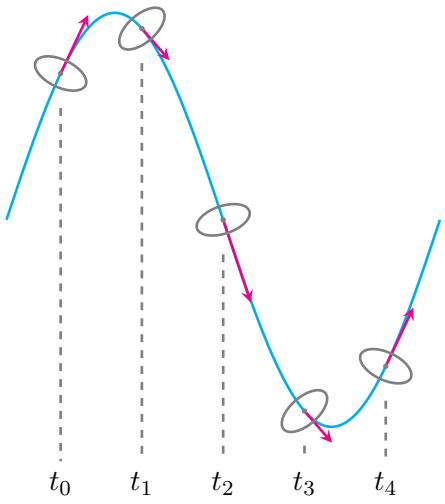


Figure 3.5: Spline trajectory in 2D. An elliptic sample follows a sine-like trajectory, its position at a time t_i is given by the spline (cyan curve) and its pose is given by the spline derivative (magenta arrows).

3.1.4 X-ray Sources and Transmission Function

High-speed experiments often use bending magnet or wiggler sources. They are included in *syris* by analytical computation of the source intensity distribution

$I_0(\vec{x}, z_1)$. The wavefield incident on the first object in the light path can be obtained by applying a spherical or the plane wave approximation from (2.3). In order to truthfully simulate the synchrotron sources, polychromaticity and spatial coherence effects described in Section 2.1.4 are implemented as well.

The complex refractive index (2.4) required to compute the transmission function from (2.6) is obtained either from the web-based interface of the CXRO [72] database, which enables energies of up to 30 keV, or the web interface of the X_{0h}[2] database enabling energies of up to 700 keV.

If an object consists only of one material we can decouple the refractive index from its internal structure. If the 3D object is defined as

$$f(\vec{x}, z) = \begin{cases} 1 & (\vec{x}, z) \in \text{object} \\ 0 & (\vec{x}, z) \notin \text{object}, \end{cases} \tag{3.2}$$

we can write its *projected thickness* as

$$p(\vec{x}) = \int f(\vec{x}, z)\, \mathrm{d}z. \tag{3.3}$$

and the transmission function as

$$T(\vec{x}) = e^{-k(\beta p(\vec{x}) - j(z + \delta p(\vec{x})))}. \tag{3.4}$$

where the projected thickness of the sample is computed from the intersections given by Algorithm 3.2 or Algorithm 3.3. If an object is composed of parts with different materials, the transmission functions of these parts are computed separately and multiplied, which accounts for the whole object.

3.1.5 Free-space Propagation

Even though we could implement the free-space propagation based on the real space propagator from (2.10), it is computationally much more efficient to compute the propagator directly in Fourier space and use the Convolution theorem [73]. Fortunately, this is feasible thanks to the angular spectrum representation of a wavefield [23], which we explain next.

Wavefield Propagation Based on its Angular Spectrum

Propagation based on the angular spectrum uses the fact that the 2D Fourier transform may be seen as the decomposition of a function into simple plane wave components. The 2D Fourier transform of a wavefield in a plane perpendicular to z is defined as

$$\tilde{u}(\vec{\xi}) = \mathcal{F}[u(\vec{x})] = \int u(\vec{x}) e^{-2\pi j \vec{\xi}\vec{x}} \mathrm{d}\vec{x} \tag{3.5}$$

[2] `http://x-server.gmca.aps.anl.gov/x0h.html`

and its inverse as

$$u(\vec{x}) = \mathcal{F}^{-1}[\tilde{u}(\vec{\xi})] = \int u(\vec{\xi}) e^{2\pi j \vec{\xi}\vec{x}} \mathrm{d}\vec{\xi} \tag{3.6}$$

where $\vec{\xi} = (\xi_x, \xi_y)$ is the 2D spatial frequency. We may look at the exponent of the inverse Fourier transform as a plane wave with the wavevector $\vec{k} = (k_x, k_y, k_z)$, $|\vec{k}| = k = 2\pi/\lambda$, where $k_x = 2\pi\xi_x$ and $k_y = 2\pi\xi_y$. Since the lateral spatial frequency $\vec{\xi}$ is related to the direction of the corresponding plane wave component, the 2D Fourier transform of a wavefield in the transverse plane is called *angular spectrum*.

Free-space propagation of each plane wave from the angular spectrum to the plane Δz corresponds to a multiplication with the phase factor $k_z \Delta z$, where

$$k_z = \sqrt{k^2 - k_x^2 - k_y^2} = \frac{2\pi}{\lambda}\sqrt{1 - (\lambda\vec{\xi})^2}. \tag{3.7}$$

The free-space propagated wavefield at Δz is

$$u(\vec{x}, \Delta z) = \int \tilde{u}(\vec{\xi}, 0) e^{j\frac{2\pi}{\lambda}\Delta z\sqrt{1-\left(\lambda\vec{\xi}\right)^2}} e^{2\pi j\vec{\xi}\vec{x}} \mathrm{d}\vec{\xi} \tag{3.8}$$

Thus, the free-space propagator from (2.10) in the Fourier space is

$$\tilde{P}(\vec{\xi}, \Delta z) = e^{jk\Delta z\sqrt{1-\left(\lambda\vec{\xi}\right)^2}}. \tag{3.9}$$

This allows us to rewrite $u_i(\vec{x}, z_i + \Delta z)$ from (2.11) in a computationally more efficient form which saves one Fourier transform of the propagator

$$u_i(\vec{x}, z_i + \Delta z) = \mathcal{F}^{-1}\left\{\mathcal{F}\left[u_{i-1}(\vec{x}, z_i) \cdot T_i(\vec{x})\right] \cdot \tilde{P}(\vec{\xi}, \Delta z)\right\}. \tag{3.10}$$

Fresnel Approximation

If the propagation distance is sufficiently larger than the lateral object extension we may use the Fresnel approximation of (2.9) [23]. It replaces the spherical wavefronts by the parabolic ones by taking into account only the 0^{th} and 1^{th} orders of the Taylor expansion of the square root into account. The r from (2.9) is approximated even further with Δz, yielding the Fresnel approximation

$$u(\vec{x}, \Delta z) = \frac{e^{jk\Delta z}}{j\lambda\Delta z} \int u(\vec{\eta}, 0) e^{\frac{jk}{2\Delta z}(\vec{x}-\vec{\eta})^2} \mathrm{d}\vec{\eta}, \tag{3.11}$$

rewritten as convolution

$$P_F(\vec{x}, \Delta z) = \frac{e^{jk\Delta z}}{j\lambda\Delta z} e^{\frac{jk}{2\Delta z}(\vec{x}-\vec{\eta})^2}. \tag{3.12}$$

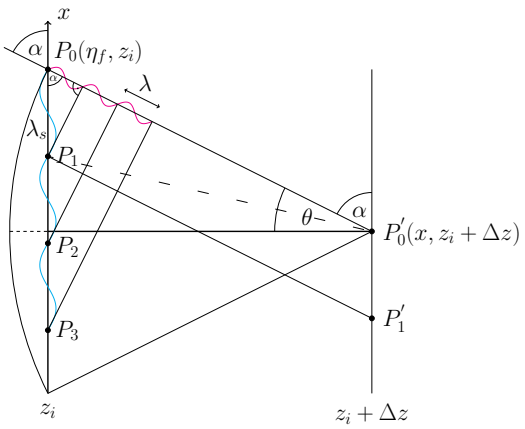


Figure 3.6: Illustration of 1D angular spectrum and one of its many plane wave components given by an X-ray wavelength λ and an inclination angle θ. The plane wave is linked to the spatial wavelength λ_s along the x-axis by $\lambda_s = \lambda/\cos(\alpha)$.

The same approximation applied on the square root in (3.9) leads to the following propagation kernel in the Fourier space

$$\tilde{P}_F(\vec{\xi}, \Delta z) = e^{jk\Delta z} e^{-j\pi\lambda\Delta z\vec{\xi}^2}. \tag{3.13}$$

which plays an important role by numerical calculations of intensity patterns, as will be described in Section 3.1.5.

Parabolic Incident Wave

The wavefront incident on the first object is in general spherical and its much better approximation than the plane wave introduced in (2.3) is a parabola. This is the same approximation as for the propagator above. If we add the parabolic phase profile to an otherwise arbitrary wavefield, we can write

$$u_i(\vec{x}, z_i) = \bar{u}_i(\vec{x}, z_i) e^{\frac{jk}{2z_i}\vec{x}^2}. \tag{3.14}$$

Let's now decompose (3.11) with respect to the object and its incident wavefield

$$u(\vec{x}, z_i + \Delta z) = \frac{e^{jk\Delta z}}{j\lambda\Delta z} \int \bar{u}_{i-1}(\vec{\eta}, z_i) \cdot T_i(\vec{\eta}) \cdot e^{\frac{jk}{2\Delta z}(\vec{x}-\vec{\eta})^2} \mathrm{d}\vec{\eta}. \tag{3.15}$$

If we now apply the parabolic incident wafefront from (3.14) we come to

$$u(\vec{x}, z_i + \Delta z) = \frac{e^{jk\Delta z}}{j\lambda\Delta z} \int \bar{u}_{i-1}(\vec{\eta}, z_i) \cdot T_i(\vec{\eta}) \cdot e^{\frac{jk}{2z_i}\vec{\eta}^2} \cdot e^{\frac{jk}{2\Delta z}(\vec{x}-\vec{\eta})^2} \mathrm{d}\vec{\eta}. \tag{3.16}$$

When we introduce the defocusing distance ΔD and magnification M [74]

$$\begin{aligned} \Delta D &= \frac{z_i \Delta z}{z_i + \Delta z} \\ M &= \frac{z_i + \Delta z}{z_i} \end{aligned}$$

and rearrange the exponent in (3.16), we come to

$$u(\vec{x}, z_i + \Delta z) = \frac{e^{\frac{jk}{2(z_i+\Delta z)}\vec{x}^2}}{\boldsymbol{M}} \cdot e^{\frac{jk(\Delta z)^2}{z_i+\Delta z}} \cdot \frac{e^{jk\boldsymbol{\Delta D}}}{j\lambda\boldsymbol{\Delta D}} \int \bar{u}_{i-1}(\vec{\eta}, z_i) \cdot T_i(\vec{\eta}) \cdot e^{\frac{jk}{2\boldsymbol{D}}\left[\left(\frac{\vec{x}}{M}-\vec{\eta}\right)^2\right]} \mathrm{d}\vec{\eta}, \tag{3.17}$$

which is the same diffraction integral as (3.15) but with changed propagation distance ΔD and the result being magnified by M [74], both emphasized by bold font in the equation. The first term accounts for the parabolic incident wavefield and the consequent amplitude drop by $1/M$. The second term is the correction factor for the changed mean propagation distance ΔD. We can write (3.17) also as convolution and derive a recursion formula similar to (2.11)

$$\begin{aligned} \bar{u}_0(\vec{x}, z_1) &= \sqrt{I_0} e^{jk\Delta z_0} \\ \bar{u}_i(M_i\vec{x}, z_i + \Delta z_i) &= \frac{e^{\frac{jk(\Delta z_i)^2}{z_i+\Delta z_i}}}{M_i} \left([\bar{u}_{i-1}(\vec{x}, z_i) \cdot T_i(\vec{x})] * P(\vec{x}, \Delta z_i)\right). \end{aligned} \tag{3.18}$$

Thus, the parabolic phase profile can be omitted during the propagation between the objects and taken into account only at the very end, after the last propagation. An important advantage of using such form of propagation is that if we are interested in the intensity of (3.18) with the added parabolic term

$$\begin{aligned} I_i(M_i\vec{x}, z_i + \Delta z_i) &= \left| \frac{e^{\frac{jk}{2(z_i+\Delta z_i)}\vec{x}^2}}{M_i} \cdot e^{\frac{jk(\Delta z_i)^2}{z_i+\Delta z_i}} \cdot \left([\bar{u}_{i-1}(\vec{x}, z_i) \cdot T_i(\vec{x})] * P(\vec{x}, \Delta z_i)\right) \right|^2 \\ &= \left| \frac{1}{M_i} \left(\bar{u}_{i-1}(\vec{x}, z_i) \cdot T_i(\vec{x})\right) * P(\vec{x}, \Delta z_i) \right|^2, \end{aligned} \tag{3.19}$$

we may omit the parabolic phase profile. This simplifies the computation and sampling criteria described in more detail in Section 3.1.8.

All of the described propagation schemes are supported in *syris* to provide free-space propagation with various levels of accuracy. E.g. when the synchrotron source is very far the Fresnel approximation from Section 3.1.5 might be sufficient but as the source gets closer, we might need the full propagator from Section 3.1.5. The following code snippet shows a white beam propagation of a sphere:

```
shape = (1024, 1024)
energies = range(15, 30) * q.keV
ps = 1 * q.um
material = make_henke('PMMA', energies)
sphere = make_sphere(shape, 256 * q.um,
                     pixel_size=ps,
                     material=material)
# Propagate to 1 m
result = propagate([sphere], shape,
                   energies, 1 * q.m, ps)
```

3.1.6 Detection Process

High-speed experiments typically use indirect detectors, which means that the X-ray photons are first converted to visible light photons by a scintillating screen. Then they are magnified by a lens and finally irradiate the sensor of a conventional camera. They are converted to electrons, amplified and converted to digital counts which are finally transferred to the computer.

Intensity right downstream the scintillator is composed of many visible light wavelengths λ_v depending on the emmission spectrum of the scintillator. The intensity superimposed over all λ_v can be written as [75]

$$I_N^v(\vec{x}, z_{N+1}) = S\left(\frac{z_1}{\Delta z_N}\vec{x}\right) * \int I_N(\vec{x}, z_{N+1}, \lambda)\left(1 - e^{-\mu(\lambda)p}\right)\frac{hc}{\lambda}L(\lambda)\,\mathrm{d}\lambda \tag{3.20}$$

where p is the scintillator thickness, $\mu(\lambda)$ its linear attenuation coefficient and $L(\lambda)$ the light yield expressing how many visible light photons are excited by absorption of one X-ray photon. The scintillator emits photons into all directions and only a part of them can be collected, which is described by the collection efficiency of a lens [75]. Furthermore, lens transmission efficiency describes how many of the collected photons actually pass through the lens. Finally, we consider the quantum efficiency of the camera which describes how many photons of a certain wavelength can the sensor convert to electrons. For this we also need the emission spectrum of the scintillator which tells us how many photons from (3.20) have which wavelength. For conciseness, we combine all of these factor into a *detector attenuation factor* A_{λ_v}.

If we take into account a thin scintillator, thus neglect its blurring effect and assume the optical system to be diffraction limited [26], we can describe the blurring of the detected image by convolution with kernel $R(\vec{x}, \lambda_v)$. If e_d^- is the mean number of electrons present without light, the total number of emmitted electrons in the camera sensor is

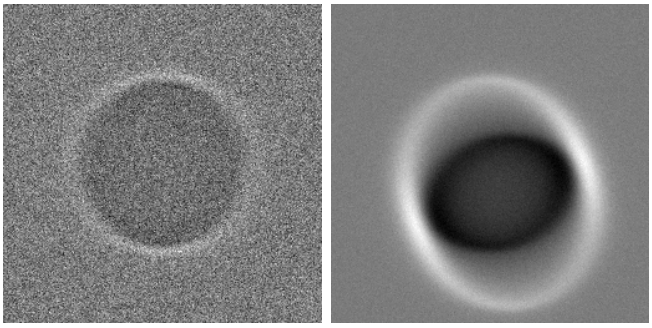
$$e^-(\vec{x}, z_{N+1}) = e_d^- + I_N^v(\vec{x}, z_{N+1}) * \int A(\lambda_v) \cdot R(\vec{x}, \lambda_v) d\lambda_v. \tag{3.21}$$

Digital counts on the camera output are given by applying the various noise sources from (2.14) to (3.21) and the noisy camera image recorded for acquisition time Δt can be written as $C(\vec{x}, \Delta t)$.

If we divide the exposure time of the k^{th} image $V_k(\vec{x})$ in a sequence into time intervals Δt in which the objects don't move more than one pixel, we can simulate motion blur by summing the camera images in these intervals

$$V_k(\vec{x}) = \sum_l C\left(\vec{x}, \Delta t_l\right). \tag{3.22}$$

Although motion blur is generally undesirable because it reduces the resolution, a reasonable compromise between signal-to-noise ratio and motion blur should be made because excessive noise also negatively impacts data analysis, see Figure 3.7.



(a) (b)

Figure 3.7: Noise vs. motion blur simulation. (a) a noisy detector image due to short exposure time, (b) less noise thanks to increased exposure time but motion blur appears.

3.1.7 Parallelization

Shape creation routines described in Section 3.1.2 operate *per-pixel*, i.e. they don't require the knowledge of the surrounding area of a pixel at which the projected thickness is currently being calculated. Also all operations which manipulate wavefields in the light path, except for noise formation, are implemented as a series of per-pixel multiplications either in the real space for pixel-wise signal change, or in the Fourier space for convolution.

Since the *per pixel* operations are independent of their neighborhoods, the power of GPUs can be greatly utilized and outperforms CPUs tremendously, as shown in Table 4.1. That is why all of the above-mentioned operations are implemented in OpenCL. We use pyfft[3], the OpenCL implementation of the Fast Fourier Transform algorithm [76] used for conversions between real and Fourier spaces.

In addition to *per pixel* parallelization we also provide `qmap`, a function that spreads *per image* computations across multiple devices for further speedup.

Table 3.1: CPU vs. GPU performance of propagating a 2D wavefield with different square dimensions (table columns) in white beam for 100 energy points. CPU is the Intel® Xeon® E5-2680 v2@2.8 GHz, GPU is the NVIDIA® GeForce® GTX Titan. The same code executed on a corresponding OpenCL platform is used.

Platform	512^2 (s)	1024^2 (s)	2048^2 (s)	4096^2 (s)	8192^2 (s)
CPU	2.377	3.903	8.078	29.626	189.393
GPU	1.951	1.965	2.195	4.084	8.995

[3] `https://github.com/fjarri-attic/pyfft`

3.1.8 Sampling

There are several potential sources of aliasing along the computational way which need to be dealt with, otherwise the error in the simulated images may be very severe. Starting with object shapes, we need to make sure that the influence region of a metaball or the smallest feature in a mesh is sufficiently larger than the pixel size so that they can be properly resolved. Once the objects are sampled sufficiently we need to consider two other important sources of aliasing in the image formation process described in Section 2.1. First is the transmission function (2.6) and second the free-space propagation (2.10). The reason is that they produce high frequency oscillations [77, 78] and if we want to resolve them our discretization must satisfy the sampling theorem [79]. Even though this is in principle relatively easy to do, we will also find ourselves in situations when we simply cannot obey the sampling theorem due to the lack of computational resources. In such cases it will be helpful to use some specific properties of the simulated physical phenomena which will enable us to relax the sampling criteria.

Transmission Function

Let's first examine the transmission function. Its attenuation effect on the wavefield is not our concern because it doesn't cause any oscillations. On the other hand, the phase shift caused by the object falls between $\pm\pi$ depending on the sample shape, which means it can create oscillations. If we expand the transmission function to

$$T_i(\vec{x}) = e^{-B_i(\vec{x})+j\phi_i(\vec{x})} = e^{-B_i(\vec{x})}\left(\cos(\phi_i(\vec{x})) + j\sin(\phi_i(\vec{x}))\right), \tag{3.23}$$

we see the sine and cosine terms which can be a source of aliasing. To satisfy the sampling theorem, we need to make sure that the phase shift $\phi_i(\vec{x})$ between two pixels is less than π. For simplicity, let's restrict ourselves to 2D space with pixel size Δx. Thus, we can write the requirement as $\Delta x \leq \pi/\mathrm{MAX}[\phi_i'(x)]$, where $\mathrm{MAX}[\phi_i'(x)]$ is the maximum derivative of $\phi_i(x)$.

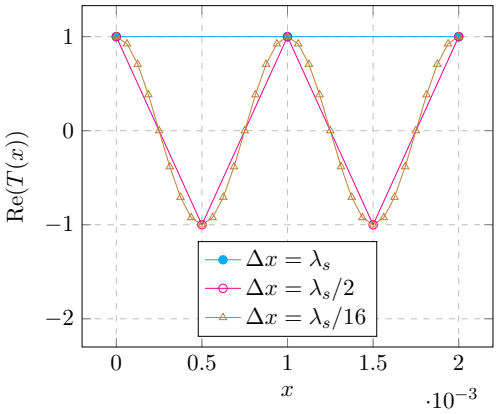
For illustration, let's take into account a 2D pure phase object (an object which doesn't attenuate X-rays, only changes the phase of the wavefront) with it's projected thickness given by the slope $f(x) = x$. Its transmission function is then

$$T_i(x) = e^{j\phi_i(\vec{x})} = e^{jk(1-\delta x)}, \tag{3.24}$$

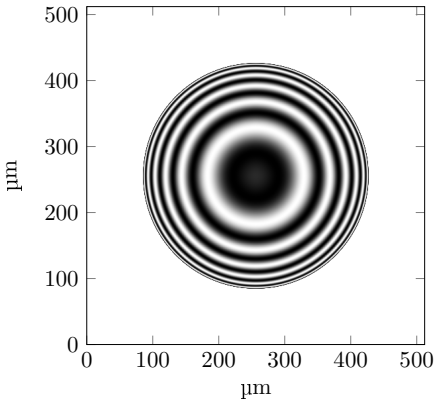
thus the derivative of the phase shift and the required pixel size are

$$\phi_i'(x) = -k\delta \implies \Delta x \leq \frac{\lambda}{2\delta}. \tag{3.25}$$

In this case the application of the transmission function creates a sine function along x with spatial wavelength $\lambda_s = \lambda/\delta$, which you can see in Figure 3.8a. Thus, we need a pixel to be half of this wavelength.



(a) Re($T(x)$) of a projected slope.



(b) Re($T(x)$) of a projected sphere.

Figure 3.8: (a) Real part of the 1D transmission function for a projected thickness given by $f(x) = x$. The function is for pixel size $\Delta x = \lambda_s/16$ resolved very nicely, still sufficiently for the sampling limit $\Delta x = \lambda_s/2$ and completely unresolved for $\Delta x = \lambda_s$, which in this case results in the object being completely invisible to the X-ray beam. (b) real part of the 2D transmission function given by a sufficiently sampled projected sphere.

Free-space Propagation

Free-space propagation described in Section 2.1.3 takes into account neighbouring points with slightly modified propagation distance. This results in spatial oscillations in the convolution kernel as we saw in Figure 2.5b. If we want to resolve these frequencies we again need to make sure that the phase shift between two adjacent pixels of a propagator is less than π. Let's again restrict ourselves to 2D for simplicity and use Figure 3.6 for explaining the sampling needs. We will consider the propagated wavefield at $P_0'(x, z_i + \Delta z)$ and limit the region which contributes to the propagation at z_i by $P_0(\eta_f, z_i)$ and analogously for the other half-plane by $P_0(-\eta_f, z_i)$. The field of view (FOV) is thus $2\eta_f$ and the maximum diffraction angle θ is the angle between P_0P_0' and the optical axis. Direction cosine with respect to the x-axis is then $\cos(\alpha) = \sin(\theta)$.

If we want to resolve all contributions to P_0' we need to make sure that the spacing Δx between $P_0(\eta_f, z_i)$ and the adjacent point $P_1(\eta, z_i)$ is such that the phase difference between the rays emerging from P_0 and P_1 is not more than $\lambda/2$ at P_0'. If $r_0 = \|P_0P_0'\|$ and $r_1 = \|P_1P_0'\|$ are the distances between P_0P_0' and P_1P_0' we can write

$$\Delta x \leq \frac{\lambda + r_1\,(1 - \cos(\Delta\theta))}{2\cos(\alpha)}, \tag{3.26}$$

where $\Delta\theta$ is the angle between P_0P_0' and P_1P_0'. Thus, the required sampling depends on the propagation distance and the maximum diffraction angle. $\Delta\theta$ in (3.26) often becomes negligible for typical synchrotron imaging use cases because the pixel size

and in turn $\Delta\theta$ is very small. As $\Delta\theta$ approaches zero, (3.26) becomes

$$\Delta x \leq \frac{\lambda}{2\cos(\alpha)}, \tag{3.27}$$

which means the required pixel size is half of the spatial wavelength λ_s discussed in Figure 3.6.

Let's now look at the sampling in the Fourier space. Since the propagator is a spherical wave which, according to the angular spectrum, can be decomposed to many plane waves we need to resolve the plane wave with the highest spatial frequency, which is the plane wave traveling in the direction of the largest diffraction angle. Based on the sampling theorem, the maximum frequency resolved by the discrete Fourier transform (DFT) is $1/(2\Delta x)$ and we need to make sure that this frequency is greater or equal to the maximum frequency of the propagator $1/\lambda_s$. Thus, the pixel size is limited by

$$\frac{1}{2\Delta x} \geq \frac{\cos(\alpha)}{\lambda} \implies \Delta x \leq \frac{\lambda}{2\cos(\alpha)}, \tag{3.28}$$

which is the same finding as for the real space case in (3.27). Thus, if we use the derived pixel size and FOV corresponding to the maximum diffraction angle, we will compute a properly sampled propagator in either of the two spaces. Once we know the maximum diffraction angle and the pixel size we can compute the number of required pixels as

$$N = \frac{2z}{\tan(\alpha)\Delta x} \tag{3.29}$$

and we can compute the discretized propagator. The two reflects the fact that the propagator center is in the middle of the discrete image and thus we need to consider two half-spaces in 2D to construct the propagator.

There are two cases leading to aliasing by propagation but before we discuss them it will be beneficial to recall the scaling property of the Fourier transform, i.e. the more a propagator stretches in one space, the more it shrinks in the other and vice versa. Moreover, we will use the term *supersampling*, which in real space means reducing Δx and increasing number of pixels by the same factor, thus the FOV remains the same. This however introduces new frequencies in the Fourier space since the largest frequency is $1/(2\Delta x)$, which leads to the *extension* of the Fourier space part that we need to consider. On the other hand, if we want to supersample the Fourier space, we reduce the spacing between the frequencies and increase the number of pixels in such a way that the highest frequency and Δx stay the same. Thus, the corresponding real space part becomes extended.

Since it is computationally more efficient to compute the propagator in the Fourier space, we will explain the aliasing on this setting. Let's first consider the case when we don't resolve all diffraction angles, which means we place Δz too far for a given pixel size, more precisely $\Delta z > N\Delta x/\tan(\theta)$. In this case we would need

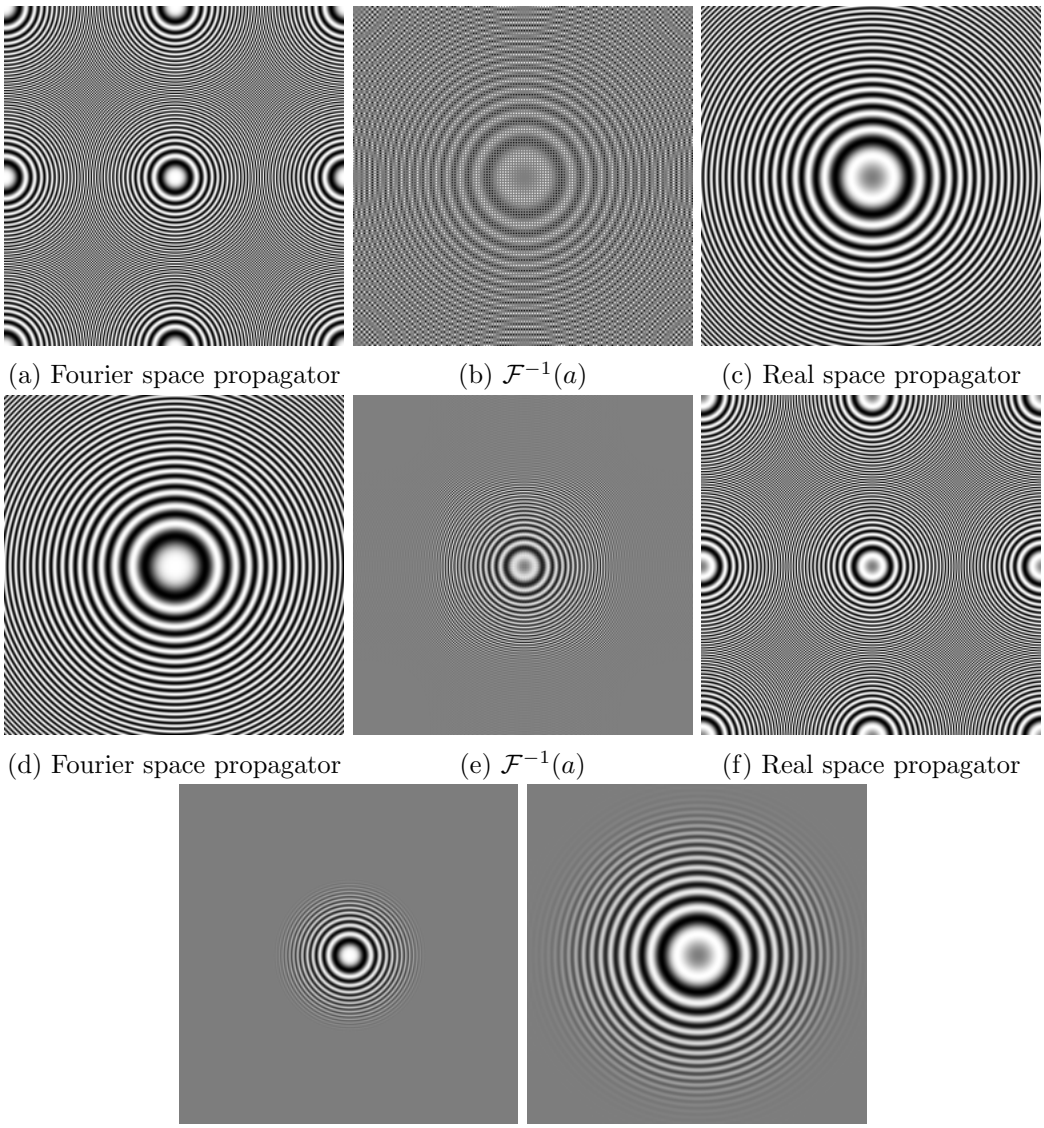
more pixels to resolve all frequencies up to the maximum frequency given by Δx, which inevitably leads to aliasing shown in Figure 3.9a. $\mathcal{F}^{-1}$ of such a propagator results in an incorrect propagator in the real space as shown ine Figure 3.9b and in extreme cases might lead to very deteriorated image quality shown in Figure 3.10a. This situation might be remedied by computing the propagator in real space and converting it to the Fourier space, or computationally more efficiently by mollifying (suppressing) the aliased frequencies as in Figure 3.9g.

Another type of aliasing occurs when the Fourier space propagator is too stretched, i.e. its real space counterpart is compacted so much that the fringes are unresolvable in real space. This is the case of Δz being too small for a given pixel size. This is the exact opposite of the previous case, i.e. the highest frequencies are not included in the Fourier space propagator. This means that we are not able to resolve the full FOV and the real space propagator parts beyond the maximum diffraction angle are aliased, shown in Figure 3.9f. An extreme case is shown in Figure 3.11 with its effect on the propagated pattern shown in Figure 3.12.

Aliasing is even more pronounced when we take into account spherical incident wave. In this case we need to consider higher diffraction angles because the mean propagation direction of every object point is given by the direction of the spherical wave in that point. Thus, λ_s must decrease and within the paraxial approximation we can write

$$\theta^{'} = \frac{\lambda}{\lambda_s^{'}} = \frac{M\eta_f}{\Delta z} \implies \lambda_s^{'} = \frac{\lambda \Delta z}{M\eta_f} \implies \lambda_s^{'} = \frac{\lambda_s}{M}, \tag{3.30}$$

where $\theta^{'}$ is the highest diffraction angle with the mean propagation direction adjusted for the spherical incident wave and $\lambda_s^{'}$ is the required wavelength based on $\theta^{'}$. We made use of $\theta \approx \tan(\theta) = \eta_f/\Delta z$ and $\theta \approx \sin(\theta) = \lambda/\lambda_s = \cos(\alpha)$. Because of the magnification, we need to propagate FOV which is M-times larger than the original and together with the reduced λ_s we can see that we need M^2 times more pixels than for the non-magnified case.

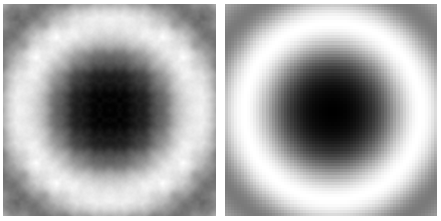


(a) Fourier space propagator (b) $\mathcal{F}^{-1}(a)$ (c) Real space propagator

(d) Fourier space propagator (e) $\mathcal{F}^{-1}(a)$ (f) Real space propagator

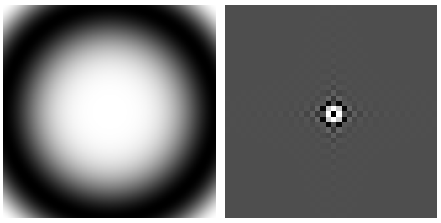
(g) Fourier space propagator (h) $\mathcal{F}^{-1}(a)$

Figure 3.9: Free-space propagator aliasing and its treatment. (a) A propagator in the Fourier space which is able to resolve only frequencies in the central half of the figure, which manifests by frequency folding. (b) is the inverse Fourier transform of (a) which yields an incorrect real space propagator. For comparison, (c) shows the correct propagator computed directly in the real space. Fourier space propagator in (d) is stretched, i.e. large diffraction angles are not present which leads to shrinking of the region we are able to resolve in the real space propagator (e). However, if the real space propagator is computed directly the regions with the unresolved diffraction angles are aliased (f). (g) is the same as (a) but with gradually mollified aliased frequencies. Its inverse Fourier transform (h) doesn't suffer from aliasing problems anymore. We can observe the mollifier graduation by slowly vanishing frequencies in (g) reflected by suppressed outer regions in (h).



(a) Aliased (b) Not aliased

Figure 3.10: Intensity image of a propagated sphere using an aliased propagator (a) and one that is properly sampled (b). We propagate far away from the original sphere in order to illustrate the aliasing, so the original shape of the sphere is not recognizable anymore.



(a) Fourier p. (b) $\mathcal{F}^{-1}(a)$

Figure 3.11: An extreme case of a stretched Fourier space propagator, which can resolve only diffraction given by the field of view $\eta_f = 2\Delta x$.

(a) Aliased (b) Supersampled (c) Supers. and decimated

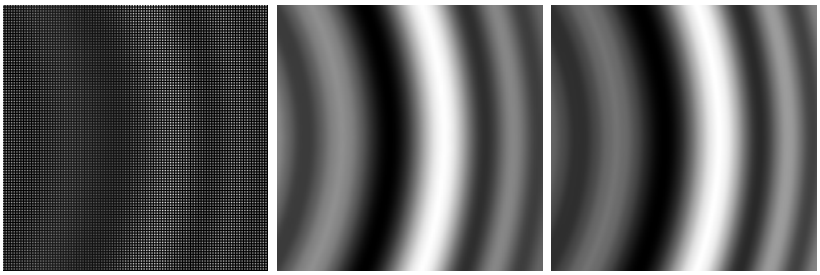
Figure 3.12: Propagation of a sphere in a setting which is able to resolve only diffraction angles bounded by $\eta_f = 2\Delta x$. (a) is the intensity obtained by using the propagator from Figure 3.11 (a). The horizontal oscillations are not physically correct and are the result of aliasing. In (b) we used 8-times supersampled propagation and in (c) we decimate the image to match the Δx in (a). This is what a detector in a real imaging system does automatically. As we can see, the edge of the sphere is pronounced in (c) but there are no high-frequency oscillations as opposed to (b) because we cannot resolve them with the chosen Δx.

Circular Convolution

We need to take great care about the circular convolution when propagating with explicitly computed spherical incident wavefield. If we want to convolve two 1D discrete signals with N and M pixels each, the region unspoiled by the circular convolution is $N - M + 1$. The propagator can have large spatial extent and so large area of the propagated result can be spoiled. This area can be very pronounced because the propagator picks up the wrapped parts of the incident wavefield, shown in Figure 3.13 (a). To remedy this, we need to limit the propagator's spatial extent and thus the maximum diffraction angle. Physically, this reflects the fact that only a limited region of the wavefield around an output pixel is capable to interfere, which means the coherence of the wavefield is reduced, which is common by X-ray imaging setups.

Let's require that the propagated unspoiled region covers the original FOV, which is $2\eta_f$. This means that we need to limit the spatial extent of the propagator to $2M\eta_f - 2\eta_f + \Delta x$. We can then compute the properly sampled incident wavefield in the plane $z_1 + \Delta z$, where z_1 is the distance between the source and the sample and the spatial extent at this plane is $2M\eta_f$. We then compute the propagator using the same sampling and grid as for the incident wavefield. It will not be aliased unless $M < 2$ and in this case we should supersample the incident wavefield in order to be able to resolve all diffraction angles within the FOV. We can now multiply the object's transmission function with the incident wavefield, convolve with the propagator and crop the result to $2\eta_f$.

A much more straightforward way to compute the spherical wave propagation is to apply (3.19), shown in in Figure 3.13 (c). There is no incident wavefield involved so we can simply take sufficiently large outlier and compute the propagated pattern as for the plane wave case, just with corrected propagation distance and intensity.



(a) Untreated Outlier (b) Treated Outlier (c) Stretched

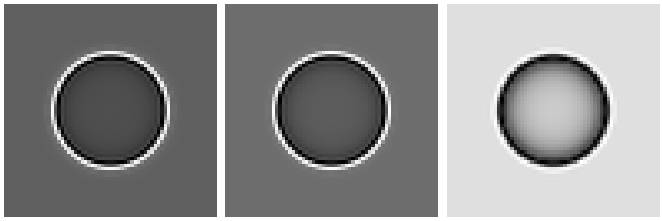
Figure 3.13: Circular convolution effect shown on a propagated sphere in cone beam with $M = 2$. Both (a) and (b) are computed by using the properly sampled incident spherical wavefield and the propagator. However, in (a) we do not suppress higher diffraction angles which leads to error caused by convolution with the wrapped incident wavefield. In (b) the propagator is restricted so that the circular convolution doesn't influence the ROI. (c) shows the stretched plane wave computation (3.18).

Memory Limitations

Propagation distance at synchrotrons is typically not larger than a few meters because the source blurring described in Section 2.1.4 increases with it as well, which in turn decreases resolution. The maximum resolution of the complete imaging system used for high-speed experiments based on indirect detector described in Section 3.1.6 is limited also by other factors and is usually not better than 1 µm. This means that the high spatial frequencies of the propagation cannot be resolved and cancel each other out. This limits the spatial extent of the wavefield we need to take into account by propagation and thus reduces the spatial extent of the propagator and the required amount of supersampling.

For instance, if we consider a camera with 64×64 pixels of size 2.5 µm×2.5 µm, a properly sampled propagator for the FOV given by the camera and distance 1 m would require 4096×4096 pixels of size 0.078 µm×0.078 µm. If we propagate using such sampling and then decimate the image to the camera pixel size we will loose the high-frequency oscillations, thus we can reduce the supersampling level. For comparison of different levels see Figure 3.14 and Figure 3.15.

The relaxed spatial extent and supersampling level have a very important practical use for situations when we work with large camera pixels and FOVs. Instead of requiring millions of pixels for proper sampling of the propagator and then decimating the image to the camera sensor size, we may split the large FOV into tiles, compute the propagation separately for each tile and merge them afterward. An example of such propagation may be found in Figure 3.20, where we used 256 tiles to propagate the wavefield by 10 cm in high resolution with 0.17 µm pixel size, decimated the propagated intensity pattern to match the desired pixel size 5.5 µm and merged the mosaic to form the final 1024×1024 image.



(a) 4096 pixels (b) 64 pixels (c) 4 pixels

Figure 3.14: Sphere propagation and decimation to the camera pixel size which limits the final resolution. The camera sensor size is 128×128 and pixel size is 2.5 µm×2.5 µm. Sample and propagator are supersampled and the propagated pattern is decimated to match the camera sensor. Supersampling required for a propagator which is not aliased in the FOV is 32, i.e. we need 4096×4096 pixels for computation. Since we loose the high frequencies due to the decimation shown in (a), we may reduce the supersampling factor and work with 512×512 images instead. In this case the propagator's extent is 64×64 pixels and the propagation gives almost identical image shown in (b). However, we cannot reduce the supersampling too much because as the propagator's spatial extent becomes too narrow it is unable to represent even the low frequency oscillations, like in Figure 3.11 and here in (c), where the propagator size is only 4×4.

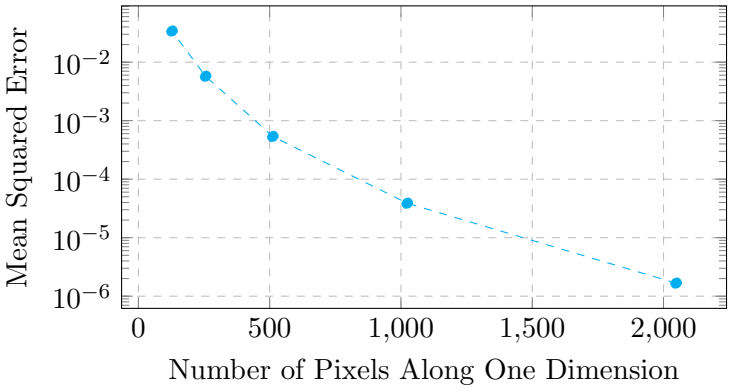


Figure 3.15: Mean Squared Error (MSE) as a function of supersampling of the wavefield and the propagator. We showed some supersampling levels in Figure 3.14 and their impact on the accuracy of the propagated intensity pattern and here we continue by comparing more of them in terms of MSE of the resulting image with respect to the image with the highest used supersampling. We need far fewer pixels combined with less supersampling than required to resolve the full propagator because the limited resolution of the imaging system cannot represent the high-frequency oscillations.

3.2 Example Experiments

We have explained the principles of the image formation at synchrotrons with some important aspects of the high-speed imaging, our simulation framework *syris* which is capable of conducting such virtual imaging experiments and now it is time to demonstrate the usefulness of the simulations for finding the optimal imaging conditions and data analysis parameters while keeping the specifics of the studied process and the data processing pipeline in mind.

3.2.1 Tomography

In this section we present a use case of *syris* that studies the tomographic reconstruction accuracy of the filtered back projection algorithm (FBP) based on different imaging conditions and how such a study can help users to choose the optimal imaging conditions for a particular experiment. This example is based on the virtual experiment in [22].

We will consider a continuous tomographic scan. The data acquisition time depends on the number of tomographic projections N and the camera exposure time t used to record one projection. If the sample changes in time, we might need to speed up the data acquisition so that its motion doesn't cause reconstruction artifacts. We can either reduce the exposure time t or the number of projections N. As we will see further, such reductions have different impacts on the reconstruction speed and accuracy and both of them might be beneficial in different situations.

First we create a 3D phantom shown in Figure 3.16. It is a triangular mesh created in Blender[4] and used by *syris* to compute the projected thickness. It consists of various shapes, sizes and materials which cause different kinds of reconstruction artifacts, e.g. streaks caused by sharp edges and beam hardening caused by strongly absorbing materials. The shapes are mostly periodic patterns with increasing frequencies which can quickly reveal maximum resolution in a specific direction. The sample is inhomogeneous and consists of aluminum, calcium, scandium and titanium, so that even the quantitative correctness of a reconstruction algorithm can be evaluated.

We will focus on the reconstruction artifacts related to motion blur and shot noise, which depend on t. We will create X-ray projections by using a monochromatic plane incident wave with 20 keV energy, place the sample very close to the detector so that the free-space propagation doesn't occur and use 1:1 conversion factor between X-ray photons and detector counts, i.e. every photon impinging on the scintillator creates one visible light photon and this photon creates one count in the camera sensor. Projections are 4 times oversampled and the PSF of the whole imaging system has a Gaussian profile with Full Width at Half Maximum (FWHM) set to one camera pixel.

We start by acquiring a virtual tomographic data set consisting of $N_0 = 1600$

[4] https://www.blender.org/

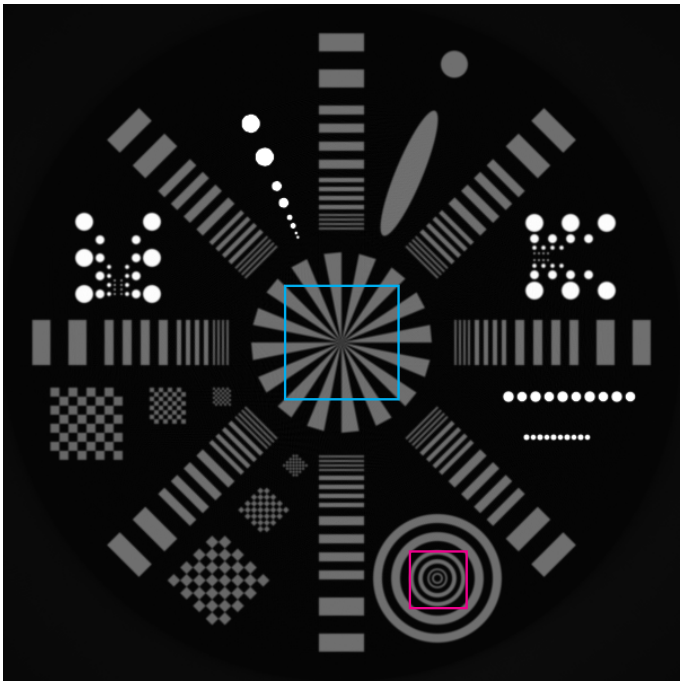
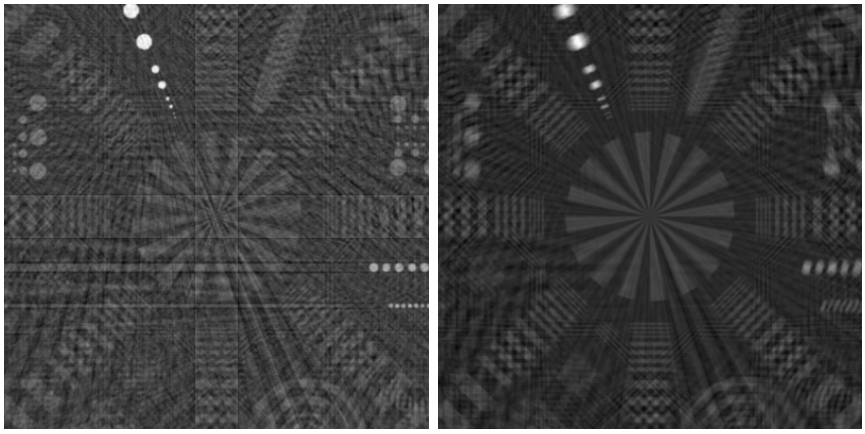


Figure 3.16: 2D cross-section of a 3D phantom constructed from a triangular mesh. It includes features of varying shapes, sizes and materials in order to provoke different reconstruction artifacts. The rectangles mark regions used for comparing the reconstruction accuracy in this section.

projections, which satisfies the angular sampling limit for our pattern [34]. Every projection is acquired in t_0 time and the number of detected counts in this time in one pixel is 128 000 in the case where the sample is outside the FOV. This number decreases based on the Beer–Lambert law with the projected sample thickness and its refractive index. The rotation speed is $\pi/(Nt_0)$ rad s^{-1}, therefore there is no rotational motion blur. To simulate the reduced acquisition time, we either decrease t or N, increase the sample rotation speed by the same factor, acquire tomographic projections, reconstruct them using FBP and compare the slices with the ground truth, which is the slice reconstructed from the starting data set. For comparison we use the Mean Squared Error (MSE) between the ground truth slice and a slice reconstructed from a data set with a reduced acquisition time. To suppress statistical fluctuations, we recompute the reduced acquisition time slice and the MSE 100 times and use the average MSE.

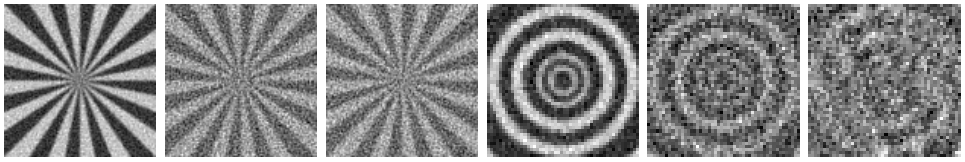
It was shown that if the noise level is the same in all projections, its power spectrum grows inversely with t or N [80]. If we use the distributivity of FBP and the fact that the noisy slice for the reduced t case is the sum of the ground truth and noise, MSE becomes the mean square of the reconstructed noise, which by the Parseval's theorem can be estimated by the mean of the noise power spectrum. If



(a) (b)

Figure 3.17: A slice reconstructed from 50 projections obtained by (a) using only every 32^{nd} angular position out of 1600. All angular positions are present in (b) but 32 consequent angular positions are superimposed on one projection due to motion blur simulation. While in (a) there are severe artifacts across the whole slice, the parts in (b) close to the rotation axis (in the middle) are quite preserved.

we now take into account that the intensity attenuated by our sample is on average considerably smaller than the one of the incident beam and neglect the noise dependence on the sample, we can finally conclude that the *MSE for the reduced t mode grows linearly with the reduction factor*, as shown in Figure 3.19.



(a) (b) (c) (d) (e) (f)

Figure 3.18: Two reconstructed slice ROIs of the phantom from Figure 3.16, (a) reconstructed from 1600 projections, (b) same number of projections but 16-times shorter t, (c) 16-times less N with the same t. The remainder shows the same conditions in a ROI further away from the rotation axis. The difference between (b) and (c) is negligible but the structure in (f) is hardly recognizable due to strong angular undersampling. On the other hand the outer ring transitions in (e) are preserved in all directions. Figure from [22].

The sample structure combined with the rotational motion blur starts to play a role for the reduced N case. Even though angular undersampling itself may cause severe artifacts across the whole slice shown in Figure 3.17, the rotational blurring may significantly suppress the error, especially when the ROI is close to the rotation axis. Indeed, the MSE for the ROI marked by the cyan rectangle in Figure 3.16 grows almost linearly as in the case of reduced t, meaning that the error stemming from the reduced number of projections is small compared to the error caused by

the increased noise, as depicted on the left of Figure 3.19.

The MSE for a ROI positioned away from the rotation axis (magenta in Figure 3.16) is shown on the right of Figure 3.19. Interestingly, it stays almost linear even for much fewer projections than needed to sample the ROI and starts deviating later. This means that we can obtain comparable reconstruction accuracy as by reducing t but by using far fewer projections. This property is especially useful for local tomography experiments and alignment of the rotation axis with the center of the studied ROI.

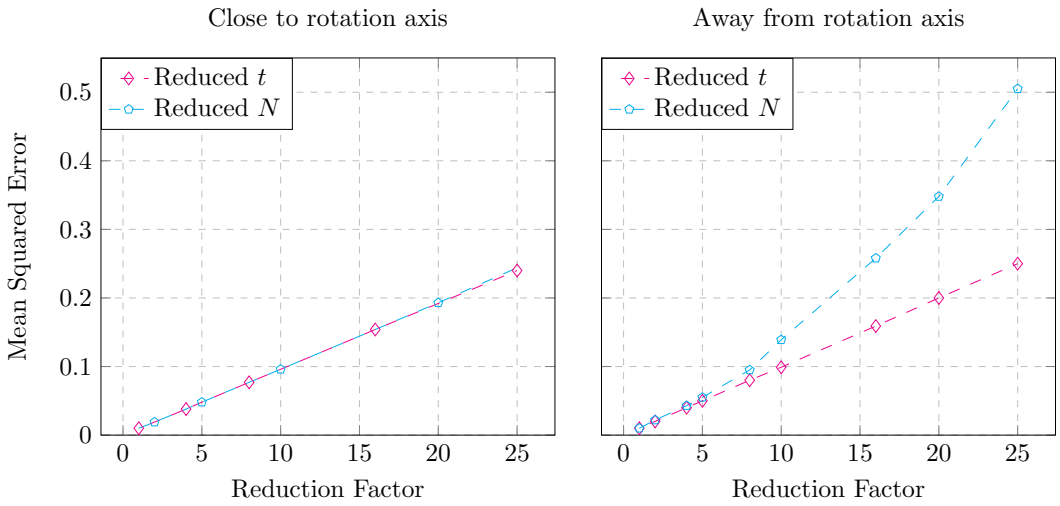
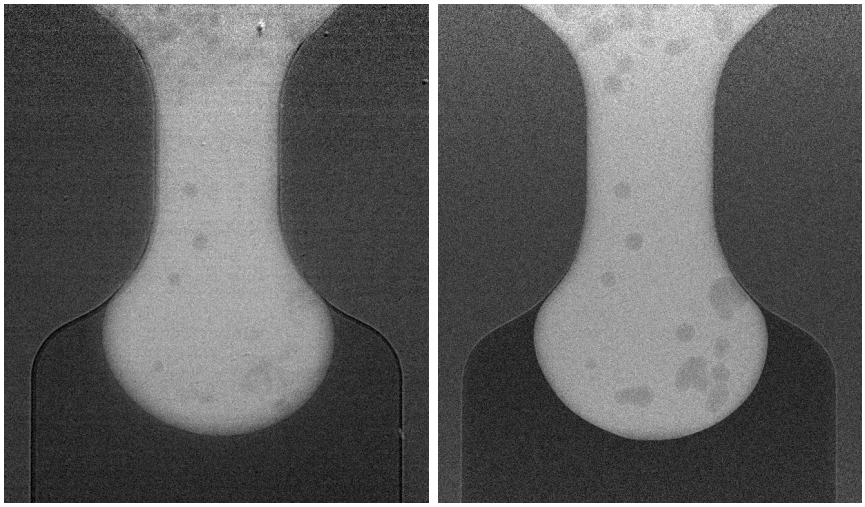


Figure 3.19: MSE of ground truth slices and slices obtained from reconstruction of projections with various reduction factors and techniques. There are two ROIs, close to the rotation axis (left, cyan in Figure 3.16) and further away from the rotation axis (right, magenta in Figure 3.16). *Reduced N* shows MSE for the case of reduced number of projections, *Reduced t* for the case of shorter exposure time t. The error caused by reducing N is almost identical to the one caused by shorter t when the ROI is close to the rotation axis. When the ROI is further away it behaves almost linearly for some time but eventually increases more rapidly due to the angular undersampling. Figure from [22].

3.2.2 Motion Estimation Algorithm Selection and Optimization of its Parameters

In this section we demonstrate how *syris* can be used to select a suitable motion estimation algorithm for the analysis of image sequences obtained from a *high-speed in-situ* X-ray imaging experiment. We will analyze the performance of various optical flow algorithms, select the best one and optimize its parameters on simulated data. Thanks to high-fidelity simulation, the parameters can be employed to analyze the real image sequence. This example is based on [22].

The experiment is a *high-speed in-situ* radiography studying particle and liquid motion in a heated semi-solid aluminum alloy. The experiment was conducted by using white beam at the ID15a beam line of ESRF [81]. The aim of the experiment was a quantitative investigation of complex flow dynamics, which requires an automated, robust and accurate motion estimation method.



(a) Real Experiment (b) Simulation

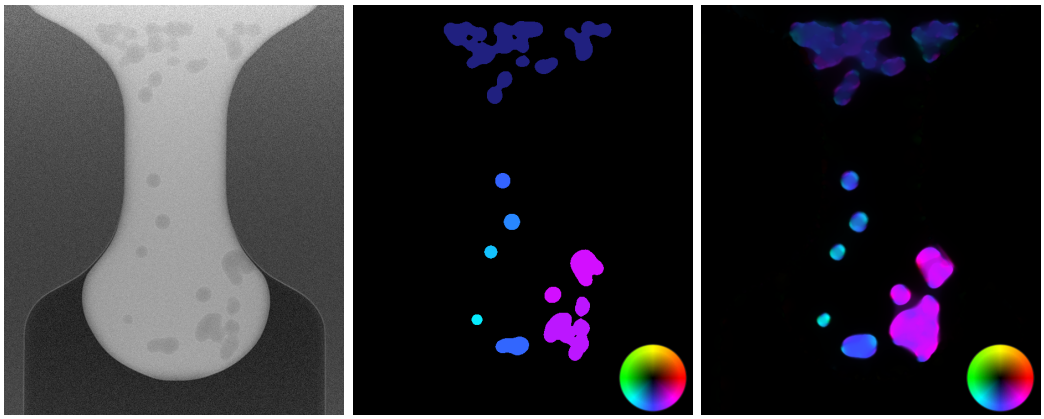
Figure 3.20: Absorbance images from a sequence. Real experiment data in (a) [81] and its simulation in (b). Figure from [22].

The virtual experiment recreates the original one by using the wiggler source properties of the ID15a beam line, meshes to create static parts of the sample and metaballs to create the particles inside the liquid. We compute the transmission functions of the sample parts and propagate the wavefield from the sample exit plane to a 100 µm thick YAG:Ce (Ce-doped $Y_3Al_5O_{12}$) scintillator where it is converted to visible light, passes through a lens with magnification 3.63 and is detected by a virtual Photron SA1 camera[5] with the effective pixel size 5.5 µm. The comparison of the real and the virtual experiment is shown in Figure 3.20.

When we have the simulated image sequence, we can perform a quantitative

[5] `http://photron.com/high-speed/cameras/fastcam-sa1-1`

comparison of some motion estimation algorithms and select the one with the best accuracy. As an input we take a pair of absorbance images obtained from the simulated radiographs with low contrast-to-noise ratio, which imposes significant challenge for motion estimation. We compare the *ground truth*, which is the flow field applied on the droplets in the simulation, and the reconstructed flow field obtained from an optical flow algorithm in Figure 3.21.



(a) Simulated image (b) Ground truth flow field (c) Estimated flow field

Figure 3.21: Motion estimation on simulated data using optical flow methods. (a) first frame of the simulated image sequence, (b) ground truth flow field, (c) computed flow field using method M_4. Color code shows the direction with the color and flow magnitude with its brightness. Figure from [22].

For comparison we choose four variational optical flow methods: Horn and Schunck M_1 [17], $M_2 = M_1$ + robust flow-driven [82], $M_3 = M_2$ + combined local-global approach [83] and $M_4 = M_3$ + intermediate flow filtering [84]. To assess the accuracy of the result we use the *endpoint error*

$$EE = \sqrt{(u_{GT} - u_{res})^2 + (v_{GT} - v_{res})^2}. \tag{3.31}$$

This metric determines the absolute difference between the ground truth $\mathbf{w}_{GT} = (u_{GT}, v_{GT})$ and the resulting $\mathbf{w}_{res} = (u_{res}, v_{res})$.

Since the static background occupies substantial part of the image, we measure the accuracy only in the vicinity of the moving particles. Performance comparison of the four optical flow methods is given in Figure 3.22 a and Table 3.2. The most accurate method is M_4 and we will further investigate its accuracy based on its parameters.

A variational optical flow model in its general form consists of two terms, a data term and a smoothness term [84]. To control the influence of both terms a special weighting parameter is used, the so-called smoothness parameter α. Optimizing this parameter is crucial in order to obtain the most accurate flow field. We can find the best value of the smoothness parameter for the selected algorithm based on the simulated data set.

Table 3.2: Comparison of four variational optical flow methods (M_1 = Horn and Schunck, M_2 = M_1 + robust flow-driven, M_3 = M_2 + combined local-global, M_4 = M_3 + flow filtering) on synthetic dataset with noise. Table from [22].

Model	Average endpoint error
M_1 = Horn and Schunck	0.664
M_2 = M_1 + robust flow-driven	0.655
M_3 = M_2 + combined local-global	0.624
M_4 = M_3 + flow filtering	**0.56**

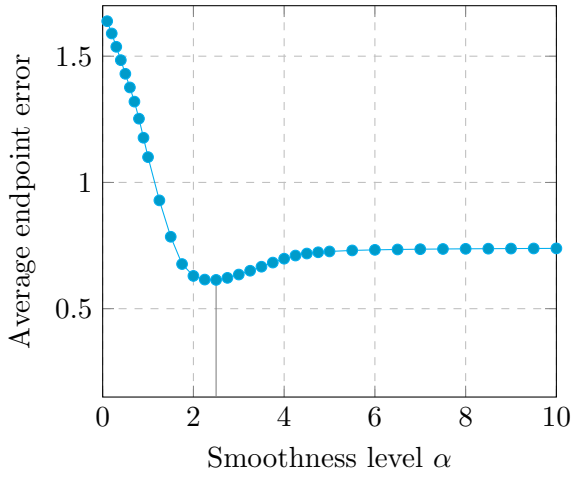


Figure 3.22: Analysis of the optical flow algorithm M_4 (Horn and Schunck, robust flow-driven, combined local-global, flow filtering). The smoothness parameter α has a global minimum. Thus, we may use the corresponding α to obtain the most accurate analysis of the real image sequence.

The average endpoint error of the optical flow algorithm M_4 with respect to different values of the smoothness parameter is shown in Figure 3.22. As we can see, the increase of the smoothness parameter starting from $\alpha = 1.0$ improves the accuracy. The best performance corresponds to $\alpha = 2.3$. Thanks to the high fidelity of the simulated data, we may use this parameter setting to process the real experimental data.

3.2.3 Grating Interferometry

This experiment demonstrates that *syris* can be used for simulations of X-ray imaging experiments which exploit specialized X-ray optics, in this case gratings. We will describe common approximations of the image formation process and show their regions of validity for various sample types.

X-ray grating interferometry [85] (XGI) uses gratings to provide multiple contrasts. We will consider an XGI setup with two gratings, one for changing the phase of the wavefield called *phase grating* (PG) and the other for blocking parts of the wavefield just before the detector called absorption grating (AG). The technique is illustrated in Figure 3.23. We will consider rectangular gratings with some period p. This means that the wavefield hits the grating for $p/2$ and for a consequent $p/2$ it doesn't. XGI is based on the self-imaging [86] effect of such a grating, i.e. at a special distance, called the Talbot distance, the distorted free-space propagated wavefield repeats itself. However, this is not of much use for us because when the wavefield of PG is repeated there is no intensity. Interestingly, at fractions of the Talbot distance the wavefield repeats itself in such a way that the periodic phase shift is converted to periodic intensity change, as shown in Figure 3.24.

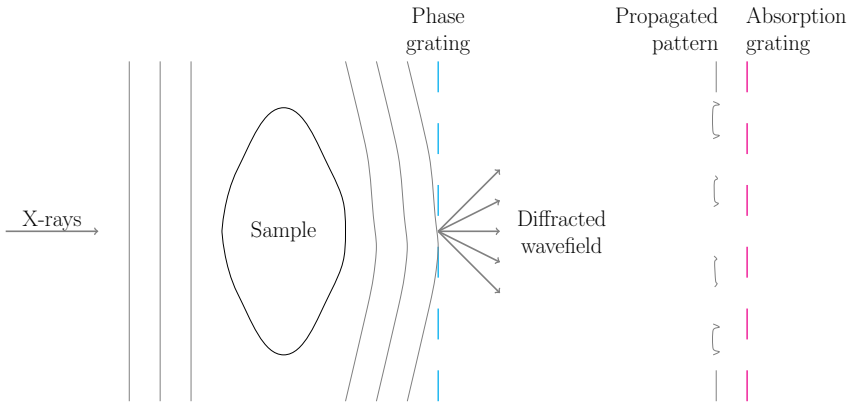


Figure 3.23: Grating interferometry setup in 2D. Incident wavefield passes through the sample, propagates to the phase grating where it is diffracted and the propagation continues to the absorption grating. One of the gratings is shifted along their periodicity which scans the propagated pattern. Such a scan in one pixel is called stepping curve and we can obtain three contrasts from it: absorption, differential phase and visibility reduction, often referred to as dark field. Absorption contrast is the mean of the stepping curve, differential phase is obtained from the shift of the propagated pattern (see the shifted grating periods in the propagated pattern) and the stepping curve visibility gives us information about the blurring caused by the sample. Here we emphasize blurring caused by the diffraction on sample edges visible as small oscillations in the propagated pattern.

We will use the stepping technique to be able to extract various contrasts, which means one of the gratings is shifted up to some periods along the grating periodicity direction (in our case the x-axis) and the intensity pattern is recorded at different such shifts. This means that the lamelas of the AG block different parts of the

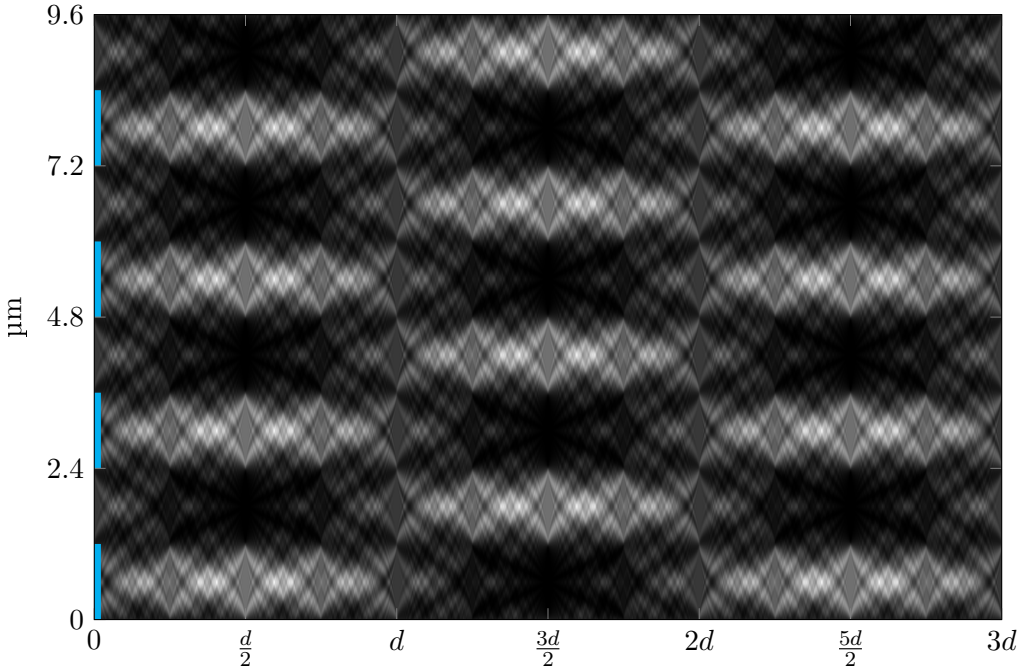


Figure 3.24: Talbot carpet, view along the optical axis from top. Phase grating which shifts the wavefield by $\pi/2$ is depicted by the blue dashed line. Intensity of the wavefield behind the grating propagated to certain distance is shown in image columns. d is the self-imaging Talbot distance. Since a phase grating doesn't change the intensity, we cannot see it at d. However, the phase modulation is transformed to intensity modulation at fractions of Talbot distances. As we can see the carpet is periodic and we can use odd multiples of the fractional Talbot distance in an experiment.

self-image. If the detector pixel size is larger than p, we will not see the lamelas but rather the integrated intensity over a region given by the pixel size and since the intensity pattern is periodic, so is the gray value in one pixel for such a stepping with period being the number of steps per one grating period (see Figure 3.25). We will call the intensity in one pixel at different stepping positions a *stepping curve*.

If we now place a sample into the X-ray light path, it will further change the wavefield and cause deviations from the self-image. First, it will attenuate the self-image based on the imaginary part of the complex refractive index and sample thickness. This is the classic absorption contrast (AC) which we would see even without XGI (compare Figure 3.26b and Figure 3.27a). In this case it is simply the average of the stepping curve. More interesting is the differential phase contrast (DPC), which is the derivative of the phase shift on the sample with respect to the grating periodicity direction. This contrast is made visible by the shift of the self-image with respect to the reference (self-image without the sample) caused by phase variations in the sample. We can easily extract this shift from the phase shift of the stepping curve. The third contrast is given by the reduced amplitude of the

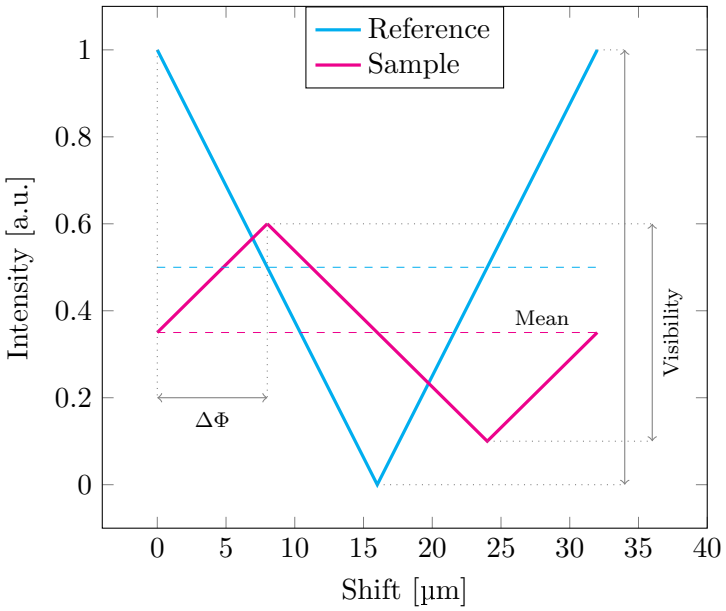


Figure 3.25: Grating interferometry stepping curve for gratings with period $p = 32\,\mu\text{m}$. Sliding one grating along its period blocks parts of light passing through the other which allows only fractions of light to reach the detector. When the shift is 0, light which comes through the first grating passes completely through the second grating, yielding maximum intensity (see the cyan reference curve). No light passes for shift $p/2$ because the gratings are in position where they block their openings completely. Phase shift introduced by the sample deflects the beam which is made visible as a shift of the grating intensity pattern. Thus, the phase stepping curve is shifted as well and we can reconstruct DPC. Absorption in the sample causes the reduced mean of the curve from which we reconstruct AC. Reduced amplitude of the curve is the result of self-image blurring and is used for DFC extraction.

stepping curve, often called visibility reduction or *dark field contrast* (DFC) and its origin is discussed below.

We can use the Fourier transform of the stepping curve to reconstruct the three contrasts in every pixel. If S is the stepping curve with the sample in the FOV and R is the stepping curve without it, the contrasts are

$$\begin{aligned} \text{AC} &= \log\left(\frac{\bar{R}}{\bar{S}}\right) \\ \text{DPC} &= \frac{p}{2\pi d}\left(\text{Arg}\left[R(m)\right] - \text{Arg}\left[S(m)\right]\right) \\ \text{DFC} &= 1 - \frac{|S(m)|\bar{R}}{|R(m)|\bar{S}}, \end{aligned} \tag{3.32}$$

where m is the number of scanned periods and the bar sign represents mean value.

The dark field contrast has lately attracted a lot of attention [87, 88, 89]. It has often been assumed that it is caused by the ultra small angle X-ray scattering

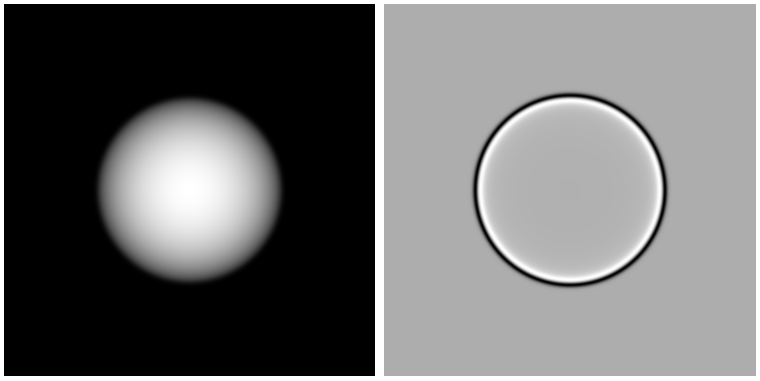
(USAXS), which blurs the resulting self-image. However, this is not the only source of the DFC. Imagine a wavefield traveling through a sphere. As a consequence of refraction, the sphere acts as a lens and can locally change the self-image period, which may give rise to fake AC and DFC due to the mismatch of the AG period and the self-image period. In fact, the situation is even more complicated, because we cannot neglect free-space propagation and the affiliated intensity oscillations at sample edges, as we have seen in Figure 2.6. These oscillation may again give rise to fake AC or DFC.

Since there might be various physical mechanisms involved behind any of the three contrasts, in the following we refer to them in terms of their definitions from (3.32) and not in terms of any physical mechanism.

2D Sphere

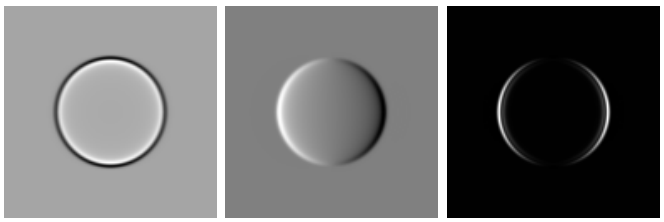
Our first example will be a 2D XGI simulation of a sphere smoothened by a Gaussian PSF with full width half maximum (FWHM) of four grating periods in order to suppress the response from the edges. We will use perfect rectangular gratings with period 2.4 µm, place the absorption grating 225 mm from the phase grating, which is the 5^{th} fractional Talbot distance (the distance when PG is repeated in terms of intensity). Phase grating shifts the phase of the perfectly monochromatic wavefield with energy 19.4 keV by $\pi/2$. The complex refractive index of the sample for the given energy is $n = 7.0845374 \times 10^{-7} + 0i$, i.e. the sample causes no absorption and the detectable intensity stems solely from the free-space propagation. The real part of the refractive index corresponds to polymethyl methacrylate (PMMA), which is a common material.

We use a monochromatic plane incident wave which interacts with the sample and the phase grating without free-space propagation between them (the sample is on the PG). Such wavefield is then propagated to the absorption grating which is assumed to be placed right before the camera sensor (again no propagation between them). We don't take into account scintillator effects and the intensity of the propagated wavefield is then masked by the stepping of the AG shifted over one grating period in 16 equidistant steps. Every intensity pattern is blurred with the PSF of the detector, which FWHM is $2p$ and this is also the camera pixel size. Such blurred pattern is then binned to the camera pixel size. We repeat this procedure also without the sample to create the reference scan and then we apply the reconstruction given by (3.32) to obtain the three contrasts depicted in Figure 3.27.



(a) (b)

Figure 3.26: (a) Smoothened sphere's projected thickness, (b) absorption contrast of the wavefield modified by the sample transmission function and propagated to 225 mm. The sample doesn't cause any absorption so the contrast stems purely from the free-space propagation.



(a) (b) (c)

Figure 3.27: Step scan of a sphere where the detector pixel size is two-times larger than the grating period. (a) absorption contrast, (b) differential phase contrast and (c) dark field contrast. Absorption contrast is mostly insensitive to the grating orientation because it is the average value of the step scan, thus the grating interaction is suppressed. On the other hand, the other two contrasts are formed thanks to the gratings and the step scan, that is why they depend on the grating periodicity direction, which is the x-axis in this case. Both contrasts disappear at the top and bottom of the sphere because the phase difference caused by the object is weak along the x-axis here.

Z-dependence

In this example we will show how the three contrasts develop as a function of the sample position which is placed between the gratings, starting from very close to the AG up to when it is placed on the PG and then upstream the PG (closer to the X-ray source than the PG). When between the gratings, the wavefield from the PG is propagated to the sample, modified by its transmission function and further propagated to the AG. When the sample is placed on the PG, like in the previous example, there is no propagation of the wavefield between the PG and the sample. Experimental conditions are the same as before, except the computations are done in 2D because of the tremendous oversampling required when we want to take high diffraction angles of the sample into account. In this case, the grating period is represented by 1024 pixels, i.e. the pixel size is 2.343 75 nm. The grating is smoothened by a Gaussian with FWHM 128 pixels. We use 2^{21} pixels to cover FOV of 4.9 mm. Obviously, this would not be computationally tractable for a 2D case because we would need 32 TiB of memory to represent a wavefield with single-precision floating point complex pixels. Unfortunately we cannot use the technique to reduce the interference region described in Section 3.1.8 because we need to resolve the large diffraction angles of the sample in order to properly compute the propagated intensity on the AG. Nevertheless, XGI setup is sensitive only along one dimension, thus we can use 2D objects with 1D projected thickness for our study. We will show the z-dependence on two samples, first of which introduces smooth phase changes and the second abrupt changes.

Before we do the actual simulations we will derive how the contrasts should look based on sample properties and some physical approximations. When we then compare the derived values with the simulations we will be able to assess how good particular approximations work for particular samples. Moreover, the deviation of the approximation from the simulated results will give us insight into which physical mechanisms are important for a particular contrast and if simplified models can be used to predict the contrast, as described below.

Since the phase of the wavefield behind the sample $\phi(x)$ is not constant it gives rise to AC which can be approximated by using the transport of intensity equation (TIE) [90]

$$\nabla_\perp \left(I(\vec{x}, z) \nabla_\perp \phi(\vec{x}, z) \right) = \frac{-2\pi}{\lambda} \frac{\partial}{\partial z} I(\vec{x}, z). \tag{3.33}$$

If we take into account sufficiently small propagation distance Δz, the partial derivative of intensity at $z = 0$ with respect to z can be approximated by

$$\frac{\partial}{\partial z} I(\vec{x}, 0) \approx \frac{I(\vec{x}, \Delta z) - I(\vec{x}, 0)}{\Delta z}. \tag{3.34}$$

If we consider the fact that the sample is a pure phase object, i.e. $I(\vec{x}, 0) = 1$, equation (3.33) simplifies to

$$1 - \frac{\Delta z \lambda}{2\pi} \Delta_\perp \phi = I(\vec{x}, \Delta z). \tag{3.35}$$

As we can see, AC grows linearly with Δz. Of course this approximation is limited by the smoothness of the sample phase and short propagation distances between the sample and AG because these properties together govern the manifestation of additional intensity fringes which simply cannot be modeled by this approximation. Examples are discussed in Figure 3.32 and in Figure 3.39.

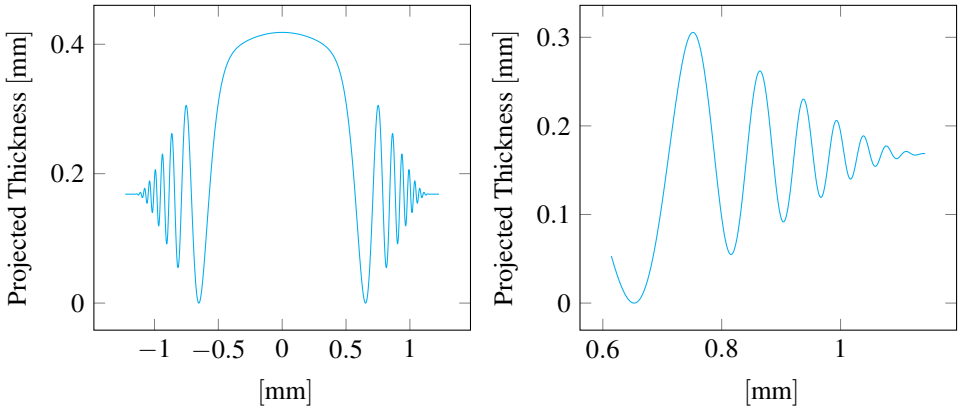
$\nabla\phi(x)$ causes the grating pattern to shift which is reflected in the DPC. Since DPC converts this shift to the beam deflection angle in the sample, the signal is constant over z. However, we must not forget that to correctly reconstruct the DPC the phase changes must be smooth, which is satisfied for smooth-phase samples when they are placed between the gratings. On the other hand, if we place even a smooth-phase sample far upstream the PG, the propagated phase on the PG will differ from the one in the sample and the reconstruction will contain artifacts. Examples of the DPC are given in Figure 3.33 and in Figure 3.40.

The $\Delta\phi(x)$ makes the propagated grating period change which in turn gives rise to DFC. To some degree it can be approximated by neglecting the higher diffraction angles. We do this by taking into account intensity of the propagated PG which is then shifted according to the beam deflection caused by the sample, i.e. we use the Snell law to compute the shifted self-image of the grating. Comparison of the DFC stemming from light propagation and the Snell approximation is discussed in Figure 3.34 and Figure 3.41.

Smoothly Varying Phase

Our first sample's projected thickness is based on $\cos(x^5)$ which increases the phase frequency, as shown in Figure 3.28. An important aspect of the sample is that it doesn't include any sharp transitions and the phase varies smoothly.

We will first show the sample geometry, then the three contrasts as a function of the sample and AG distance and last again the contrast signals, but selected for some points along the x-axis and compared with the approximations described above.



(a) (b)

Figure 3.28: $\cos(x^5)$ based projected thickness in (a) and its crop in (b). This geometry is interesting because it causes smooth phase variations with increasing frequency.

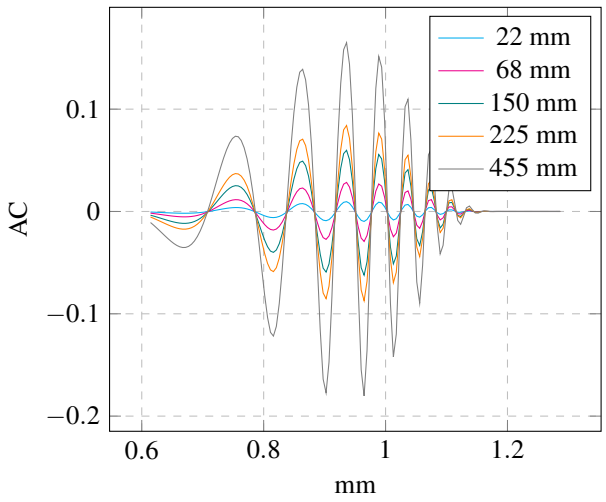


Figure 3.29: AC from the step scan for the sample in Figure 3.28. Various sample distances from the AG are shown in different colors. Interference pattern caused by the free-space propagation develops over distance.

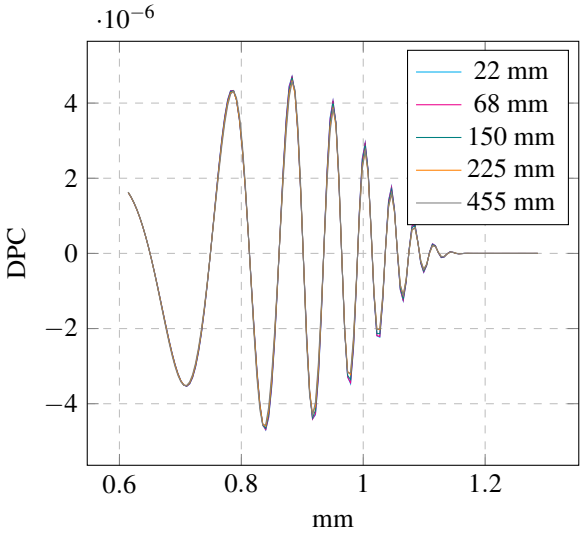


Figure 3.30: DPC from the step scan. Various sample distances from the AG are shown in different colors. The signal doesn't change along z because it reflects the beam deflection angle given by the sample. This in general doesn't hold for the case when the sample is upstream the PG but in our case the phase varies slowly with propagation so the phase of the wavefield propagated from the sample to the PG is almost the same as the phase of the wavefield right downstream the sample.

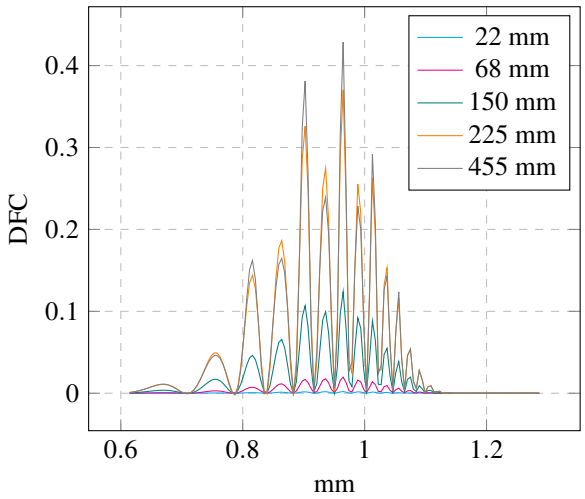


Figure 3.31: DFC from the step scan. Various sample distances from the AG are shown in different colors. The further the sample is from the AG the more the wavefield distortion manifests via free-space propagation, which leads to reduced visibility and in turn greater DFC. Since DFC is related to the phase, it doesn't change much when the sample is upstream the PG because the propagated phase is similar to the original one, as described in Figure 3.37.

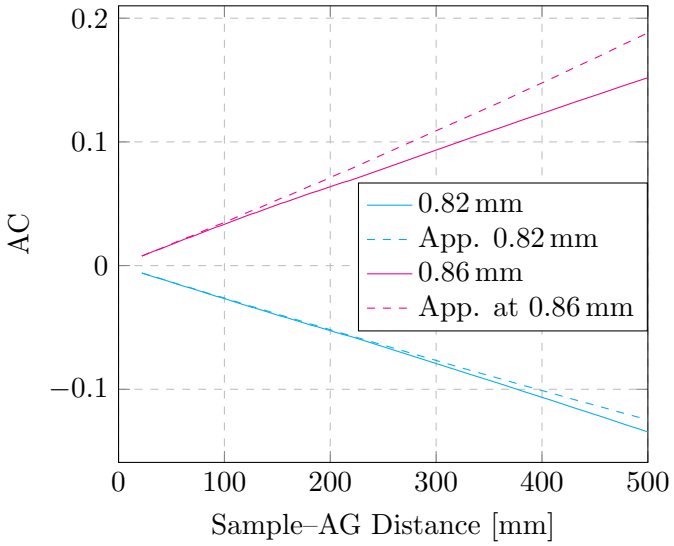


Figure 3.32: AC z-dependence for two neighbouring peaks in the AC. Solid lines are the values reconstructed from the step scan and the dashed lines are the TIE-approximations with no gratings taken into account, i.e. the sample is propagated to the AG and decimated to match the detector pixel size. The sample is rather smooth with no abrupt transitions, thus this approximation holds quite well even for larger distances.

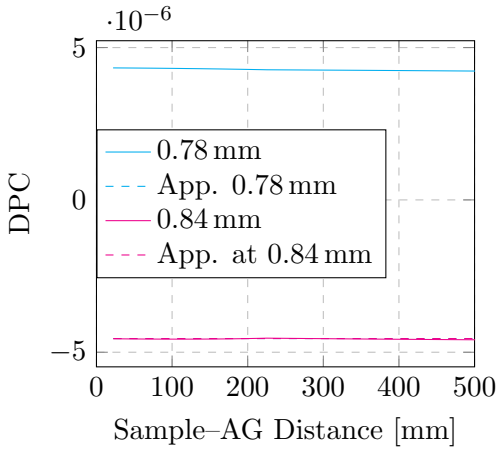


Figure 3.33: DPC z-dependence for two neighbouring peaks with different sign in the DPC. Solid lines are the values from the step scan and the dashed lines are computed from the sample phase. The reconstruction and computed values are so close because the phase of the sample doesn't change much with propagation.

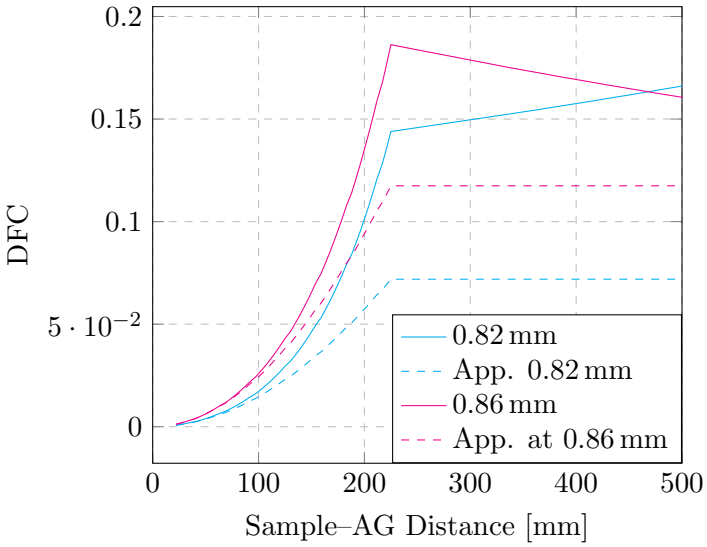
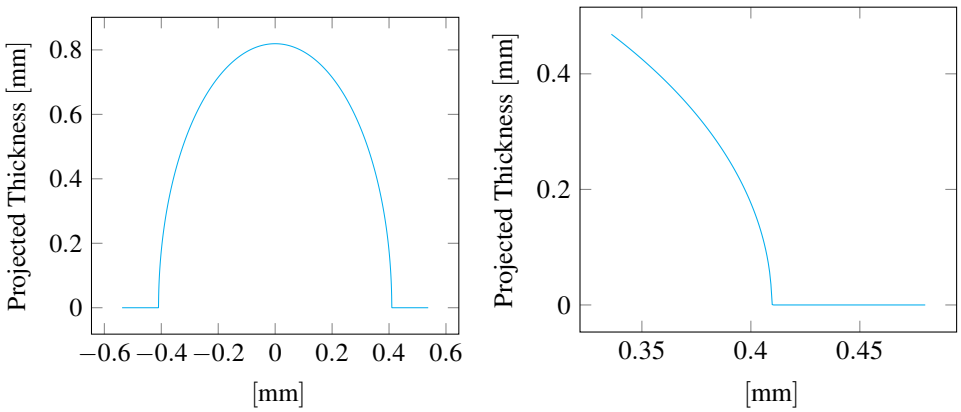


Figure 3.34: DFC z-dependence for two neighbouring peaks on DFC. Solid lines are the values from the step scan and dashed lines are the DFC values from a step scan when propagation was exchanged by shifted intensity pattern of the PG. The shift was computed by Snell law from the sample phase. As we can see, the approximation follows the reconstruction but it isn't a perfect fit, i.e. the DFC doesn't stem only from the period change but also the higher diffraction orders play a role.

Sphere

Unlike the sample before, we will now use a sphere with only a slightly smoothened edge to introduce strong wavefield perturbation. The smoothening is done by a Gaussian PSF with FWHM 0.125 of the grating period. As we will show further, abrupt phase transition introduced by such samples has large impact on the contrast formation, and more importantly, on the validity of the approximations one makes when tries to describe the contrast formation.



(a) (b)

Figure 3.35: Projected thickness of a smooth sphere in (a) and its crop in (b). The sphere is convolved with a Gaussian PSF with FWHM 0.125 grating period, which suppresses sharp edges. Nevertheless, the phase transition is still quite abrupt and gives rise to a strong DFC.

As we have seen, the image formation process for the XGI method is quite complex even when many imaging modalities are not taken into account (e.g. coherence properties, grating imperfections, ...) and it is hard to approximate the contrast formation, however not impossible within certain limits. Simulations which we made here show that the three contrasts depend differently on the sample position and for samples with smooth phase variations the AC grows linearly, DPC stays constant even for the case when the sample is upstream the PG and DFC grows nonlinearly.

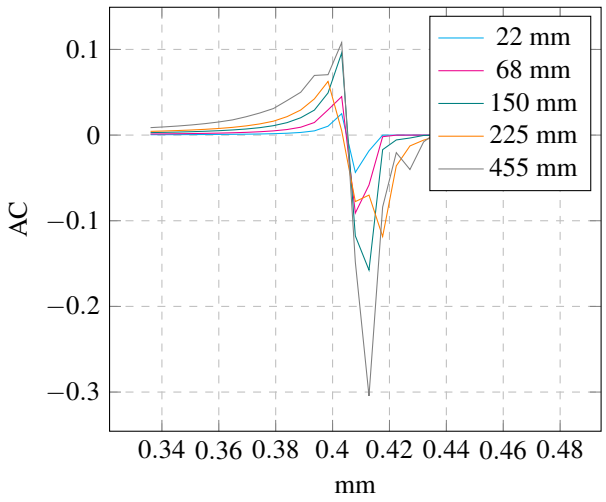


Figure 3.36: AC from the step scan for the sample in Figure 3.35. Various sample distances from the AG are shown in different colors. Sample edge is enhanced by typical free-space-propagation-induced, positive-negative alternations.

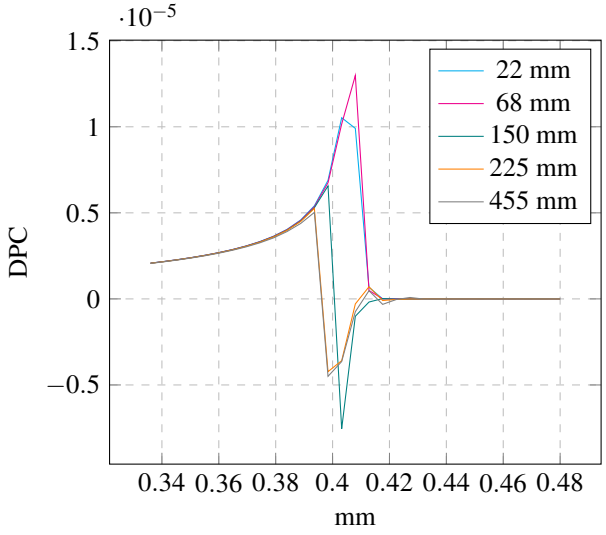


Figure 3.37: DPC from the step scan. Various sample distances from the AG are shown in different colors. The phase shift can be retrieved only within $\pm\pi$ which is often a problem, especially for samples with large phase shift, like our sphere. The reconstructed DPC is *wrapped* and needs to be further treated to represent the sample.

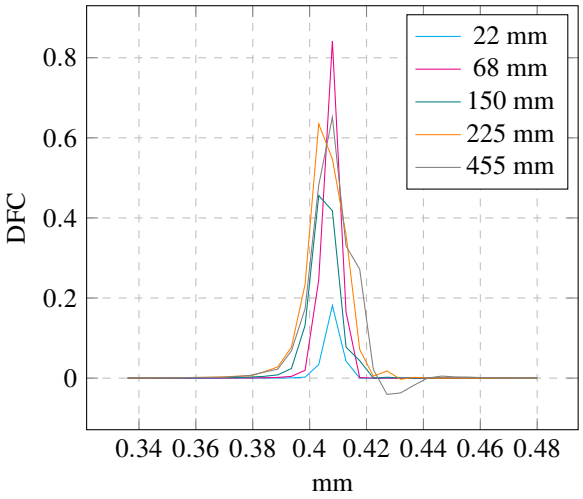


Figure 3.38: DFC from the step scan. Various sample distances from the AG are shown in different colors. The strong phase transition at the edge makes the beam deflect away from one detector pixel, i.e. there is no intensity in such a pixel, thus the visibility goes to 0 and DFC is almost saturated.

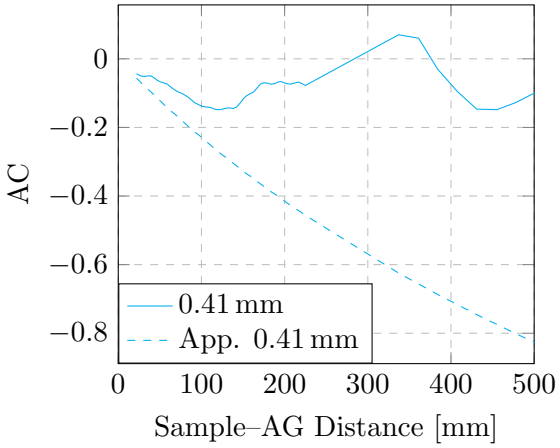


Figure 3.39: AC z-dependence at the sample edge. Solid line are the values reconstructed from the step scan and the dashed line are the TIE-approximations with no gratings taken into account, i.e. the sample is propagated to the AG and decimated to match the detector pixel size. As we can see, the TIE-approximation is no longer valid because the sharp edge makes the sample phase oscillate as it propagates, which in turn gives rise to additional intensity fringes. These fringes cancel out in the detector pixel because they are too close to each other and the total recorded intensity doesn't change much.

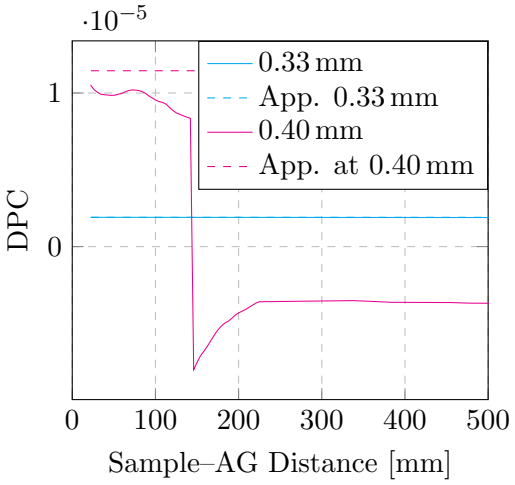


Figure 3.40: DPC z-dependence for two points of the sample, one in the region with small phase change (cyan) and another on the sample edge (magenta). Solid lines are the values from the step scan and the dashed lines are computed from the sample phase. The reconstruction and computed values show good agreement for the region with smooth phase transition close to the center of the sphere. However, the phase at the edge is wrapped, made visible by the sharp transition in the plot.

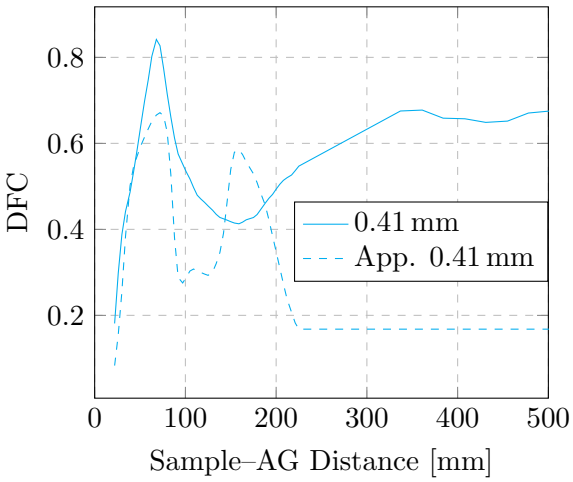


Figure 3.41: DFC z-dependence at the sample edge. Solid line are the values from the step scan and dashed line are the DFC values from a step scan when propagation was exchanged by shifting the self-image based on the sample phase and Snell law. The signal profile is different than before with its maximum when the sample is between the gratings. Moreover, the approximation fails sooner than before because the free-space propagation contributions from the surrounding area of the edge cannot be neglected and they lead to complex intensity patterns which cannot be easily approximated.

3.3 Summary

As we have seen in the examples above, based on a close approximation of a sample or a process and high-fidelity simulation of the image formation process, we can use synthetic data to select optimal data processing parameters for the analysis of the real image data. Moreover, we can investigate the impact of various imaging conditions on the data processing pipeline and use the findings to select the optimal combination of the experimental and data processing parameters, which can be used for the initialization of a real experiment.

We have also seen that simulations are useful to investigate advanced imaging methods and validate approximations of the image formation process. Thus, they can be used by the image processing community to test hypotheses about new imaging methods and to develop and benchmark new reconstruction and analysis algorithms.

Chapter 4

Low Latency 3D Material Structure Reconstruction

In this section we address research Question 2: *How to speed up 3D reconstruction so that it can be used to drive an experiment?*

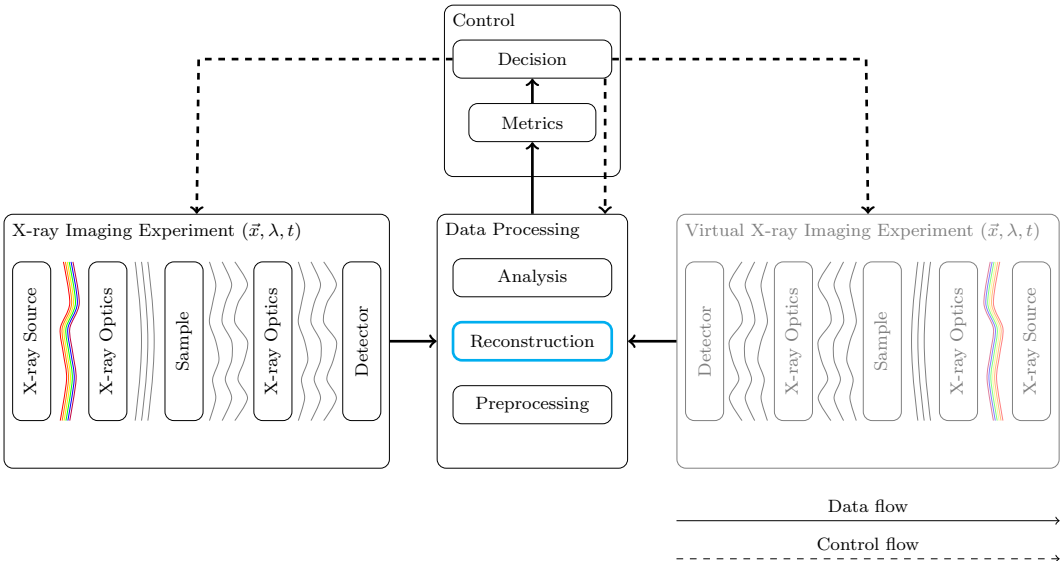


Figure 4.1: Our focus in this chapter, fast 3D *reconstruction* highlighted in cyan.

The 3D reconstruction within the scope of our system is highlighted in Figure 4.1. If we want to speed it up in such a way that its output can be used for *online* data analysis, we need to consider the fact that the projections come to the data processing pipeline *one-by-one* and not all at the same time. Thus, we can start the reconstruction before the projection acquisition finishes. The time which defines how fast we can react to the measurement process based on the 3D reconstruction is given by the time an algorithm needs to update the already partially reconstructed

3D volume from a projection or a small subset of them. We will call this time *latency* and we will seek an algorithm and its implementation which minimize it.

To do this we will first describe various 3D reconstruction algorithms. They use different mathematical approaches which define their time complexity and that in turn defines the lowest achievable latency. Once we find an algorithm which minimizes latency, we need to investigate the performance optimization possibilities, so that we can provide a fast implementation and integrate it into our data acquisition scheme from Chapter 1 and use it for providing 3D reconstruction-based feedback to the experiment.

4.1 3D Reconstruction Algorithms

There are more ways to reconstruct tomographic and laminographic data sets, all of which have their advantages and disadvantages. We will now seek a technique which enables fast reconstruction with low latency, so that the 3D volume can be used to drive the course of an experiment. We will explain basic tomographic reconstruction techniques and select the most suitable one for our needs. We will then shortly explain its extension for the laminographic case.

4.1.1 Algebraic Reconstruction

We can consider the discrete case of (2.16)

$$\mu p\left(i\right) = \sum_{w} \Delta x \mu \left[w\cos\phi, w\sin\phi\right], \tag{4.1}$$

where i is the pixel position in the projected row, discrete x and y are obtained from the polar coordinates explained in Figure 2.8 and based on the rotation angle ϕ, Δx is the pixel size, the situation is depicted in Figure 4.2. If we consider all projection angles we come to a large linear system of equations. If N is the detector row width and M is the number of projections, the system has $N \times M$ equations just for one slice. We can try to solve these equations algebraically [35, 36, 37, 38], but they can have infinitely many solutions, so we cannot guarantee that our reconstruction is correct. On the other hand, we might incorporate some knowledge about the sample and experimental setup to obtain more accurate results and speed up the computation. Common iterative algebraic reconstruction techniques (ART) are however too computationally expensive (time complexity $\mathcal{O}\left(kMN^2\right)$) for being employed in a low-latency feedback system. k in the time complexity is the number of iterations.

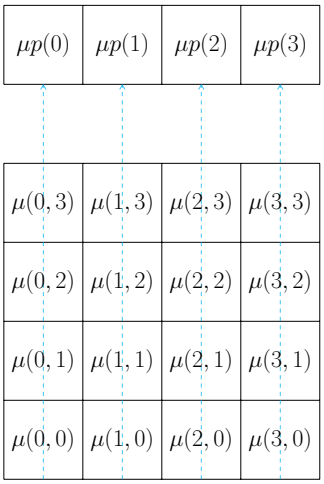


Figure 4.2: A discrete object slice and its projection onto a detector row $\mu p(i)$. All projections form NM linear equations, where N is the detector row width and M is the number of projections.

4.1.2 Fourier-based Reconstruction

3D reconstruction can be also done by using the Fourier slice theorem [26]. We will need the 2D Fourier transform with spatial frequencies u and v

$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-2\pi j(xu+yv)}\,\mathrm{d}x\mathrm{d}y. \tag{4.2}$$

Further, we will make use of the polar representation of the Fourier space, thus we can represent a line across the space which crosses zero frequency by a frequency w and an angle, let's use the tomographic ϕ. Thus, we obtain the spatial frequencies from the polar space by $u = w \cdot cos\phi$ and $v = w \cdot sin\phi$ and we can rewrite (4.2) as

$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-2\pi wj(x\cos\phi+y\sin\phi))}\,\mathrm{d}x\mathrm{d}y. \tag{4.3}$$

Fourier transform uses simple *rotated* 2D sine and cosine basis functions given by the complex exponent to transform a function from real space to frequency (Fourier) space. The important property of the basis functions is that they are essentially one dimensional, only extended to the second dimension and rotated. When we perform a Fourier transform, it does not matter if we rotate our sample $f(x,y)$ or the basis functions.

Let's now compute $F(u,v)$ along a line in the Fourier space given by its polar representation defined in (4.3). It is crucial to realize that the X-ray direction given by the tomographic angle ϕ is perpendicular to the oscillation direction of all basis functions along the line in the polar space given by ϕ, as shown in Figure 4.3. This means that these *basis functions along the X-ray direction are constant* and if we

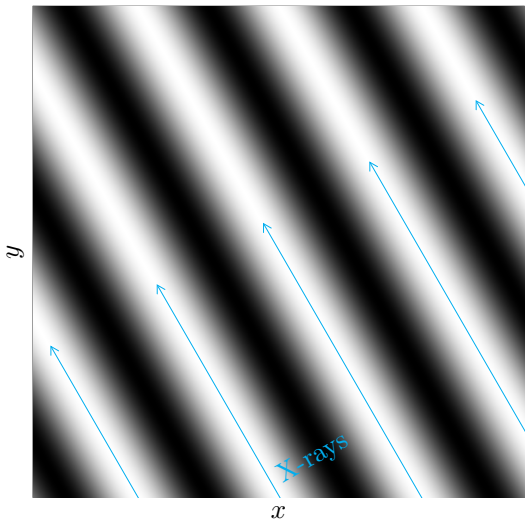


Figure 4.3: Fourier slice theorem illustration. The basis functions $e^{-2\pi wj(x\cos\phi+y\sin\phi)}$ of the 2D Fourier transform are essentially one dimensional, only extended to the second dimension and rotated, in this case $\phi = -30°$ and $w = 4$. This, combined with the fact that we may exchange the rotation of the basis functions for the sample rotation, means that the X-ray integration for angle ϕ is perpendicular to the periodicity of basis functions rotated by ϕ. Since the basis functions in this direction are constant for one ray, we can directly use the fact that the X-ray beam integrates our sample in this direction and we can compute a line in the 2D Fourier space of a slice by 1D Fourier transform of the projected row. If we acquire projections for all ϕ required to populate the 2D Fourier space of a slice, we will be able to reconstruct the sample by simply performing the inverse 2D Fourier transform.

want to compute the Fourier transform along our line we only need to integrate the sample in this direction, which is what the X-rays do for us, and perform one dimensional Fourier transform with respect to w. If we acquire tomographic projections for angles up to 180°, we will populate the whole 2D Fourier space and we can reconstruct our sample.

More formally, the sample projection for a rotation angle ϕ is

$$p_\phi(x) = \int_{-\infty}^{\infty} f\left(x', y'\right)\, \mathrm{d}y \tag{4.4}$$

where x' and y' are rotated coordinates given by the tomographic angle ϕ

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}. \tag{4.5}$$

Fourier transform of a projection is $S_\phi(w) = \mathcal{F}[p_\phi(x)]$.

Let's exchange the basis function rotation by sample rotation and set $\phi = 0$ for the basis functions, thanks to which we can come to the following relation between

the projection $p_\phi(x)$ and the 2D Fourier transform of the slice $F(u, v)$

$$\begin{aligned}
F(u,v) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-2\pi j(xu+yv)}\,\mathrm{d}x\mathrm{d}y \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-2\pi wj(x\cos\phi+y\sin\phi)}\,\mathrm{d}x\mathrm{d}y \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x',y')e^{-2\pi jxw}\,\mathrm{d}x\mathrm{d}y && \phi = 0 \text{ for the basis functions} \\
&= \int_{-\infty}^{\infty}\left(\int_{-\infty}^{\infty} f\left(x',y'\right)\,\mathrm{d}y\right)e^{-2\pi jxw}\,\mathrm{d}x \\
&= \int_{-\infty}^{\infty} p_\phi(x)e^{-2\pi jxw}\,\mathrm{d}x = S_\phi(w) && \text{we used (4.4).}
\end{aligned} \tag{4.6}$$

We can now apply the inverse Fourier transform and obtain the slice of our sample. Although straightforward and much faster than ART, we need to perform a 2D Fourier transform every time we add a projection to the Fourier space. Its time complexity is $\mathcal{O}\left(N^2\log(N)\right)$ and for low latency reconstruction it would be more beneficial to have an algorithm which can update the reconstructed slice based on every incoming projection without the need to invert the Fourier space every time.

4.1.3 Filtered Back Projection

Filtered Back Projection (FBP) algorithm enables updating the volume without the need of transforming the 2D Fourier space for every accounted projection. It is based on performing the inverse Fourier transform

$$f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)e^{2\pi wj(x\cos\phi+y\sin\phi)}\,\mathrm{d}u\mathrm{d}v \tag{4.7}$$

using the polar coordinate system. When we change the integration limits we need to consider the Jacobian determinant of the conversion between the two spaces

$$|J| = \begin{vmatrix} \dfrac{\partial u}{\partial w} & \dfrac{\partial u}{\partial \phi} \\ \dfrac{\partial v}{\partial w} & \dfrac{\partial v}{\partial \phi} \end{vmatrix} = \begin{vmatrix} \cos\phi & -w\sin\phi \\ \sin\phi & w\cos\phi \end{vmatrix} = w\cos^2\phi + w\sin^2\phi = w. \tag{4.8}$$

Thus, we need to apply a linear correction factor given by the frequency w, which is where the "filtered" part of the algorithm name comes from. The integration is now

$$f(x,y) = \int_0^{2\pi}\int_0^{\infty} w\,F(w,\phi)\,e^{2\pi wj(x\cos\phi+y\sin\phi)}\,\mathrm{d}w\mathrm{d}\phi\,. \tag{4.9}$$

Splitting the angular integral into two halves and using the fact that the Fourier transform of a real signal is symmetric allow us to rewrite this integral as

$$f(x,y) = \int_0^{\pi} \left(\int_{-\infty}^{\infty} |w|\, F(w,\phi)\, e^{2\pi w j (x\cos\phi + y\sin\phi)}\, \mathrm{d}w \right) \mathrm{d}\phi\,. \tag{4.10}$$

Finally, we can replace $F(w,\phi)$ by $S_\phi(w)$ and summarize that the filtered back projection algorithm processes projections by taking their 1D Fourier transforms, filters them with $|w|$ in the Fourier space, transforms them back to real space and looks up the value in the filtered projection at the rotated coordinate $x\cos\phi + y\sin\phi$ given by current ϕ. In other words, for one pixel x, y in a slice, the FBP looks up one pixel in every projection given by the rotation angle and sums up all such pixels. Another way to look at this is to imagine what happens to one projection for all points in one slice. It is simply "smeared" across the 2D slice under ϕ.

The discrete version of (4.10) for a pixel x, y, projection width N pixels, physical pixel size Δx and M such projections we can write

$$f(x,y) = \frac{\pi}{M\Delta x} \sum_{m=0}^{M-1} \sum_{k=-N/2}^{N/2} \left|\frac{k}{N}\right| S_m\left(\frac{k}{N}\right) e^{2\pi j \frac{k}{N}\left(x\cos\left(m\frac{\pi}{M}\right) + y\sin\left(m\frac{\pi}{M}\right)\right)}. \tag{4.11}$$

Even though the time complexity of FBP, $\mathcal{O}\left(MN(\log N + N)\right)$ is high, we can employ the fact that it takes time until the next projection comes when we want to reconstruct a data set on-the-fly. Thus, for one projection we have $\mathcal{O}\left(N(\log N + N)\right)$, which is lower than $\mathcal{O}\left(N^2 \log(N)\right)$ by the Fourier-based method. Therefore, we choose this algorithm for usage by experiment automation.

Filtered Back Projection For Laminography

We will use the laminographic back projection algorithm because it is more general and can be used by more experiments with no performance loss compared to the tomographic special case, as we will see later in Section 4.2.

The tilted rotation axis by laminography makes the reconstruction slightly more complicated than by the tomographic case. One slice of the sample is spread across multiple detector rows, which we consider by the following transformation matrix [16]

$$\begin{pmatrix} d_x \\ d_y \end{pmatrix} = \begin{pmatrix} \cos(\boldsymbol{\phi}) & 0 & \sin(\boldsymbol{\phi}) & c_x \\ \sin(\boldsymbol{\phi})\cos(\theta) & \sin(\theta) & -\cos(\boldsymbol{\phi})\cos(\theta) & c_y \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \\ 1 \end{pmatrix}, \tag{4.12}$$

which prescribes how to transform a 3D voxel in the reconstructed volume v_x, v_y, v_z into the detector pixel d_x, d_y. Offset c_x, c_y is used to define the origin, and thus the rotation axis with respect to the detector plane. Another difference for the laminographic case is that we need to acquire projections over 360° because the

projection at some angle ϕ is not the horizontally flipped projection at $\phi+\pi$ anymore. Last, we need to consider a modified filter $\sin\theta|w|$ [16]. Apart from these changes, the course of the algorithm is the same and can be described by

Algorithm 4.1: Laminographic back projection

Input: Filtered projections $P(m, d_x, d_y)$, laminographic angle θ, pixel size Δx
Output: 3D Volume

```
1 for v_x ← 0 to N − 1 do
2     for v_y ← 0 to N − 1 do
3         for v_z ← 0 to N − 1 do
4             for m ← 0 to M − 1 do
5                 φ = m 2π/M
6                 d_x = v_x cos φ + v_y sin φ + c_x
7                 d_y = v_x sin φ cos θ − v_y cos φ cos θ v_z sin θ + c_y
8                 V(v_x, v_y, v_z) += P(m, d_x, d_y)
9             V(v_x, v_y, v_z) *= 2π/(MΔx)
```

4.2 Algorithm Optimization

Once we have selected Algorithm 4.1 for experiment automation, it is time for its efficient implementation on a suitable hardware platform. We will focus only on the back projection part and not the Fourier transform required for the filtering because it has been extensively studied and there are libraries which provide its efficient implementation[1].

Back projection has very high time complexity $\mathcal{O}\left(MN^3\right)$. Fortunately, the computations for different voxels are independent of each other, thus the algorithm can be efficiently parallelized. Since GPUs have shown their potential in 3D reconstruction [10, 11], we support their utilization by implementing the algorithm in OpenCL. Moreover, it is not vendor-specific, as opposed to CUDA®, and it can run on other hardware platforms, e.g. CPUs. OpenCL is also used by the UFO framework [20] used for image processing at various synchrotrons. We will implement the algorithm within this framework because it already efficiently implements projection filtering and it will be easier for the users if they can use our algorithm from a well-established platform. Detailed information about the integration can be found in Chapter 5.

There are various aspects of the algorithm which need to be considered. It is bandwidth-limited, which means that there are relatively few computations compared to the amount of projection data that needs to be transferred to the kernel, which we address in Section 4.2.2. Another problem is the speed difference between the system main memory and the video memory. For instance, commonly

[1] `https://github.com/clMathLibraries/clFFT`

used DDR3-1600 memory transfer rate is $12.8\,\mathrm{GB\,s^{-1}}$, which is only a fraction of the $288.4\,\mathrm{GB\,s^{-1}}$, which can be achieved with the GDDR5 memory of NVIDIA® GeForce GTX Titan. This issue is very important especially for multi-GPU systems and will be discussed in detail in Section 4.2.3.

4.2.1 Performance Measurement Procedure

Throughout this section we will present various performance measurements for different versions of the developed algorithm, comparison of various video cards and platforms. Apart from latency, we will use giga updates per second GU/s [46] as the performance metric. Unlike data throughput in terms of $\mathrm{MiB\,s^{-1}}$, this metric takes into account the volume size *and* the number of projections. It is the number of additions to all voxels in the resulting volume per second divided by 1×10^9, where the number of additions is given by the number of projections. For example, a reconstruction of a $100 \times 100 \times 100$ volume from 1000 projections in 1 s results in 1 GU/s.

We use laminographic angle 65° and either OpenCL events for precise measurements of kernel durations or the system time when we take into account the data transfer to the video card. If not stated otherwise, all measurements are repeated 30 times, averaged and the standard deviation is shown by error bars.

4.2.2 Kernel Optimization

In this section we will specify basic strategy for increasing the throughput of Algorithm 4.1. Based on that we will be able to design the kernel itself, decide which loops will be handled by OpenCL, choose appropriate data structures and consider some strategies to decrease the number of required computations.

Number of Projections per Kernel Invocation

The highest performance impact has the number of processed projections per kernel invocation, we will call this number *burst* or B. If we want to reconstruct a volume of size N^3 from M projections, we need to read $M \cdot N^3$ pixels and update the volume $M/B \cdot N^3$ (let's for now neglect the fact that we actually need to read more pixels because of the interpolation). The factor M/B comes from the fact that we can write the value from all processed projections in the kernel to a temporary variable which resides in a register with much faster access time than the one to global memory. We update the global memory only once at the end of the kernel. Thus, to achieve the highest throughput we should maximize B, but of course we cannot actually reach M due to the limited size of global memory.

Typical data sets nowadays consist of projections with 2048×2048 pixels and if we want to acquire enough data to satisfy the sampling theorem, we need 6434 of them. If we take into account single-precision floating point numbers to represent the filtered values, we need more than 100 GiB of memory, which we nowadays don't

have. Even if we could read all the projections into memory, we wouldn't want to because then the GPU would need to wait until all the data is copied before it starts execution, which wastes time. Moreover, if we are concerned about latency and the DAQ speed is slow we want to choose small B because the GPU is idle most of time and the fewer the projections we upload, the sooner they will be processed. For fast DAQ we rather need to increase B to maximize the throughput.

Execution and Data Structures

Let's start with choosing the loops which we will let OpenCL to handle for us. The maximum number is three and we choose the first three loops which iterate over the volume because not all vendors support image arrays as kernel arguments, thus we couldn't access the input projections in the kernel. This leaves us to handle the loop over the projections, like in [46]. Due to the image array limitation, we provide separate kernels with B input images.

To ensure our kernel achieves the highest possible performance we need to consider its scheduling by hardware. Read operations from global memory typically take hundreds of cycles and subsequent arithmetic operations are postponed until the data is ready, which is a problem in our case because there are relatively few arithmetic operations per one read. Fortunately, read latency can be suppressed by having many subgroups per kernel invocation. GPU divides work groups into these subgroups and swaps between them based on data availability [65]. This way, when a subgroup comes to the point when it needs to wait many cycles to obtain data from global memory, the scheduler replaces it by another subgroup for which data has just been fetched. To ensure that there is enough work for the GPU between the data reads in our kernel, we need to maximize the number of subgroups.

The ratio between the number of subgroups which simultaneously reside on the GPU and the maximum number of subgroups which can potentially reside there is called *occupancy*. It is not only limited by how many threads we allow the GPU to execute at once but also by the amount of registers each thread requires, shared memory and other aspects. In our case the occupancy decreases with increasing B and the performance suffers. This places another upper limit on B. However, occupancy varies from one video card to another because of their different properties, so it is impossible to select a universal B or work group size which leads to the best performance on a given card. For example, on NVIDIA® GeForce GTX Titan based on the Kepler™ architecture, full occupancy can be maintained by using at maximum 24 projections per kernel, which means it is the best M/B ratio and it gives the highest throughput. Comparison with other burst modes is depicted in Figure 4.4.

Once we know the occupancy α and the time t_1 required to process one voxel in a kernel with $B_1 = 1$, we can estimate the performance of a kernel with burst B_x. By B_1 we need to read one projection pixel and write one volume voxel per thread. Since the read is cached it will take shorter than the write. Let's say the read takes $t_r = t_1/c$ time and let's neglect the arithmetic operations to compute the

pixel position. Thus, the write time is $t_w = t_1 \cdot (1 - 1/c)$. Using B_x projections in a kernel and decreasing the amount of write operations to the global memory by the same amount leads to kernel duration

$$t_{B_x} = \frac{B_x t_r + t_w}{\alpha} = \frac{t_1}{\alpha c} \left(B_x + c - 1 \right). \tag{4.13}$$

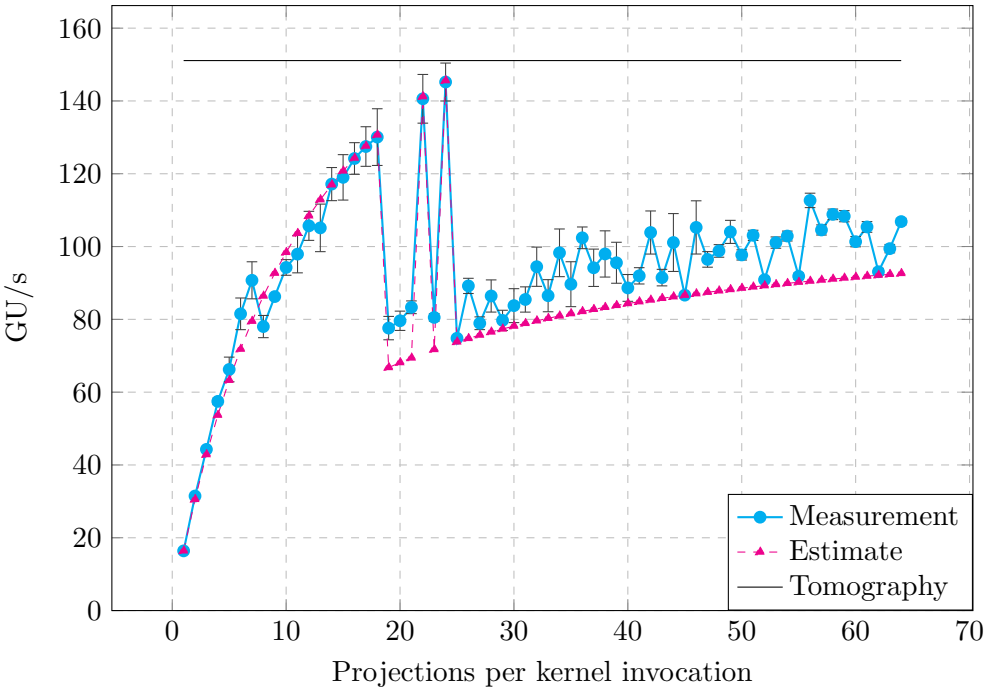


Figure 4.4: Measured GU/s (cyan) as a function of the number of projections per kernel invocation. The video card is NVIDIA® GeForce GTX Titan. Theoretical values (magenta) calculated based on (4.13) and $c = 13.5$ provide a good estimate about performance development. Tomographic back projection (black) has $M/B = 1$, thus presents the performance limit for laminographic back projection. Performance improves until burst 18 because the M/B ratio increases and the occupancy is at its maximum. Full occupancy between B 19 to 65 is reached only for $B = 22$ and $B = 24$. It dropped by half for the other bursts because the compiler used more registers per thread than allowed. The best performance at $B = 24$ is caused by the best M/B ratio.

Let's now consider which data structures and memory spaces are best suited for the data we need. When a GPU processes neighboring voxels in parallel it results in looking up neighboring projection pixels, i.e. the data we need to access is localized, thus is can be heavily cached. Caching can be leveraged by a specialized OpenCL image structure, which employs the texture memory of video cards. Moreover, as we can see from Algorithm 4.1, the resulting projection pixel is not an integer number and we need to interpolate between neighboring pixels. Nearest neighbor

interpolation is simple and leads to the look up of just one pixel but results in far less accurate results which is why we will use linear interpolation instead. Thanks to efficient texture memory caching and specialized texture fetch instructions, linear interpolation is where GPUs truly excel.

The kernel further requires the sine and cosine of the rotation angle for every processed projection. Since all threads need the same sine and cosine values for one projection and computation of the trigonometric functions is expensive, it is much faster to pre-compute them on the host side and use a lookup table than to compute them in the kernel. We can speed up the look up by using constant memory space, especially suited for situations when all threads read the same value because a read by one thread can be shared with neighboring threads, so the amount of reads is reduced and this memory space is also cached [64].

With the optimizations made in Section 4.2.2 and here, we were able to achieve 145.42 GU/s for $B = 24$. We also measured the performance of an efficient tomographic back projection algorithm[2] with $M/B = 1$, which causes the minimum number of writes to the global memory. The performance on the same system as we used for Figure 4.4 was 151.09 GU/s. This means that there is almost no penalty for generalizing tomographic back projection to laminography and the algorithm is well suited for both, high throughput and low latency reconstructions.

4.2.3 Data Transfer Optimization

Once we optimized the kernel code, it is time to investigate the overall performance including data transfer to the video card. Since the kernel requires a lot of data, we will try to decrease this amount and make the transfer as fast as possible.

As we have seen in (4.12), the required projection region depends on the volume shape, position with respect to the origin, tomographic and laminographic angle. These are all known values which we can calculate for every projection and upload only the relevant region to the video card, by which we can save a lot of data copying, especially for small laminographic angles. For instance, if the laminographic angle is 90°, which is the case of tomography, we need only one detector row for the reconstruction of one slice. The required amounts of projection data for reconstructing just one slice are shown in Figure 4.5.

If we want to reconstruct more slices, we can define a scale factor s which changes the volume shape in such a way that the amount of voxels stays the same. If we consider a projection region with dimensions $x \times y$, we can reconstruct volume $x \times y \times x$ (the volume respects the coordinate system where z is the beam propagation direction, so one slice has dimensions $x \times z$). The scaled volume is thus $x/s \times y s^2 \times x/s$. We can compare various s and their impact on the required amount of transferred projections for different laminographic angles and select the best s for a particular case. A 2D map $s \times \theta$ showing the amount of required data is shown in Figure 4.6.

[2] `https://github.com/ufo-kit/ufo-filters`

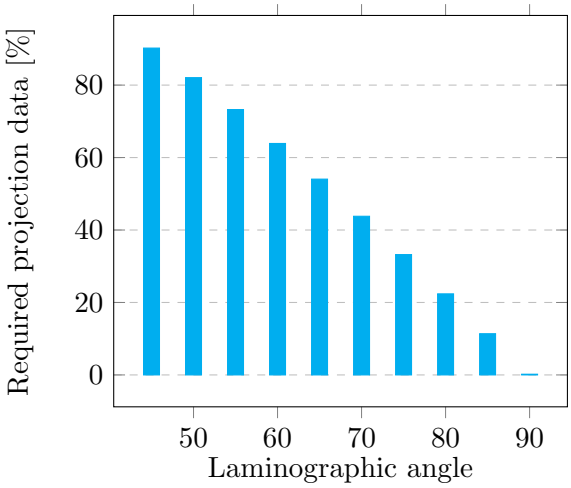


Figure 4.5: Ratio of the required projection data for back projection and its total amount for different laminographic angles. Projection size is 1024×1024 and volume size is $1024 \times 1 \times 1024$ (one slice). The greater the laminographic angle, the smaller projection region we need, i.e. that we can significantly reduce the amount of data transferred to a video card if we don't copy complete projections but only their relevant regions.

Time complexity of back projection grows linearly with the number of projections. This means that if we increase the number of projections processed per kernel then both the data transfer and computation times will increase proportionally, which doesn't help us. However, we can use the fact that the algorithm's time complexity grows with the square of the projection width. If we up-size the volume to $mx \times y \times mx$, the corresponding projection region is $mx \times y$, thus the ratio between the computation and the data transfer time grows linearly with $m \cdot t_k/t_c$, where t_k is the time needed to back project one projection pixel to one voxel and t_c is the time required to copy one projection pixel to the video card. This observation can be used to determine such volume shape which makes $t_k \geq t_c$, thus hides the time needed to transfer the projections if we use double buffering, i.e. the kernel works on one set of projections while the other set is being transferred. We can see the comparison between measured and computed t_k/t_c ratios in Figure 4.7.

Once we make sure that we don't copy the unnecessary data we want to maximize the upload speed to the video memory. We can achieve this by using page-locked [64] (pinned) memory. This kind of memory cannot be swapped out to the disk and can be used directly without any delays for data transfer to the video memory as opposed to the non-locked memory, which content first needs to be copied to a pinned buffer and only then transferred to the GPU. Whereas using pinned memory can speed up the data transfer by a factor of up to 3.7 (from $2.9\,\mathrm{GiB\,s^{-1}}$ to $10.8\,\mathrm{GiB\,s^{-1}}$) on a system with NVIDIA® video cards and DDR3-1600 main memory, an AMD system doesn't show any improvement with throughput $6.5\,\mathrm{GiB\,s^{-1}}$ in both cases. Latencies for various burst modes with data transfer optimization including copying

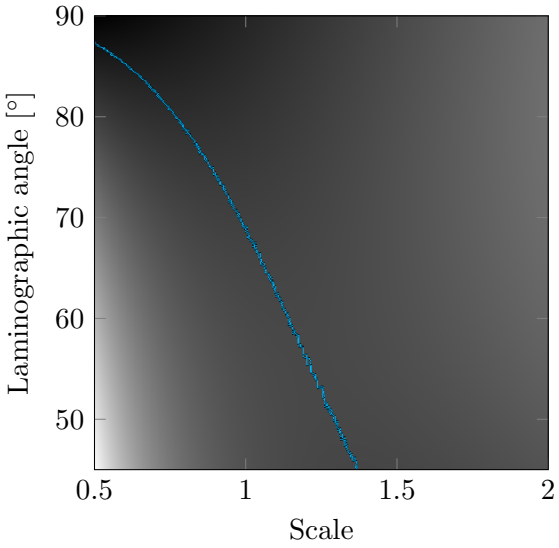


Figure 4.6: Optimal reconstructed volume shape as a function of a scaling factor and laminographic angle. Scaling factor shrinks or enlarges the width and height of a slice by the same amount and counteracts on the number of slices, so that the number of voxels stays the same. E.g. scale 2 changes the shape of a $1024 \times 1024 \times 1024$ cube to $512 \times 4096 \times 512$. A cube is best suited for laminographic angle 63.45°, for other angles it is worth slightly changing the volume shape.

only required projection parts and using pinned memory is depicted in Figure 4.9.

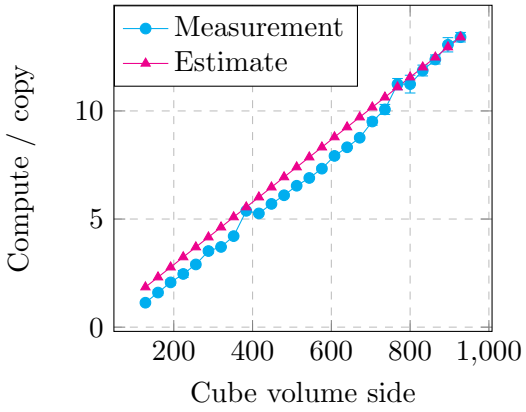


Figure 4.7: Measured and estimated ratio between the kernel computation time and the data transfer time. Measured time is the sum of the duration of the kernel and the data copy based on OpenCL events. Reconstructed volume was a cube. The linear $m \cdot t_k/t_c$ behavior of the ratio for the case when we extend the reconstructed region in the lateral x direction by m can be used to hide data transfers by overlapping computation and data transfer.

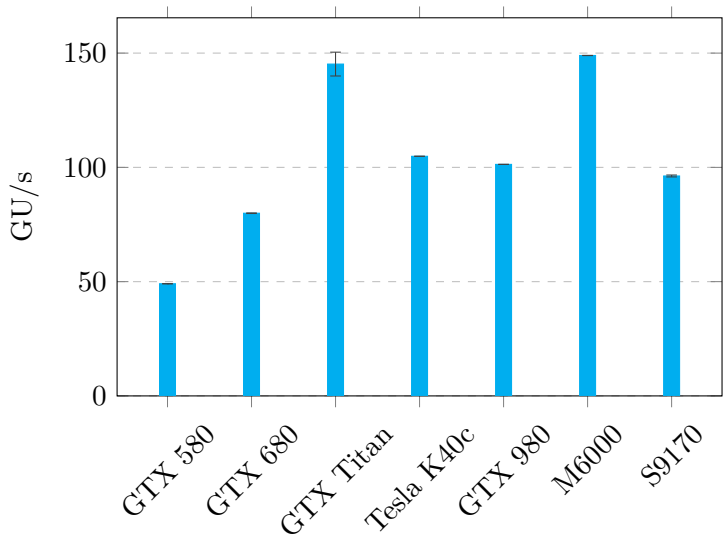


Figure 4.8: Performance comparison between various video cards from various manufacturers.

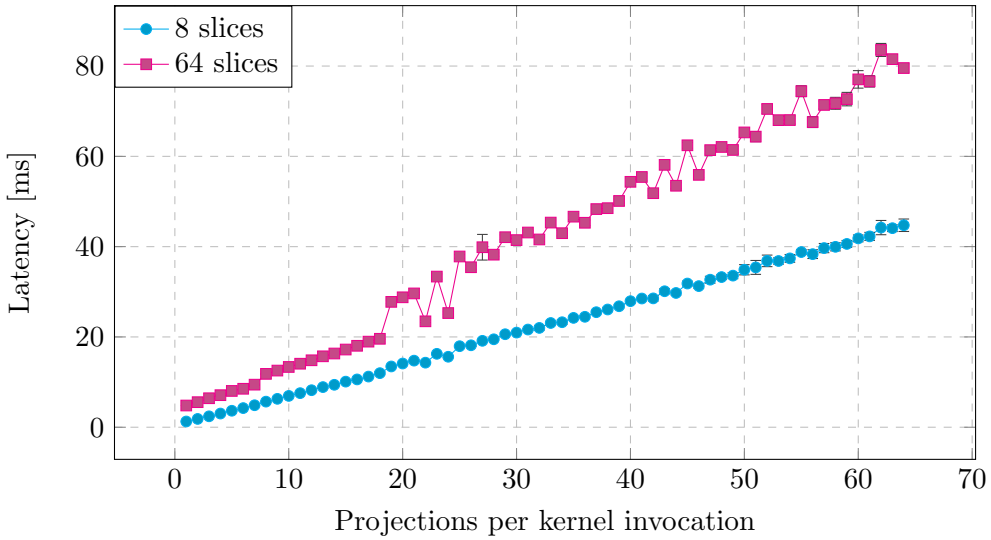


Figure 4.9: Latency as a function of B on a system with NVIDIA® GeForce GTX Titan, average memory throughput 10 GiB s^{-1}, projections of size 1024×1024 and various volume sizes. When we reconstruct volume $1024 \times 8 \times 1024$ (cyan) computation time is shorter than the transfer time and the overall latency grows almost linearly with B. When the volume increases to $1024 \times 64 \times 1024$ (magenta), the computation dominates the transfer and the differences between processing various amount of projections per kernel invocation start to be visible.

4.2.4 Multi-GPU Systems

One can further increase the back projection performance by employing multi-GPU systems. In order to fully utilize the computational resources we need to make sure that the time required to copy *all* projection parts required for one kernel invocation to *all* graphic cards is not longer than the duration of *one* kernel, since the kernels execute in parallel. If we know t_k and t_c described in Section 4.2.3 and we want to reconstruct volume $x \times y \times z$, use M projections which in total require the transfer of P pixels from host to the video card and we also know the maximum size of usable video memory, we can predict the number of fully utilized video cards G in the system by

$$G = \frac{M \cdot t_k \cdot xyz}{P \cdot t_c}. \tag{4.14}$$

If we want one video card to reconstruct one full volume $x \times y \times x$, we need to copy the full projections to the video card and (4.14) simplifies to

$$G = \frac{t_k \cdot x}{t_c} \tag{4.15}$$

If the reconstructed volume is a cube we can write

$$G = \frac{t_k \cdot \sqrt[3]{V}}{t_c}, \tag{4.16}$$

where V is the number of voxels that fit into the video memory, thus $\sqrt[3]{V}$ is the volume side.

Next to GU/s, G is a very important parameter for choosing the correct video card type when we want to build up a multi-GPU system in such a way that all cards are fully utilized. We can see a comparison of various video cards in Figure 4.10.

We have measured the real performance on three multi-GPU systems to validate (4.14). First system has seven NVIDIA® GeForce GTX Titan cards, second one contains six NVIDIA® GeForce GTX 580 cards and the third one holds four AMD FirePro™ S9170 cards. In the first two cases, Figure 4.11 and Figure 4.12, we used such volume shapes that G varied for various optimization strategies. In Figure 4.13, G for no optimization was around 4, which means at the limit of utilizing all cards and as we can see, there are no differences between various strategies.

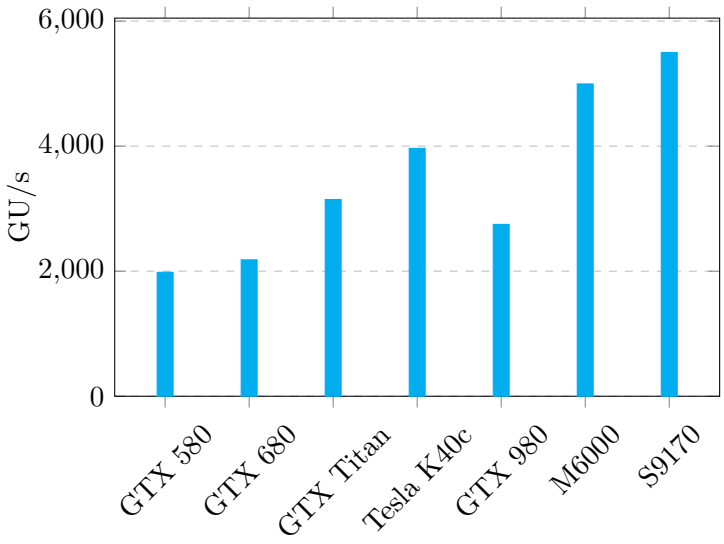


Figure 4.10: Theoretical GU/s computed as $G \cdot GU/s$ by using (4.16). We used $10\,\mathrm{GiB\,s^{-1}}$ to compute t_c, which is a typical transfer speed on the systems which were used for benchmarking in this section.

Table 4.1: Video cards used for benchmarks in this section and their most important characteristics. Names are shorthand for NVIDIA® GeForce GTX 580, NVIDIA® GeForce GTX 680, NVIDIA® GeForce GTX Titan, NVIDIA® Tesla® K40c, NVIDIA® GeForce GTX 980, NVIDIA® Quadro® M6000, AMD FirePro™ S9170.

Device	Architecture	Cores	Memory [GB]	Bandwidth [$\mathrm{GB\,s^{-1}}$]
GTX 580	Fermi	512	1.5	192.4
GTX 680	Kepler	1536	2	192.2
GTX Titan	Kepler	2688	6	288.4
Tesla K40c	Kepler	2880	12	288
GTX 980	Maxwell	2048	4	224
M6000	Maxwell	3072	24	317
S9170	Graphics Core Next	2816	32	320

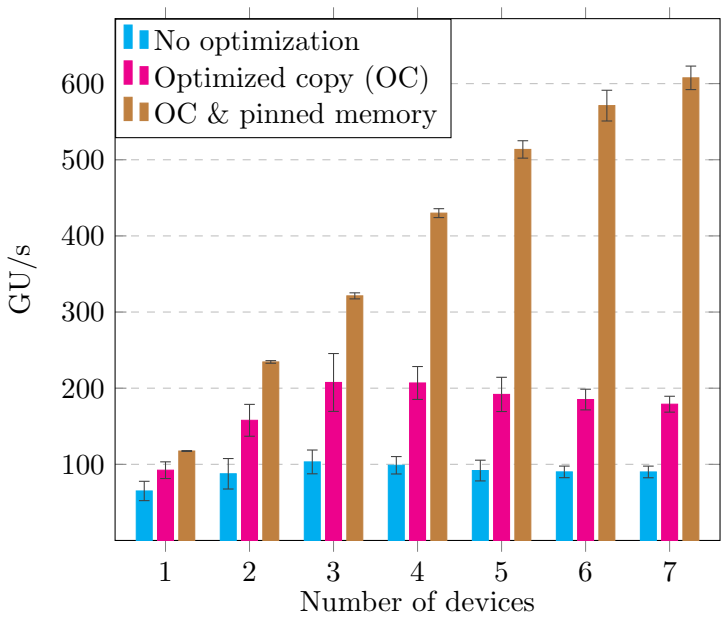


Figure 4.11: Reconstruction speedup with various techniques of increasing the data throughput on a system with seven NVIDIA® GeForce GTX Titan cards. We used projections of size 2048×2048 and volume with size $928 \times 928 \times 928$. Copying complete projections without the usage of pinned memory in cyan causes poor speedup because the time spent by transferring the data to the GPUs is much greater than the back projection time ($G = 0.6$). Copying only the required projection region (magenta) improves the performance ($G = 1.6$). The best speedup is achieved by copying only the required region combined with the usage of pinned memory (brown) ($G = 9.3$).

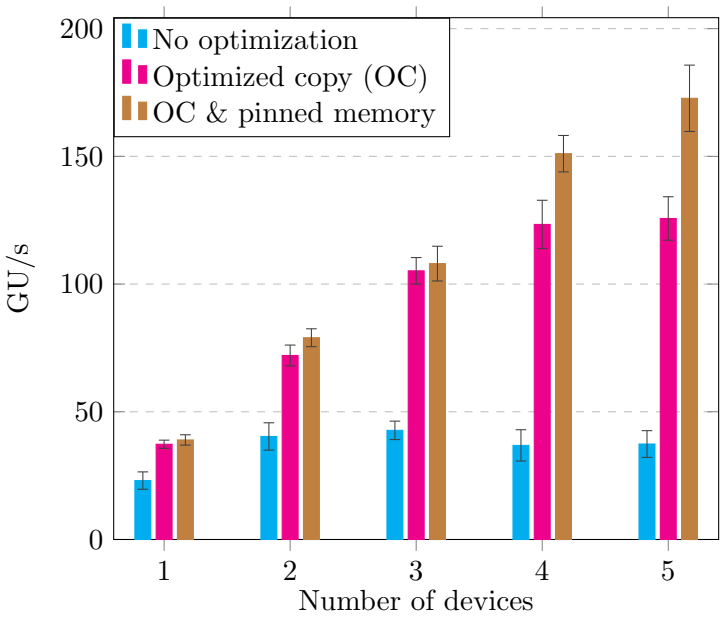


Figure 4.12: Reconstruction speedup with various techniques of increasing the data throughput on a system with five NVIDIA® GeForce GTX 580 cards. We used projections of size 2048×2048 and volume with size $640 \times 640 \times 640$. In this case the respective G values were 1.45, 8.3 and 10.2.

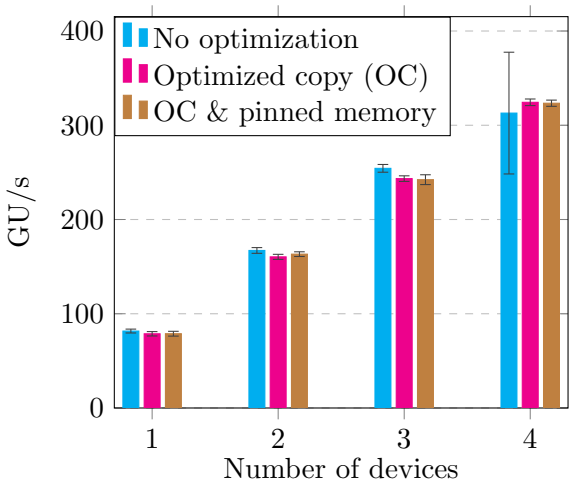


Figure 4.13: Reconstruction speedup on a system with four AMD FirePro™ S9170 cards. We used projections of size 2048×2048 and volume with size $928 \times 928 \times 928$. Already the worst G based on (4.14), given by copying full projections without the usage of pinned memory is 4.1, i.e. we could utilize all cards and there is no difference between various optimizations.

4.3 Summary

In this section we reviewed multiple algorithms which reconstruct sample structure in 3D. We selected filtered back projection for implementation because it allows fast update of the partially reconstructed volume from a subset of projections. Our algorithm is able to reconstruct data sets of laminographic experiments, which use tilted rotation axis and may be seen as a generalization of tomography.

We used OpenCL to implement the algorithm and we investigated how to minimize the amount of the projection data which needs to be copied to the video card. This allowed us to increase the compute/copy ratio, which is especially important by multi-GPU systems. Processing more projections per one kernel invocation and the usage of suitable memory spaces enabled us to achieve nearly the same data throughput as by highly optimized tomographic back projection algorithm and at the same time keep the latency low. This makes the algorithm suitable for usage in an image-based control system.

Chapter 5

Image-based Automation of X-ray Imaging Experiments

As we discussed in the previous chapter, the image-based feedback may require 3D sample structure reconstruction. Since the sample alignment information needs to be determined for correct reconstruction, in this chapter we will investigate problems related to research Question 3: *How to determine the sample alignment information from the acquired data?* Answering this question will enable us to use the 3D reconstruction to adjust the course of an experiment.

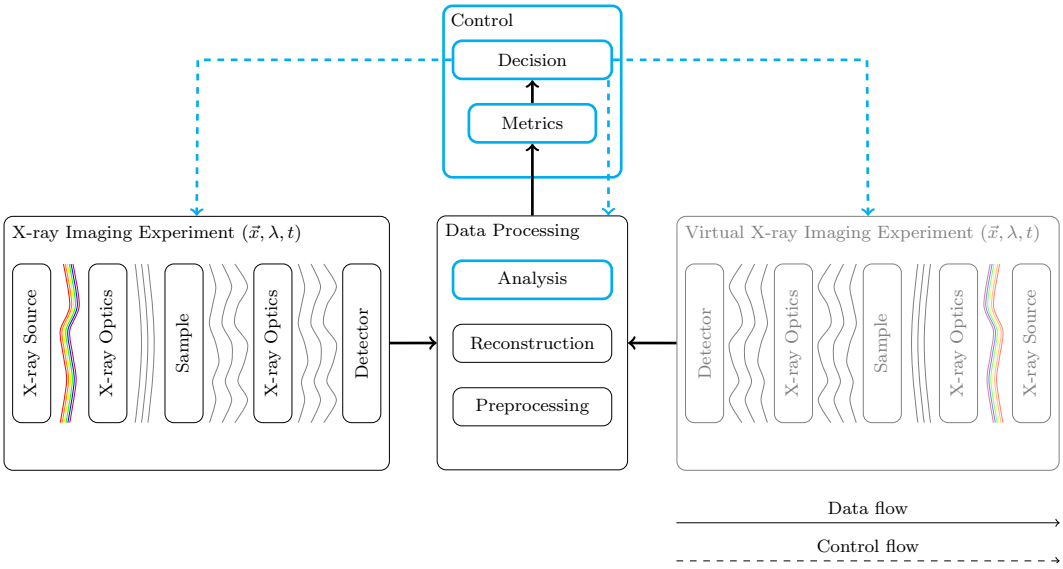


Figure 5.1: Our focus in this chapter, experiment automation and image-based control enabled by *online* data analysis highlighted in cyan.

We will further address Question 4: *How to enable image-based control and automation of a broad range of synchrotron X-ray imaging experiments while keeping*

the response time of the system low? Thus, we will look for existing building blocks which could be used to construct the desired *autonomous low-latency DAQ system.* We will employ a high-performance computing framework [20] for fast data processing, a Python-based beam line control system [21] which uses a specialized library for fast access to camera images and allows us to feed them into the data processing framework. We will integrate our 3D reconstruction algorithm from Chapter 4 into the data processing framework, which will enable us to use it in the *online* image-based feedback loop. We will implement a general autonomous data acquisition scheme within the control system and provide common beam line preparation tasks, namely focusing and finding the tomographic rotation axis.

5.1 Determination of the Sample Alignment

Before we get to the experiment automation itself, we need to consider the kind of data we will use for the automation. Since 2D projections might not be enough to make image-based decisions, we will enable both, 2D and 3D reconstruction-based feedback to the experiment. For the 3D case, we will use a generalized version of our algorithm from Chapter 4.

The back projection Algorithm 4.1 has several parameters related to the sample alighment which have to be chosen correctly. We need to know the 3D rotation axis direction which may be obtained from intrinsically rotating the sample coordinate system around the laminographic angle θ (pitch, around x-axis) and the roll angle ψ (around z-axis). For the final 3D pose of the sample we need a third rotation given by the yaw angle ξ (around y-axis), which rotates the sample around the rotation axis. However, this angle is just an offset to the tomographic rotation angle ϕ used to acquire projections and it merely rotates the sample in a slice, i.e. it doesn't cause alignment artifacts. Further, we need to know the rotation axis origin with respect to the x-axis c_x. The origin with respect to the y-axis c_y vertically offsets the sample in the slices but doesn't cause alignment artifacts.

The artifacts stemming from incorrect θ, ψ or c_x manifest as blurring of the slices because a point in the sample spreads out across several voxels. We can use metrics based on statistical tools to detect such artifacts [91], which we will describe below. To find the correct parameters, one can use optimization techniques or simply scan the parameters and select the ones which minimize some metric. Because we cannot save all projections in the video card memory, the determination of a correct parameter value is much faster when we back project a volume with many parameter values at once than when we back project one slice at a time for an adjusted parameter given by an optimization procedure. This stems from the fact that we need to copy relatively large amount of data between the main memory and the video card to reconstruct just one slice, as described in Chapter 4. When we reconstruct many slices at once, the compute/copy ratio increases with the number of reconstructed slices. For this reason we will extend our algorithm to be able to reconstruct 3D volumes of slices with various parameter values.

We adjust the algorithm from Chapter 4 to enable reconstructions of slices with various alignment parameters discussed above. Instead of back projecting slices along the y-axis, we can keep y constant and vary one of the parameters above. This way, we obtain volume $x \times p \times z$, where $p \in \{c_x, \theta, \psi\}$. For this we need to generalize the transformation matrix from (4.12):

$$\begin{pmatrix} \cos\psi\cos\phi & -\sin\psi & \cos\psi\sin\phi & c_x \\ \sin\theta\sin\psi\cos\phi + \cos\theta\sin\phi & \sin\theta\cos\psi & \sin\theta\sin\psi\sin\phi - \cos\theta\cos\phi & c_y. \end{pmatrix} \quad (5.1)$$

Once we have the volume $x \times p \times z$, we may apply some metric and look for its extrema to determine the best value of p.

We use motor positions from the experiment as initial values of the alignment parameters during the finding procedure. First, we need to look for the sample, thus reconstruct slices $x \times y \times z$. Afterward we scan the remaining parameters and iterate to refine the result.

5.1.1 Metrics

We will now describe and compare the behavior of various metrics on different alignment parameters. We will compare the metrics behavior with respect to the rotation axis on a simulated laminography data set and a real tomography data set. We will also investigate their behavior with respect to the laminographic angle on simulated data.

Standard Deviation

When a voxel in the volume is smeared due to a wrong parameter setting, its grey value is distributed across several voxels, which means that their individual intensity will be lower than the intensity of the correctly reconstructed voxel. This broadens the histogram which can be detected by standard deviation σ. Greater σ stands for better parameter values.

$$\sigma = \sqrt{E[(I(x,y) - \mu)^2]}, \quad (5.2)$$

where $I(x,y)$ stands for grey value in a voxel, E the expected value and μ the mean of the volume.

Kurtosis

Kurtosis κ is another measure sensitive to the histogram shape, in particular its "tailedness". It is defined as

$$\kappa = \frac{E[(I(x,y) - \mu)^4]}{\left(E[(I(x,y) - \mu)^2]\right)^2} - 3. \quad (5.3)$$

Median of Absolute Deviation

MAD is also sensitive to the shape of the histogram but is more robust with respect to statistical outliers than the previous two metrics because it uses the median of the data. First, we compute the median grey value of an image, then we subtract it from individual pixels, take the absolute value and compute the median again

$$\mathrm{MAD} = \mathrm{MED}\left[|I(x,y) - \mathrm{MED}(I(x,y))|\right] \tag{5.4}$$

Since sharp images lead to more concentrated histograms, i.e. lower MADs, we need to minimize this metric.

Sum of the Absolute Gradient

SAG is sensitive to the ring artifacts produced by incorrect rotation axis setting (Figure 5.2c). The radius of the rings is given by the rotation axis misalignment. As the rings become smaller, SAD becomes smaller as well. It is defined as

$$\mathrm{SAD} = \sum_{x,y} |\nabla_x(I(x,y)) + \nabla_y(I(x,y))| \tag{5.5}$$

Entropy

Entropy is another metric used in practice [91, 92] and again works on image histogram. Similar to κ, it relies on the fact that there is background in the reconstructed volume. More accurate parameter values are reflected in the decrease of this metric because the histogram part which belongs to the sample becomes narrower. The metric is defined as

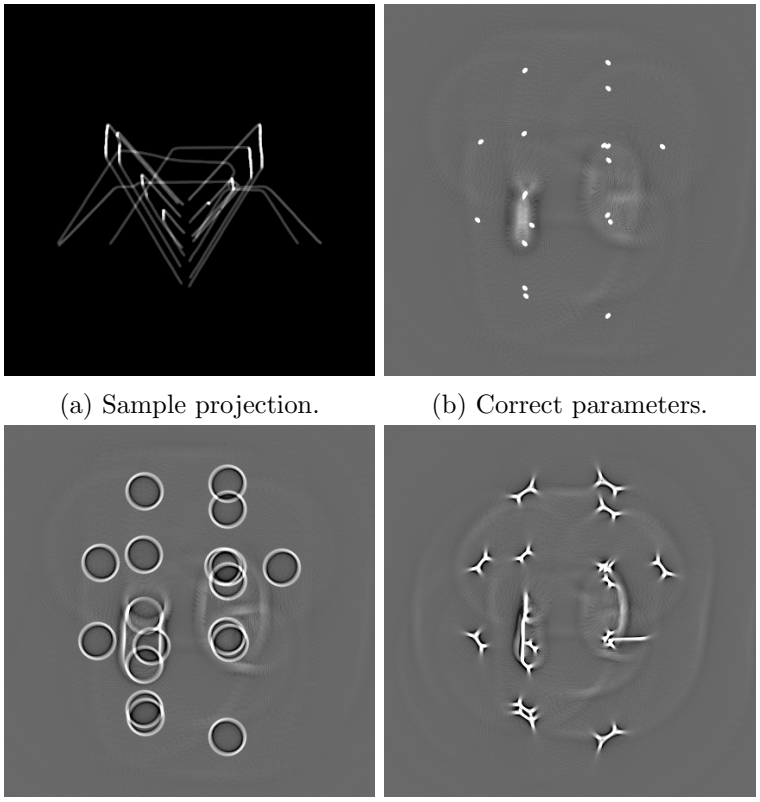
$$H = -\sum_{x,y} P[I(x,y)] \log_2 P[I(x,y)], \tag{5.6}$$

where $P[I(x,y)]$ is the probability of the occurrence of the grey value $I(x,y)$ in pixel (x,y), which is the histogram value of that grey value divided by the amount of pixels in the image.

5.1.2 Simulated Data Set

We will show the behavior of metrics from Section 5.1.1 on a simulated laminographic data set shown in Figure 5.2. The simulation was based on the segmentation of diffraction loops from [33]. We used laminographic angle $\theta = 75°$ and projected the sample under various rotation angles which make up the laminographic data set.

We reconstructed the data set with various settings of the rotation axis and θ and show the behavior of the metrics in Figure 5.3 and Figure 5.4. σ, κ and SAG metrics are very sensitive to the correct rotation axis setting. The peak of MAD is more flat, which makes the metric less decisive. Entropy shows slight oscillations around the maximum, which makes it difficult to choose the correct axis from the region

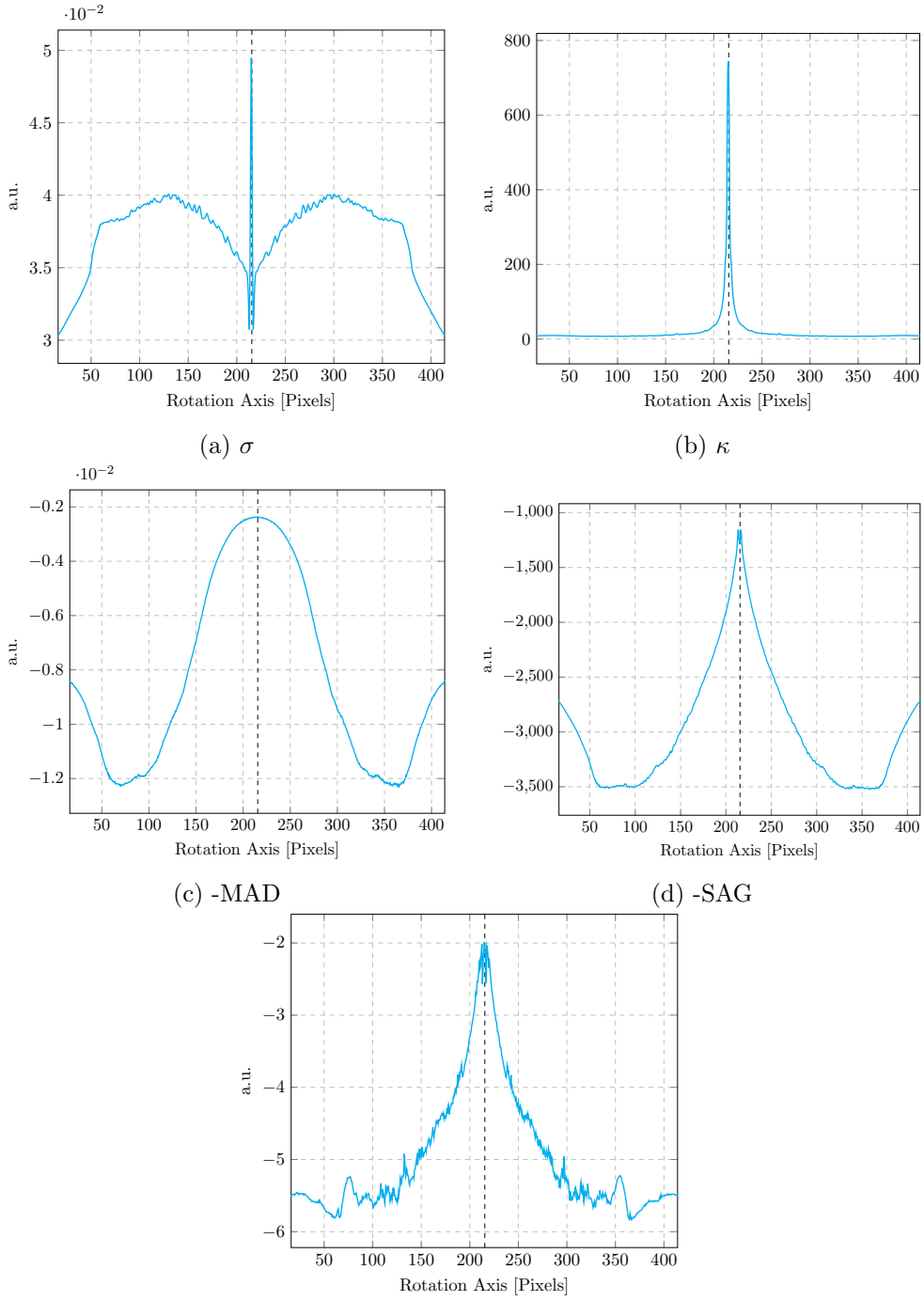


(a) Sample projection. (b) Correct parameters.

(c) Wrong rotation axis. (d) Wrong laminographic angle.

Figure 5.2: 3D reconstruction with various parameter misconfigurations. The data set is a simulation of a dislocation loop network based on data from [33]. Sample tilt was $\theta = 75^\circ$. Rotation axis in (c) is shifted by 20 pixels but θ is correct. In (d), θ is off by 5° but the rotation axis is correct. (b) to (d) are reconstructed slices.

surrounding the peak. σ, κ and SAG are also good for choosing the laminographic angle. σ shows a clear peak, however the regions far away from the maximum start to oscillate and might spoil the result. On the other hand, κ behaves very well also in this case. The peak of SAD is more flat than in the rotation axis case but it is still usable. Entropy has a local minimum in the correct laminographic angle and as well as MAD fails in this case.

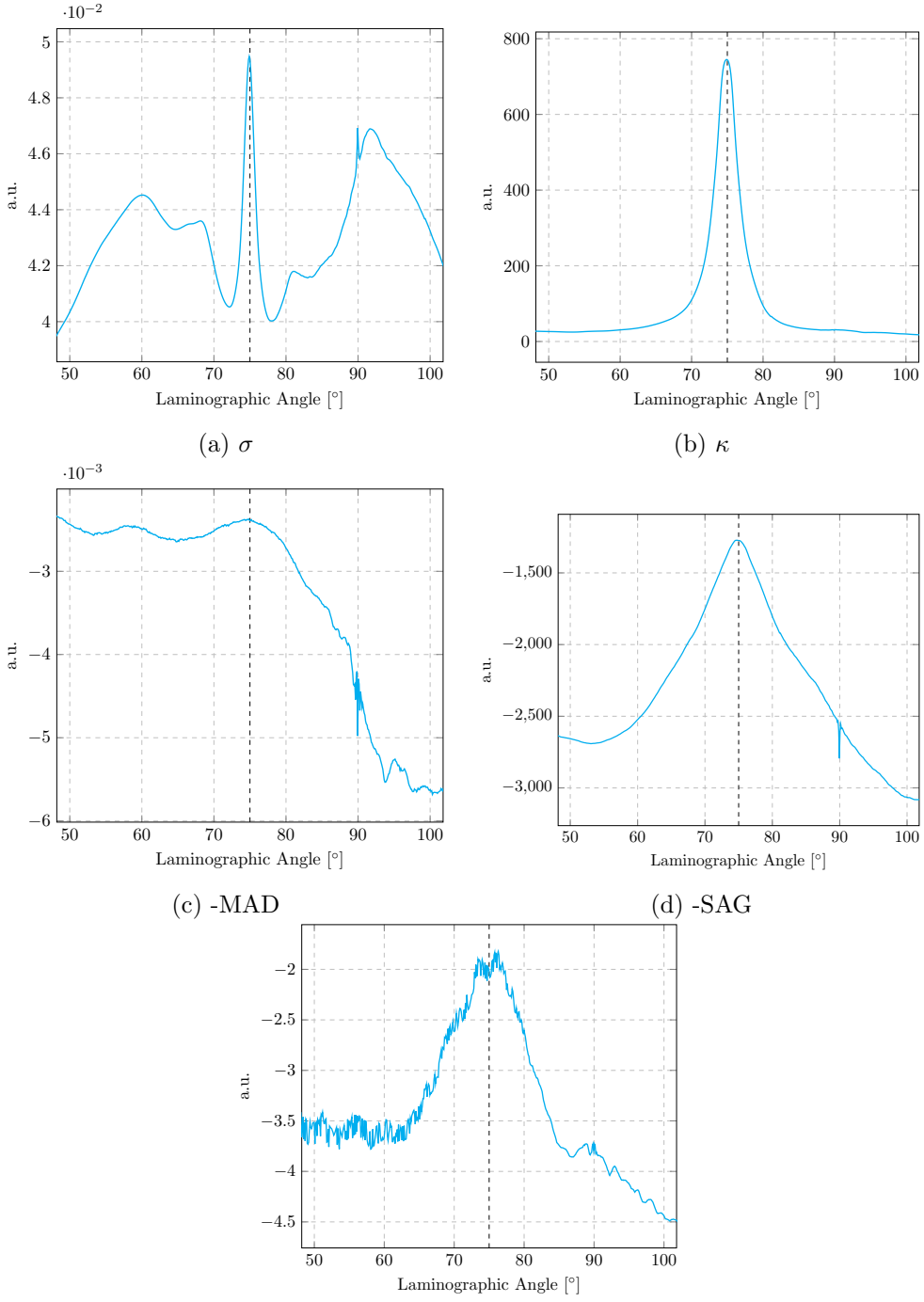


(a) σ (b) κ

(c) -MAD (d) -SAG

(e) -H

Figure 5.3: Metrics applied on varying rotation axis setting for data set from Figure 5.2. The most decisive are σ in (a), κ in (b) and SAG in (d). The true rotation axis is shown by the black dashed line. All metrics show the correct parameter value at their maximum for clarity, which is why we take the inverse in (c), (d) and (e).



(a) σ (b) κ

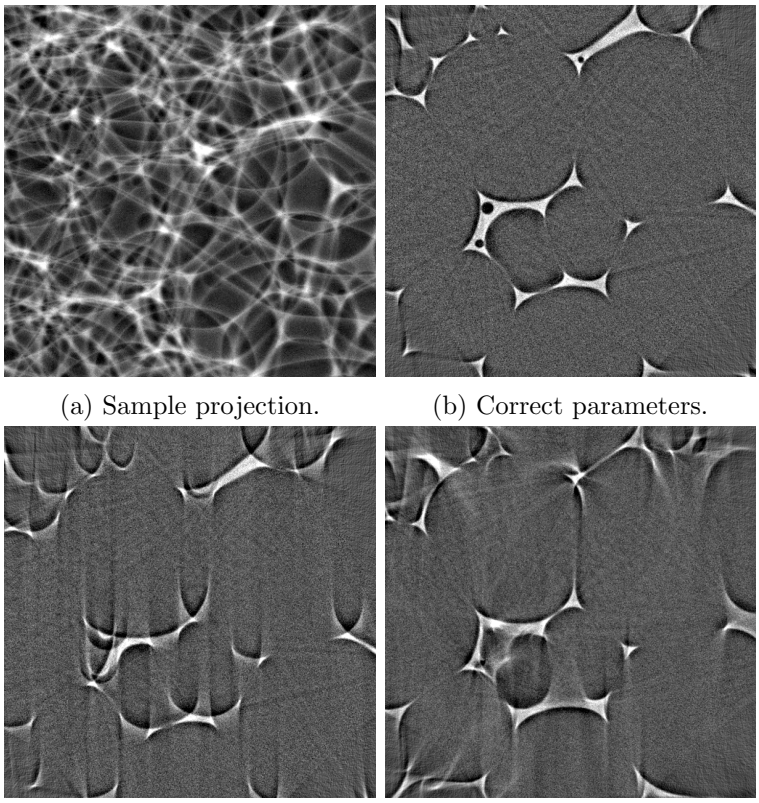
(c) -MAD (d) -SAG

(e) -H

Figure 5.4: Metrics applied on varying laminographic angle. κ in (b) is the most sensitive one with no oscillations in the outlier regions. The data set is again from Figure 5.2. The true θ is shown by the black dashed line.

5.1.3 Real Data Set

Even though we have seen which metrics work well on simulated data, the results might differ for real data, which is why we will apply the metrics on a real tomographic data set of a liquid bubble foam shown in Figure 5.5, which is especially challenging because the tomography was localized around a ROI in the sample. This means that parts of the sample actually leave the detector FOV during rotation which causes reconstruction artifacts. In particular, we will apply the metrics on slices with varying rotation axes and show the results in Figure 5.7.

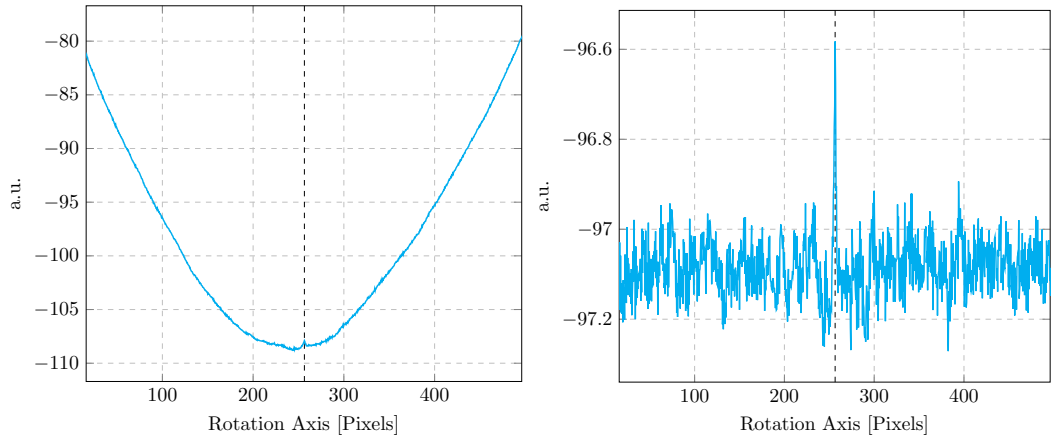


(a) Sample projection. (b) Correct parameters.

(c) Wrong rotation axis. (d) Wrong laminographic angle.

Figure 5.5: 3D reconstruction with various parameter misconfigurations. The data set is a real scan of a liquid foam. Sample tilt was $\theta = 90^\circ$. Rotation axis in (c) is shifted by 20 pixels but θ is correct. In (d), θ is off by 5° but the rotation axis is correct. (b) through (d) are reconstructed slices.

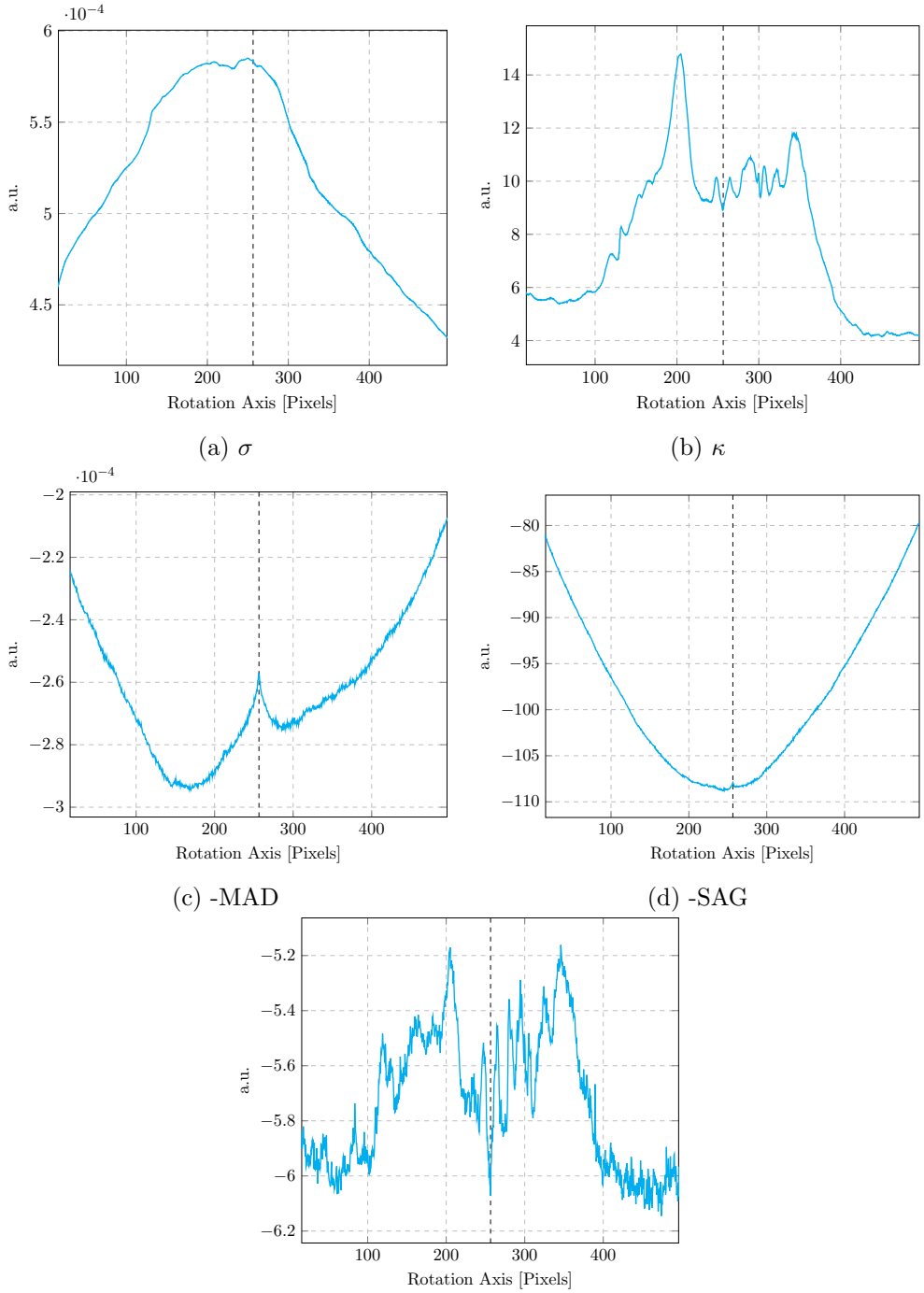
When we search for a parmeter too far from its correct value the reconstruction will be very blurred and sample features might overlap or even end up outside of the reconstructed volume. This will be reflected on the metric behavior and might lead to false parameter determination as shown in Figure 5.7d. As we can see, there is a global tendency for the SAG metric with only a little peak around the correct parameter value, which means we cannot just use a global minimum or maximum



(a) -SAG. (b) -SAG without low frequencies.

Figure 5.6: Filtering low frequencies from the SAG metric (a) removes the global tendency and enables us to use the global maximum to determine the correct reconstruction parameter again (b).

for extracting the correct parameter value. However, e.g. for the SAG metric, the global tendnecy varies more gradually than the sharp peak around the correct parameter, which we can exploit. We filter out the low frequencies from the metric which enables us to use the global maximum again, see Figure 5.6b.



(a) σ (b) κ

(c) -MAD (d) -SAG

(e) -H

Figure 5.7: Metrics applied to various rotation axes for the data set in Figure 5.5. σ in (a) provides the correct parameter value, however the peak is flat, thus susceptible to error. κ in (b) in this case fails, MAD in (c) is sensitive to the correct parameter but we cannot use the global maximum because there is a global tendency in the metric, the same applies for SAG in (d). Entropy in (e) fails. The true rotation axis is shown by the black dashed line.

5.2 Experiment Control

Experiment automation requires a lot of software components which are on the one hand self-contained to improve system's modularity, but on the other hand can communicate with low latency in order to enable prompt feedback to the experiment hardware. We will not implement the whole software stack but rather use suitable existing components that fit our needs, extend and combine them in such a way that we will be able to automate experiments.

5.2.1 System Components

High-speed experiments typically require scintillating screen to convert X-rays to visible light and a conventional visible light camera. These cameras are shipped with their own control software. However, these are closed products which cannot be used to control the cameras from our DAQ system. Luckily, the vendors often provide a Software Development Kit (SDK), which can be used to access the camera programatically. *libuca*[1] is a GLib[2]-based library used to access cameras in a unified way, i.e. it creates a camera abstraction layer with an API which fits a large variety of cameras. Individual cameras are then implemented as plugins and used via the general API. Thanks to GLib, it provides various programming language bindings, including Python.

UFO framework [20] is a distributed data processing framework which we will use to perform the data processing tasks. It is also built on GLib and it consists of *ufo-core*[3], which takes care of connecting various processing nodes together and carries out the execution of the final data processing pipeline. *ufo-filters* are plugins which implement concrete image processing tasks. We implement our laminographic back projection algorithm from Chapter 4 as one these tasks as well so that we can use it in combination with the framework's preprocessing and filtering capabilities.

concert [21] is a Python-based control system which brings together device access, *libuca*, *ufo-core* and *ufo-filters*. Whereas *libuca* serves for fast camera access, other beam line devices, e.g. motors, are either accessed via TANGO [59] or specialized communication protocols. The role of *concert* is to control complete beam lines, from simple device parameter adjustment to performing scans and conducting complete experiments. Tight integration of direct camera access and high-performance computing into *concert* makes it unique and suitable for low-latency experiment automation. We will use it to implement our automation procedures and conduct experiments in Chapter 6.

[1] `https://github.com/ufo-kit/libuca`
[2] `https://developer.gnome.org/glib/`
[3] `https://github.com/ufo-kit/ufo-core`

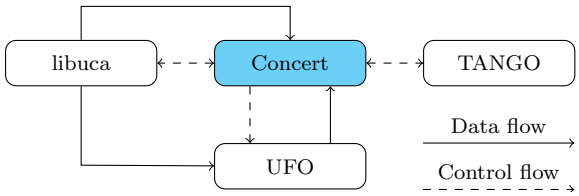


Figure 5.8: Experiment automation scheme with concrete software components. *concert* controls cameras via *libuca* and the rest of beam line devices via TANGO. It also receives control information from these components, e.g. camera frame rate or motor positions. *libuca* sends images to the UFO framework and *concert* provides processing parameters. Processing results are read back to *concert* and used to make decisions about the course of an experiment. Image stream must be available also in the control system itself for live preview.

5.2.2 Data Acquisition Implementation

The data acquisition typically consists of recording the sample itself, but also the normalization data required for more precise sample reconstruction. For example, one records a set of *flat fields*, which are images of the beam without the sample used to remove the background. In this context, *Experiment* is a set of procedures which acquire all the necessary data for sample investigation. Users might need to quickly change the DAQ procedure during beam time due to new ideas or specific sample needs, which is why an experiment has to be modeled in a very flexible way.

For this reason, we will use the general concepts in *concert* to describe experiments by finer building blocks, which will allow us to specify the acquisition and processing pipelines of various data kinds. Experiments themselves will be responsible for connecting these building blocks and enable the users to add, remove or even swap them dynamically during an experiment.

Coroutines

concert uses *coroutines* to effectively model data processing pipelines. *Coroutine* [93] is a generalization of *subroutine* and once it is invoked, it enables us to give control to another part of a program and resume later without actually exiting the coroutine. Thus, its internal state is preserved between subsequent control gains. This is very useful for processing data streams because the coroutine doesn't have to process all items at once (e.g. in a for loop) when its internal state influences the processing.

Coroutines in Python are implemented by *generators*. They are functions which contain `yield` somewhere in their body instead of the `return` statement. Generators can be data *producers*, i.e. they yield values further processed by another part of the program, or they can be *consumers*, i.e. they obtain values by using `yield` on the right side of an expression. Consumers process the obtained values and either send them forward, which makes them *processors*, or they might terminate the stream, which makes them *sinks*. A comparison between conventional functions and coroutines is depicted in Figure 5.9.

```
def produce():
    for i in range(10):
        yield i

def square(number):
    return number ** 2

def consume(item):
    print item

for number in produce():
    consume(square(number))
```

(a) Conventional approach.

```
def square(consumer):
    while True:
        number = yield
        consumer.send(number ** 2)

def consume():
    while True:
        item = yield
        print item

square(consume())
```

(b) Coroutine approach.

Figure 5.9: A *producer* generator `produce` provides numbers which are conventionally processed by another functions in (a). In (b) the order of function calls is reversed where the *processor* coroutine `square` forwards modified data to the *sink* `consume`. Initialization of coroutines and data injection to them is left out in (b) for brevity.

Acquisitions

Although coroutines provide an elegant way to describe image processing pipelines, they have one major limitation. Once a data producer finishes, it is not possible to restart it. This means that we need to recreate the pipeline again when new data is ready. In order to do that, we need to have access to objects defining the behavior of coroutines *before* they are actually turned into them at run-time. For instance, the `square` function from Figure 5.9b is turned into a coroutine once it is invoked, i.e. if we save the *reference* to this function we can reuse it multiple times.

`Acquisition` class stores these references for whole data acquisition pipelines, which specify the DAQ of a specific data kind (e.g. flat fields or tomographic projections). It consists of references to a data *producer* and multiple *consumers*, which perform various tasks from writing raw data to the disk to 3D sample reconstruction. Once the acquisition is invoked, it connects the producer to consumers, i.e. it creates a coroutine-based data acquisition pipeline. The internal structure of class `Acquisition` is depicted in Figure 5.10.

Experiments

`Experiment` class consists of more `Acquisition` class instances and executes the whole DAQ process. It enables adding, removing and even swapping acquisitions to make the data acquisition scheme as flexible as possible. An example of an experiment which uses acquisitions from Figure 5.10 is depicted in Figure 5.11.

```
class Acquisition(object):
    def __init__(self, producer, consumers):
        self.producer = producer
        self.consumers = consumers

    def connect(self):
        started = []
        for consumer in self.consumers:
            started.append(consumer())

        for item in self.producer():
            for consumer in started:
                consumer.send(item)

def combine():
    return square(consume())

acq = Acquisition(produce, [combine])
```

Figure 5.10: `Acquisition` class for re-using coroutines.

Addons

`Addon` class encapsulates complex functionality of recurring data consumers. For instance, during an imaging experiment we want to store the raw data to the disk. Let's say we have three acquisitions in an experiment, *darks*, *flats* and *projections* and we want to store images from all three acquisitions. In this case we may use the `ImageWriter` subclass of the `Addon` class by simply writing

```
writer = ImageWriter(experiment.acquisitions, walker)
```

`writer` automatically *attaches* image writing consumers to all acquisitions in the `experiment`. It uses a `walker` instance, which creates subdirectories based on the acquisition type. We will not describe its specifics because they are not important for our example. Just like image writing, one can implement complex addons used for advanced data pre-processing or 3D reconstruction.

```
class Experiment(object):
    def __init__(self, acquisitions):
        self.acquisitions = acquisitions

    def add(self, acquisition):
        self.acquisitions.append(acquisition)

    def remove(self, acquisition):
        self.acquisitions.remove(acquisition)

    def swap(self, first, second):
        first_index = self.acquisitions.index(first)
        second_index = self.acquisitions.index(second)
        self.acquisitions[first_index] = second
        self.acquisitions[second_index] = first

    def run():
        for acq in self.acquisitions:
            acq.connect()

foo = Acquisition(produce, [combine])
bar = Acquisition(produce, [consume])
ex = Experiment()
ex.run()
# Change the order of acquisitions
ex.swap(foo, bar)
```

Figure 5.11: `Experiment` class connects more acquisitions and enables users to change them dynamically.

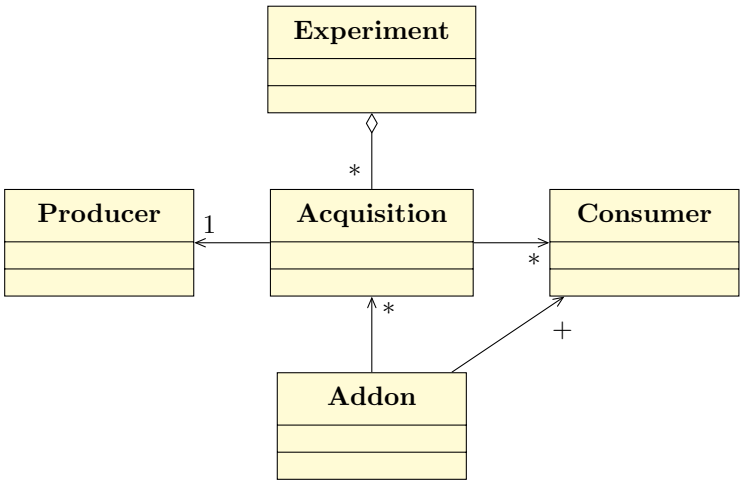


Figure 5.12: Classes used for conducting an experiment. `Acquisition` connects one `Producer` to multiple `Consumer` instances. An `Experiment` consists of multiple acquisitions. Complex consumer behavior is encapsulated by the `Addon` class, which can attach itself to many `Acquisition` instances.

Closed Loop Optimization

Experiments can be used together with some optimization strategy in a closed loop depicted in Figure 5.13. This is the image-based feedback to the experiment. Measured data is evaluated based on some metric and if it doesn't meet the requirements some experimental parameter is adjusted and the data is remeasured. This is repeated in a loop until the metric is satisfied.

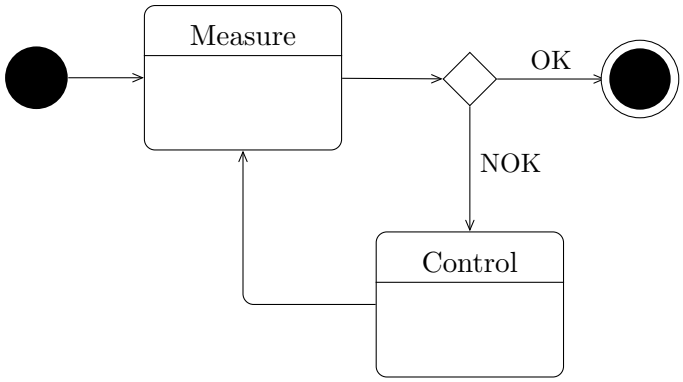


Figure 5.13: Closed loop execution scheme as a UML state diagram. *Measured* data is evaluated and if it doesn't meet certain requirements *Control* adjusts some experimental parameter. This is repeated until the data meets the requirements.

A good example for image-based optimization of some parameter is focusing. It requires a camera and a motor which shifts the focus plane perpendicularly to the camera sensor plane. The aim is to find such position of the focus motor which minimizes some sharpness measure, found by some optimization algorithm. We implemented such focusing strategy in *concert*, the user has to provide a camera, a motor, sharpness measure and one of the optimization algorithms based on the ones from scipy[4].

Another example is sample alignment for a tomographic scan. The rotation axis needs to be perfectly aligned with the y-axis, so that a point in the sample remains in the same height in every projection. We can perform the alignment based on the fact that a circle obtained by rotating a point around the rotation axis projected onto a detector becomes an ellipse. If we want to align the axis, we need to know the roll angle ψ and the pitch angle θ of the ellipse, which may be obtained from finding its parameters. If we write an ellipse in the conic section form

$$\begin{aligned} Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0 \qquad (5.7)\\ \begin{pmatrix} x & y \end{pmatrix} \boldsymbol{A_{33}} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} D & E \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = 0 \\ \begin{pmatrix} A & B/2 \\ B/2 & C \end{pmatrix} = \boldsymbol{A_{33}}, \end{aligned}$$

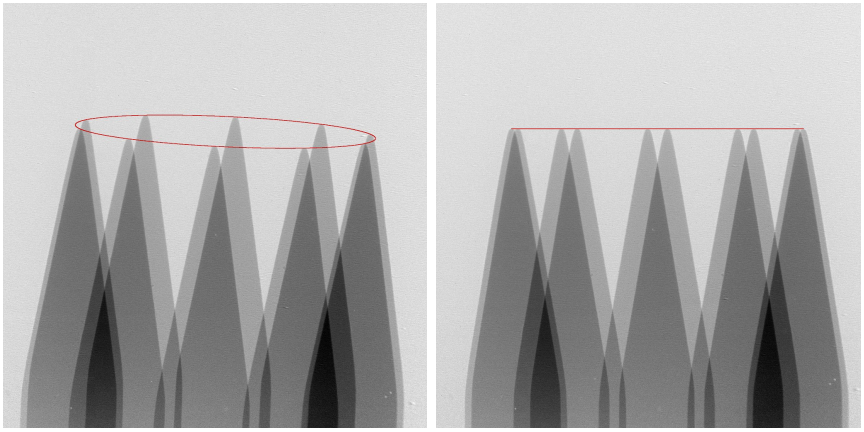
[4] `https://docs.scipy.org/doc/scipy-0.18.1/reference/optimize.html`

we may use the singular value decomposition [94] to determine the parameters in (5.7) from the measured (x, y) points. Further singular value decomposition of the matrix $\boldsymbol{A_{33}}$ allows us to find the angles θ and ψ. If `matrix` is a matrix with rows given by (5.7), the angles can be computed in Python as in Figure 5.14.

```
import numpy as np

u, s, v = np.linalg.svd(matrix)
a, b, c, d, e, f = v[-1]
a_33 = np.array([[a, b / 2],[b / 2, c]])
u, s, v = np.linalg.svd(a_33)
pitch_angle = np.arcsin(np.sqrt(s[1] / s[0]))
roll_angle = np.arctan(v[1, 1] / v[1, 0])
```

Figure 5.14: Roll and pitch angle computation by fitting an ellipse to projections at various positions around the rotation axis.



(a) Misaligned rotation axis. (b) Aligned rotation axis.

Figure 5.15: Rotation axis alignment. The sample is a steel needle which is easily segmentable. It is rotated around the misaligned rotation axis, depicted in (a) by superimposing images from various rotation positions. Extracted needle tips are used to fit the ellipse and determine the roll ψ and pitch θ angles. Sample stage is rotated based on these angles in order to align the rotation axis with the y-axis required by tomography, depicted in (b). Figure taken from [21].

In practice, we insert an easily segmentable object into the beam, rotate it around the axis of rotation and extract the same point of the object from all the projections, which gives us the ellipse positions. Then we fit the ellipse and use the angle between its major axis and the x-axis to roll the rotation axis and the ratio between the major and minor axes to adjust the pitch of the axis. Because of the projection ambiguity, we cannot know if we should adjust the pitch angle in the positive or negative direction, so we need to select one and remeasure the ellipse points. If θ becomes

larger we need to move to the other direction. At the end, it is good to remeasure the ellipse points once more to make sure it turns into a degenerate case, a line, which means that θ is correct and if the line is aligned with the x-axis, also ψ is correct and we may commence the tomographic scan. In addition to tomography, this technique may be used also for aligning the rotation axis for other geometries, like laminography.

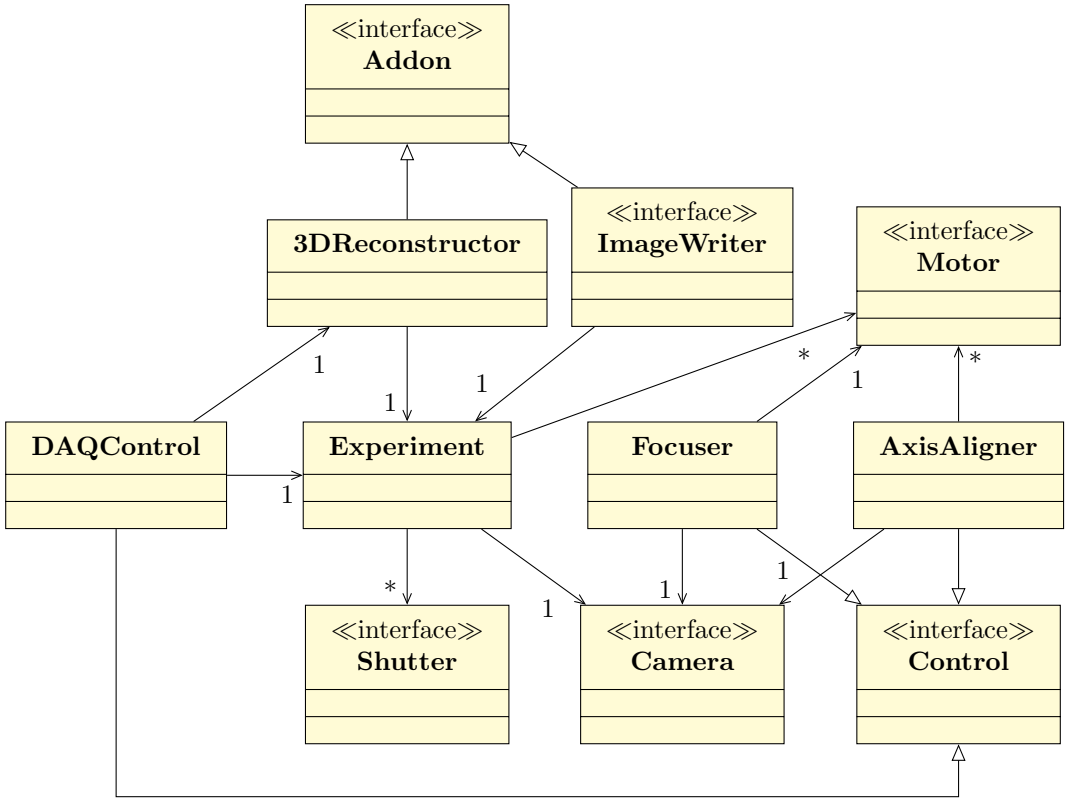


Figure 5.16: High-level simplified UML class diagram of a DAQ control based on 3D reconstruction. The `3DReconstructor` class automatically finds the sample alignment information and reconstructs 3D volumes. The `DAQControl` implements some experiment-specific closed loop for 3D reconstruction-based automation.

5.3 Summary

In this section we investigated the ability of various image metrics to detect the correct sample alignment based on laminographic reconstruction.

We then chose suitable components to build our autonomous data acquisition system. We used *libuca* for camera access and we integrated our 3D reconstruction algorithm from Chapter 4 into *UFO framework*, which is a high-performance computing framework. We used *concert* for programmatic access to the mentioned

components and for device control. We extended it by a set of classes and procedures required for image-based experiment control and automation. We then used the developed system to implement focus finding and rotation axis alignment routines.

Thanks to the usage of high-performance computing and flexible design of the overall system, it can be used to automate and control a broad range of experiments based on *online* data processing. An assembly for conducting a 3D reconstruction-driven experiment is depicted in Figure 5.16.

Chapter 6

Conducted Demonstrator Experiments

In this chapter we conduct several experiments to demonstrate the usage of our autonomous DAQ system from Chapter 5 and check the correctness of simulations from Chapter 3. First, we will conduct an autonomous tomography experiment which adjusts the camera frame rate based on the rate of change in a liquid foam in Section 6.1. Afterward we will show an autonomous *high-throughput* tomography experiment in Section 6.2, which scans many similar biological samples and reconstructs them on-the-fly to check the quality of the acquired data. We will also conduct an *interactive* laminography experiment in Section 6.3, which enables us to follow an *in-situ* process in 3D. Our last experiment in Section 6.4 shows that the physical principles included in simulations in Chapter 3 enable us to predict the outcome of an X-ray imaging experiment which uses specialized optical elements in order to provide multiple contrasts from one measurement.

6.1 Frame Rate Optimization Applied to Tomography

The goal of this experiment is to demonstrate the 3D reconstruction-based automation capabilities of the system described in Chapter 5. This section is based on a high-speed tomography of a liquid foam which changes its structure over time, presented in [95].

Bubbles in such foam rupture or merge at unknown rates. If we want to record a tomogram which captures the sample at a certain state without reconstruction artifacts, we need to make sure that the sample doesn't change while we acquire all projections. Since we don't know how fast the sample changes, we could record the data with the maximum available rotation speed and camera frame rate, which together define the maximum rate of change that we are able to follow with our DAQ hardware. However, this approach has a major drawback in terms of the data quality. If we maximize the recording rate, we inevitably decrease the amount of captured photons which leads to noisy data sets and makes the data analysis much harder.

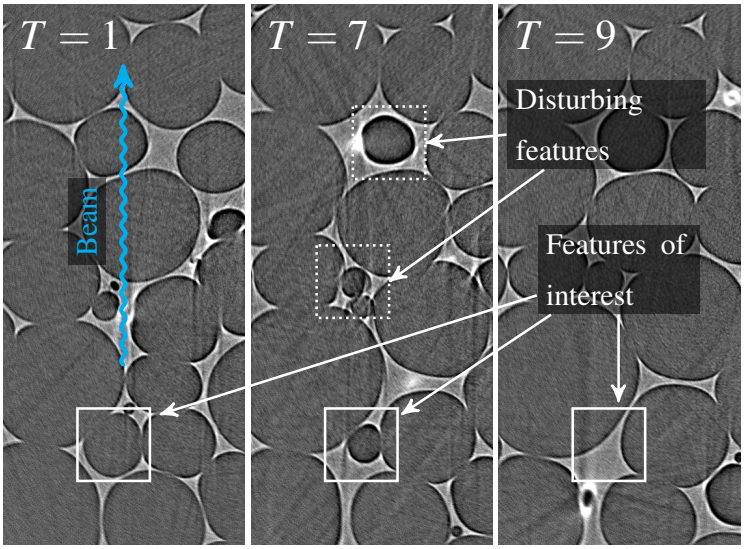


Figure 6.1: Tomographic slices T depicting the same region at different stages of the foaming process. Figure appeared in [95].

Thus, it is much more desirable to use image analysis to determine the *slowest* recording speed which gives us artifact-free 3D reconstruction and simultaneously maximizes the SNR. With our system from Chapter 5, we can do this iteratively, i.e. record a data set, assess the image quality based on some metric and adjust the recording speed until the metric is satisfied.

If we are interested in a specific ROI of our sample it is not sufficient to use projections for assessing the data quality because the beam propagation dimension is lost in them. Imagine that we are interested in the ROI specified by the solid rectangle in Figure 6.1. The projection of this ROI is a part of a detector row and it contains all the changes along the beam direction. Thus, if the sample changes outside of this ROI along the beam direction, we won't be able to recognize this change from the change in the ROI. For example the dotted rectangles from Figure 6.1 are superimposed on the ROI in the solid rectangle. For this reason, it is much more reliable to use 3D reconstruction for data comparison. In Figure 6.2, we use the correlation coefficient

$$r = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2}\sqrt{\sum_i (y_i - \bar{y})^2}} \tag{6.1}$$

to compare projection-based and slice-based correlation between consecutive data sets from Figure 6.1. Whereas the projection-based correlation fails due to the disturbing features in Figure 6.1, the slice-based correlation decays smoothly as the bubble in the solid rectangle in Figure 6.1 disappears.

Hence, our goal is to find an optimum between foam stability and imaging quality in an automatic way based on comparing tomographic slices. To realize this we

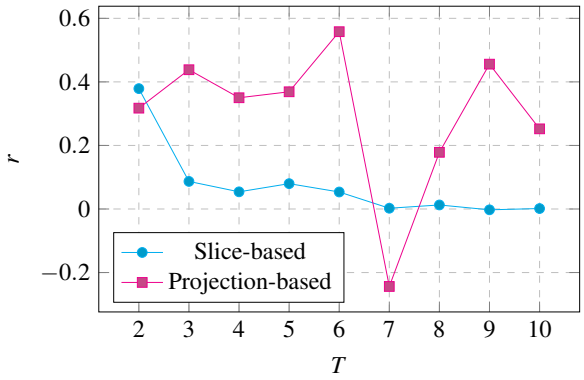


Figure 6.2: Slice- and projection-based correlation coefficient r between the first and the n-th tomogram depicted in Figure 6.1. Figure taken from [95].

use *libuca* for fast access to camera images, UFO framework for fast tomographic reconstruction and the closed loop acquisition scheme described in Section 5.2.2. We use these components to optimize the recording speed, i.e. the camera frame rate and rotation speed. We start by acquiring two tomograms one after another with slow acquisition speed, reconstruct slices on-the-fly and compare them in terms of the correlation coefficient r defined above. If r does not exceed a threshold, it means that the two slices from consecutive tomograms are not similar enough and we double the frame rate and adjust the rotation speed accordingly. After that we acquire a new pair of tomograms and continue like this until the threshold is reached.

To test our hypothesis we set the correlation coefficient to an empirically determined value 0.7. The automated DAQ process stopped after three iterations which used 200, 400 and 800 frames per second, respectively. The corresponding correlation coefficients were 0.295, 0.397 and 0.744. This means that our slice-based feedback approach was able to converge towards stable, similar reconstructions of a specific ROI, which would not have been feasible with a projection-based correlation because of the lack of the third dimension in the projections.

6.2 High-throughput Tomography with Online Reconstruction

This example demonstrates our system from Chapter 5 on an automated high-throughput tomography experiment. Our objective is to acquire many tomographic data sets of similar biological samples (Figure 6.4) and automatically reconstruct full tomograms on-the-fly in such a way that the reconstruction will not be the bottleneck of the data acquisition.

We use a sample changer which can hold up to 49 samples, the pco.dimax camera with 50 frames/s to acquire 3000 tomographic projections of size 2016×2016 and two bytes per pixel. The camera is connected to the acquisition computer by the

CameraLink interface and is set to a mode when it first records images into an on-camera buffer. They can be downloaded to the computer after the acquisition is finished. Such mode is necessary because the frame rate combined with the frame size would require download speed 406 MB s^{-1}, whereas the interface limit is 255 MB s^{-1}. As the projections are being downloaded they are simultaneously stored on the disk and streamed via a 10 Gigabit Ethernet network to a powerful reconstruction server with 7 NVIDIA® GeForce GTX Titan video cards.

Such a setting enables us to partially parallelize hardware positioning and fully parallelize data acquisition and reconstruction. When we acquire all data for one sample, we can start the data download from the camera. During this time we cannot record another data set, so we at least use it to place the next sample on the rotation stage. Once all the data is stored on disk and streamed to the reconstruction server, reconstruction of the current data set and acquisition of another one may start.

After the server receives a data set it finds the sample by looking for edges in a projection. This is done by applying the SAG metric from Section 5.1.1 to projection rows. The row which maximizes the metric is selected for finding the rotation axis. We vary the rotation axis by hundred pixels around the approximate value given by the rotation stage position with respect to the detector. Axis position which minimizes the SAG metric is selected as the correct one and used to reconstruct the whole volume. We use the tomographic filtered back projection algorithm from *ufo-filters*[1] to maximize the reconstruction throughput and make sure the server finishes the reconstruction of the complete volume before the next data set arrives.

To conduct the experiment in a way we have described, we integrate the sample changer into *concert* and combine it with the DAQ system described in Section 5.2.2. *concert* orchestrates the DAQ on an acquisition computer, i.e. changes samples, acquires tomographic scans and writes images to the disk. We also implement a special data consumer which sends the projection stream to the reconstruction server.

Thanks to the usage of high-performance computing in combination with powerful hardware, the bottleneck of this experiment was the data acquisition and not the reconstruction part. This means that we were able to reconstruct a complete 3D volume of one sample before the scan of the next one finished and we reduced the time needed to acquire and reconstruct all data to 64 % of the time which would be required to do all the steps sequentially. The achieved speedup is very close to the theoretical limit 62 %, which is given by the fact that we have to acquire and stream data sequentally, but we can always reconstruct a data set in parallel with the acquisition of the next one.

This data acquisition scheme will be very important in the future for quality assurance and will enable unsupervised experiments, as well as increase the ratio of beam time used for data acquisition and dead time.

[1] `https://github.com/ufo-kit/ufo-filters`

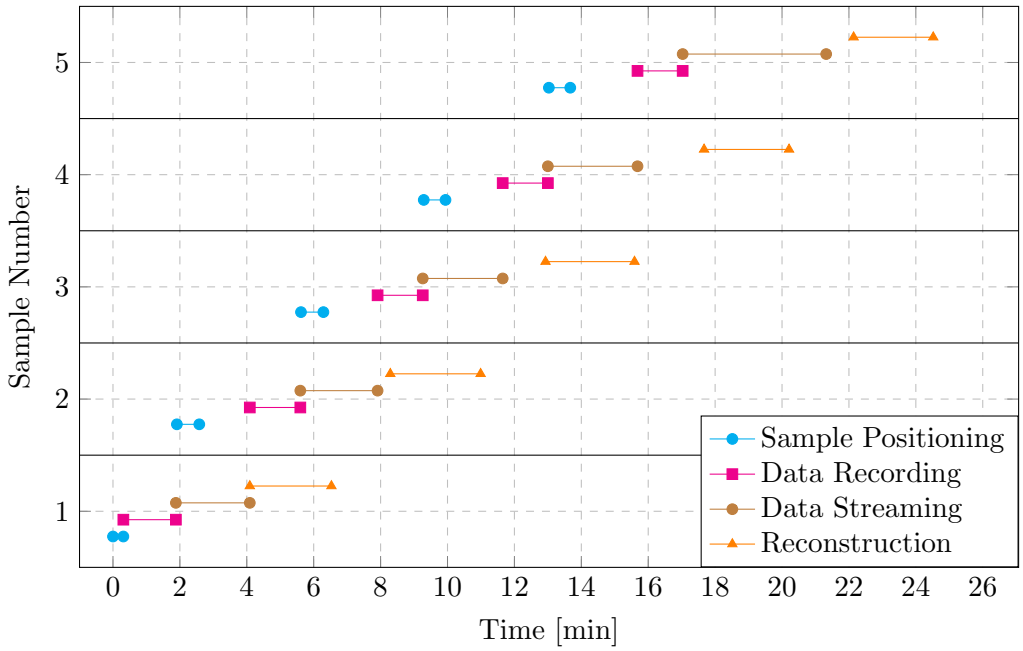
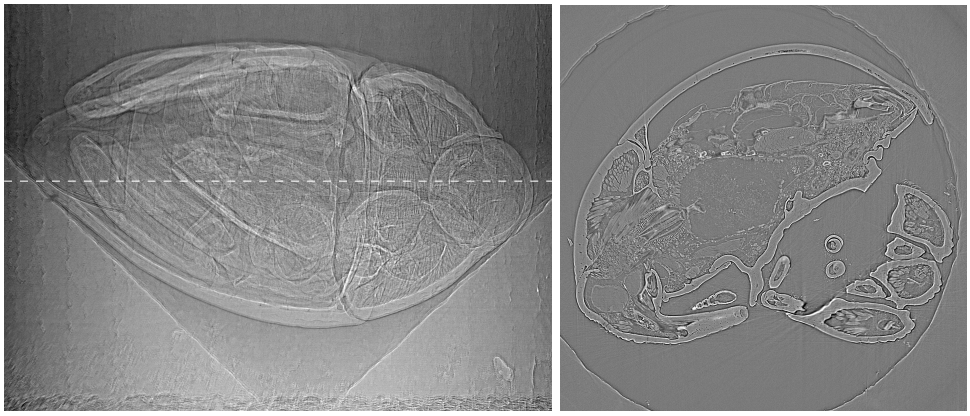


Figure 6.3: Experiment course for five samples. *Sample positioning* (cyan) is the placement of the sample on the rotation stage, *data recording* (magenta) includes normalization and projection data storage in the camera buffer, *data streaming* (brown) is the time required to send a data set to the reconstruction server and *reconstruction* is the duration of the 3D reconstruction of the complete volume, including the determination of sample position and the correct rotation axis. The reconstruction time is shorter than the recording and streaming time, thus the bottleneck of our experiment is the data acquisition and not the reconstruction.

6.3 ROI Positioning Applied to Laminography

This experiment, presented in [95], demonstrates the application of our system to *in-situ* laminography measurement of damage evolution in an aluminum sheet as a function of applied load force. The task of our system is to acquire data and provide fast 3D reconstruction of the sample to check its position and data quality.

Online 3D reconstruction is particularly important for laminography because the tilted rotation axis makes it more difficult to correctly position the sample if it is larger than the FOV of the detector and exhibits low contrast or periodic features. For such samples, it might happen that the ROI leaves the FOV during rotation, leading to missing data in projections and deteriorated reconstruction quality. An example is shown in Figure 6.5, where the sample with a notch (Figure 6.5a) was scanned, dismounted from the rotation stage in order to apply force on the material and repositioned back on the stage. Second scan in Figure 6.5b didn't capture the reference notch position anymore [96], which made data correlation impossible. Another example in Figure 6.6 shows blurring caused by local stress relaxation,



(a) Sample projection. (b) Reconstructed slice

Figure 6.4: Projection of a Trigonopterus weevil in (a) and a reconstructed slice in (b) marked by dashed line in (a).

which manifests in the reconstruction as double edge artifacts. It is very hard to see the subtle changes in the projections caused by the relaxation process, as opposed to the reconstructed slice in Figure 6.6. Thus, it would be very beneficial to have at least some slices shortly after the data is acquired.

The experiment was conducted at the ID19 beam line of the ESRF. The material microstructure is resolved by a detector based on an optical microscope [97], which magnifies the image of a 8.7 µm thick LSO:Tb scintillator onto a scientific CMOS camera (pco.edge), yielding an effective pixel size of 0.65 µm.

During a scan, *concert* broadcasts the incoming data stream simultaneously to the data storage and to our compute framework for reconstruction. The reconstructed central slice is investigated right after the scan to judge whether the acquired data is not blurred and the scan successful.

Figure 6.7 shows a properly aligned AA21xx alloy for a local shear loading case, in (a) without loading and in (b) under shear load. Despite the presence of inherent laminographic artifacts [98], we can observe the elongation of the pores in the material, which are roughly aligned with the principal shear force direction marked with arrows. The notch shape itself changed as well due to large plastic deformation.

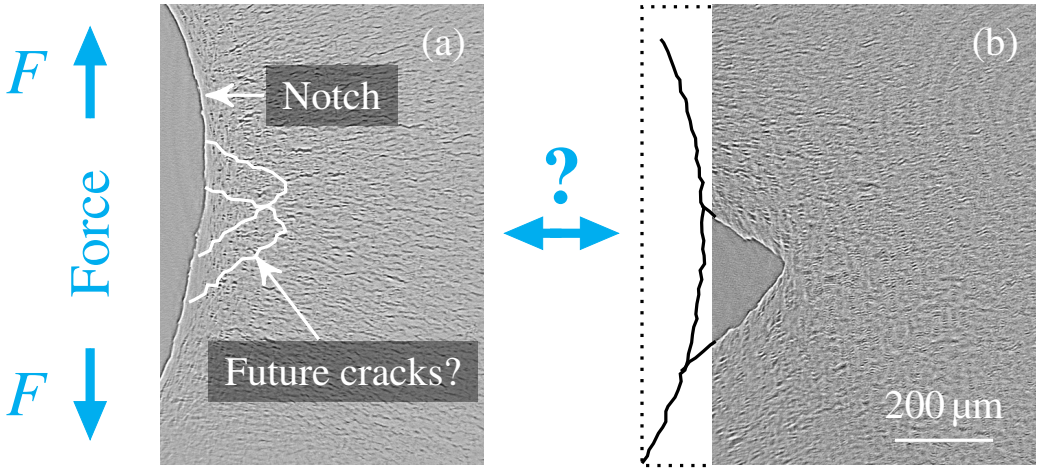


Figure 6.5: 2D section of reconstructed 3D *in situ* laminography data showing damage in a polyamide 6 specimen. The notch is visible in (a) but after sample repositioning due to applied force it is lost in (b). Figure from [95].

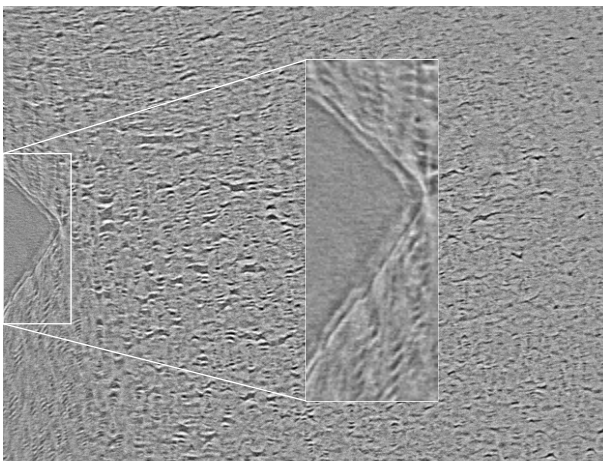


Figure 6.6: Double edge artifacts in a 2D section of a reconstructed 3D laminographic volume. The cause is local stress relaxation which is hard to see in projections. Hence, fast 3D reconstruction is needed to reveal such artifacts during the experiment, when it is still possible to reacquire the data if necessary. Local stress relaxation Figure from [95].

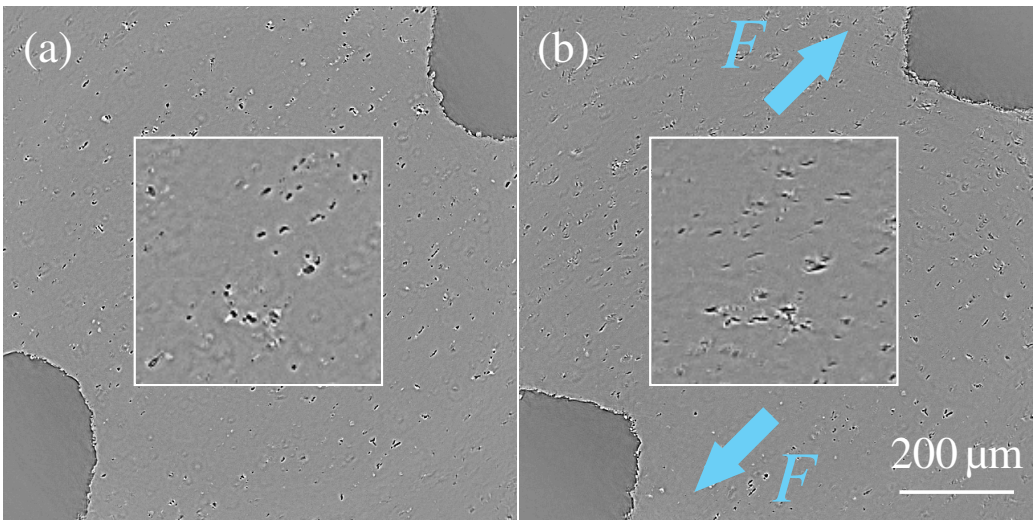
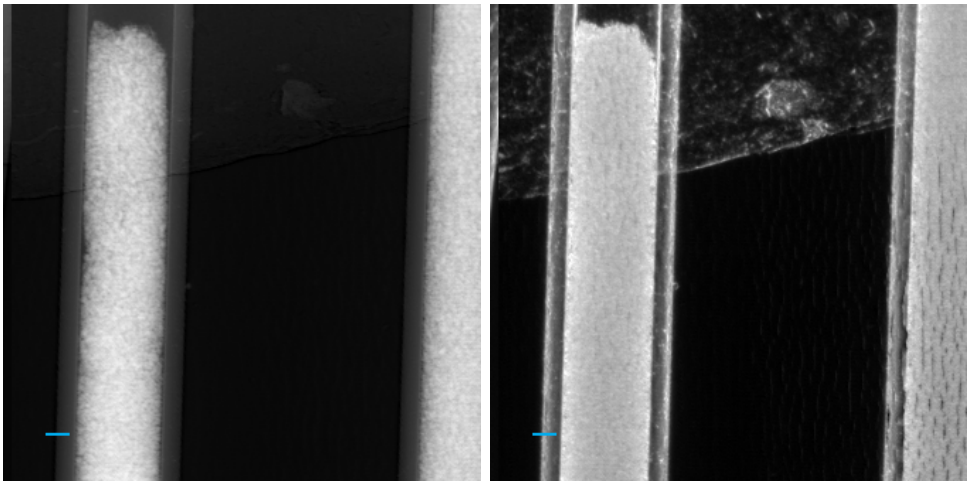


Figure 6.7: 2D section of an *in-situ* laminographic scan showing two well-aligned notches at mid-thickness in a 1 mm thick Al-alloy sample and a magnification of the smaller structures (a) in the undeformed state, and (b) *in situ* after local shear loading induced by force F. Figure appeared in [95].

6.4 Signal Z-dependence in Grating Interferometry

In Section 3.2.3, we simulated grating interferometry (GI) experiments to see how the contrast changes with various sample distances from the absorption grating (AG). In this section, we will compare the simulation with a real experiment to demonstrate that *syris* can be used to simulate X-ray imaging experiments with complex contrast formation mechanisms based on special optical elements.

We perform grating interferometry step scans as described in Section 3.2.3 with the sample positioned at various distances from the absorption grating (AG) to see the development of the absorption contrast (AC) and the dark field contrast (DFC). The sample is a glass capillary and we will focus on its edge, which according to Section 3.2.3 should give rise to AC and DFC contrast change based on the sample distance from the AG.

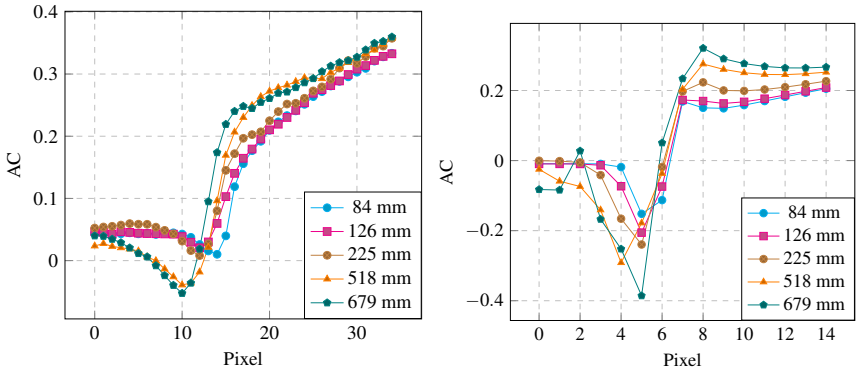


(a) AC (b) DFC

Figure 6.8: Absorption (a) and dark field (c) contrasts of a glass capillary placed right in front of the phase grating. The cyan line is used for analyzing the contrast dependence on the propagation distance in Figure 6.11.

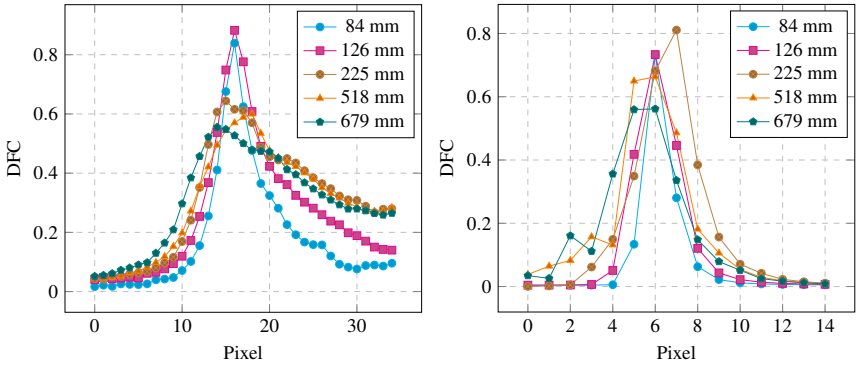
The experiment was conducted at the ID19 beam line of the ESRF because of its excellent coherence properties, which allow us to achieve strong contrast by grating interferometry. X-ray source was an undulator with beam energy 19.4 keV. Both gratings, the phase grating (PG) and the absorption grating (AG) had period 4.8 µm. PG was made of nickel and caused phase shift of the beam which travelled through its lamellas by $\pi/2$. It was placed 225 mm from the AG, which is the 5^{th} self-imaging distance. AG was made of gold and its lamellas acted as beam stoppers. It was placed directly in front of a CCD detector with 5 µm effective pixel size. The samples were placed 84 mm, 126 mm, 225 mm, 518 mm and 679 mm from the AG. AC and DFC obtained from the step scan are shown in Figure 6.8. The simulation uses the same sample, beam and grating properties as were used by the real experiment.

As we can see in Figure 6.9 and Figure 6.11a, AC grows with z which is a consequence of the fact that the free-space propagation gives rise to intensity modulations around sample edges observable in AC and simulation shows good agreements with real measurement.



(a) Data (b) Simulation

Figure 6.9: Absorption contrast profile obtained from real data in (a) and from simulation in (b).

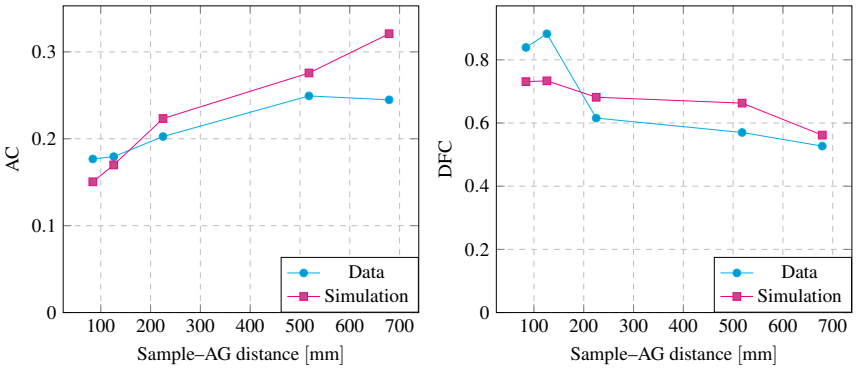


(a) Profile line (b) Signal at $x = 16$

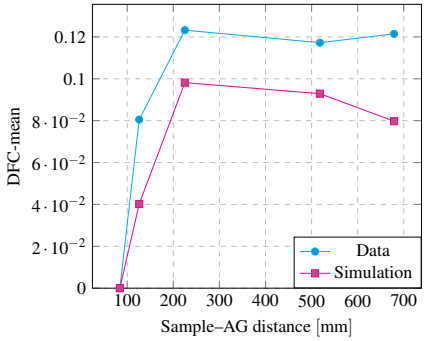
Figure 6.10: Dark field contrast profile obtained from real data in (a) and from simulation in (b).

The peak of the DFC profile depicted in Figure 6.10 decreases as the sample is positioned further away from the AG, shown in Figure 6.11b. However, this doesn't mean that the contrast is lost. The peak with increasing z becomes broader, thus the DFC signal is spread across several pixels. This is well reflected in Figure 6.11c, where we show the mean of the profile line normalized by its minimum. As we can see, the mean DFC actually rises as long as the sample is between the gratings, which is again well captured in the simulation.

The differences between the simulation and real measurement arise from defects



(a) Profile line (b) Signal at $x = 15$



(c) Signal at $x = 15$

Figure 6.11: AC signal at various sample distances two pixels right from the DFC peak in (a), DFC peak value in (b) and relative mean of the DFC profile in (c).

in the imaging setup, e.g. source blur caused by the extended source size, polychromaticity, grating and optical system imperfections. Despite these differences, the development of the contrasts in simulation shows good agreement with the real measurement thanks to the physical mechanisms included in *syris*. To achieve even closer match one could include the imperfections mentioned above. However, this is beyond the scope of our demonstration here.

6.5 Summary

In this chapter, we demonstrated the usage of our data acquisition system from Chapter 5 by conducting various experiments. Our first experiment was tomography of a changing liquid foam. We optimized the camera frame rate based on tomographic reconstruction and found an optimal data acquisition speed, which on the one hand enabled artifact-free 3D reconstruction, and on the other hand high SNR in the acquired projections.

Our second experiment was a *high-throughput* tomography. We used our system to exchange biological samples, determine sample alignment and use it to reconstruct

full 3D volumes *online*. Simultaneous data acquisition and processing enabled us to achieve sample throughput 12 samples/h.

The next experiment was *in-situ* laminography of damage evolution in aluminum sheets. Our system acquired data and performed *online* laminographic reconstruction which enabled us to check the sample alignment, data quality and ROI positioning.

The last experiment was grating interferometry of a glass capillary scanned at various distances from the detector. We compared the real experiment with simulated data obtained from our simulation framework *syris* and confirmed that it can be used to simulate X-ray imaging experiments with complex image formation mechanisms.

Chapter 7

Conclusion

The increasing complexity of experiments for high sample throughput, *in-situ* and even *in-vivo* imaging of structure evolution during technological or biological processes, combined with the fact that synchrotron beam time is a limited resource calls for robust and autonomous data acquisition schemes.

Experimental conditions and data processing algorithms need to be matched in such a way, that the analysis of the acquired data provides sufficient accuracy to answer a particular scientific question about the studied sample or process. To ensure the success of an experiment and save valuable beam time, it is important to first select the initial experimental conditions and data processing algorithms prior to the real measurement, and to then refine them *online* based on fast analysis of the acquired data.

This thesis addresses challenges related with modern X-ray imaging experiments, namely their preparation by *synthetic* experiments, their automation concerning the whole data acquisition and analysis pipeline and the control of both, the technological or biological process under study and the image diagnostics process based on *online* data analysis.

We investigated problems related to the implementation of X-ray imaging simulation and developed *syris*, a comprehensive framework which can be used to simulate a broad range of X-ray imaging experiments, including dynamics. Thanks to the extensibility of the framework, users can adjust existing modules or include more physical principles to investigate *novel imaging techniques*. In its current implementation, *syris* takes into account many important aspects of imaging experiments, namely the full light path from the X-ray source to the camera electronics, various approaches for the creation of sample shapes and motion in order to support up to 4D high-speed experiments. Because of the high computational cost, we chose approximations which on the one hand enable high-fidelity simulations and on the other hand fast, parallelized implementation in OpenCL. These properties make *syris* unique. One of its use cases is finding of suitable combinations of experimental and data processing conditions before or during the actual measurement in order to save valuable beam time. Moreover, it can be used by the image processing commu-

nity for rapid development of new and more robust pre-processing, reconstruction and analysis algorithms.

To enable 3D reconstruction-based feedback to the experiment, we investigated various reconstruction algorithms and selected filtered back projection for implementation because it enables us to quickly update volumes based on an incoming stream of projection data. In order to support many experiment types, we chose the laminographic geometry with tilted rotation axis. Implementation in OpenCL in combination with various optimization strategies enabled us to achieve 96 % of the performance of a state of the art tomographic back projection algorithm. We further investigated and implemented a strategy to reduce the number of transferred projection data to a video card, which combined with the usage of page-locked memory enabled us to increase the compute/copy ratio significantly, which led to better utilization of multi-GPU systems.

We extended a high-level control system *concert* by a versatile data acquisition scheme which enables image-based automated experiment control. Furthermore, we integrated our 3D reconstruction algorithm from Chapter 4 within a high-performance data processing framework, programmatically accessible from *concert*, thus the reconstruction algorithm can be used by our system. We also investigated various metrics for the determination of the sample alignment based on 3D reconstruction and integrated them into our system. We also employed our system to implement procedures for image focusing and rotation axis alignment.

For the demonstration of the capabilities of our system, we used it to conduct four experiments. During a tomography experiment, we optimized the camera frame rate to obtain 3D reconstruction without motion blur, which is important for successful further data analysis. We conducted a high-throughput experiment with a sample changer to automatically acquire many tomographic data sets and checked their quality based on fast 3D reconstruction. Thanks to overlapping acquisition and 3D reconstruction, we were able to achieve effective throughput 12 samples/hour. We also used our system during a laminography experiment to interactively control ROI positioning and data quality. Finally, we compared the grating interferometry simulations with real measurements to show that one can use *syris* to predict the impact of specialized X-ray optics to the contrast formation.

This work pushes the X-ray imaging field forward by enabling autonomous data acquisition schemes driven by image-based feedback. We saw that simulations can provide a good estimate about the experiment outcome, which can be used to optimize both the experimental and data processing parameters in order to initialize the real experiment. Our autonomous data acquisition scheme enables us to fine-tune the experimental and data processing parameters on-the-fly during the measurement in order to obtain the best data quality for answering a particular scientific question.

Future Outlook

Even though *syris* is able to conduct complete X-ray imaging experiments, there are various directions for its extension. The amount of X-ray instrumentation is

tremendous and one can include various X-ray sources and optical elements. Another interesting feature would be to incorporate precise dynamics of sample processes to increase the fidelity of sample behavior, e.g. fluid dynamics simulations. We see the most important and beneficial use case of *syris* in creating an open database of several experiment-specific data sets together with ground truth data. The imaging community could use this database to develop, train and benchmark novel algorithms.

Despite that we provide a versatile implementation of an algorithm for the 3D reconstruction of sample structure capable of reconstructing tomographic and laminographic data, it would be beneficial to extend the algorithm to support the cone beam geometry, which would be appreciated by the imaging community part which uses laboratory X-ray sources.

There are also more image-based control routines which could be automated, e.g. beam shape and spectrum adjustment based on the contrast in the acquired images.

As of now, the automation scheme is general and can be used for various experiments. Moreover, our system can be used to create specifically optimized automated workflows which would fit the needs of a particular experiment. Such workflows will enable new kinds of *high-speed* experiments which cannot be controlled by a human operator due to the fast data acquisition. Thus, our work will be an important cornerstone of experiments which will reveal new information about fast processes in life science and material research.

Bibliography

[1] Sebastian Thrun. Toward robotic cars. *Communications of the ACM*, 53(4):99–106, 2010.

[2] Tamim Asfour, Kristian Regenstein, Pedram Azad, J Schroder, Alexander Bierbaum, Nikolaus Vahrenkamp, and Rüdiger Dillmann. Armar-iii: An integrated humanoid platform for sensory-motor control. In *2006 6th IEEE-RAS international conference on humanoid robots*, pages 169–175. IEEE, 2006.

[3] Wilhelm Conrad Röntgen. On a new kind of rays. *Science*, pages 227–231, 1896.

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

[5] Jens Als-Nielsen and Des McMorrow. *Elements of modern X-ray physics*. Wiley, 2011.

[6] JW Jung, JS Lee, N Kwon, SJ Park, S Chang, J Kim, J Pyo, Y Kohmura, Y Nishino, M Yamamoto, et al. Fast microtomography using bright monochromatic x-rays. *Review of Scientific Instruments*, 83(9):093704, 2012.

[7] L Salvo, P Lhuissier, M Scheel, S Terzi, M Di Michiel, E Boller, JA Taylor, AK Dahle, and M Suéry. 3d in situ imaging of aluminium alloys during solidification. *Transactions of the Indian Institute of Metals*, 65(6):623–626, 2012.

[8] Donal P Finegan, Mario Scheel, James B Robinson, Bernhard Tjaden, Ian Hunt, Thomas J Mason, Jason Millichamp, Marco Di Michiel, Gregory J Offer, Gareth Hinds, et al. In-operando high-speed tomography of lithium-ion batteries during thermal runaway. *Nature communications*, 6, 2015.

[9] Tomy dos Santos Rolo, Alexey Ershov, Thomas van de Kamp, and Tilo Baumbach. In vivo x-ray cine-tomography for tracking morphological dynamics. *Proceedings of the National Academy of Sciences*, 111(11):3921–3926, 2014.

[10] Peter B Noël, Alan M Walczak, Jinhui Xu, Jason J Corso, Kenneth R Hoffmann, and Sebastian Schafer. Gpu-based cone beam computed tomography. *Computer methods and programs in biomedicine*, 98(3):271–277, 2010.

[11] Suren Chilingaryan, Alessandro Mirone, Andrew Hammersley, Claudio Ferrero, Lukas Helfen, Andreas Kopmann, Tomy dos Santos Rolo, and Patrik Vagovič. A gpu-based architecture for real-time data assessment at synchrotron experiments. *Nuclear Science, IEEE Transactions on*, 58(4):1447–1455, 2011.

[12] Vincent Van Nieuwenhove, Jan De Beenhouwer, Francesco De Carlo, Lucia Mancini, Federica Marone, and Jan Sijbers. Dynamic intensity normalization using eigen flat fields in x-ray imaging. *Optics express*, 23(21):27975–27989, 2015.

[13] David Paganin, SC Mayo, Tim E Gureyev, Peter R Miller, and Steve W Wilkins. Simultaneous phase and amplitude extraction from a single defocused image of a homogeneous object. *Journal of microscopy*, 206(1):33–40, 2002.

[14] Julian Moosmann, Ralf Hofmann, Andrei Bronnikov, and Tilo Baumbach. Nonlinear phase retrieval from single-distance radiograph. *Optics express*, 18(25):25771–25785, 2010.

[15] AC Thompson, J Llacer, L Campbell Finman, EB Hughes, JN Otis, S Wilson, and HD Zeman. Computed tomography using synchrotron radiation. *Nuclear Instruments and Methods in Physics Research*, 222(1):319–323, 1984.

[16] L. Helfen, A. Myagotin, P. Mikulík, P. Pernot, A. Voropaev, M. Elyyan, M. Di Michiel, J. Baruchel, and T. Baumbach. On the implementation of computed laminography using synchrotron radiation. *Review of Scientific Instruments*, 82(6):063702, 2011.

[17] B.K.P. Horn and B.G. Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

[18] Dzung L Pham, Chenyang Xu, and Jerry L Prince. Current methods in medical image segmentation 1. *Annual review of biomedical engineering*, 2(1):315–337, 2000.

[19] A. Munshi, B. Gaster, T.G. Mattson, J. Fung, and D. Ginsburg. *OpenCL programming guide*. Addison-Wesley Professional, 2011.

[20] Matthias Vogelgesang, Suren Chilingaryan, Tomy dos Santos Rolo, and Andreas Kopmann. Ufo: A scalable gpu-based image processing framework for on-line monitoring. In *Proceedings of The 14th IEEE Conference on High Performance Computing and Communication & The 9th IEEE International Conference on Embedded Software and Systems (HPCC-ICESS)*, HPCC '12, pages 824–829. IEEE Computer Society, 6 2012.

[21] M. Vogelgesang, T. Farago, T. dos Santos Rolo, A. Kopmann, and T. Baumbach. When hardware and software work in concert. In *Proceedings of the 14th International Conference on Accelerator & Large Experimental Physics Control Systems*, ICALEPCS '13, 2013.

[22] Tomas Farago, Petr Mikulík, Alexey Ershov, Matthias Vogelgesang, Daniel Hänschke, and T Baumbach. Syris: A flexible and efficient framework for simulation of x-ray imaging of dynamic processes. *To be submitted to the Journal of Synchrotron Radiation*.

[23] Joseph W Goodman. *Introduction to Fourier optics*. Roberts and Company Publishers, 2005.

[24] Max Born and Emil Wolf. *Principles of optics: electromagnetic theory of propagation, interference and diffraction of light*. CUP Archive, 1999.

[25] Standard for characterization of image sensors and cameras.

[26] B. Jaehne. *Digital image processing*. Number Bd. 1. Springer, 6th edition, 2005.

[27] Sébastien Harasse, Wataru Yashiro, and Atsushi Momose. Iterative reconstruction in x-ray computed laminography from differential phase measurements. *Opt. Express*, 19(17):16560–16573, Aug 2011.

[28] Timm Weitkamp, Christian David, Oliver Bunk, Jens Bruder, Peter Cloetens, and Franz Pfeiffer. X-ray phase radiography and tomography of soft tissue using grating interferometry. *European Journal of Radiology*, 68(3):S13–S17, 2008.

[29] V Altapova, J Butzer, TdS Rolo, P Vagovic, A Cecilia, J Moosmann, J Kenntner, J Mohr, D Pelliccia, VF Pichugin, et al. X-ray phase-contrast radiography using a filtered white beam with a grating interferometer. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 648:S42–S45, 2011.

[30] Martin D. de Jonge and Stefan Vogt. Hard x-ray fluorescence tomography – an emerging tool for structural visualization. *Current opinion in structural biology*, 20:606–614, 2010.

[31] Feng Xu, Lukas Helfen, Heikki Suhonen, Dan Elgrabli, Sam Bayat, Péter Reischig, Tilo Baumbach, and Peter Cloetens. Correlative nanoscale 3d imaging of structure and composition in extended objects. *PLoS ONE*, 7(11):e50124, 2012.

[32] Wolfgang Ludwig, Soeren Schmidt, Erik Mejdal Lauridsen, and Henning Friis Poulsen. X-ray diffraction contrast tomography: a novel technique for three-dimensional grain mapping of polycrystals. i. direct beam case. *Journal of Applied Crystallography*, 41(2):302–309, 2008.

[33] D Hänschke, L Helfen, V Altapova, A Danilewsky, and T Baumbach. Three-dimensional imaging of dislocations by x-ray diffraction laminography. *Applied Physics Letters*, 101(24):244103, 2012.

[34] Avinash C.. Kak and Malcolm Slaney. *Principles of computerized tomographic imaging*. IEEE press, 1988.

[35] Richard Gordon, Robert Bender, and Gabor T Herman. Algebraic reconstruction techniques (art) for three-dimensional electron microscopy and x-ray photography. *Journal of theoretical Biology*, 29(3):471–481, 1970.

[36] Peter Gilbert. Iterative methods for the three-dimensional reconstruction of an object from projections. *Journal of theoretical biology*, 36(1):105–117, 1972.

[37] Anders H Andersen and Avinash C Kak. Simultaneous algebraic reconstruction technique (sart): a superior implementation of the art algorithm. *Ultrasonic imaging*, 6(1):81–94, 1984.

[38] Kees Joost Batenburg and Jan Sijbers. Dart: a practical reconstruction algorithm for discrete tomography. *IEEE Transactions on Image Processing*, 20(9):2542–2553, 2011.

[39] DW Shattuck, J Rapela, E Asma, A Chatzioannou, Jinyi Qi, and RM Leahy. Internet2-based 3d pet image reconstruction using a pc cluster. *Physics in Medicine and Biology*, 47(15):2785, 2002.

[40] MD Jones, R Yao, and CP Bhole. Hybrid mpi-openmp programming for parallel osem pet reconstruction. *IEEE Transactions on nuclear science*, 53(5):2752–2758, 2006.

[41] Nicolas Gac, Stéphane Mancini, Michel Desvignes, and Dominique Houzet. High speed 3d tomography on cpu, gpu, and fpga. *EURASIP Journal on Embedded systems*, 2008:5, 2008.

[42] Maraike Schellmann, Sergei Gorlatch, Dominik Meiländer, Thomas Kösters, Klaus Schäfers, Frank Wübbeling, and Martin Burger. Parallel medical image reconstruction: from graphics processors to grids. In *International Conference on Parallel Computing Technologies*, pages 457–473. Springer, 2009.

[43] WJ Palenstijn, KJ Batenburg, and J Sijbers. Performance improvements for iterative electron tomography reconstruction using graphics processing units (gpus). *Journal of structural biology*, 176(2):250–253, 2011.

[44] Mark E Davison. The ill-conditioned nature of the limited angle tomography problem. *SIAM Journal on Applied Mathematics*, 43(2):428–448, 1983.

[45] L Helfen, T Baumbach, Petr Mikulík, D Kiel, P Pernot, P Cloetens, and J Baruchel. High-resolution three-dimensional imaging of flat objects by synchrotron-radiation computed laminography. *Applied Physics Letters*, 86(7):071915, 2005.

[46] A. Myagotin, A. Voropaev, L. Helfen, D. Hanschke, and T. Baumbach. Efficient volume reconstruction for parallel-beam computed laminography by filtered backprojection on multi-core clusters. *Image Processing, IEEE Transactions on*, 22(12):5348–5361, 12 2013.

[47] Alexey Voropaev, Anton Myagotin, Lukas Helfen, and Tilo Baumbach. Direct fourier inversion reconstruction algorithm for computed laminography. *IEEE Transactions on Image Processing*, 25(5):2368–2378, 2016.

[48] B Lai and F Cerrina. Shadow: A synchrotron radiation ray tracing program. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 246(1-3):337–341, 1986.

[49] O Chubar and P Elleaume. Accurate and efficient computation of synchrotron radiation in the near field region. In *proc. of the EPAC98 Conference*, pages 1177–1179, 1998.

[50] Franz Schäfers. The bessy raytrace program ray. In *Modern Developments in X-Ray and Neutron Optics*, pages 9–41. Springer, 2008.

[51] E Bergbäck Knudsen, Andrea Prodi, Jana Baltser, Maria Thomsen, P Kjær Willendrup, M Sanchez del Rio, Claudio Ferrero, Emmanuel Farhi, Kristoffer Haldrup, Anette Vickery, et al. Mcxtrace: a monte carlo software package for simulating x-ray optics, beamlines and experiments. *Journal of Applied Crystallography*, 46(3):679–696, 2013.

[52] Xianbo Shi, Ruben Reininger, M Sanchez del Rio, and Lahsen Assoufid. A hybrid method for x-ray optics simulation: combining geometric ray-tracing and wavefront propagation. *Journal of synchrotron radiation*, 21(4):669–678, 2014.

[53] Konstantin Klementiev and Roman Chernikov. Powerful scriptable ray tracing package xrt. In *SPIE Optical Engineering+ Applications*, pages 92090A–92090A. International Society for Optics and Photonics, 2014.

[54] Lili Zhou, KS Clifford Chao, and Jenghwa Chang. Fast polyenergetic forward projection for image formation using opencl on a heterogeneous parallel computing platform. *Medical physics*, 39(11):6745–6756, 2012.

[55] P Wenig and S Kasperl. Examination of the measurement uncertainty on dimensional measurements by x-ray computed tomography. In *Proceedings of 9th European Conference on Non-Destructive Testing (ECNDT), Berlin, Germany*, 2006.

[56] Michael A Sherman, Ajay Seth, and Scott L Delp. Simbody: multibody dynamics for biomedical research. *Procedia Iutam*, 2:241–261, 2011.

[57] Hrvoje Jasak, Aleksandar Jemcov, and Zeljko Tukovic. Openfoam: A c++ library for complex physics simulations. In *International workshop on coupled methods in numerical dynamics*, volume 1000, pages 1–20, 2007.

[58] Joyce E Farrell, Feng Xiao, Peter B Catrysse, and Brian A Wandell. A simulation tool for evaluating digital camera image quality. In *Electronic Imaging 2004*, pages 124–131. International Society for Optics and Photonics, 2003.

[59] A Götz, E Taurel, JL Pons, P Verdier, JM Chaize, J Meyer, F Poncet, G Heunen, E Götz, A Buteau, et al. Tango a corba based control system. *ICALEPCS2003, Gyeongju, October*, 2003.

[60] Jose Gabadinho, Antonia Beteva, Matias Guijarro, Vicente Rey-Bakaikoa, Darren Spruce, Matthew W Bowler, Sandor Brockhauser, David Flot, Elspeth J Gordon, David R Hall, et al. Mxcube: a synchrotron beamline control environment customized for macromolecular crystallography experiments. *Journal of synchrotron radiation*, 17(5):700–707, 2010.

[61] Leo R Dalesio, MR Kraimer, and AJ Kozubal. Epics architecture. In *ICALEPCS*, volume 91, pages 92–15, 1991.

[62] Kevin Mader, Federica Marone, Christoph Hintermüller, Gordan Mikuljan, Andreas Isenegger, and Marco Stampanoni. High-throughput full-automatic synchrotron-based tomographic microscopy. *Journal of synchrotron radiation*, 18(2):117–124, 2011.

[63] M.J. Flynn. Some computer organizations and their effectiveness. *Computers, IEEE Transactions on*, 100(9):948–960, 1972.

[64] J. Sanders and E. Kandrot. *CUDA by example: an introduction to general-purpose GPU programming*. Addison-Wesley Professional, 2010.

[65] David B Kirk and W Hwu Wen-mei. *Programming massively parallel processors: a hands-on approach*. Newnes, 2012.

[66] Yoshihiro Kanamori, Zoltan Szego, and Tomoyuki Nishita. Gpu-based fast ray casting for a large number of metaballs. In *Computer Graphics Forum*, volume 27, pages 351–360. Wiley Online Library, 2008.

[67] James F Blinn. A generalization of algebraic surface drawing. *ACM transactions on graphics (TOG)*, 1(3):235–256, 1982.

[68] Algorithms. *Commun. ACM*, 7(6):347–349, June 1964.

[69] Thomas van de Kamp, Patrik Vagovič, Tilo Baumbach, and Alexander Riedel. A biological screw in a beetle's leg. *Science*, 333(6038):52–52, 2011.

[70] Tomas Möller and Ben Trumbore. Fast, minimum storage ray/triangle intersection. In *ACM SIGGRAPH 2005 Courses*, page 7. ACM, 2005.

[71] Carl De Boor, Carl De Boor, Carl De Boor, and Carl De Boor. *A practical guide to splines*, volume 27. Springer-Verlag New York, 1978.

[72] Burton L Henke, Eric M Gullikson, and John C Davis. X-ray interactions: photoabsorption, scattering, transmission, and reflection at e= 50-30,000 ev, z= 1-92. *Atomic data and nuclear data tables*, 54(2):181–342, 1993.

[73] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.

[74] Peter Cloetens. *Contribution to Phase Contrast Imaging, Reconstruction and Tomography with Hard Synchrotron Radiation*. PhD thesis, Vrije Universiteit Brussel, 1999.

[75] P-A Douissard, Angelica Cecilia, Thierry Martin, Valentin Chevalier, Maurice Couchaud, Tilo Baumbach, Klaus Dupré, Markus Kuhbacher, and Alexander Rack. A novel epitaxially grown lso-based thin-film scintillator for microimaging using hard synchrotron radiation. *Journal of synchrotron radiation*, 17(5):571–583, 2010.

[76] GD Bergland. A guided tour of the fast fourier transform. *Spectrum, IEEE*, 6(7):41–52, 1969.

[77] Maciej Sypek, Michal GoÂ, et al. Image multiplying and high-frequency oscillations effects in the fresnel region light propagation simulation. *Optical Engineering*, 42(11):3158–3164, 2003.

[78] Earl J Kirkland. *Advanced computing in electron microscopy*. Springer Science & Business Media, 2010.

[79] Claude E Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, 1949.

[80] Stephen J Riederer, Nobert J Pelc, and David A Chesler. The noise power spectrum in computed x-ray tomography. *Physics in medicine and biology*, 23(3):446, 1978.

[81] S Zabler, A Ershov, A Rack, F Garcia-Moreno, T Baumbach, and J Banhart. Particle and liquid motion in semi-solid aluminium alloys: A quantitative in situ microradioscopy study. *Acta Materialia*, 61(4):1244–1253, 2013.

[82] Nils Papenberg, Andrés Bruhn, Thomas Brox, Stephan Didas, and Joachim Weickert. Highly accurate optic flow computation with theoretically justified warping. *Int. J. Comput. Vision*, 67(2):141–158, April 2006.

[83] Andrés Bruhn, Joachim Weickert, and Christoph Schnörr. Lucas/kanade meets horn/schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, 61(3):211–231, 2005.

[84] Deqing Sun, Stefan Roth, and MichaelJ. Black. A quantitative analysis of current practices in optical flow estimation and the principles behind them. *International Journal of Computer Vision*, 106(2):115–137, 2014.

[85] Atsushi Momose, Wataru Yashiro, Yoshihiro Takeda, et al. X-ray phase imaging with talbot interferometry. *Biomedical Mathematics: Promising Directions in Imaging, Therapy Planning, and Inverse Problems*, pages 281–320, 2009.

[86] Henry Fox Talbot. Lxxvi. facts relating to optical science. no. iv. *The London and Edinburgh Philosophical Magazine and Journal of Science*, 9(56):401–407, 1836.

[87] Wataru Yashiro and Atsushi Momose. Effects of unresolvable edges in grating-based x-ray differential phase imaging. *Optics express*, 23(7):9233–9251, 2015.

[88] Johannes Wolf, Jonathan I Sperl, Florian Schaff, Markus Schüttler, Andre Yaroshenko, Irene Zanette, Julia Herzen, and Franz Pfeiffer. Lens-term- and edge-effect in x-ray grating interferometry. *Biomedical optics express*, 6(12):4812–4824, 2015.

[89] Thomas Koenig, Marcus Zuber, Barbara Trimborn, Tomas Farago, Pascal Meyer, Danays Kunka, Frederic Albrecht, Sascha Kreuer, Thomas Volk, Michael Fiederle, et al. On the origin and nature of the grating interferometric dark-field contrast obtained with low-brilliance x-ray sources. *Physics in medicine and biology*, 61(9):3427, 2016.

[90] Michael Reed Teague. Deterministic phase retrieval: a green's function solution. *JOSA*, 73(11):1434–1441, 1983.

[91] Tilman Donath, Felix Beckmann, and Andreas Schreyer. Automated determination of the center of rotation in tomography data. *JOSA A*, 23(5):1048–1057, 2006.

[92] Doga Gürsoy, Francesco De Carlo, Xianghui Xiao, and Chris Jacobsen. Tomopy: a framework for the analysis of synchrotron tomographic data. *Journal of synchrotron radiation*, 21(5):1188–1193, 2014.

[93] Melvin E Conway. Design of a separable transition-diagram compiler. *Communications of the ACM*, 6(7):396–408, 1963.

[94] Walter Gander, Martin J Gander, and Felix Kwok. *Scientific Computing-An Introduction using Maple and MATLAB*, volume 11. Springer Science & Business, 2014.

[95] Matthias Vogelgesang, Tomas Farago, Thilo F Morgeneyer, Lukas Helfen, Tomy dos Santos Rolo, A Myagotin, and T Baumbach. Real-time image-content-based beamline control for smart 4d x-ray imaging. *Journal of Synchrotron Radiation*, 23(5), 2016.

[96] Yin Cheng, Lucien Laiarinandrasana, Lukas Helfen, Henry Proudhon, Olga Klinkova, Tilo Baumbach, and Thilo F Morgeneyer. 3d damage micromechanisms in polyamide 6 ahead of a severe notch studied by in situ synchrotron laminography. *Macromolecular Chemistry and Physics*, 2016.

[97] P A Douissard, A Cecilia, X Rochet, X Chapel, T Martin, T van de Kamp, L Helfen, T Baumbach, L Luquot, X Xiao, J Meinhardt, and A Rack. A versatile indirect detector design for hard x-ray microimaging. *Journal of Instrumentation*, 7(09):P09016, 2012.

[98] Feng Xu, Lukas Helfen, Tilo Baumbach, and Heikki Suhonen. Comparison of image quality in computed laminography and tomography. *Opt. Express*, 20(2):794–806, Jan 2012.

Nomenclature

ALU Arithmetic Logic Unit

API Application Programming Interface

APS Advanced Photon Source

ART Algebraic reconstruction technique

AVX Advanced Vector Extensions

CLI Command Line Interface

CORBA Common Object Request Broker Architecture

CPU Central Processing Unit

CUDA Compute Unified Device Architecture

EPICS Experimental Physics and Industrial Control System

ESRF European Synchrotron Radiation Facility

FBP Filtered Back Projection

FOV Field of View

FOV Field of View

FPGA Field Programmable Gate Array

FWHM Full Width at Half Maximum

GPGPU General Purpose GPU

GPU Graphics Processing Unit

GUI Graphical User Interface

GUPS Giga updates per second

MAD Median of Absolute Deviation

MIMD Multiple Instructions Multiple Data

MISD Multiple Instructions Single Data

OpenCL Open Computing Language

PSF Point Spread Function

SAG Sum of the Absolute Gradient

SDK Software Development Kit

SISD Single Instruction Single Data

SLS Swiss Light Source

SMT Simultaneous Multithreading

SNR Signal to Noise Ratio

SPMD Single Program Multiple Data

SSE Streaming SIMD Extensions

TIE Transport of Intensity Equation

TOMCAT TOmographic Microscopy and Coherent rAdiology experimenTs

USAXS Ultra Small Angle X-ray Scattering

XGI X-ray Grating Interferometry