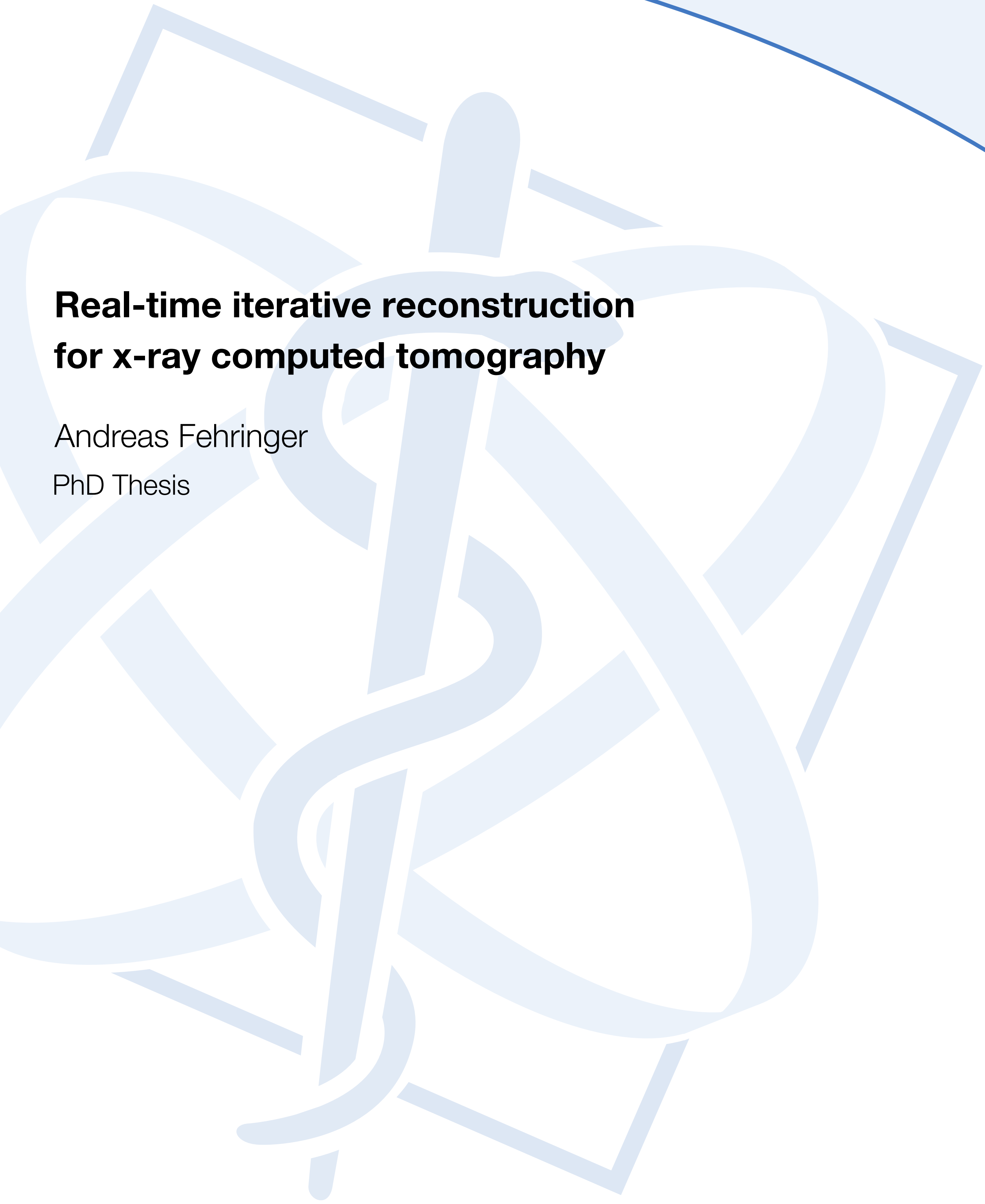
Lehrstuhl für Biomedizinische Physik
Technische Universität München



Real-time iterative reconstruction for x-ray computed tomography

Andreas Fehringer

PhD Thesis

Technische Universität München

TECHNISCHE UNIVERSITÄT MÜNCHEN
Physik Department
Lehrstuhl für Biomedizinische Physik

Real-time iterative reconstruction for x-ray computed tomography

Andreas Fehringer

Vollständiger Abdruck der von der Fakultät für Physik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:	Prof. Dr. Martin Zacharias
Prüfer der Dissertation:	1. Prof. Dr. Franz Pfeiffer
	2. Priv.-Doz. Dr. Tobias Lasser

Die Dissertation wurde am 08.01.2019 bei der Technischen Universität München eingereicht und durch die Fakultät für Physik am 16.09.2019 angenommen.

Abstract

We introduce a scientific reconstruction framework for tomographic reconstruction including an integral concept to make statistical iterative reconstruction a real-time approach for high-resolution computed tomography (CT). The practical consequence of our findings is a multiplication of the throughput in future high-resolution CT systems. The presented methods show how to make use of projective geometry and a heterogeneous single-computer system for this purpose. They include a way to formulate the problem efficiently and divide it into several components. The first set of optimization approaches deals with the synergy of the required components, the second with each component itself and the third with the Ordered-Subsets Momentum solver. It is shown in each step how the performance of our approach is connected to the current hardware limits. A low-statistics micro-CT measurement and a high-statistics reference are used to prove that our method is able to decrease the acquisition time at least by a factor of four at the same image quality. The timing results in about $42\,\mathrm{s}$ per iteration and an overall reconstruction time of $24\,\mathrm{min}$ for 18 iterations on the 2k-cubed voxel cone-beam volume. This is about 17 times faster than the total low-statistics acquisition time.

Zusammenfassung

Wir stellen ein Framework für tomografische Rekonstruktion im wissenschaftlichen Umfeld vor, das ein umfassendes Konzept enthält, um statistisch iterative Rekonstruktion ein Echtzeitverfahren für hochauflösende Computertomographie (CT) zu machen. In praktischer Konsequenz ermöglichen unsere Erkenntnisse eine Multiplikation des Durchsatzes künftiger hochauflösender CT Systeme. Die vorgestellten Methoden zeigen, wie man mittels projektiver Geometrie und einem heterogenen Einzelcomputersystem genanntes Ziel umsetzen kann. Sie beinhalten eine effiziente Formulierung des Problems und einen Weg, es in mehrere Bestandteile aufzuteilen. Die erste Gruppe der vorgestellten Optimierungsansätze behandelt das Zusammenspiel dieser Bestandteile, die zweite die Bestandteile selbst und die dritte das Ordered-Subset Momentum Lösungsverfahren. In jedem Schritt wird gezeigt, wie die Leistungsfähigkeit unserer Ansätze mit den Fähigkeiten aktueller Hardware zusammenhängt. Eine Mikro-CT Messung mit geringer Statistik und eine Referenzmessung mit hoher Statistik werden herangezogen, um unter Beweis zu stellen, dass durch unsere Methode die Messzeit bei gleichbleibender Bildqualität um mindestens Faktor vier verringert werden kann. Die Zeitmessung ergab etwa $42\,\mathrm{s}$ pro Iteration und eine Rekonstruktionszeit von insgesamt $24\,\mathrm{min}$ für 18 Iterationen auf dem 2k-Voxelkubus mit Kegelstrahlgeometrie. Dies ist etwa 17 mal schneller als die gesamte Messzeit des verwendeten Testdatensatzes.

For in him [, Jesus Christ,] all things were created:
things in heaven and on earth,
visible and invisible,
whether thrones or powers or rulers or authorities;
all things have been created through him and for him.
He is before all things, and in him all things hold together.

The Apostle Paul
(Col 1:16-17)

Contents

Abstract v

Contents x
List of Figures xii
List of Algorithms xiii

1 Preface 3
1.1 What the world is all about 4
1.2 Faith and the difference it makes for research 5
2 Introduction 9
2.1 Motivation 10
2.2 Overview 11

I Theory 13
3 X-ray computed tomography 17
3.1 Data acquisition and challenges from the real world 18
3.2 Analytical reconstruction 21
3.3 Iterative reconstruction 25
4 Projective geometry 31
4.1 Mathematical background 32
4.2 Projection matrices in CT 33
4.3 Some special use cases 36
5 Heterogeneous computing 45
5.1 Basic concepts 46
5.2 GPGPU computing 49

II Implementation 53
6 The building blocks 57
6.1 Preliminary considerations 58
6.2 The kernels 61
6.3 Handling big data 64
6.4 Remarks on the OpenCL framework 68
7 The reconstruction framework 71
7.1 Visions and their realization 72
7.2 An overview 74
8 Integral optimization 79
8.1 The right components and their synergy 80
8.2 Component-wise optimization 82
8.3 Optimizing the minimizer 86

III Results 91
9 Real-time SIR for micro-CT 95
9.1 Constraints and potential 96
9.2 Benchmark and profilings 98

Outtake 107

Appendix 109
A Derivations of the cost function 110
B A possible definition of the Euler angles 113
C Computing the parallel-beam direction vector 114
D Optimizing the workgroup size without considering transfers 115

Acknowledgments 119
Picture credits 120
List of publications 122
Bibliography 129

List of Figures

1 A camel supposed to fit through the eye of a needle.. 4
2 A witness to early western research.. 5
3 An early x-ray image. 10
4 The development of x-ray sources over time.. 19
5 Interaction cross section over the energy 20
6 A simple image and its sinogram 23
7 FBP with noise and undersampling 23
8 Geometry required for the FDK reconstruction.. 25
9 The Huber penalty function. 27
10 Two straight rails touch at infinity... 32
11 The 2-D projective and Euclidean space. 33
12 A schematic of the CT geometry with all relevant measures 34
13 A fully-automated rectified reconstruction 37
14 Microprocessor evolution over the last 45 years 46
15 Theoretical CPU and GPU single-precision peak performance over the last decade 47
16 The GPU memory layout 50
17 The hierarchy inside the Geometry class 61
18 Two basic types of tomographic projectors. 62
19 Data splitting for cone-beam CT computations on GPUs.................................. 64
20 Mean runtime for the gradient on heterogeneous hardware.............................. 82
21 Performance of FP and BP over the workgroup shape..................................... 86
22 Performance of the BP over the walk parameter 87
23 A reference for the convergence of SIR 88
24 Runtime for different numbers of subsets in SIR 89
25 Several quality measures over the interations..100
26 Benchmark quality reference: A real sample 101
27 Profiling SIR..103
28 Outtake 107
29 Performance of over the workgroup shape (FP, BP without transfers) 115

List of Algorithms

1 The filtered backprojection (FBP) 22
2 The method of Feldkamp-Davis-Kress (FDK) 25
3 Statistical iterative reconstruction (SIR) with a Gaussian noise model 26
4 A popular regularization term 27
5 Bit-reversal sorting implemented in Python 29
6 Useful operations with projection matrices 35
7 Fully automated sample rectification 38
8 Assessing the maximum number of float32 terms in a sum 49
9 The OS-OGM 2 implemented in Python 80
10 Derivatives of the cost function 111

1. Preface

What is this world all about? - This fundamental question about life usually evokes two different kinds of reactions: One is laughter. Many people resigned on finding an answer. The other is struggle. Struggle means, accepting the challenge and starting to search because a meaningless world is unacceptable. During my time at university I found that many of my fellow students were, just like me, on the second track. This question was the driving force for my studies.

Picture: A tribute to Michelangelo's famous painting

The creation of Adam. ©

Before starting the scientific examination of my topic *real-time iterative reconstruction for x-ray computed tomography* I would like to explain my motivation for doing science. The intention of an author is crucial for the interpretation of any text. We know that there is a great difference whether an author intended to write a fictional scenario or a factual report.

Scientific texts are usually not fiction. However, the motivation and assumptions of the author still play an important role. History has shown that research was often carried out with predefined goals or limits. These were implied by wrong convictions of the scientific community at a certain time, the personal interests of the scientist or those of his or her funders.

Among the conclusions I was allowed to draw from my years of study, the engagement in science and everything around it, I learned that the underlying beliefs are crucial to science and all other areas of life.

1.1. What the world is all about

> Jesus answered, "If you want to be perfect, go, sell your possessions and give to the poor, and you will have treasure in heaven. Then come, follow me." When the young man heard this, he went away sad, because he had great wealth. Then Jesus said to his disciples, "Truly I tell you, [...] it is easier for a camel to go through the eye of a needle than for someone who is rich to enter the kingdom of God." When the disciples heard this, they were greatly astonished and asked, "Who then can be saved?" Jesus looked at them and said, "With man this is impossible, but with God all things are possible." [1]

Figure 1 A camel supposed to fit through the eye of a needle.
An illustration of the famous figure of speech used by Jesus Christ to describe the chances of a rich man to enter the kingdom of God by his own means. © by Judith Ganter.

This very story was the beginning of my concrete search for the meaning of this world and especially of my life. I do not remember where I first heard it, but these powerful words took root deeply in my mind. They did not stop bothering me for two or three years until I finally found answers. Later, I was astonished to notice that all of these words had already been included in this exact same story in Matthew 19 in the Bible.

Leaving your treasures behind and following Christ in order to enter the kingdom of God. This seems, at first sight, to be God's answer to the purpose of life in this passage. But what would that mean? Does God require to abstain from all pleasures? Does a life devoted to Christ mean to become a priest or joining a monastery? And what about the camel not fitting through the eye of a needle? (Figure 1 tries to meet the picture in my mind.) Is there any chance at all? Which standard has to be applied to life? What is good and how good is good enough? Or is there a god just throwing dice in the end [2]?

In 2006, I started to study physics. It is the mother and still the most fundamental of all

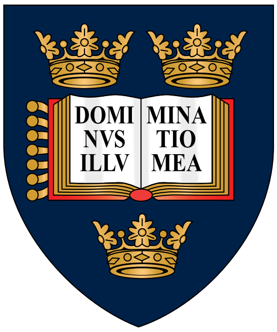


Figure 2 A witness to early western research.
The inscription of the coat of arms of Oxford University says *The Lord is my Light*. In the Bible, the coming to the light is related to finding the truth.

natural sciences and I expected to find reliable answers to life there. Contrary to my hopes, however, my thirst did not get satisfied in one of the great lectures of this renowned university. It was in a small group of students reading the Bible. The answer I found is brilliantly summarized in John's world-famous verses:

> For God so loved the world that he gave his one and only Son, that whoever believes in him shall not perish but have eternal life. [3]

And in Paul's letter to the Ephesians:

> For it is by grace you have been saved, through faith—and this is not from yourselves, it is the gift of God—not by works, so that no one can boast. [4]

Among all the things I studied and worked for only the words from the Bible could satisfy my need for answers. Yes, I can say, it turned my life from vanity into magnificence. A savior that makes imperfect, selfish and weak human beings match his majesty and generous character is exactly what this world lacks. His grace alone is sufficient. We are created to benefit from it and, therefore, give all glory to him and no more to ourselves. This is the key fitting into the hole. The conviction of absolute truth, justification by Christ's grace alone, and the relationship to the living God have not only become the answer to an intellectual question for me, but the basis and motivation for my life and especially my research.

1.2. Faith and the difference it makes for research

During my studies I was able to attend a one-week seminary at one of the oldest universities: Cambridge University. I was amazed by the multitude of reminders of an almost lost thinking: college names like *Trinity*, *St. John's*, *Christ's*, or *Jesus*, and the chapel as the central building of each school—just a few witnesses that remained from the beginning of western research and higher education.

Another interesting piece of very early academic heritage spatially not too far away is shown in figure 2. It is the coat of arms of Oxford University. The Latin inscription says *The Lord is my Light*. This quote from the beginning of Psalm 27 reminds us in its original context of the Lord as savior and source of all power. Further, it hints to one of the names of the Lord Jesus Christ, who is called *The Light of the World* which reveals the truth according to the gospel of John. Jesus explains that the truth can be found by remaining in his word and that only the truth can set men free. [5]

I consider this statement to be one of the most important foundations of education and research, especially in times where access to the truth and freedom of speech is heavily restricted for many people. Usually, we tend to first think of some remote dictatorial regimes where freedom of speech is restricted. But even in our culture a liberal and self-centered worldview is increasingly propagated without allowing rational discussion. This is an alarming circumstance.

Last year, the 500th anniversary of the Reformation in Germany reminded us once again of the Christian roots of our culture and the basic Christian convictions. The German state secretary Thomas Rachel said in an interview to the German federal ministry for education and research (BMBF):

> „Die Freiheit der Forschung vor jeder unsachgemäßen und fachfremden Bevormundung wurzelt für mich genuin im protestantischen Freiheitsverständnis und in dem durch die Rechtfertigungslehre begründeten Menschenbild.“ [6]
>
> (The freedom of research before any inappropriate and non-specialist dictation is for me genuinely founded in the Protestant understanding of freedom and in the conception of the human being originating from the doctrine of justification.)

Only the truth will set us free. And research should be all about finding and defending it. The existence of absolute truth equally valid for all people—independent of their location, time and beliefs—is the very basis of science. Theories are only valid if they can globally be confirmed or falsified. Unfortunately, this very foundation of free science is nowadays profoundly attacked by modern relativism and lifestyle. If truth as the highest goal is put aside, rational research gets replaced by arbitrary results. The most dangerous threads for truth probably remained the same throughout history: self realization, career prospects, personal success and wealth which are propagated as modern lifestyle. They are base motives deep inside human nature. I am, for sure, not excluded from their influence. But, if at all, the way to overcome them can only be with a perspective beyond mankind.

Coming back to the the old Oxford statement *The Lord is my Light*, it has often been argued that a biblical worldview led to modern science [7, for example]. The biblical understanding of truth perfectly meets the requirements. The God of the Bible did not remain mere theory. He *became flesh and made his dwelling among us* [8] and if we search for him with all our mind and heart, as he calls us to do, he reveals himself. He can be experienced, which is the very nature of truth. This experience allowed early pioneers in science to be confident that in the same way as God reveals himself as a person, he also reveals the natural laws of his creation for those who search. And they were right. Science is known to work great. Today, however, it is important to note that it is not the only access to truth and by definition limited [9, for example]. It can teach us the *how*'s of this world, but we need the creator himself to tell us the *why*'s.

My personal motivation for doing science is to give glory to my Lord and creator by searching for the truth, maybe revealing a little more of his genius, and by knowing that all good thoughts are not achievements, but gifts.

2. Introduction

When Wilhelm C. Röntgen discovered x-rays in 1895 [10] and received the first Nobel price for Physics in 1901, he was not yet aware of the great implications of his discovery. These *new kind of rays* allow deeper insights in astronomy as well as nano technology. Their greatest value has been found in medical applications, where x-rays have played an important role in diagnostics since the very beginnings.

Picture: Wilhelm C. Röntgen imaged with visible light. 🄯

2.1. Motivation

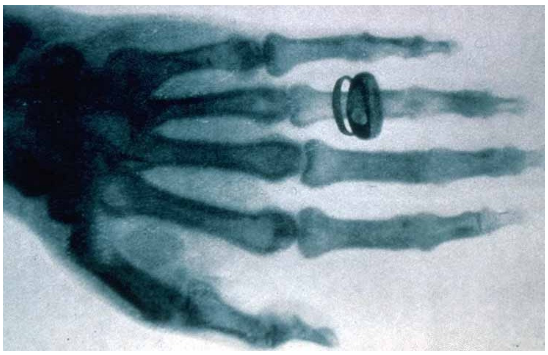


Figure 3 An early x-ray image.
W. C. Röntgen imaged the hand of a fellow professor, Albert von Kölliker, in 1896.

Today, same as 120 years ago when Röntgen took the image shown in figure 3, x-ray physics is still an active field of research. In 1979, another Nobel Price is awarded to Sir Godfrey N. Hounsfield and Allan M. Cormack for their part in developing a technique based on x-rays called computed tomography (CT) [11]. A computer algorithm helps to reconstruct a three-dimensional (3-D) map of the local x-ray attenuation coefficients inside a sample from a set of transmission images. Only one decade after the first clinical CT was built, the foundation for another important application was lain: x-ray micro tomography (micro CT) [12]. Since then, synchrotrons and laboratory sources have provided great 3-D insights into the microscopic and even nanoscopic world. Today it plays a major role in industrial non-destructive testing with a steeply increasing number of applications.

Fast analytical reconstruction approaches made it possible to make CT a wide-spread technique that can be found in most modern hospitals around the world. Their high demands on the acquisition process, especially concerning noise and angular sampling, drew the attention of the CT community already very early to iterative techniques [13, 14]. The high computational requirements have, however, almost let it fall into oblivion again during the last decades. Only recently the vast computing power started to make those techniques attractive again and they have become a very active field of research. Especially the massive development of graphic processor units (GPU) pushed by the gaming industry lead to great advances in cheap computing power. Some basic iterative features are thus already implemented in commercial devices in the clinics. A long-term case study showed that these methods allow to reduce the dose of the hazardous radiation during the acquisition [15].

A very recent and promising class of new approaches is model-based reconstruction. It directly takes into account the physical effects known to lower the image quality in order to receive the most realistic image. Statistical iterative reconstruction (SIR) is, in contrast to many commercial solutions [16], fully iterative. It is able to model the whole process and drastically reduce the high noise that has been characteristic for CT images over decades [17]. Advanced models further allow to tackle source and detector blurring [18], time-resolved measurements [19] and task-based reconstructions like material decomposition [20]. First commercial realizations were implemented recently [21].

It is still a long way to go until the full capabilities of SIR will be exploited. Active exchange with the leading manufacturers of medical and industrial CTs over the last few years has shown that the main obstacles for broader application are usability and reconstruction time. The work described in this thesis emerged from the investigation of these two obstacles.

2.2. Overview

The main result of this work is that SIR can be run as real-time application especially for the big data sizes relevant in industry and science. For that means an integral pipeline is presented that goes from investigating available methods and selecting the components meeting best the requirements to their implementation from scratch, optimization and synchronization. In the end, some basic showcases prove the value of these efforts for micro-CT applications.

Before going into details, it is important to understand the driving idea behind SIR, its functional principle and its potential. Part I, the theory, therefore provides all the background information required. After a short introduction into x-ray physics and x-ray imaging, it concentrates on tomographic reconstruction (chapter 3). There are two different basic approaches: The filtered backprojection (FBP), the by far most successful representative of very fast analytical approaches and statistical iterative reconstruction (SIR), the currently most promising representative of the family of high-quality iterative techniques. The first part also includes an introduction to projective geometry (chapter 4), a very powerful mathematical tool to describe the geometric parameters of a CT scan and consider it in reconstruction. It ends with an introduction to heterogeneous computing (chapter 5), which is computing on different hardware according to a specific task. The configuration that turned out most suitable for the implementation presented below is general-purpose computation on graphics processing units (GPGPU). It supports massive parallelization on compact and cheap hardware as well as a very high data throughput.

Part II, the implementation, introduces a fast, but flexible SIR framework. It tries to meet the requirements of a scientific environment, where research not only takes place on reconstruction methods but also on CT applications. Therefore, it tries to combine generality with ease of use and flexibility with speed. The compute-intensive parts can be reduced to several common core components, which are the same for most algorithms. A fast GPGPU framework is introduced holding the most important core components (chapter 6). Those are the tomographic forward (FP) and back projector (BP) and a general local regularizer. Subsequently, a generic SIR framework is presented with its modular structure, usability concept and an overview of investigated methods (chapter 7). The implementation part closes with an integral optimization of the whole implementation with respect to maximum performance which is especially important for big data sets (chapter 8).

The thesis closes with part III, the results from the suggested methods. It demonstrates that the benefits of SIR were made accessible for full-size micro-CT data sets. Further, the real-time performance of the presented implementation is proven for a data set with a volume size of $(2048)^3$ voxels (2k-cubed) (chapter 9).

Part I

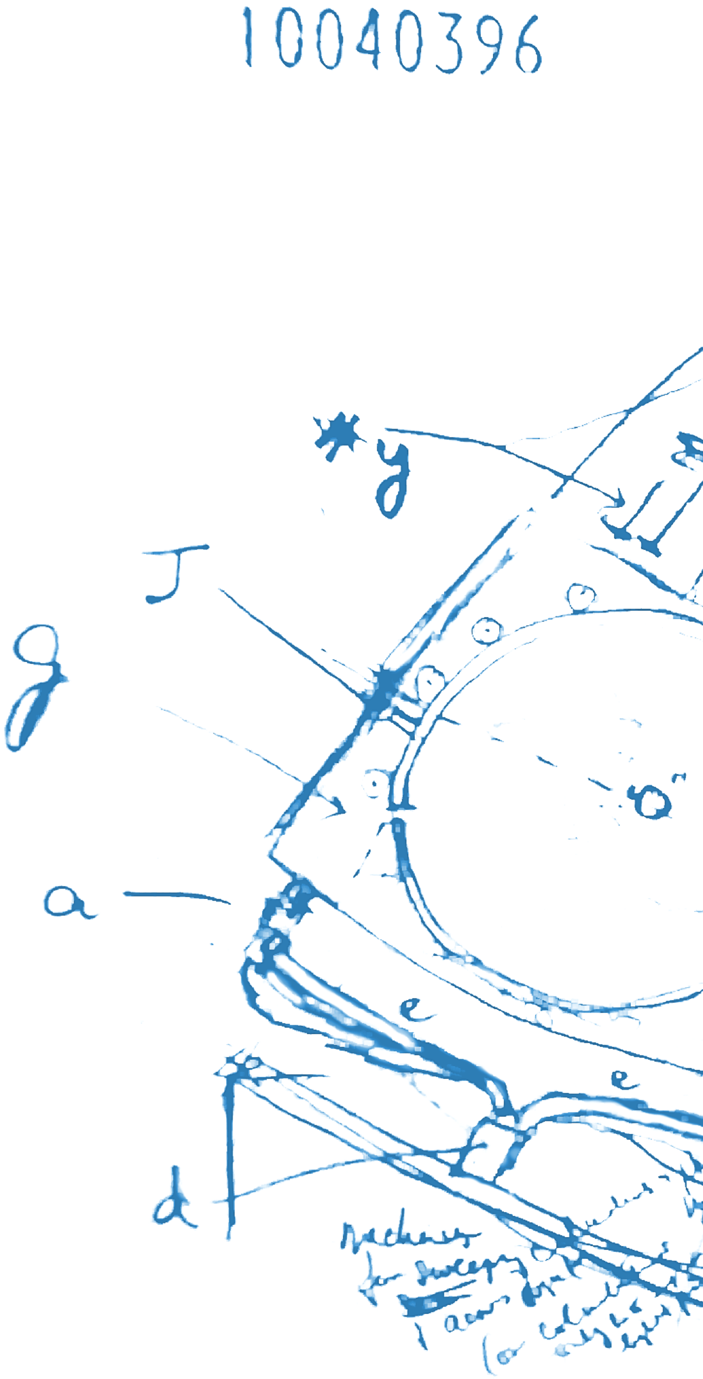
Theory

The first part provides the physical and mathematical background required for the methods presented later. First, there is an introduction to x-ray computed tomography (chapter 3) explaining the basics of tomographic data acquisition with x-rays and two very common approaches for tomographic reconstruction. Second, there is an introduction to projective geometry (chapter 4) that deals with the mathematics of projecting a 3-D setting into a plain with a given camera geometry. It is shown, how it can be used for computed tomography and some special use cases. Third and last, there is an introduction to heterogeneous computing (chapter 5). After some short thoughts about computing on multiple architectures and parallel computing, the GPU architecture and ways to use it for general-purpose computations (GPGPU) are investigated.

3. X-ray computed tomography

X-ray computed tomography is an imaging modality for non-destructive 3-D investigation. It is very common in medical applications and becomes increasingly popular for non-destructive testing in industry and science.

The most common alternatives for non-destructive examination are x-ray radiographs, light microscopy, ultrasound imaging and magnetic resonance imaging (MRI). In contrast to x-ray CT, the last three do not carry the risk of radiation damage. For radiographs that risk is reduced by a lot. Moreover, light microscopy, ultrasound and radiographic imaging are much cheaper. The great asset of x-ray CT is its by far most accurate spatial information. Especially the three cheap methods allow only very limited 3-D investigation. Compared to MRI, x-ray CT is a lot faster, cheaper and more accurate.



Picture: A sketch drawn by Sir Godfrey N. Hounsfield. 🅭

3.1. Data acquisition and challenges from the real world

X-ray computed tomography allows to create a 3-D map of coefficients that describe how strongly the x-rays interact with the material. This map allows to distiguish different materials or mass densities. The working principle of CT consists of two basic steps. The first is data acquisition in which the desired quantity can be measured only indirectly. The second is reconstruction on a computer where the 3-D map of coefficients is retrieved from the measurement. Gray-scale images representing 2-D planes through the volumetric data are the most common way to assess this map for humans.

The basic setup for a tomographic scan consists of an x-ray source placed on one side of a specimen, a 2-D detector placed at the opposite side and a mechanical mechanism that either rotates the specimen in the middle perpendicularly to the x-ray beam or, vice versa, the two devices around the specimen.

Detailed information about x-ray sources, the interaction of x-rays with matter, and the different acquisition techniques described can be found in literature. [22, for example]

3.1.1. The x-ray source

There are several ways to generate x-rays. They differ in complexity and brilliance. The brilliance is defined as the number of emitted photons per time divided by the source spot size, the solid angle of the emitted photons and the bandwidth of their spectrum. Another common measure is the flux, which is the number of photons per time and unit area.

The simplest way to generate x-rays is to shoot electrons onto a metal target. X-rays then either come from the negative acceleration of electrons that are stopped inside the material or they come from bound electrons filling up vacant places of the inner atomic shell, where an electron was kicked out before. The radiation coming from the first process has a broad energy spectrum and is called *Bremsstrahlung*. The radiation coming from the second process is called characteristic radiation. Its sharp energy peaks are characteristic to the target material, because they are determined by the energy levels of the atomic electron shell.

The most common implementation of that idea is the x-ray tube. Thermal electrons are accelerated and focused onto a rotating metal target inside a vacuum closure. The spectrum and flux can be varied by the acceleration voltage and the electron current. Preventing the target from melting requires external cooling. Usually a rotating anode is used to help distributing the heat. Its advantages made that source the preferred choice for most applications. It is comparatively small, easy to operate and cheap. Disadvantages are, however, the broad energy spectrum, the big source spot size and the limited flux due to the heat it produces.

In CT applications, the broad spectrum can lead to so-called beam-hardening artifacts as discussed later. The extend of the source spot which is necessary to avoid heat damages on the anode causes blurring. A variant of that type of source invented to circumvent that problem is the micro-focus tube. It has a smaller spot size that comes with the cost of a reduced flux. In turn, it requires longer exposure times in order to obtain equivalently high photon statistics and low image noise. Another advancement is the liquid metal jet tube [23]. It uses a stream of liquid metal as anode and thus tries to bypass the problem of anode damage. Ideally, the flux can thus be improved by about an order of magnitude.

Sources with a higher brilliance can only be found in larger scientific facilities. Synchrotron sources can have more then ten orders of magnitude higher brilliance than a tube. X-rays are generated from accelerating very fast electrons in different kind of magnets. Free electron lasers promise rather twenty orders of magnitude. The expensive and limited resources are of course reserved for special applications. An overview of the available brilliance over time is given in figure 4.

A promising, neither table-top, nor facility-filling new lab-based source is the compact light source [24, 25]. It takes advantage of inverse Compton scattering by colliding an electron beam with a laser beam and exceeds conventional sources by one or two orders of magnitude in brilliance.

3.1.2. X-rays and their interaction with matter

X-rays are electro-magnetic radiation with a wavelength in the Ångström range ($\sim 10^{-10}\,\mathrm{m}$). Similar to visible light they can be described as photon particles or waves. The interaction with matter can take place in several ways.

Elastic (or coherent) scattering is an interaction of an x-ray photon with a charged particle. It can happen with single particles or whole structures, for example a crystal. The energy of the scattered x-ray remains constant but its direction changes.

An inelastic (or incoherent) scatter event, also called Compton scattering, describes the collision of a photon with another particle. In contrast to elastic scattering, energy is transferred from the photon to the other particle. The difference of the wavelength of the in- and outgoing photon can be expressed as

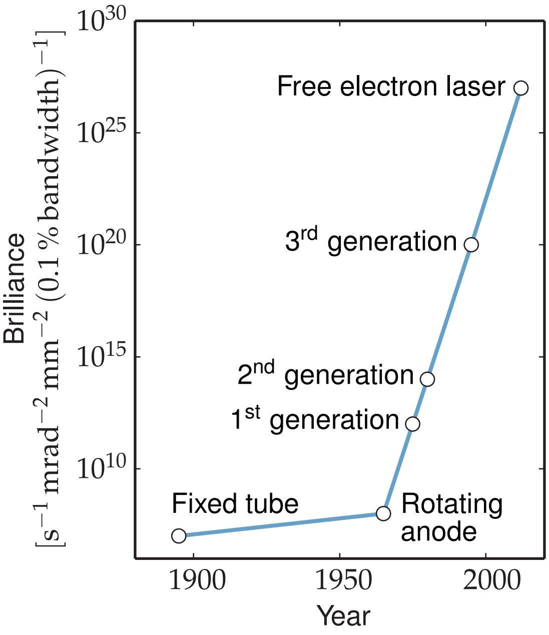


Figure 4 The development of x-ray sources over time. Als-Nielsen et al. illustrate the maximum available brilliance of x-ray sources since x-rays were discovered. The graph was taken from their book [22, fig. 1.1, p. 2]. © by John Wiley and Sons.

$$\lambda' - \lambda = \lambda_C (1 - \cos\psi),$$
$$\text{with} \quad \lambda_C = \frac{h}{m_e c},$$

where λ_C is the Compton wavelength and ψ the scatter angle.

X-rays can also be absorbed by an atom. Its energy is transferred to an electron which is then able to overcome its binding and leave the atom. This process is known as photoelectric effect.

An overview over the cross sections of these effects and some additional higher-energy interactions is given in figure 5. Furthermore, x-rays can be refracted and reflected. In contrast to visible light, the real part of the refractive index is smaller than one.

3.1.3. Generating image contrast

Computed tomography always considers the transmission of x-rays through a sample.

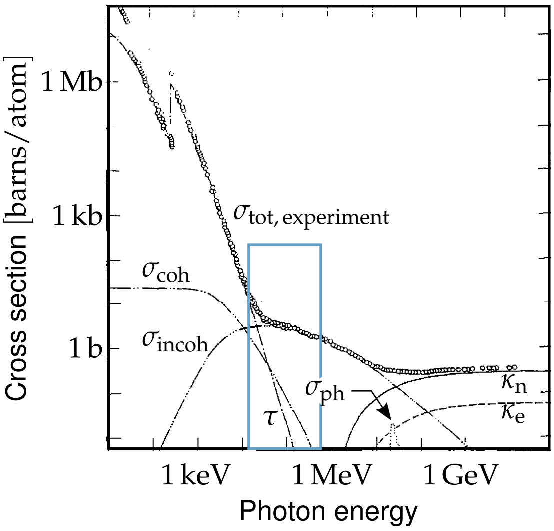


Figure 5 Interaction cross section over the energy. The graph shows measurement points of the total photon cross section in carbon σ_{tot} over the photon energy. The cross sections of the underlying interactions are: the atomic photo effect τ, coherent scattering σ_{coh} and incoherent (Compton) scattering σ_{incoh}, nuclear photo absorption $\sigma_{\text{ph.n.}}$, and nuclear- and electron-field pair production κ_n, κ_e. The underlying plot was taken from from Hubbell et al. [26, fig. 1, p. 1024]. The region relevant for most x-ray tomography applications is highlighted with the blue frame. © by AIP Publishing.

There are several possibilities to gain image contrast from the different ways of interaction, but also some obstacles.

The most common acquisition mode takes advantage of the attenuation. Its strength is described by the attenuation coefficient μ which is dependent on the atomic number of the material the x-rays penetrate. Lambert and Beer described the outgoing intensity I of a ray along a direction z relative to its incoming intensity I_0 as it goes through matter:

$$I = I_0 \exp\left(-\int \mu(z)\, \mathrm{d}z\right)$$

The line integrals p which are required for tomographic imaging can be retrieved by computing

$$p = \int \mu(z)\, \mathrm{d}z = -\log\left(\frac{I}{I_0}\right)$$

after measuring the transmitted intensity I through a sample and a reference intensity I_0 without sample.

The map of intensity values $I(u,v)$ of a whole region described by the spatial coordinates u and v is called *intensity image* throughout this work, the corresponding reference intensity map $I_0(u,v)$ is called *flatfield* and the corresponding map of line integrals is called a projection $p(u,v)$ or tomographic view.

The attenuation of x-rays regards the photons as particles. But also the wave-optical properties can be exploited. One very successful way to get complementary information to the bare attenuation is grating interferometry. A grating is placed into the beam generating an interference pattern which is then sampled by a second grating [27]. Three different images can be extracted from the measurement. The first matches the attenuation image described above. The second is a differential projection containing the total phase shift of the wave through the sample and thus describing the sum of the local refractive index values. The third image, called darkfield, contains information about small-angle scattering inside the sample which causes incoherence of the beam. It is only sensitive to scattering perpendicular to the orientation of the gratings and therefore allows to visualize the direction of substructures far below the image resolution by rotating the sample relative to the gratings. All three signals are very commonly used for tomographic imaging. Adding a third grating close to the source makes this technique even possible with conventional x-ray tubes [28].

Tomographic reconstruction is also common for propagation-based acquisition techniques. The intensity of the propagated wavefront behind the sample is recorded either in the nearfield or farfield. Phase-retrieval algorithms can then extract the projection images for attenuation contrast and phase contrast.

The way x-rays interact with matter results not only in opportunities, but also in some obstacles for generating image contrast. The first is scattering. Most tomographic models assume rays to be straight lines through the image. If rays are scattered by a small angle or multiple times within the sample, they are likely detected at an unexpected spot and appear as noise or blur on the projections images.

A second is beam-hardening. It occurs only for polychromatic x-ray spectra like it is emitted from an x-ray tube. As the x-ray attenuation coefficient μ usually becomes smaller with the x-ray energy, lower-energy photons are absorbed more than higher-energy photons. This effect causes the effective spectrum to be changed while the beam traverses the sample. The hardened effective spectrum then causes less photons to be absorbed than in the original spectrum. The law of Lambert-Beer does not account for this effect.

The third obstacle is quantum noise. As x-rays are photons, they follow the Poisson statistics. If only a few photons penetrate the sample at all, these quantum fluctuations generate significant noise in the images.

3.1.4. Detecting x-rays

Similar to visible light, x-rays can be detected either via analog or digital approaches. Analog x-ray films, which are still very popular in radiography, are not suited well for tomography which depends on digital reconstruction. Detailed information about x-ray detectors can be found in literature. [29, 30, for example]

On the digital side, x-rays are either first converted to visible light pulses in a scintillator which can then be detected for example by a charge-coupled device (CCD) camera or to electron hole pairs in a semiconductor with some read-out electronics.

There are two fundamental principles for counting the x-rays. Integrating detectors sum up the total electric charge in each pixel over time and are read out after a certain frame time. The more recent photon-counting detectors are able to convert every single photon into an electronic pulse. Each pixel has a separate electronic circuit with an integer counter deciding which events to consider.

Photon-counting detectors have many advantages for computed tomography. The most prominent one is that they do not suffer from electronic noise. Through their working principle, they directly reflect the Poisson photon statistics, which can be modeled in advanced reconstruction approaches. Another advantage comes from the individual pulses which are proportional to the x-ray energy. Thresholds in the discriminator thus allow to make the detector sensitive for different energy ranges without additional effort. There are very promising results for spectral imaging with those devices [31]. The limiting factor for a wide application are the limited count rates and the high costs.

3.2. Analytical reconstruction

The idea behind computed tomography, as already stated in the introduction, is creating a 3-D map of local attenuation coefficients from a set of transmission views through a sample. For simplicity, but without loss of generality, the reconstruction part uses the terminology of attenuation contrast imaging. More detailed information about analytical tomographic reconstruction can be found in literature. [32, 33, for example]

3.2.1. The tomographic problem

Analytical reconstruction tries to find an analytically exact inversion formula of the tomographic problem. In 1917, more than half a century before the first CT reconstruction, Johann K. A. Radon already found the required mathematical framework as a theoretical result [34]. It describes the Radon transform, which is the relationship between a 2-D function $\mu\,(x,z)$ and a defined set of line integrals

$$\begin{aligned} p\,(\varphi, r) &= \int_{\varphi} \mu\,(x,z)\,\mathrm{d}l \\ &= \iint\limits_{-\infty}^{+\infty} \mu\,(x,z)\;\delta\,(x\;\cos\varphi + \\ &\qquad\qquad z\;\sin\varphi - r)\;\mathrm{d}x\,\mathrm{d}z \end{aligned}$$

over this function, the so-called sinogram or Radon space. It is parameterized with the angle φ under which the line integral was taken and the offset r. A visual example of the Radon transform is shown in figure 6.

The Radon transform holds exactly for a slice in y of a 3-D image volume $\mu\,(x,y,z)$ in CT and the corresponding set of projection images $p\,(\varphi,u,v)$, where $u = r$ and $v = y$. The choice of the coordinates u and v describes a parallel-beam geometry. That means that there is no geometrical magnification and all rays from the source traverse the sample on parallel paths hitting the projection plane perpendicularly. More possible geometries will be described in chapter 4 about projective geometry.

3.2.2. The filtered backprojection (FBP)

There is an exact inversion formula for the Radon transform holding for a discrete number of projections that are equally spaced in φ. The so-called filtered backprojection (FBP) is the most common way to formulate it. It allows a very efficient implementation on a computer.

Algorithm 1 The filtered backprojection (FBP) [32, 33]

A Fourier filter step

$$p^{\mathrm{f}}\,(\varphi,r) = \left(\mathcal{F}_{\mathrm{r}}^{-1}\,\left(|\omega|\;\mathcal{F}_{\mathrm{r}}\,p\right)\right)(\varphi,r)$$

is followed by a backprojection

$$\begin{aligned} \mu\,(x,z) &= \int_{0^\circ}^{180^\circ} p^{\mathrm{f}}\,(\varphi,r)\;\mathrm{d}\varphi \\ \text{with}\quad r &= x\;\cos\varphi + z\;\sin\varphi. \end{aligned}$$

The FBP goes back to the Fourier slice theorem that relates the sinogram $p\,(\varphi,r)$ with the image $\mu\,(x,z)$ like this:

$$\left(\mathcal{F}_{\mathrm{r}}\,p\right)(\varphi,r) = \left(\mathcal{F}_{\mathrm{x,z}}\,\mu\right)(-r\;\sin\varphi, r\;\cos\varphi)$$

It states that the 1-D Fourier transform of the sinogram along r equals the 2-D Fourier transform of the image in polar coordinates. The problem with implementing a direct Fourier inversion method is that the Fourier slice theorem is defined on a polar grid, but standard implementations of the Fourier transform require data on a rectangular grid.

It can be shown, that the Fourier slice theorem can be solved for $\mu\,(x,z)$ by changing to polar integration variables for the 2-D Fourier transform. The result can be written as shown in algorithm 1.

The first step describes a filter in Fourier domain. Because the weight $|\omega|$ is the absolute frequency in Fourier space, this filter is usually referred to as ramp filter. Another name is Ram-Lak filter. The second step can be thought of smearing each column of the filtered sinogram $p^{\mathrm{f}}\,(\varphi,r)$ over the image $\mu\,(x,z)$ under its corresponding angle φ. The symmetry of the full sinogram described in figure 6 makes it possible to omit its second

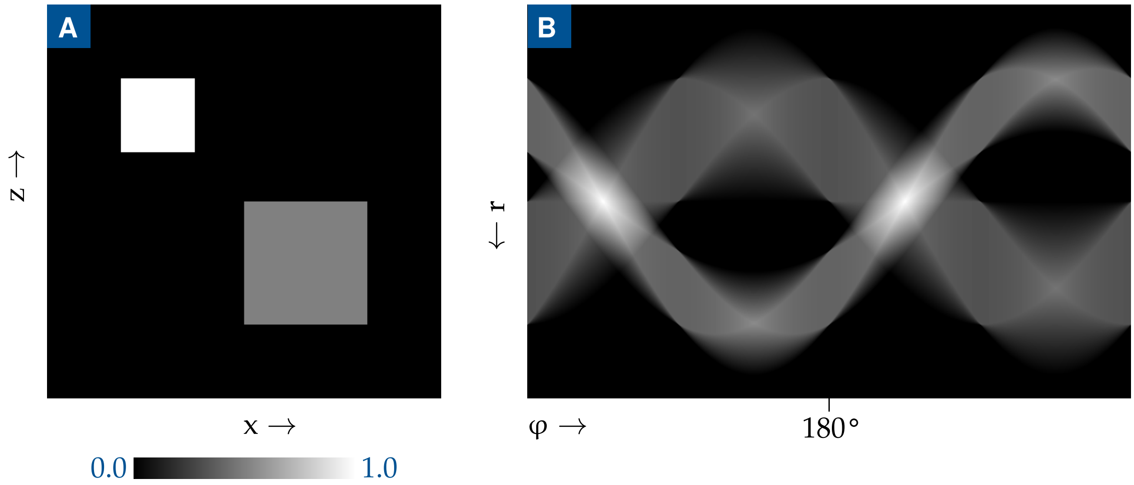


Figure 6 A simple image (A) and its sinogram (B).
The left image (A) represents a simple function $\mu(x,z)$ in gray values. The smaller rectangle has a constant value of 1.0 (white), the bigger one of 0.5 and the background of 0.0 (black) as the gray-scale bar below indicates. The right image (B) is the representation of its Radon transform $p(\varphi, r)$.
The left-most column of the sinogram ($\varphi = 0\,°$) shows the result for integrating $\mu(x,z)$ along the z axis. The further integration angles φ then continue counter-clockwise around the image. After the first quarter ($\varphi = 90\,°$), the sinogram shows the result for integrating along negative x and in the center ($\varphi = 180\,°$) for integrating along negative z. The last column ($\varphi \to 360\,°$) is close towards integrating along positive z again.
A closer look at the sinogram reveals its symmetry. The second half $[180\,°, 360\,°[$ equals the first half $[0\,°, 180\,°[$ with flipped dimension r.

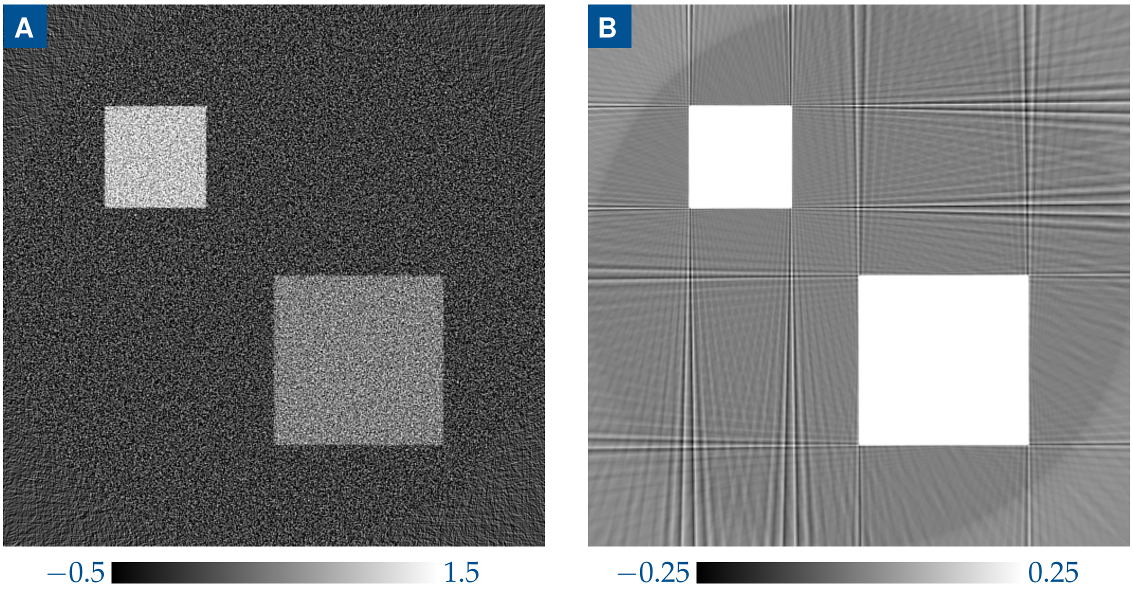


Figure 7 FBP with noise (A) and undersampling (B).
The two images are both FBP reconstructions of the sinogram shown in figure 6B. The perfect reconstruction is expected to look like the phantom in figure 6A. The window of values shown in the images is displayed in the gray-scale bar below each image.
The left image (A) is the reconstruction from the original sinogram with additional $6\,\%$ Gaussian noise with respect to the maximum sinogram value. The standard deviation of the smaller rectangle in the reconstruction is around $35\,\%$. This is an example of how the analytically exact FBP amplifies noise.
The right image (B) is the FBP from the four-fold undersampled data set. That means that only every fourth column in the sinogram was considered. The result are high-frequency streak artifacts in extension of sharp edges that become worse towards the border of the image.

half. In practical measurements taking the projections only in the range $\varphi = [0°, 180°[$ is called a half scan. For a full scan the symmetry implies that opposing views hold the same information. For discrete, uniform angular sampling it is therefore reasonable to take an odd total number of views.

The form of the filter function induces a great disadvantage of the FBP. As the mathematically exact solution demands that high frequencies have to be amplified most, the reconstruction results of real measurements often suffer from high-frequency noise. An example is shown in figure 7A. There is a set of common heuristic modifications of the ramp filter weighting high frequencies less than the exact solution suggests, but their improvements in noise reduction always come with the cost of an overall reduced sharpness.

Implementing the FBP on a computer requires transforming all the integrals into discrete sums. The 2-D or 3-D functions become maps of discrete pixels. Details can be found in chapter II about the implementation. Kak and Slaney showed that the number of discrete projections $\#\varphi$ and the number of pixels $\#r$ have to fulfill the following relationship because of the Nyquist sampling criterion [32]:

$$\#\varphi \geq \frac{\pi}{2} \#r.$$

If there are fewer projections, undersampling artifacts occur in the outer part of the reconstruction, where the given relationship is not satisfied. This effect can be seen in figure 7B. Even worse than sparse sampling artifacts are limited angle artifacts. They occur if the projections are not distributed equally in angular space, but a whole wedge is missing.

3.2.3. The method of Feldkamp, Davis, and Kress (FDK)

For medical or laboratory setups the parallel-beam assumption does not hold anymore. The rays emitted from the source point and going to the either flat or curved 2-D detector rather illuminate a cone-shaped area inside the sample. This is why this type of geometry is also called cone-beam. Tuy states that in this case exact reconstruction is only possible if all planes intersecting the object also intersect the trajectory of the source [35]. This is for example given for a helical trajectory, but not for a circular one.

Feldkamp, Davis, and Kress (FDK) developed a popular heuristic approximation for circular trajectories correcting most of the errors introduced by the cone-beam geometry [33, 36]. It can be seen as an extension to the FBP and is only exact in the one slice of the reconstruction that lies in the same plane as the source trajectory. Algorithm 2 shows its definition and figure 8 the relevant geometrical measures.

First, the filter step is modified. Thereby, $p(\varphi, u, v)$ is $p(\varphi, r)$ from the FBP (algorithm 1) trivially extended by the second dimension v of a flat detector parallel to the axis of rotation. The notation of the projection weights w_F assumes that the origin of the detector coordinates u and v is at the central ray, namely the only one hitting the detector perpendicularly. Parameter d_SD is the distance from the source to the detector. The weight considers the slope of each ray through the volume. This slope can also be expressed by the cosine of the angle κ between the actual ray and the central ray. Most descriptions in literature [36, 32, 33, 37, for example] use u and v as coordinates in the plane parallel to the detector going through the center of rotation and not in the detector plane itself,

Algorithm 2 The method of Feldkamp-Davis-Kress (FDK) [36, 32, 33]

The FBP filter step and backprojection get modified

$$p^{\mathrm{f}}_{\mathrm{FDK}}(\varphi,u,v) = w_{\mathrm{F}}(u,v)\left(\mathcal{F}^{-1}_{u}\left(|\omega|\,\mathcal{F}_{u}\right)\right)\,p(\varphi,u,v)$$

$$\mu_{\mathrm{FDK}}(x,y,z) = \frac{1}{2}\int_{0^\circ}^{360^\circ} w_{\mathrm{BP}}(\varphi,x,z)\;p^{\mathrm{f}}_{\mathrm{FDK}}(\varphi,u,v)\;\mathrm{d}\varphi$$

with the weights

$$w_{\mathrm{F}}(u,v) = \frac{d_{\mathrm{SD}}}{\sqrt{d^2_{\mathrm{SD}}+u^2+v^2}} = \cos\kappa$$

$$w_{\mathrm{BP}}(\varphi,x,z) = \left(\frac{d_{\mathrm{SC}}}{d_{\mathrm{SX}'}}\right)^2 = \left(\frac{d_{\mathrm{SC}}}{d_{\mathrm{SC}}+x\,\cos\varphi - z\,\sin\varphi}\right)^2.$$

The detector coordinates u and v can be computed from the image coordinates x, y and z as described later in chapter 4 about projective geometry.

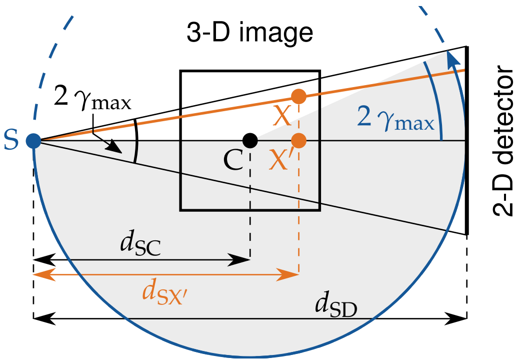


Figure 8 Geometry required for the FDK reconstruction.
The sketch shows the relative positions of the x-ray source S, the 3-D image volume and the 2-D detector in the source plane of a cone-beam geometry. Further important points are the center of rotation C and X′, the orthogonal projection of a point X onto the central ray. All measures depending on X are highlighted in orange. The distances d_{SC}, $d_{\mathrm{SX}'}$ and d_{SD} are required to compute the Feldkamp weights. The circular source trajectory (blue) is indicated by the dashed line. A circular arrow and its enclosed area in gray label the segment of 180° plus twice the maximum fan angle $\gamma_{\max}$ needed for a short scan with Parker weights.

like in this thesis. Therefore, they use d_{SC}, the distance from the source to the center of rotation, instead of d_{SD}.

Second, the back projection is also modified. Of course also the image volume $\mu(x,z)$ from the original formulation in algorithm 1 has to be extended by the third dimension. For consistency with the later definitions, the axis y is the one parallel to the axis of rotation. The weight w_{BP} is proportional to the density of rays in the diverging cone at a certain point (x,y,z) of the volume. Therefore, it relates the distance between the source and the center of rotation d_{SC} to the distance $d_{\mathrm{SX}'}$ from the source to the projection of the actual point X onto the central ray.

Because of the divergent rays in the u direction, the FDK is only consistent for a whole 360° scan. There is, however, another modification allowing to reduce the scan to $180^\circ + 2\,\gamma_{\max}$ by introducing the so-called Parker weights [38]. The additionally required angular wedge, compared to an FBP half scan, equals the opening angle of the cone in u direction and is twice the maximum fan angle $\gamma_{\max}$.

3.3. Iterative reconstruction

Iterative reconstruction (IR) is an alternative way to solve the tomographic problem. It does not aim for a direct inversion formula but improves the result by iterative updates, starting from a first guess. The updates are computed from comparing how well the current image estimate fits the projection data. IR is generally more robust to imperfections in the measurements, especially noise or undersampling. Recent techniques also can consider different kind of prior knowledge. Details can be found in literature [17, 39, for example].

There are different types of iterative reconstruction methods and also different ways to motivate them. Simple iterative approaches like the algebraic reconstruction technique (ART) were already investigated in the early years of CT [13, 14]. The focus of these studies will be on a more recent and general technique called statistical iterative reconstruction (SIR).

3.3.1. Statistical Iterative Reconstruction (SIR)

SIR is a model-based approach. It allows to model physical properties directly into the reconstruction. One of its most powerful assets is an underlying noise model. In contrast to analytical approaches, it considers that measured projection data is never the perfect Radon transform of a 3-D function because of noise. Strictly spoken, analytical approaches do not hold in the presence of noise, because the measured projections do not form a consistent Radon space in a mathematical sense. SIR, in contrast, allows deviations within a certain noise range.

A simple, but very common example is noted in algorithm 3. The vector μ holds all discretized voxel values of the attenuation map $\mu(x, y, z)$ that is reconstructed and the vector $\boldsymbol{p}$ the discretized projection pixel values for all views $p(\varphi, u, v)$. $\boldsymbol{A}$ is a matrix performing a discrete Radon transform. It holds the weights by which each voxel element from the image vector μ contributes to a given pixel element of the projection vector $\boldsymbol{p}$.

Algorithm 3 Statistical iterative reconstruction (SIR) with a Gaussian noise model [17, 39]

The cost function

$$\mathcal{L}_{\mathrm{G}}(\mu) = \frac{1}{2}\,\|\boldsymbol{A}\cdot\mu - \boldsymbol{p}\|_w^2 + \lambda\, R(\mu)$$

is minimized by a numerical solver

$$\mu_{\mathrm{opt}} = \arg\min_{\mu} \mathcal{L}(\mu)\,.$$

The notation $\|\centerdot\|_w^2$ denotes an element-wise weighted quadratic norm.

The first addend of the cost function is the data fidelity term, short data term. It is the weighted squared difference of the virtual projections $\boldsymbol{A}\cdot\mu$, generated from the current guess μ, and the given projections $\boldsymbol{p}$. The weights w hold the statistical reliability of each data point $\boldsymbol{p}$. The data term is derived from a statistical noise distribution as shown below. It allows solutions with different noise realizations according to the underlying noise model. In the example Gaussian noise was assumed.

3.3.2. Regularization

The second addend in the cost function is the regularization term $R(\mu)$. It allows to add prior knowledge to the reconstruction in order to receive the most realistic attenuation map possible.

CT reconstruction is mathematically an ill-posed problem. Therefore, many different attenuation maps lead to a good agreement with the measured data in the presence of noise. Prior knowledge can help to select a physically plausible solution. As many solutions are very noisy and real objects are usually partly homogeneous, typical regularization terms are smoothness constraints. The more sophisticated terms additionally try to detect and exclude sharp edges from the smoothness assumption.

The parameter λ defines the weight of data term and regularization. Estimated too high, the result becomes patchy or blurry. Estimated too low, the result becomes noisy or contains artifacts. This unknown parameter is one of the greatest obstacles for iterative reconstruction.

The Huber regularization is one of the most common choices. It belongs to the class of Gibbs neighborhood priors that apply a penalty function $\Psi\left(t\right)$ on the value difference, more exactly on the gradient, between each voxel i and all its spacial neighbors $\mathcal{N}_i$ weighted by their inverse spacial distance $\Delta_{i,n}$. Algorithm 4 shows how it is defined.

Small differences are considered as noise and thus punished with a quadratic function, great differences as edges which are punished with a linear function. The whole penalty is plotted in figure 9.

Algorithm 4 A popular regularization term

The Gibbs neighborhood prior

$$\mathrm{R_G}\left(\boldsymbol{\mu}\right) = \frac{1}{2} \sum_{i\in[\boldsymbol{\mu}]} \sum_{n\in\mathcal{N}_i} \frac{1}{\Delta_{i,n}} \Psi\left(\frac{\mu_i - \mu_n}{\Delta_{i,n}}\right)$$

with a Huber penalty

$$\Psi_{\mathrm{H}}^{\gamma}\left(t\right) = \begin{cases} 1/2\,\gamma^{-1}\,t^2 & \text{for } |t| < \gamma \\ |t| - \gamma/2 & \text{else.} \end{cases}$$

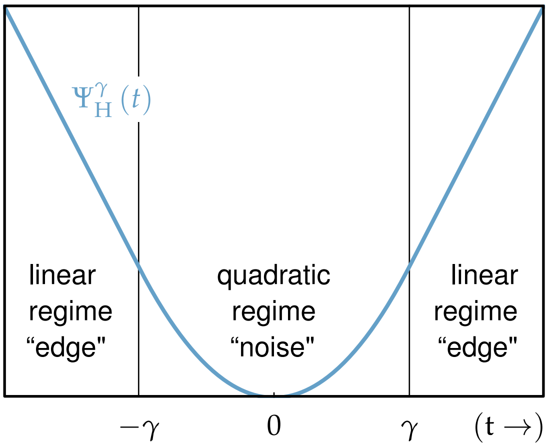


Figure 9 The Huber penalty function.
Small differences t between neighboring voxels are considered to be noise and result in a quadratic penalty. Large differences fall into the linear regime in order to preserve real edges. The parameter γ defines the transition between both regimes.

3.3.3. Derivation of the data term

As stated above, SIR is built on a model for the noise statistics. Different statistics result in different data terms. Electronic noise as well as photon quantum noise in the limit of many photons both follow a Gaussian distribution. It has been shown that in that case the Gaussian distribution can also be assumed for the line integrals $\boldsymbol{p}$ after applying the inverse law of Lambert-Beer in the measured intensities. [40]

Let $\boldsymbol{\mathcal{P}}$ be a set of Gaussian random variables distributed around the mean values $\bar{\boldsymbol{p}}$ with standard deviations $\boldsymbol{\sigma}$. Given a set of virtual projections $\bar{\boldsymbol{p}}\left(\boldsymbol{\mu}\right) = \boldsymbol{A} \cdot \boldsymbol{\mu}$ from a current image estimate $\boldsymbol{\mu}$, the probability of observing a certain noise realization $\boldsymbol{p}$ can be written as

$$\mathrm{P}\left(\boldsymbol{\mathcal{P}} = \boldsymbol{p}\,|\boldsymbol{\mu}\right) =$$
$$\prod_{j\in[\boldsymbol{p}]} \mathrm{P}\left(\mathcal{P}_j = p_j\,|\boldsymbol{\mu}\right) =$$
$$\prod_{j\in[\boldsymbol{p}]} \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{\left(p_j - \bar{p}_j\left(\boldsymbol{\mu}\right)\right)^2}{2\sigma_j^2}\right).$$

This term is also called the likelihood $\mathrm{L}\left(\boldsymbol{\mu}\,|\boldsymbol{p}\right)$ for $\boldsymbol{\mu}$ to be the right parameters that lead to the measured result $\boldsymbol{p}$.

Maximizing this likelihood results in $\boldsymbol{\mu}_{\mathrm{ML}}$, one of the solutions from the set of different attenuation maps that maximize the probability $\mathrm{P}\left(\boldsymbol{\mathcal{P}} = \boldsymbol{p}\,|\boldsymbol{\mu}\right)$. For noisy data $\boldsymbol{p}$ and without regularization this can be a very noisy and thus unrealistic solution.

In practice, it is more common to minimize the negative log-likelihood in order to get rid of the exponential function. It reads

$$\begin{aligned} l(\boldsymbol{\mu}\,|\boldsymbol{p}) &= -\log \mathrm{L}(\boldsymbol{\mu}\,|\boldsymbol{p}) \\ &= -\sum_{j\in[\boldsymbol{p}]} \log\left(\sqrt{2\pi\sigma_j^2}\right) + \\ &\quad \sum_{j\in[\boldsymbol{p}]} \frac{1}{2\sigma_j^2}\left(p_j - \{\boldsymbol{A}\cdot\boldsymbol{\mu}\}_j\right)^2 \end{aligned}$$

in the Gaussian case. Neglecting the first term, which is independent from $\boldsymbol{\mu}$, the whole problem can be simplified to

$$\begin{aligned} \boldsymbol{\mu}_{\mathrm{ML}} &= \arg\min_{\boldsymbol{\mu}} l(\boldsymbol{\mu}\,|\boldsymbol{p}) \\ &= \arg\min_{\boldsymbol{\mu}} \left(\frac{1}{2}\,\|\boldsymbol{A}\cdot\boldsymbol{\mu} - \boldsymbol{p}\|_w^2\right) \end{aligned}$$

with $w_j = \sigma_j^{-2}$. This is exactly the data term of the SIR cost function in algorithm 3.

The other common noise model in CT is derived from Poisson statistics. It is more compute-intensive, but very valuable for low-statistics measurements where the Photon quantum noise dominates. The according cost function reads

$$\mathcal{L}_{\mathrm{P}}(\boldsymbol{\mu}) = \sum_{j\in[\boldsymbol{I}]} I_j \log\left(\bar{I}_j(\boldsymbol{\mu})\right) - \bar{I}_j(\boldsymbol{\mu})$$

with

$$\bar{\boldsymbol{I}}(\boldsymbol{\mu}) = \boldsymbol{I}_0 \exp\left[-\boldsymbol{A}\cdot\boldsymbol{\mu}\right] + \boldsymbol{I}_{\mathrm{b}}$$

and can be derived from the Poisson distribution

$$\mathrm{P}(\boldsymbol{\mathcal{P}} = \boldsymbol{I}\,|\boldsymbol{\mu}) = \prod_{j\in[\boldsymbol{I}]} \frac{\mathrm{e}^{-\bar{I}_j(\boldsymbol{\mu})}\,\left(\bar{I}_j(\boldsymbol{\mu})\right)^{I_j}}{I_j!},$$

in the same way as shown for the Gaussian case.

The Poisson data model is only valid directly on the vector $\boldsymbol{I}$ of the discretized x-ray intensities $I(\varphi, u, v)$, not on the vector of line integrals $\boldsymbol{p}$. Therefore, the Lambert-Beer law can be found included into the forward model $\bar{\boldsymbol{I}}(\boldsymbol{\mu})$. The additional intensity $\boldsymbol{I}_{\mathrm{b}}$ is the mean number of background events coming for example from photon scattering.

3.3.4. Solvers

A solver is an iterative algorithm that can find the minimum of a given cost function. It starts with an initial guess $\boldsymbol{\mu}^{(0)}$ and converges step by step towards a final solution $\boldsymbol{\mu}_{\mathrm{opt}}$. In iteration n, the next step $\boldsymbol{\mu}^{(n)}$ is most generally computed from the function $\mathcal{L}(\boldsymbol{\mu})$ and its derivatives, evaluated at the current guess $\boldsymbol{\mu}^{(n-1)}$, and information from previous steps. Common functions required by different solvers are given in appendix A.

A very good general-purpose solver for CT is the non-linear conjugate gradient algorithm (NLCG) [41]. Depending on the exact implementation, it requires the gradient and sometimes the denominator of the cost function.

There are also specialized solvers for the SIR cost function. Probably the most popular one is the ordered-subset separable paraboloidal surrogates (OS-SPS) algorithm [42, 43] and its variants. It requires only the curvature and the gradient. The expensive curvature can be precomputed.

Ordered subsets can improve its convergence. They enable iterations on only a subset of projections $\boldsymbol{p}_m$ for subset m instead of considering the whole vector $\boldsymbol{p}$ which saves compute time. Ideally, the contribution of such a (sub-)iteration is equal to an iteration that considers the full data set. Therefore, the term iteration always means subset iteration in this thesis. In literature, this term sometimes refers to a full cycle of iterations until

Algorithm 5 Bit-reversal sorting [42, 43] implemented in Python.

```python
def _rev_bits(n):
    # reverse the bit order of an integer
    return int('{:032b}'.format(n)[::-1], 2)

def sorted_bit_rev(a_):
    as_ = [ _rev_bits(n) for n in a_ ]
    as_ = sorted(as_)
    as_ = [ _rev_bits(n) for n in as_ ]
    return as_

sorted_bit_rev([0, 1, 2, 3, 4, 5, 6, 7])
```

Output: `[0, 4, 2, 6, 1, 5, 3, 7]`

all projections were considered once. The optimum number of views in a subset is quasi independent from the total number of views. Mathematically, conversion is only guaranteed if at least the last iteration operates on the full data set. The best order in which the views are stored before being destributed to subsets is the bit-reversal order [42, 43]. It is sketched in algorithm 5. This order guaranties that neighbor views are at maximum angular distance and, for helical geometry, an equal distribution of the views within a subset over all turns, which is important for the turns to be updated simultaniously. A final iteration using all views can be used to guarantee convergence.

Recently, momentum support was added to the OS-SPS solver (OS-MOM) which takes into account information from the previous steps for computing the updates [44]. The technique was able to gain about an order of magnitude in the speed of convergence. It also prefers larger subsets, i. e. few hundred instead of a few dozens, which is good for parallelization. The most recent variant is the OS-OGM [45], which is proven to include the fastest linear combination of momentums possible. Later, algorithm 9 on page 80 will show a simple implementation.

Closure

Computed tomography is able to provide great insights into almost any sample without destroying it. This chapter summarized the basic x-ray physics and computer algorithms for understanding the working principle of CT and related challenges.

The available ways for generating x-ray projections leave many choices. Tube sources, in contrast to a synchrotron, are fast, small and cheap but generate beam hardening due to their polychromatic spectrum. The size of their focal spot decides further between either a high spatial resolution or a high flux and fast acquisitions. Absorption contrast measurements are simple to implement, but the phase or darkfield signal adds much more valuable information. Integrating detectors are cheap and can handle high count rates, but photon-counting detectors do not suffer from electronic noise and can intrinsically separate energy ranges.

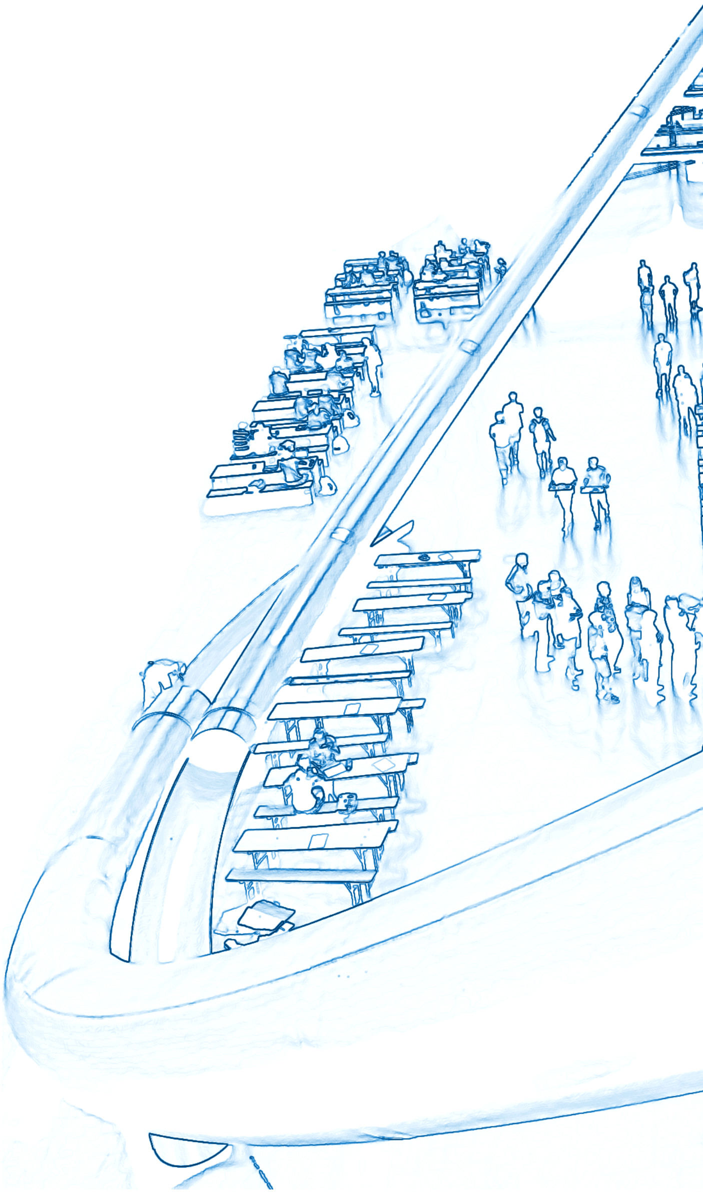
There are two classes of reconstruction algorithms. The first are analytical approaches like the FBP and FDK. They provide fast and accurate results, but have high requirements on the quality of the acquisition that often cannot be met. Each imperfection like noise, missing data or beam hardening generates sometimes severe image artifacts.

The second class are the iterative techniques. They have the option to model the physical effects correctly, can additionally include prior information and perform task-based reconstructions. The costs for these advantages are an increased computational effort and an additional regularization parameter which is hard to tune.

4. Projective geometry

Projective geometry is a powerful mathematical tool to describe and compute on perspective images in computer vision. August F. Möbius, long before computers even existed, laid the foundation of this kind of mathematics [46]. Today it plays a big role in learning computers how to “see” with a camera.

The great asset for CT comes with its generic and straight forward way of describing arbitrary cone-, fan- or parallel-beam geometries. All required computations can be expressed with vectors and matrices which can be evaluated very quickly on a computer.



Picture: A perspective view of the parabolic slides inside the TUM university building. © mediaTUM

4.1. Mathematical background

Projective geometry is an extension to Euclidean geometry. It was introduced, because Euclidean coordinates do not allow to represent points at infinity. Everything described in this section can in detail be found in the book *Multiple View Geometry* [47].

As an illustration of the problem, one can imagine a photograph of the two rails along a straight railroad track on a plain terrain as shown in figure 10. The two parallel rails intersect at the horizon. This is because their common point of infinity is part of the photograph. It is commonly said that parallels intersect at infinity, but Euclidean geometry offers no way to describe it.

4.1.1. Homogeneous coordinates

Projective geometry introduces a new way to write the coordinates of a point in space. Those coordinates $\tilde{x}$ are called homogeneous coordinates and map to their Euclidean counterpart x by the relation

$$\tilde{x} = k \begin{pmatrix} x \\ 1 \end{pmatrix} \Leftrightarrow x \quad \text{for } k \neq 0.$$

In 2-D space, for example, the Euclidean coordinate $x = (u, v)^\mathsf{T}$ can be expressed in homogeneous coordinates with $\tilde{x} = (u, v, 1)^\mathsf{T}$ by adding an additional element 1. In projective geometry, the coordinates $(\tilde{u}, \tilde{v}, 1)^\mathsf{T}$ and $(k\,\tilde{u}, k\,\tilde{v}, k)^\mathsf{T}$ represent the same point. Therefore, mapping a homogeneous coordinate $\tilde{x} = (\tilde{u}, \tilde{v}, \tilde{w})^\mathsf{T}$ back to the Euclidean means extracting the first two divided by the last one $x = (\tilde{u}/\tilde{w}, \tilde{v}/\tilde{w})^\mathsf{T}$. It becomes obvious, that the the latter conversion only works for $\tilde{w} \neq 0$. This is where the points at infinity come into play. They are exactly all points in homogeneous coordinates with $\tilde{w} = 0$. The first three entries describe the direction of that point in space, the last one represents its infinite distance from the origin. The homogeneous coordinates of a point are never all zero.

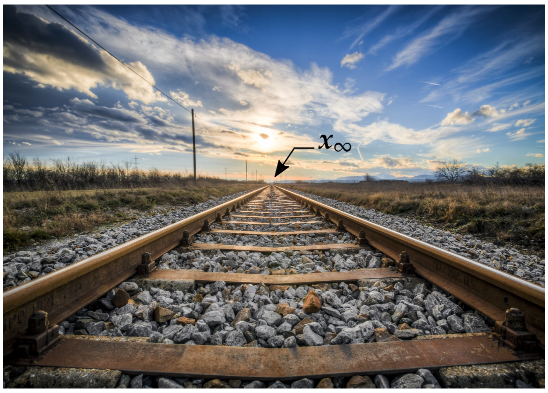


Figure 10 Two straight rails touch at infinity.
The perspective photograph shows how two straight parallel lines intersect at the point of infinity x_∞. This point cannot be described by Euclidean coordinates. 🅭

A very useful illustration explaining how the 2-D projective space $\mathbb{P}^2$ relates to the 2-D Euclidean space $\mathbb{R}^2$ by illustrating it in the 3-D Euclidean space $\mathbb{R}^3$ is shown in figure 11.

4.1.2. Camera projection

Taking a photograph, for example with a CCD camera, can be seen as a perspective projection of the 3-D world into a 2-D plane, which is the sensor. Homogeneous coordinates allow a very simple description of that process. Coordinates in $\mathbb{R}^3$, which are usually used to represent points in the 3-D world, can easily be extended to coordinates in $\mathbb{P}^3$. The latter includes additionally all points at infinity. A so-called projective transformation can then be applied to map the 3-D to a 2-D projective space $\mathbb{P}^3 \rightarrow \mathbb{P}^2$. In contrast to an affine transformation, which could as well be described in Euclidean space, this transformation can map points at infinity to any other point. This way it is able to map points at infinity into the 2-D projection as required for perspective

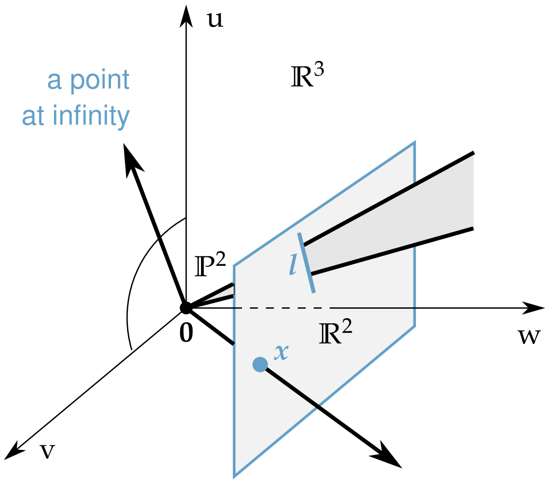


Figure 11 The 2-D projective and Euclidean space. Hartley et al. suggest this illustration of the relationship between the 2-D projective space $\mathbb{P}^2$ and the 2-D Euclidean space $\mathbb{R}^2$ [47, fig. 2.1, p. 29]. The projective space can be seen as the pencil of rays and all planes through the origin $\mathbf{0}$ in $\mathbb{R}^3$, the Euclidean space as the plane $w = 1$. Every point x in $\mathbb{R}^2$ corresponds exactly to one ray of $\mathbb{P}^2$ and every line $\boldsymbol{l}$ to one plane, respectively.
The plane $w = 0$ represents the line at infinity holding all points at infinity. Those can only be represented in $\mathbb{P}^2$, but not in $\mathbb{R}^2$. © by the Cambridge University Press.

photographs like the one shown in figure 10.

The transformation of a point $\tilde{X} \in \mathbb{P}^3$ to the corresponding point $\tilde{x} \in \mathbb{P}^2$ can be written with help of a 3×4 projection matrix $\tilde{\boldsymbol{P}}$:

$$\tilde{x} = \tilde{\boldsymbol{P}} \cdot \tilde{X}$$

This matrix holds all the geometric camera properties, its position and orientation within the 3-D world.

4.2. Projection matrices in CT

Same as for cameras, also the projections of a cone-beam CT can be described by projection matrices [48]. This section explains our way of constructing such a matrix and introduces some useful operations with it.

Each single projection of a scan has to be described with one corresponding matrix $\tilde{\boldsymbol{P}}$ consisting of the following components. The relevant coordinate systems and measures of a tomographic setup are introduced in figure 12. The coordinate system $(x, y, z)^\mathsf{T}$ of the image volume equals the array indices of the discrete voxel representation in a computer. Its origin is in the center of the voxel with index $[0, 0, 0]^\mathsf{T}$. Assuming cuboid voxels, the units are normalized to the voxel dimensions $\Delta x, \Delta y, \Delta z$. The coordinate system $(u, v)^\mathsf{T}$ of the virtual detector is build in the same way using units of the usually rectangular detector pixels $\Delta u, \Delta v$.

4.2.1. The intrinsic parameters

A 3×3 camera calibration matrix $\tilde{\boldsymbol{K}}$ is able to describe the intrinsic geometry of a camera or a tomographic setup, respectively. Mapping a point $\tilde{X} = (x, y, z, 1)^\mathsf{T}$ inside a sample to the point $\tilde{x} = (u, v, 1)^\mathsf{T}$ on the x-ray detector can be written as

$$\begin{pmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{pmatrix} = \tilde{\boldsymbol{K}} \cdot \begin{pmatrix} \mathbb{1}^3 & \mathbf{0} \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

$$\text{with} \quad \tilde{\boldsymbol{K}} = \begin{pmatrix} \alpha_\mathrm{u} & s & p_\mathrm{u} \\ & \alpha_\mathrm{v} & p_\mathrm{v} \\ & & 1 \end{pmatrix}$$

where $\mathbb{1}^3$ is the 3-D identity matrix.

The scaling parameters are given by

$$\alpha_\mathrm{u} = \frac{d_\mathrm{SD}}{\Delta u} \quad \text{and} \quad \alpha_\mathrm{v} = \frac{d_\mathrm{SD}}{\Delta v}$$

containing d_SD, the analogue to the focal length of a camera, and the detector pixel dimensions. As α_u and α_v are dimensionless, the measures required to compute it can be given in arbitrary but matching units.

The offset $\boldsymbol{p} = (p_\mathrm{u}, p_\mathrm{v})^\mathsf{T}$ describes the position of the principal point in detector coordinates. If it is the center of the detector, it follows from the total number of detector pix-

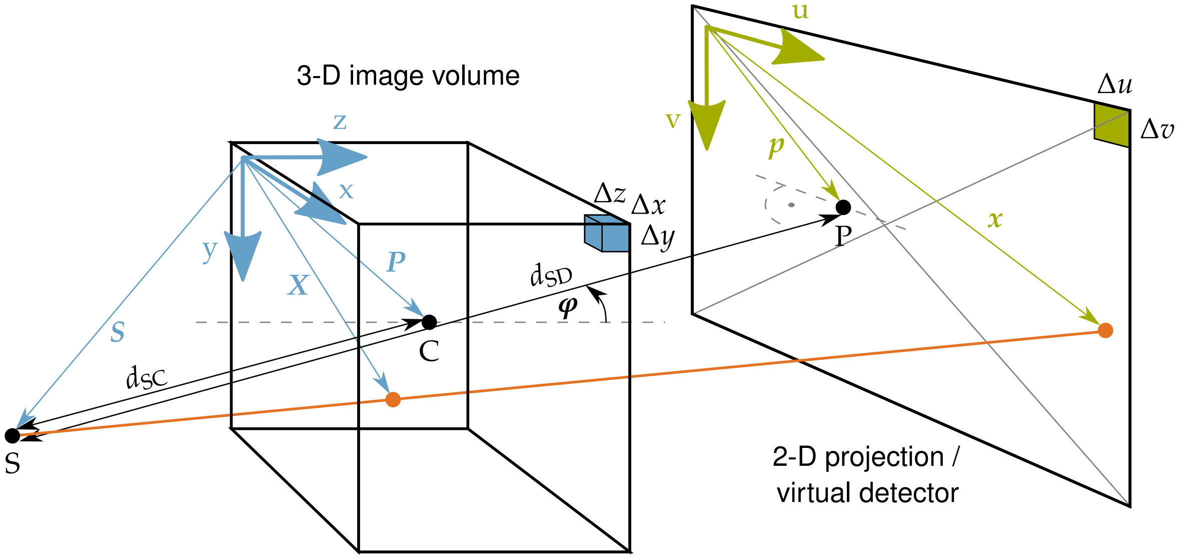


Figure 12 A schematic of the CT geometry with all relevant measures.
The perspective projection of a voxel at position X within the 3-D image volume to the corresponding pixel x on the 2-D virtual detector can be described with a $\mathbb{P}^3 \rightarrow \mathbb{P}^2$ projective transformation $\boldsymbol{P}$ using homogeneous coordinates. This schematic shows the coordinate systems, points and measures relevant to describe the whole CT geometry.
The 3-D image coordinate system (x, y, and z) with corresponding vectors is printed in blue and the one on the 2-D projections (u and v), in green. The first one uses units of image voxels, which are Δx, Δy and Δz, the second one units of detector pixels Δu and Δv.
The distance between the x-ray source S and the center of rotation C of the CT setup is called d_SC and d_SD is the distance between the source S and the detector. The principal point P is defined as the intersection of the central ray with the detector plane. It is the foot of a perpendicular of S on the detector. The 3-D angular rotation of that ray with respect to the z-axis of the image coordinate system is described by the Euler angles $\boldsymbol{\varphi}$. A possible choice is given in appendix B.

els $\#u$ and $\#v$:

$$\begin{pmatrix} p_\mathrm{u} \\ p_\mathrm{v} \end{pmatrix}_\mathrm{central} = \frac{1}{2}\begin{pmatrix} \#u - 1 \\ \#v - 1 \end{pmatrix}.$$

The -1 is caused by the fact, that the position of a pixel is measured at its center.

If the skew parameter s is not 0, the detector becomes trapezoidal. This can be beneficial for reconstructing parallel resorted projections [32, subsec. 3.4.3, pp. 92-93] from a fan-beam helical scan.

4.2.2. The extrinsic parameters

The extrinsic parameters describe the position and orientation of the source relative to the image coordinate system. It does not play a role if the source and the detector indeed rotate around the sample, or if, for practical reasons, the sample is rotated within a stationary setup. For simplicity, the origin of the image coordinate system is for a moment assumed at the center of rotation C and its units arbitrary units. It is then called the world coordinate system. The error will be corrected afterwards.

The rotation matrix $\boldsymbol{R}_\mathrm{W}$ describes the rotation of the source with respect to the world coordinates and the source translation vector S_W its position. The unrotated source is then given by $\boldsymbol{t}_\mathrm{W} = -\boldsymbol{R}_\mathrm{W} \cdot S_\mathrm{W}$, where $-\boldsymbol{R}_\mathrm{W} = \boldsymbol{R}_\mathrm{W}^{-1}$. It reads $(0, 0, d_\mathrm{SC})^\intercal$ for a CT setup, where C lies on the line between S and P.

Together with the intrinsic parameters, the whole projection transformation reads

$$\tilde{x} = \tilde{\boldsymbol{K}} \cdot \begin{pmatrix} \boldsymbol{R}_\mathrm{W} & t_\mathrm{W} \end{pmatrix} \cdot \tilde{\boldsymbol{X}}_\mathrm{W}.$$

The 3×4 projection matrix for world coordi-

Algorithm 6 Useful operations with projection matrices

The **detector coordinates** of a point in the image under the view described by $\tilde{\boldsymbol{P}}$ result in

$$\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{P}} \cdot \tilde{\boldsymbol{X}}$$

given its homogeneous coordinates $\tilde{\boldsymbol{X}} = (x, y, z, 1)^{\mathsf{T}}$.

The **position of the source** in image coordinates can be obtained from

$$\tilde{\boldsymbol{P}} \cdot \tilde{\boldsymbol{S}} \overset{!}{=} \boldsymbol{0}$$

$$\Rightarrow \begin{pmatrix} \tilde{S}_{\mathrm{x}} \\ \tilde{S}_{\mathrm{y}} \\ \tilde{S}_{\mathrm{z}} \\ \tilde{S}_{\omega} \end{pmatrix} = \begin{pmatrix} \det(\ \tilde{\boldsymbol{p}}_{\mathrm{y}} \quad \tilde{\boldsymbol{p}}_{\mathrm{z}} \quad \tilde{\boldsymbol{p}}_{\omega}\ ) \\ -\det(\ \tilde{\boldsymbol{p}}_{\mathrm{x}} \quad \tilde{\boldsymbol{p}}_{\mathrm{z}} \quad \tilde{\boldsymbol{p}}_{\omega}\ ) \\ \det(\ \tilde{\boldsymbol{p}}_{\mathrm{x}} \quad \tilde{\boldsymbol{p}}_{\mathrm{y}} \quad \tilde{\boldsymbol{p}}_{\omega}\ ) \\ -\det(\ \tilde{\boldsymbol{p}}_{\mathrm{x}} \quad \tilde{\boldsymbol{p}}_{\mathrm{y}} \quad \tilde{\boldsymbol{p}}_{\mathrm{z}}\ ) \end{pmatrix}$$

where $\tilde{\boldsymbol{p}}_n$ denotes the column n of $\tilde{\boldsymbol{P}}$.

The **main propagation direction** is

$$\nu = \underset{n \in \{\mathrm{x},\mathrm{y},\mathrm{z}\}}{\arg\max} \left|\{\boldsymbol{m}^{\mathrm{w}}\}_n\right|$$

with the central ray $\boldsymbol{m}^{\mathrm{w}}$ which is also called principle axis vector. It is the last row of $\tilde{\boldsymbol{M}}$, the left 3×3 submatrix of $\tilde{\boldsymbol{P}}$.

A **ray through the image** according to Joseph's method [49] starting at the detector coordinates $\tilde{\boldsymbol{x}} = (u, v, 1)^{\mathsf{T}}$ reads

$$\boldsymbol{X}(\xi) = \boldsymbol{X}_0 + \xi\, \boldsymbol{U}_0 \quad \text{with } \xi \in \mathbb{R}.$$

It requires a reference point given by the pseudo inverse and a direction vector:

$$\begin{aligned} \tilde{\boldsymbol{X}}_0 &= \tilde{\boldsymbol{P}}^{+} \cdot \tilde{\boldsymbol{x}}, \\ \boldsymbol{U} &= \boldsymbol{X}_0 - \boldsymbol{S}. \\ \boldsymbol{U}_0 &= \boldsymbol{U} / \{\boldsymbol{U}\}_\nu \end{aligned}$$

The latter is normalized to its component ν. Adding $\boldsymbol{U}_0$ to any point on the ray (i. e. incrementing ξ by one) thus steps into the next slice of voxels perpendicular to ν.

Entry and exit point of the ray, i. e. the first and last voxel position in ν, follow from

$$\begin{aligned} \{\boldsymbol{X}(\xi)\}_\nu = \{\boldsymbol{X}_0\}_\nu + \xi_{\min} \{\boldsymbol{U}_0\}_\nu &\overset{!}{=} 0 \\ \Rightarrow \xi_{\min} = -\{\boldsymbol{X}_0\}_\nu \quad \text{as } \{\boldsymbol{U}_0\}_\nu &\overset{\mathrm{def}}{=} 1 \\ \xi_{\max} = \xi_{\min} + \#\nu - 1 & \end{aligned}$$

where $\#\nu$ is the number of voxels along ν.

---

nates is accordingly defined as

$$\tilde{\boldsymbol{P}}_{\mathrm{W}} = \tilde{\boldsymbol{K}} \cdot \begin{pmatrix} \boldsymbol{R}_{\mathrm{W}} & \boldsymbol{t}_{\mathrm{W}} \end{pmatrix}.$$

The principle described in the next section (4.3) allows to easily carry out the transformation from world to image voxel coordinates [37]:

$$\tilde{\boldsymbol{P}} = \tilde{\boldsymbol{P}}_{\mathrm{W}} \cdot \tilde{\boldsymbol{T}}_{\mathrm{W}}$$

with

$$\tilde{\boldsymbol{T}}_{\mathrm{W}} = \begin{pmatrix} \Delta x^{-1} & & & -P_{\mathrm{x}}\,\Delta x \\ & \Delta y^{-1} & & -P_{\mathrm{y}}\,\Delta y \\ & & \Delta z^{-1} & -P_{\mathrm{z}}\,\Delta z \\ & & & 1 \end{pmatrix}$$

The inverse image voxel dimensions scale the arbitrary units of the world coordinate system to image voxel units. They have to be given in the same units as the entries of $\boldsymbol{t}_{\mathrm{W}}$ above. The rescaled negative vector $\boldsymbol{P}$, given in image coordinates, causes the right shift of the origin.

Summarizing, the projection matrix $\tilde{\boldsymbol{P}}$ directly allows to take image voxel coordinates and returns detector pixel coordinates. By constructing it in the presented way, the distance d_{SD} and the detector pixel shape can be given in arbitrary but matching units. The same holds for d_{SC} and the image voxel shape.

4.2.3. Operations

The whole geometry of a CT setup can be described with a set of projection matrices. It is therefore convenient to deduce all the required operations from those matrices. Algorithm 6 gives an overview of useful standard operations. Chapter 6 shows how they can be used to carry out tomographic forward and back projections.

4.2.4. Parallel-beam geometry

Parallel-beam geometry can also be represented by projection matrices [47, pp. 166–174]. The concepts described for affine cameras can directly be transferred to parallel-beam CT.

The affine projection matrix assumes the source S is at infinity. It writes

$$\tilde{\boldsymbol{P}}_{\mathrm{W},\infty} = \tilde{\boldsymbol{K}}_{\infty} \cdot \begin{pmatrix} \{\boldsymbol{R}_{\mathrm{W}}\}^{\mathrm{x}} & t_{\mathrm{W,x}} \\ \{\boldsymbol{R}_{\mathrm{W}}\}^{\mathrm{y}} & t_{\mathrm{W,y}} \\ \mathbf{0}^{\mathsf{T}} & 1 \end{pmatrix}$$

where $\{\boldsymbol{R}_{\mathrm{W}}\}^{m}$ denotes the m-th row of the rotation matrix and $t_{\mathrm{W},m}$ the m-th component of the source translation vector $\boldsymbol{t}$. That means that $\boldsymbol{R}_{\mathrm{W}}$ and $\boldsymbol{t}_{\mathrm{W}}$ lack their last row. This can be found by investigating the limit $d_{\mathrm{SC}} \to \infty$. The corresponding intrinsic camera calibration matrix is

$$\tilde{\boldsymbol{K}}_{\infty} = \begin{pmatrix} \alpha_{\mathrm{u},\infty} & s & \\ & \alpha_{v,\infty} & \\ & & 1 \end{pmatrix}$$

with

$$\alpha_{\mathrm{u},\infty} = \Delta u^{-1} \quad \text{and} \quad \alpha_{v,\infty} = \Delta v^{-1}$$

lacking the principal point $\boldsymbol{p}$ and the distance d_{SD} between source and detector.

Checking if $\boldsymbol{m}^{\mathrm{w}} = \mathbf{0}$ allows to identify an affine projection matrix. Modifications of $\tilde{\boldsymbol{P}}_{\mathrm{W},\infty}$ like $\tilde{\boldsymbol{T}}_{\mathrm{W}}$ and the ones described in the subsequent section work just the same as for any other projection matrix.

However, some of the operations described in algorithm 6 differ for an affine geometry. As a reminder, in a parallel-beam geometry all rays have the same direction and hit the detector perpendicularly. Instead of using the principal axis vector $\boldsymbol{m}^{\mathrm{w}}$, which is $\mathbf{0}$ here, this common direction of all rays can be found by solving

$$\boldsymbol{M}_{2\times 3} \cdot \boldsymbol{D} \overset{!}{=} 0$$

for $\boldsymbol{D}$ where $\boldsymbol{M}_{2\times 3}$ is the upper left 2×3 submatrix of $\tilde{\boldsymbol{P}}_{\infty}$. Appendix C shows a solution for $\boldsymbol{D}$ and its derivation. The position of the source is the point at infinity under the same direction, namely $\tilde{\mathrm{S}}_{\infty} = (\boldsymbol{D}, 0)^{\mathsf{T}}$. The main propagation direction is accordingly

$$\nu_{\infty} = \underset{n \in \{\mathrm{x,y,z}\}}{\arg\max} \left|\{\boldsymbol{D}\}_{n}\right|$$

and the direction vector needed to follow a ray through the volume $\boldsymbol{U}_{\infty} = \boldsymbol{D}$.

4.3. Some special use cases

A projection matrix can be multiplied with any other matrix and stays a valid projection matrix as long as the left-hand 3×3 submatrix does not become singular [47, following from the result on p. 157]. In this section we show how this property adds a lot of power to the projection matrices in CT for various use cases.

4.3.1. Sample rectification

Modifying the projection matrices allows to directly reconstruct into a rotated or shifted image volume. This can be very beneficial if the reconstructed object would be off-center or slanted with respect to the image voxel grid.

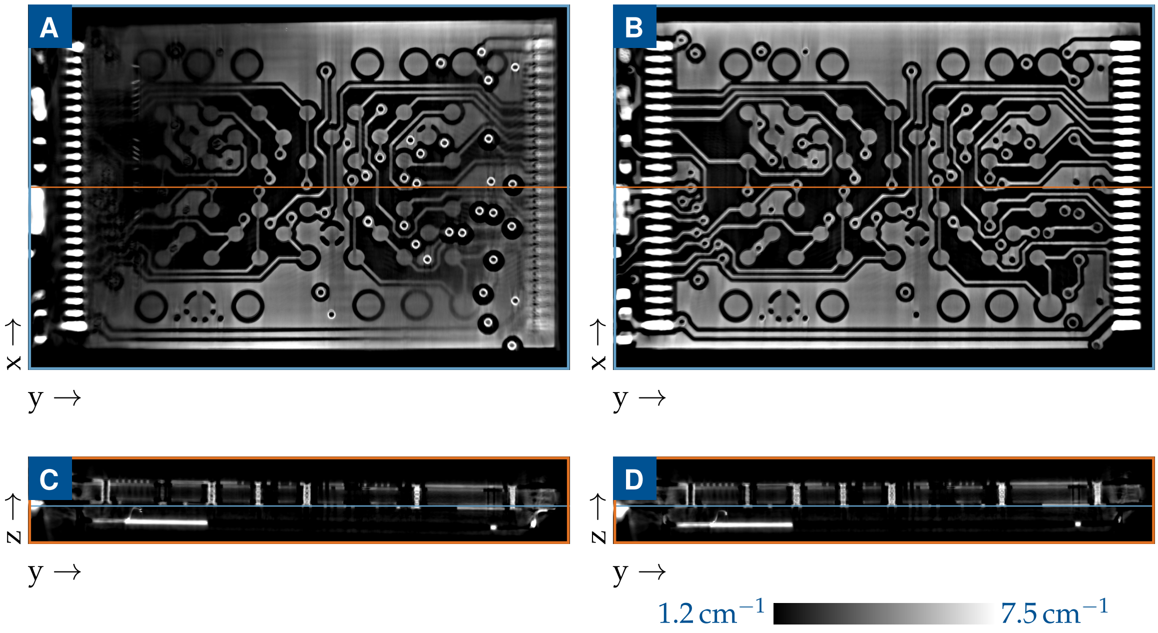


Figure 13 A fully-automated rectified reconstruction.
Two different reconstructions of the same micro-CT measurement show the benefit of aligning the sample to the voxel grid. The sample is an electronic chip, which was obviously mounted into the sample holder with a slight slant. The upper row shows coronal slices (A, B, blue frame) that are marked in the sagittal slices below (C, D, orange frame) by a blue line and vice versa. On the left is the result without alignment. The conducting path shown in the coronal slice (A) is mixed with the bonds to another layer in the right half of the image and with air in the left half. The reason is that the layers of the micro chip are not aligned with the voxel grid as shown in the sagittal slice (C).
On the right is the automatically rectified counterpart. The projection matrices used for reconstruction intrinsically included the required volume transformation without any extra interpolation. The coronal slice (B) contains clearly only the thin conducting path of interest, which is also visible in the sagittal slice (D). Different layers of the micro chip can easily be investigated by moving through the slices perpendicular to z. The parameters required for rectification, as defined in algorithm 7, resulted in $I_{\mathrm{bg}} = 0.950$, $\varphi_{\max \mathrm{sl}} = 77.7°$, $\theta_{\mathrm{sl}} = 1.096°$, and $\varphi_{\min \mathrm{th}} = -0.417°$.

The modified projection matrix

$$\tilde{\boldsymbol{P}}_{\mathrm{shift,rot}} = \tilde{\boldsymbol{P}} \cdot \tilde{\boldsymbol{T}}_{\mathrm{shift,rot}}$$

$$\text{with} \quad \tilde{\boldsymbol{T}}_{\mathrm{shift,rot}} = \begin{pmatrix} -\boldsymbol{R} & -\Delta P \\ \mathbf{0}^{\mathsf{T}} & 1 \end{pmatrix}$$

allows to rotate the volume by the rotation matrix $\boldsymbol{R}$ and shift it by the vector ΔP without any additional interpolation or computational effort directly within a tomographic reconstruction.

An example using this modification is algorithm 7. It explains a transformation that can be used to rectify a sample within the 3-D grid of voxels. Moreover, it suggests how to guess the required parameters fully-automated from the acquired transmission views. All the exemplary plots are results from rectifying the sample shown in figure 13. The great benefit of this estimation is that no prior tomographic reconstruction is involved. That makes it fast and practicable for large data sizes. Furthermore, it allows to know the required volume size before reconstruction. That allows to reconstruct only a subvolume and save a lot of compute time.

The assumptions of this algorithm are equidistant transmission views with a value range of $[0, 1]$ and moderate noise taken in an angular range of either $[0°, 180°[$ or $[0°, 360°[$. The sample must not exceed the u range of the detector in any view.

For the sample in figure 13, the parameters given in the caption indicate that already very small slant angles result in a bad slicing. With a sample height of $\#y = 1850\,\mathrm{vx}$, the accu-

Algorithm 7 Fully automated sample rectification

Adjusting a sample within a 3-D grid of voxels can be achieved by modifying $\tilde{\boldsymbol{P}}$ with the following rotation matrix:

$$\boldsymbol{R} = \mathbf{R}_{\varphi_{\text{min th}}} \cdot \mathbf{R}_{-\varphi_{\text{max sl}}} \cdot \mathbf{R}_{\theta_{\text{sl}}} \cdot \mathbf{R}_{\varphi_{\text{max sl}}}$$

From right to left, it performs the following steps:

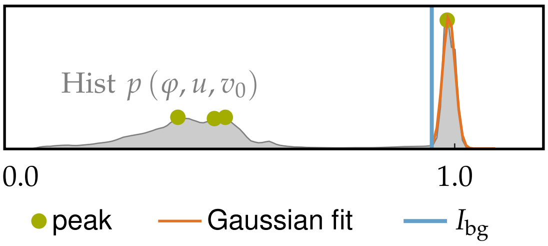
1. Rotate the volume by $\varphi_{\text{max sl}}$ around y so that the orientation of the sample with maximum slant is along z.
2. Rotate around z by the slant θ_{sl} so that the sample is upright.
3. Rotate back by $-\varphi_{\text{max sl}}$ around y.
4. Rotate the sample by $\varphi_{\text{min th}}$ around y so that the dimension of its thinnest extent is along z.

The required angles can be found by investigating the set of transmission views:

I. Generate a **mask**

$$M(\varphi, u, v) = \begin{cases} 1 & \text{for } I(\varphi, u, v) > I_{\text{bg}} \\ 0 & \text{else} \end{cases}$$

that is 1 where the sample is and 0 in the background. An according threshold I_{bg} can be guessed from the histogram:



II. Create an angle-dependent **broadness profile** curve

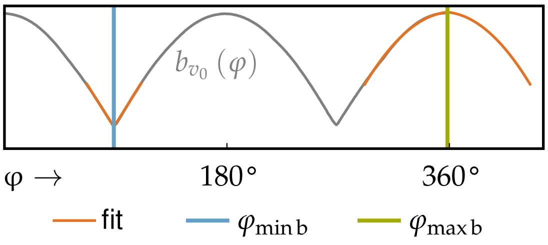
$$b_{v_0}(\varphi) = \int M(\varphi, u, v_0)\, \mathrm{d}u$$

by summing up the mask within any slice of interest v_0. The mean over several slices gives a more robust result.

III. The minimum $\varphi_{\text{min b}}$ of that profile corresponds to the narrowest view of an elongated sample. The view along the **minimum thickness** is therefore

$$\varphi_{\text{min th}} = \varphi_{\text{min b}} - 90°.$$

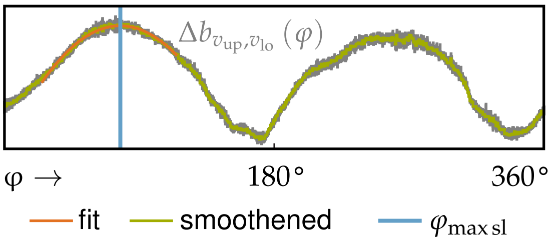
The minimum and the maximum values of the profile give also a hint for the optimum crop size in u, z and x, respectively:



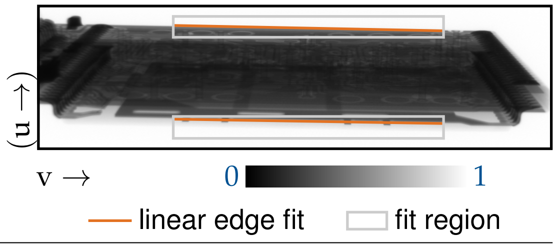
IV. The **angle of maximum slant** can be found by computing the difference between the masks in an upper and in a lower region of the sample

$$\Delta b_{v_{\text{up}}, v_{\text{lo}}}(\varphi) = \int \left| M(\varphi, u, v_{\text{up}}) - M(\varphi, u, v_{\text{lo}}) \right| \mathrm{d}u$$

and finding its maximum $\varphi_{\text{max sl}}$:



V. Finding the **slant angle** θ_{sl} can be done by a linear fit on the edges of the sample in the transmission view closest to $\varphi_{\text{max sl}}$. To compensate for perspective distortions, the edge on both sides can be investigated. The slant follows from the average of the two slopes:



racy of the slant angle has to be

$$\begin{aligned}\Delta\theta_{\mathrm{sl}} &= \tan^{-1}\frac{\#z}{\#y}\\ &= \tan^{-1}\frac{1\,\mathrm{vx}}{1850\,\mathrm{vx}} = 0.031\,°\end{aligned}$$

so that the reconstructed slices in z do not mix.

4.3.2. Correcting degenerated cone-beam geometries

There are some tomographic applications that differ a lot from the classical cone-beam CT geometry, where a source rotates around a sample on an either circular or helical trajectory. One on of them is tensor tomography [50]. It requires the views to cover most of the Euler space around the sample. Projection matrices are perfect to describe the geometry in such a situation. But, for a highly performance-optimized implementation like it is presented in part II, it is very useful if some general constraints on the geometry can be assumed.

The first beneficial assumption is that the main propagation direction ν is only either x or z. A view with main direction y can easily be converted into one with main direction z by changing the corresponding columns in the projection matrix

$$\tilde{\boldsymbol{P}}_{\mathrm{x}\leftrightarrow\mathrm{z}} = \tilde{\boldsymbol{P}}\cdot\tilde{\boldsymbol{T}}_{\mathrm{x}\leftrightarrow\mathrm{z}},$$

$$\text{with}\quad \tilde{\boldsymbol{T}}_{\mathrm{x}\leftrightarrow\mathrm{z}} = \begin{pmatrix}1 & & & \\ & & 1 & \\ & 1 & & \\ & & & 1\end{pmatrix}.$$

To compensate the error, the image volume can be resorted if necassary, so that y and z are swapped back. It would even be possible to restrict the main propagation direction to only one single dimension this way. But, as resorting is expensive, it would substantially slow down the reconstruction for standard geometries.

The second useful assumption is that the image coordinate y and the detector coordinate v have the same sign. That means, that the detector is not defined upside-down. There is an easy test if that is the case by projecting the unit vector $\tilde{e}_{\mathrm{y}} = (0,1,0,0)^{\mathsf{T}}$ and checking if

$$\operatorname{sign}\{\tilde{x}\}_{\mathrm{u}} = \operatorname{sign}\{\tilde{\boldsymbol{P}}\cdot\tilde{e}_{\mathrm{y}}\}_{\mathrm{u}} = \operatorname{sign} p_{\mathrm{u,y}}$$

is negative. In other words, the sign of the element u,y in the projection matrix tells the detector orientation.

It is obvious that an upside-down detector can be corrected with a $180\,°$ rotation of the affected view. The necessary modification of the projection matrix in order to compensate that operation

$$\tilde{\boldsymbol{P}}_{\mathrm{rot180}} = \tilde{\boldsymbol{T}}_{\mathrm{rot180}}\cdot\tilde{\boldsymbol{P}},$$

$$\text{with}\quad \tilde{\boldsymbol{T}}_{\mathrm{rot180}} = \begin{pmatrix}-1 & & \#u-1\\ & -1 & \#v-1\\ & & 1\end{pmatrix}$$

contains not only the rotation, but additionally has to apply an overall shift. The reason is that 2-D pixel arrays in a computer are always indexed with positive integer numbers. A rotation therefore changes the origin of the detector coordinate system. The pixel with index $(0,0)^{\mathsf{T}}$, for example, has the index $(\#u-1,\#v-1)^{\mathsf{T}}$ in the rotated array if $\#u$ and $\#v$ are the total number of pixels in u and v.

Third, some setups define the orientation of the detector vice versa so that its surface normal does not look along the ray but towards it. The result is that the u axis is mirrored resulting in very strange cone-beam artifacts if defined wrong in a tomographic reconstruction.

Similar to the transformation $\tilde{\boldsymbol{T}}_{\mathrm{rot180}}$ above, the corresponding correction reads

$$\tilde{\boldsymbol{P}}_{\mathrm{flip\,u}} = \tilde{\boldsymbol{T}}_{\mathrm{flip\,u}} \cdot \tilde{\boldsymbol{P}},$$

$$\text{with} \quad \tilde{\boldsymbol{T}}_{\mathrm{flip\,u}} = \begin{pmatrix} -1 & & \#u-1 \\ & 1 & 0 \\ & & 1 \end{pmatrix}.$$

An interesting property of all three transformations $\tilde{\boldsymbol{T}}_{\mathrm{x}\leftrightarrow\mathrm{z}}$, $\tilde{\boldsymbol{T}}_{\mathrm{rot180}}$ and $\tilde{\boldsymbol{T}}_{\mathrm{flip\,u}}$ is that both are their own inverse. That helps to transform the pseudo inverse of the projection matrix required to follow a ray through the image:

$$\begin{aligned} \tilde{\boldsymbol{P}}^{+}_{\mathrm{x}\leftrightarrow\mathrm{z}} &= \left(\tilde{\boldsymbol{P}} \cdot \tilde{\boldsymbol{T}}_{\mathrm{x}\leftrightarrow\mathrm{z}}\right)^{+} = \tilde{\boldsymbol{T}}_{\mathrm{x}\leftrightarrow\mathrm{z}} \cdot \tilde{\boldsymbol{P}}^{+} \\ \tilde{\boldsymbol{P}}^{+}_{\mathrm{rot180}} &= \left(\tilde{\boldsymbol{T}}_{\mathrm{rot180}} \cdot \tilde{\boldsymbol{P}}\right)^{+} = \tilde{\boldsymbol{P}}^{+} \cdot \tilde{\boldsymbol{T}}_{\mathrm{rot180}} \\ \tilde{\boldsymbol{P}}^{+}_{\mathrm{flip\,u}} &= \left(\tilde{\boldsymbol{T}}_{\mathrm{flip\,u}} \cdot \tilde{\boldsymbol{P}}\right)^{+} = \tilde{\boldsymbol{P}}^{+} \cdot \tilde{\boldsymbol{T}}_{\mathrm{flip\,u}} \end{aligned}$$

4.3.3. Corrections for rebinned data

In order to improve statistics or for reconstructing a coarser-sampled overview, several pixels or voxels are sometimes summed up in one big pixel or voxel. This technique is called *rebinning*. The factor by which a data set is scaled down (or up) is usually an integer so that interpolation is avoided.

If the image volume was binned down by the factors $\boldsymbol{K} = (k_{\mathrm{x}}, k_{\mathrm{y}}, k_{\mathrm{z}})^{\mathsf{T}}$ for the different dimensions, the corresponding projection matrix can be deduced from the original by applying

$$\tilde{\boldsymbol{P}}_{\mathrm{bin}\,\boldsymbol{K}} = \tilde{\boldsymbol{P}} \cdot \tilde{\boldsymbol{T}}_{\mathrm{bin}\,\boldsymbol{K}},$$

$$\text{with} \quad \tilde{\boldsymbol{T}}_{\mathrm{bin}\,\boldsymbol{K}} = \begin{pmatrix} \operatorname{diag}(\boldsymbol{K}) & \boldsymbol{\Pi} \\ \boldsymbol{0}^{\mathsf{T}} & 1 \end{pmatrix}$$

$$\text{and} \qquad \boldsymbol{\Pi} = \frac{1}{2}\boldsymbol{K} - \frac{1}{2}.$$

As the voxel size changes, the center of the voxels defining the position grid, including the origin, change, too. This fact is compensated by the shift $\boldsymbol{\Pi}$.

If the set of projections was binned down by the factors $\boldsymbol{k} = (k_{\mathrm{u}}, k_{\mathrm{v}})^{\mathsf{T}}$, the corresponding correction reads quite similar:

$$\tilde{\boldsymbol{P}}_{\mathrm{bin}\,\boldsymbol{k}} = \tilde{\boldsymbol{T}}_{\mathrm{bin}\,\boldsymbol{k}} \cdot \tilde{\boldsymbol{P}},$$

$$\text{with} \quad \tilde{\boldsymbol{T}}_{\mathrm{bin}\,\boldsymbol{k}} = \begin{pmatrix} \operatorname{diag}\left(\boldsymbol{k}^{-1}\right) & \boldsymbol{\pi} \\ \boldsymbol{0}^{\mathsf{T}} & 1 \end{pmatrix}$$

$$\text{and} \qquad \boldsymbol{\pi} = \frac{1}{2}\boldsymbol{k}^{-1} - \frac{1}{2}.$$

Rebinning both volumes involves both transformations. Simple relations describe the inverse transformations

$$\begin{aligned} \tilde{\boldsymbol{T}}^{-1}_{\mathrm{bin}\,\boldsymbol{K}} &= \tilde{\boldsymbol{T}}_{\mathrm{bin}\,\boldsymbol{K}^{-1}} \\ \tilde{\boldsymbol{T}}^{-1}_{\mathrm{bin}\,\boldsymbol{k}} &= \tilde{\boldsymbol{T}}_{\mathrm{bin}\,\boldsymbol{k}^{-1}}, \end{aligned}$$

which correspond to rebinning with inverse factors.

4.3.4. Computing the Feldkamp weights

In order to make a CT reconstruction framework as versatile as possible, it preferably is able to reconstruct from arbitrary projection matrices and requires no further information about the geometry. The assumptions made by the FDK algorithm must of course still be fulfilled. They include in particular equidistant views and a circular trajectory. Unfortunately, we found that deducing the Feldkamp weights only from the matrices does not work in general. The reason is that the projection matrices do not change if the whole geometry is stretched along one axis (including the pixel and voxel sizes), but the Feldkamp weights do. That issue can be solved by additionally providing the absolute pixel dimensions Δu, Δv and voxel dimensions Δx, Δy, Δz.

The two Feldkamp weighting factors were introduced in algorithm 2 on page 25. The first one can be written as

$$w_{\mathrm{F}}\left(\boldsymbol{x}_{\mathrm{W}}\right) = \frac{d_{\mathrm{SD}}}{\sqrt{d_{\mathrm{SD}}^2 + \left\|\boldsymbol{x}_{\mathrm{W}}\right\|^2}}$$

where $\boldsymbol{x}_{\mathrm{W}}$ describes a position on the detector in orthogonal detector coordinates with a real-world aspect ratio measured from the principal point. These coordinates can be obtained from the detector array indices $\boldsymbol{x}$ by

$$\begin{aligned} \boldsymbol{x}_{\mathrm{W}} &= \boldsymbol{T}_{\mathrm{px}} \cdot \left(\boldsymbol{x} - \boldsymbol{p}\right) \\ \text{with}\quad \boldsymbol{T}_{\mathrm{px}} &= \begin{pmatrix} \Delta u & -s \\ & \Delta v \end{pmatrix}. \end{aligned}$$

The required coordinates of the principal point $\boldsymbol{p}$ can again be computed from $\tilde{\boldsymbol{p}} = \tilde{\boldsymbol{M}} \cdot \boldsymbol{m}^{\mathrm{w}}$ [47, pp. 160f].

Unfortunately, the distance from the source to the detector d_{SD} is still unknown. It can be retrieved by investigating the following alternative definition of the first Feldkamp weight. Considering the fact that the weight is the absolute steepness of a ray $\boldsymbol{U}_{\mathrm{W}}$ relative to the central ray $\boldsymbol{m}_{\mathrm{W}}^{\mathrm{w}+}$, it can also be written as

$$w_{\mathrm{F}}\left(\boldsymbol{x}\right) = \cos\kappa = \cos\sphericalangle\left(\boldsymbol{U}_{\mathrm{W}}, \boldsymbol{m}_{\mathrm{W}}^{\mathrm{w}+}\right)$$

with the cosine of the angle between two vectors

$$\cos\sphericalangle\left(\boldsymbol{A}, \boldsymbol{B}\right) = \frac{\boldsymbol{A} \cdot \boldsymbol{B}}{\|\boldsymbol{A}\|\|\boldsymbol{B}\|}.$$

Both vectors, $\boldsymbol{U}_{\mathrm{W}}$ and $\boldsymbol{m}_{\mathrm{W}}^{\mathrm{w}+}$, describe the direction of a ray through the image volume and must have coordinates with a real-world aspect ratio in order to obtain the correct angle. They can be computed from the corresponding vectors $\boldsymbol{U}$ and $\boldsymbol{m}^{\mathrm{w}+}$ in image voxel coordinates by applying the transformation

$$\boldsymbol{T}_{\mathrm{vx}} = \operatorname{diag}\left(\Delta x, \Delta y, \Delta z\right)^{\mathsf{T}}.$$

The vector

$$\boldsymbol{m}^{\mathrm{w}+} = \operatorname{sign}\left(\det \boldsymbol{M}\right)\, \boldsymbol{m}^{\mathrm{w}}$$

denotes the direction of the principal ray with positive orientation [47, p. 162] pointing from the source to the detector. $\boldsymbol{U}$ is the direction of the ray of interest and can be computed from $\boldsymbol{x}$ as shown before in algorithm 6 on page 35.

Comparing the two different definitions or, alternatively, a simple geometrical consideration yields

$$\begin{aligned} d_{\mathrm{SD}} &= \frac{\left\|\boldsymbol{x}_{\mathrm{W}}\right\|}{\tan\kappa} \\ &= \frac{\left\|\boldsymbol{x}_{\mathrm{W}}\right\|}{\sqrt{\cos\sphericalangle\left(\boldsymbol{U}_{\mathrm{W}}, \boldsymbol{m}_{\mathrm{W}}^{\mathrm{w}+}\right)^{-2} - 1}} \end{aligned}$$

using $\tan\kappa = \sqrt{\left(\cos\kappa\right)^{-2} - 1}$. Inserting any detector point $\boldsymbol{x} \neq \boldsymbol{p}$ by deriving $\boldsymbol{x}_{\mathrm{W}}$ and $\boldsymbol{U}$ from it allows now to obtain d_{SD}.

The second part of the Feldkamp weights

$$w_{\mathrm{BP}}\left(\boldsymbol{X}\right) = \left(\frac{d_{\mathrm{SC}}}{d_{\mathrm{SX}'}}\right)^2$$

is the inverse ray density at a certain point $\boldsymbol{X}$ of the image volume. The distance $d_{\mathrm{SX}'}$ is measured from the source to the projection of that image point to the principal axis. Therefore, it can alternatively be written as

$$d_{\mathrm{SX}'} = \boldsymbol{U}_{\mathrm{W}} \cdot \frac{\boldsymbol{m}_{\mathrm{W}}^{\mathrm{w}+}}{\left\|\boldsymbol{m}_{\mathrm{W}}^{\mathrm{w}+}\right\|}$$

projecting the direction of the ray $\boldsymbol{U}_{\mathrm{W}}$ onto the positive principal axis $\boldsymbol{m}_{\mathrm{W}}^{\mathrm{w}+}$. Both have to be given in coordinates of real-world aspect ratios as above. The direction of the ray passing through $\boldsymbol{X}$ can be computed from $\boldsymbol{U} = \boldsymbol{X} - \boldsymbol{S}$, where $\boldsymbol{S}$ is the position of the source.

Computing the distance d_{SC} from the source to the center of rotation is also possible from the projection matrices. Assuming a circular trajectory as required for FDK, the center of rotation can be obtained by intersecting one or more principal rays. Writing two corresponding rays g and h leaving from the principal point as

$$\begin{aligned} \text{g:} \quad & \boldsymbol{X}_{\mathrm{g}}\left(\xi_{\mathrm{g}}\right) = \boldsymbol{X}_{0,\mathrm{g}} + \xi_{\mathrm{g}}\,\boldsymbol{U}_{\mathrm{g}} \\ \text{h:} \quad & \boldsymbol{X}_{\mathrm{h}}\left(\xi_{\mathrm{h}}\right) = \boldsymbol{X}_{0,\mathrm{h}} + \xi_{\mathrm{h}}\,\boldsymbol{U}_{\mathrm{h}} \end{aligned}$$

with $\xi_{\mathrm{g}}, \xi_{\mathrm{h}} \in \mathbb{R}$, similar to the ray in algorithm 6 on page 35, the parameters ξ'_{g} and ξ'_{h} for the points where the rays intersect (or come closest) can be computed as

$$\begin{pmatrix} \xi'_{\mathrm{g}} \\ \xi'_{\mathrm{h}} \\ \xi'_{\mathrm{n}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{U}_{\mathrm{g}} & -\boldsymbol{U}_{\mathrm{h}} & \boldsymbol{U}_{\mathrm{g}} \times \boldsymbol{U}_{\mathrm{h}} \end{pmatrix}^{-1} \cdot \left(\boldsymbol{X}_{0,\mathrm{h}} - \boldsymbol{X}_{0,\mathrm{g}}\right).$$

The third component ξ'_{n} is not of interest. Inserting one of the two parameters into its ray equation yields the desired center of rotation

$$\boldsymbol{C} = \boldsymbol{X}_{\mathrm{g}}\left(\xi'_{\mathrm{g}}\right)$$

and its distance to the source

$$d_{\mathrm{SC}} = \left\| \boldsymbol{S}_{\mathrm{W}} - \boldsymbol{C}_{\mathrm{W}} \right\|_2$$

after transforming the coordinates again to the real-world aspect ratio.

If the image voxels are cubic, the detector pixels are square and the absolute value of d_{SC} is not of interest, the real-world transformations can be simplified to

$$\begin{aligned} \boldsymbol{T}_{\mathrm{vx,sq}} &= \mathbb{1}^3 \\ \boldsymbol{T}_{\mathrm{px,sq}} &= \frac{\Delta x}{\Delta u}\,\mathbb{1}^2 \end{aligned}$$

because only the relative coordinate ratios matter for the Feldkamp weights. This simplification allows to normalize the projection matrices such that $\|\hat{\boldsymbol{m}}^{\mathrm{w}}\| = d_{\mathrm{SC}}^{-1}$ [51] and $\det \hat{\boldsymbol{M}} > 0$. The second Feldkamp weight then reduces to

$$w_{\mathrm{BP,sq}}\left(\boldsymbol{X}\right) = \left(\boldsymbol{U} \cdot \hat{\boldsymbol{m}}^{\mathrm{w}}\right)^{-2}.$$

Closure

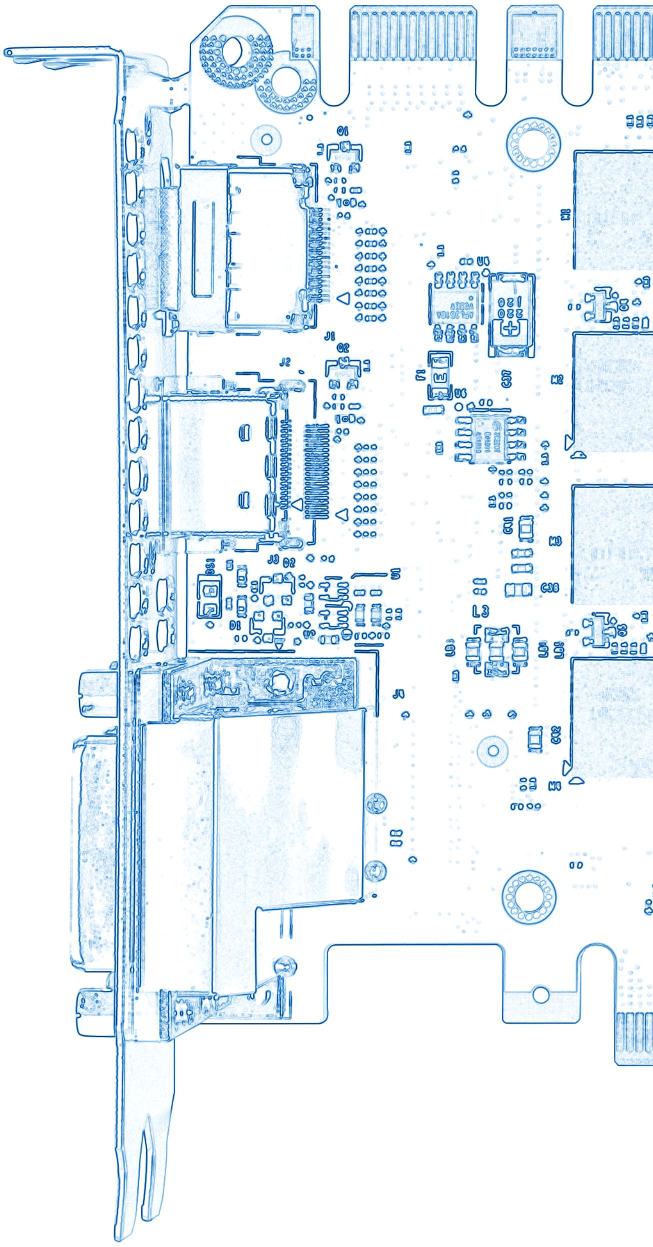
Projective geometry is a great asset for CT reconstruction. The projection matrices provide a simple and direct linear mapping from voxel to pixel indices including all the required geometric properties in an unambiguous way. All linear operations necessary for CT reconstruction are expressed by linear algebra and therefore promise a fast implementation on a computer. (An example for a non-linear operation is the reconstruction from a curved detector, which would have to be applied separately.)

The nature of the projection matrices allows to combine them with any coordinate transformation in matrix form. This feature was demonstrated at the example of an intrinsic sample rectification.

A shortcoming is that it is sometimes hard to mix projective geometry with the conventional understanding of geometry including distances and angles. There is also a geometric property that is not encoded in the projection matrices, namely the real-world aspect ratio of the coordinates for non-uniform voxels and pixels. Until now, we saw only the Feldkamp weights suffering from these two problems, but showed that it is still possible to compute them by adding the correct voxel and pixel dimensions.

5. Heterogeneous computing

One major challenge for CT reconstruction is that complex algorithms have to be applied to huge amounts of data, especially for iterative techniques. This chapter introduces concepts to get the maximum computing performance from modern computer systems. There are usually different types of processors available, namely the central processing unit (CPU) and the graphics processing unit (GPU), allowing a task-based hardware choice. After discussing the basic concepts of heterogeneous computing and parallelization, the GPU architecture is discussed in more detail. A special focus is put on General-purpose computation on GPUs (GPGPU). It allows to use the GPU as a general-purpose device for highly-parallel computations not only restricted to graphic applications.



Picture: The inner part of a modern GPU device. 🅭

5.1. Basic concepts

For a proper understanding of heterogeneous computing it is useful to have a look on the development of the computing power over the last years as depicted in figures 14 and 15. At the time when the clock frequency could not be pushed anymore, multi-core CPUs started to be developed. Parallel computing became a necessity to exploit their full performance. At the same time, the gaming industry pushed the development of highly-parallel GPU devices that are able to cover the high demands of their graphics engines.

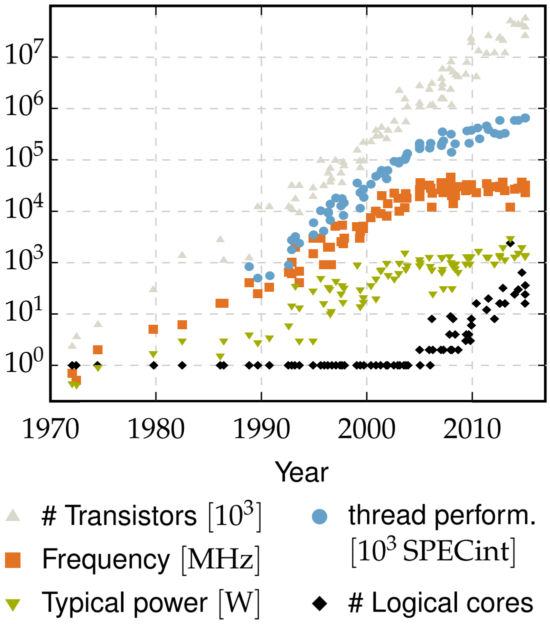


Figure 14 Microprocessor evolution over the last 45 years.
Rupp [52] collected several characteristic performance parameters of microprocessors over the last decades. The trend shows that at the time the clock frequency stopped increasing, multi-core processors became popular. The single-thread performance was measured in terms of the SPECint standard [53]. 🅭

5.1.1. Parallel computing

Parallelization means writing a computer program in a way that a task is split into several subtasks that can be executed simultaneously. Modern processors like CPUs and GPUs allow several different ways to realize that concept. One independent work unit is called a thread.

The degree to which a program can be parallelized depends on the task. If a subtask depends on the result of another subtask, they necessarily have to run in sequence. Furthermore, two threads are not allowed to modify the data at the same position in memory at the same time. This would cause a so-called data race, which means that it is not defined which, if any, of the two modifications is applied. For a fixed data size, the maximum speedup of a program is given by Amdahl's law

$$S = \frac{1}{1 - p + p/s}$$

where p is the fraction of parallelizable code measured by its runtime before parallelization and s is the degree of parallelization the code allows [54, 55]. The speedup therefore not simply scales with the number of parallel work units, but depends on the non-parallel overhead of the program. Its ultimate limit is $(1-p)^{-1}$ given by the serial parts. In image processing, p is usually very close to 1 because the operations can often by applied to each pixel or voxel independently.

A parallel program is in general not strictly deterministic. Because of the limited floating-point accuracy, the numerical result after several operations on a floating-point variable depends on their order. But, the order in which threads are executed is usually not defined. Often it is not even defined, which threads run in parallel and which in sequence. Fortunately, these errors are usually small enough to be neglected, especially if the accuracy is sufficient for a serial version of the program.

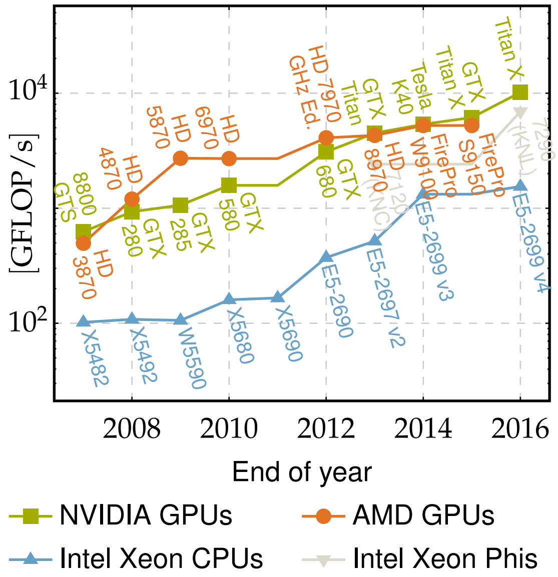


Figure 15 Theoretical CPU and GPU single-precision peak performance over the last decade.
Rupp [56] also provides an overview of the number of single-precision floating-point operations per second (FLOP/s) that could be performed by CPU and GPU processors during the last years. All data points are labeled with the corresponding device name. The GPU is roughly always an order of magnitude ahead. 🅭

5.1.2. Types of parallelism

CPUs and GPUs offer different types of parallelism suitable for different kind of tasks. There are various frameworks to make use of them.

The most high-level type is computing in a cluster or in a grid. Thereby, the workload is distributed over several independent computers connected via network or the Internet. This type is therefore highly scalable, but the communication between the different compute nodes is slow compared to other types. Common frameworks to use this kind of parallelism are the *Open Message Parsing Interface* [57] (Open MPI) or specialized grid computing solutions like the *Berkeley Open Infrastructure for Network Computing* (BOINC) [58].

The next type, multi-threading, is probably the most wide-spread one. Several threads are assigned to the different cores of a CPU allowing to simultaneously compute on completely independent tasks. On top, server mainboards usually allow to install two to four CPUs to increase the number of available cores. The *Hyperthreading* technology by *Intel* additionally allows to share a core between two threads increasing the speedup for threads that do not fully occupy the core all the time. The different threads share the same memory space and can thus communicate very quickly, but the amount of parallelization is very limited. Common frameworks allowing to use multi-threading are

- *Open Multi-Processing* (OpenMP) [59], an application programming interface (API) supported by many C, C++ and Fortran compilers,
- *POSIX threads* (pthreads) [60],
- OpenMPI, and
- the *Open Computing Language* (OpenCL) [61], a powerful API to make use of many different heterogeneous computing systems.

The streaming multiprocessor of a modern GPU also allows executing multiple threads. Compared to a CPU processor, the number of threads is roughly two orders of magnitude higher reaching several thousands, but the threads are not entirely independent from another. If two threads within a so-called *wavefront*, i. e. a set of usually 32 threads, fall into different code paths, each code path has to be evaluated sequentially. This effect is called branching or wavefront divergence. There are also restrictions on the access patterns for the different types of memory in order to keep them simultaneous. Details will be explained in the next section. The most common frameworks to use the GPU for computing are *CUDA* [62], an API which is limited to *Nvidia* devices, and OpenCL. Current main boards allow up to four GPU devices within one compute server. Parallelization over multiple GPU devices is called multi-GPU computing.

The last type of parallelization is usually offered by all modern CPUs. They provide instructions for vector parallelization which allow *Single Instruction, Multiple Data* (SIMD) execution. A whole vector of memory elements is modified at once by the same operation. One recent set of vector instructions are the *Advanced Vector Extensions 2* (AVX 2) [63] supporting $256\,\mathrm{bit}$ operations, which are, for example, eight float32 numbers at once. A disadvantage is that proper vectorization can hardly be reached without a lot of extra programming effort, which often makes the code even dependent on a certain compiler or hardware. Modern compilers already provide very limited support for automatic vectorization.

5.1.3. Multiple abstraction layers

Code for high-performance computing (HPC) often needs to be written in a low-level, machine-oriented compute language like C or C++ in order to reach the desired performance or take advantage of the APIs discussed above. In contrast, low-level languages, like Python, R or Matlab, provide great usability, are easy to read and write and are equipped with many tools for fast data analysis.

Fortunately, problems often can be divided into an algorithmic part and some generic operations that require a lot of computing power. That allows to keep the algorithmic part simple, user friendly and easy to modify by using a high-level language and move the performance-critical, static parts into fast C/C++ modules. A wrapper function cares for the interaction. Two possible APIs to combine, for example, Python with C are the *ctypes* Python module [64] extending Python with the functionality to use C/C++ libraries or the *Python/C API* [65] allowing to add a Python interface in the C/C++ code.

5.1.4. Data types

Different hardware used for heterogeneous computing supports different sets of data types. Recent CPU processors usually have native support for float32, float64, which are also called single and double floating-point precision, and various integer types. GPUs have good support for integer types, float32 and float16, which is called half precision. Double precision is often available, but usually significantly slower.

The precision required for a certain task depends on its numerical error. It is caused by rounding the result of each operation to the next floating-point or integer number that can be represented. For CT reconstruction, many of the demanding operations are big sums of values in the same order of magnitude. Algorithm 8 provides a C snippet which increments a floating-point variable by 1 until the representation of the incremented number equals its predecessor. It turns out that the result is

$$n_{\mathrm{max\,elem}} = 2^{M+1}$$

where M is the floating-point significand. This makes perfect sense, because numbers higher than this value require an exponent that is greater than one and thus have a spacing greater one. More results are displayed in table 1. The numbers suggest that neither half- nor single-precision are suited to sum up whole tomographic data sets without extra effort as they usually have several hundred to thousand voxels cubed. But, they are suited

Algorithm 8 Assessing the maximum number of float32 terms in a sum. (C code)

```c
#include <stdio.h>

int main() {
    float  s = 0.;
    while (s++ != s);
    printf("element: %d\n", (int)s);

    return 0;
}
```

Output: `element: 16777216`

Precision	Max. # of elements	Equivalent
half	$2^{11} = 2048$	$\approx 12.7^3$
single	$2^{24} = 16777216$	$= 256^3$
double	$2^{53} = 9007199254740992$	$\approx 208064^3$

Table 1 Estimate of the maximum number of elements in a floating-point sum depending on the precision with a volume equivalent.

to sum over one single dimension.

5.1.5. Choosing the right hardware

The variety of available HPC modalities raises the question for the best choice. Tomographic reconstruction is a very demanding computational operation, especially for high-resolution data sets. For example, the micro-CT data set of the electronic chip that was shown in figure 13 on page 37, has 3999 views of $(2048\,\mathrm{px})^2$ occupying $62.5\,\mathrm{GiB}$ of memory in float32 representation. The reconstruction result is a volume of $(2048\,\mathrm{vx})^3$ consuming $32.0\,\mathrm{GiB}$. In other words it is a linear system with $[p] = 1.7 \times 10^{10}$ equations and $[\mu] = 8.6 \times 10^9$ unknowns. Automatic cropping as introduced in algorithm 7 on page 38 can reduce the sizes to $41.8\,\mathrm{GiB}$ for p and $4.2\,\mathrm{GiB}$ for μ in this case, but the computational effort stays very demanding.

Cluster or grid computing involves expensive devices and a big infrastructure. Cloud computing is another possible solution, but additionally requires transferring the data to a remote service via the Internet and is thus very limited by the available bandwidth. That is why those options can usually be ruled out for CT reconstruction, especially as the required data still fits into the memory of a single-server system.

In a single-server system there are typically two options: Multi-threading on CPUs, possibly with SIMD operations, or multi-GPU computing. Figure 15 illustrated that the number of single-precision floating-point operations per second $(\mathrm{FLOP/s})$ possible on GPUs has constantly been kept about an order of magnitude over the available high-performance server CPUs during the last years. Because of their wide distribution in the consumer market, GPU devices are furthermore under rapid development and available at low costs. This is why the GPU should definitely be considered for CT. Exploiting its high performance is however restricted to highly-parallel applications and limited by some special architecture properties described in the next section.

5.2. GPGPU computing

GPGPU stands for *general-purpose computation on GPU devices*. The nomenclature used to describe the different concepts in this section corresponds to the *Open Compute Language* (OpenCL) [61]. It introduces only the basic ideas. More details about how to use those GPU devices in practice can be found in literature [66, 67, 68, 69, 70, for example] or, concerning CT, in part II of this thesis.

5.2.1. Architecture and parallelization

The streaming multiprocessor of a GPU has the ability to process up to several thousand threads executing basically the same operations on different data. Each thread processes *one work item* at a time.

The work items are further organized in *workgroups* of user-defined size. All work items in a workgroup are guaranteed to run in parallel at the same time. Depending on the size of the multiprocessor, several workgroups can also run in parallel. The optimum workgroup size is thus a multiple of the wavefront size and a divider of the total number of threads available, which, itself, is commonly a multiple of 512.

A *kernel* function defines the work for each work item depending on its unique integer identifier.

5.2.2. Memory and data transfer

The various types of memory available on a GPU device offer different options to optimize the data flow depending on the application. As thousands of threads run in parallel, fast data flow is probability the most critical issue for GPU programming. Figure 16 holds an overview of the available memory types, their scope, size and speed.

Each type of memory has special properties. Global memory accounts for almost the whole memory available, is visible from all work items and persistent between two kernel executions. However, it has a latency of several hundred cycles for cache misses and is best accessed only coalesced by many work items in succession. Latency and transfer time can be hidden by computations in some applications.

The very limited constant memory is read-only for the GPU, but has a very good caching behavior. It is especially fast if all work items access the same value at once.

Images can be seen as one-, two-, or three-dimensional arrays within global memory. A special data arrangement allows to cache values that are spatially close to the previous access. Images furthermore support floating-point indices with optional hardware interpolation and indices that are out-of-range offering various ways to treat the borders.

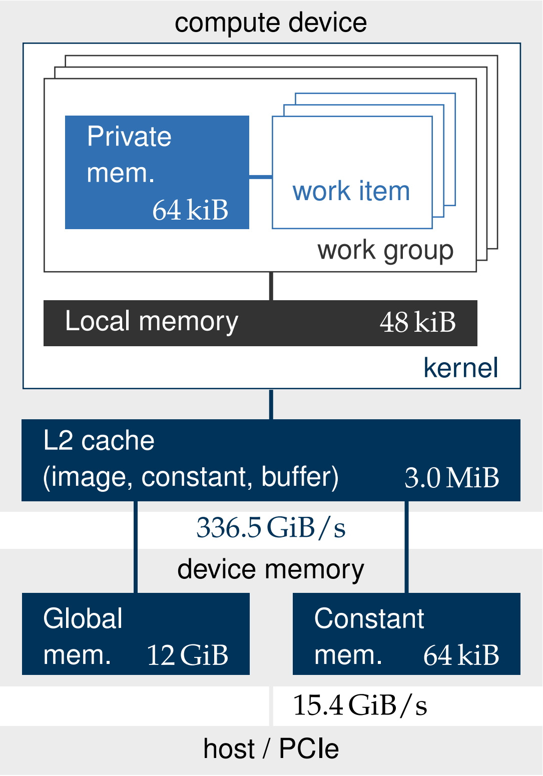


Figure 16 The GPU memory layout.
Each solid rectangle stands for a physical device and each framed rectangle for a memory scope in the OpenCL programming scheme. The exemplary sizes shown are the theoretical values for a *Nvidia GeForce GTX Titan X* graphics board [71] connected via PCIe 3.0 16x.
Each work item has a set of registers available (i. e. private memory) which are only persistent during its runtime. All work items within a work group have access to the same local memory and all work items within a kernel call or subsequent kernel calls to the same global, image (i. e. texture) and constant memory. The latter three are located off-chip in the device memory and connected via the on-chip L2 cache. Data can be exchanged with the host memory via PCIe.
Off-chip memory access is several hundred times slower [70, 72] than on-chip memory access. The communication with the host is more than a magnitude slower than access to the device memory.

The very fast but limited local and private memory spaces are only visible during the runtime of a work item. Local memory is shared within a whole work group.

Transferring data from or to host memory is by far the slowest process in the chain. It is limited by the throughput of the *Peripheral Component Interconnect Express* (PCIe) bus. That is the reason why small computations are usually faster on a CPU than on a GPU device. Modern devices offer ways to pull or push data while still computing on other data. That way, the transfer times for compute-intensive problems that can be split into several chunks of data, can be hidden almost entirely. This technique plays a major role in the implementation presented later.

A last way to store data on a GPU can be exploited for variables that do not change during the whole program execution. As the GPU kernels are compiled during the runtime of the program, it is possible to pass variables as macros to the compiler. This way they consume neither extra memory nor transfer time during the kernel run. In some cases it is even possible that the optimization algorithms of the compiler can save operations or reduce the size of the program by knowing the values at compile time. This is especially the case if those macro variables control branches. If a kernel is called many times with a limited set of different macro variables, it is possible to store a compiled kernel for each macro set and reuse it.

5.2.3. Profiling

Profiling allows to analyze and optimize the performance of a computer program by measuring some parameters during its runtime. Those parameters can for example be the required memory or the transfer- and runtimes of individual subtasks or components in the program. The analysis reveals for example parts with a bad performance, a bottleneck in the data transfer or waiting times for parallel processes. Depending on their impact on the overall performance, those slow parts can then pointedly be optimized.

Closure

Since the upper limit of the clock frequency in processing units has remained constant for many years, parallel computing became a necessity for compute-intensive tasks. Heterogeneous computing is the idea to choose different hardware depending on the task. Comparing the advantages and disadvantages of the available types of parallelization, it turns out that multi-threading and GPU computing are very good choices for CT reconstruction. The GPU architecture offers many different features to maximize the data flow.

Part II

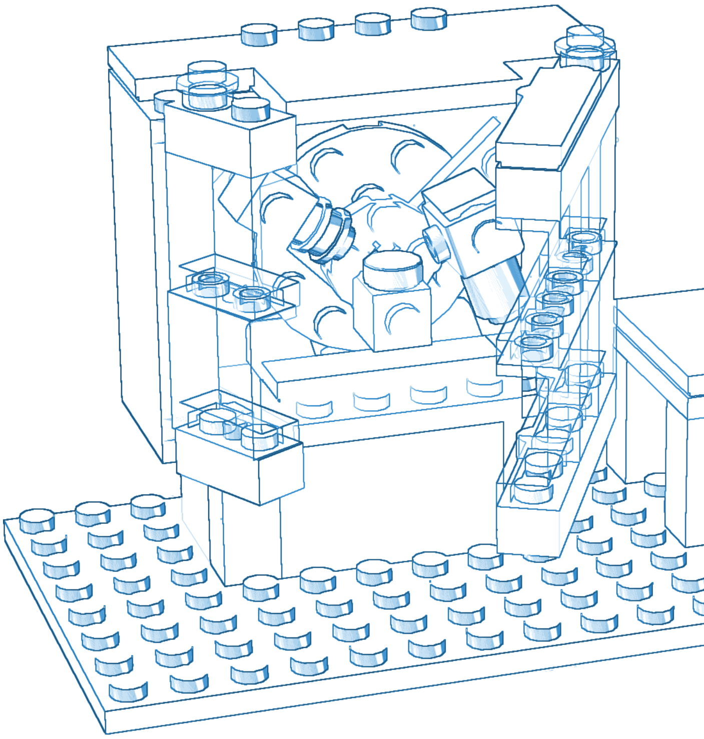
Implementation

In history, iterative reconstruction had not played a role in practical applications for a long time because of its high computational demand. The rapid hardware developments of the recent years gave it a new revival. This part about the implementation of a high-performance CT reconstruction framework puts the concepts from the previous part into practice and explains the various details required to make SIR convenient even for large data sets. First, the core components are introduced, which can be seen as the building blocks of the framework including, among others, the GPU implementation of the tomographic forward and back projector (chapter 6). Second, an overview of the whole reconstruction model is given, especially how it is possible to keep it both, flexible and fast (chapter 7). Third, an integral optimization of the whole framework performed to fine-tune it for maximum performance (chapter 8).

6. The building blocks

The core components investigated in this chapter are the basic building blocks for our CT framework. It holds all the computationally demanding operations and shows how to implement them best on heterogeneous hardware. The performance of these basic elements determine in the end the performance and therefore the convenience of iterative reconstruction.

Some preliminary considerations at the beginning explain how to identify those core components and how data is organized best. Next, the beating heart of the implementation is investigated in detail, namely the GPU kernel functions and, subsequently, the way quasi nonstop computation can be guaranteed despite the limited memory resources of GPU devices. The chapter closes with some practical remarks on the OpenCL framework and a systematic optimization of the kernel and transfer parameters.



Picture: LEGO x-ray machine.

© by the Boise State University

6.1. Preliminary considerations

The idea to identify some compute-intensive static core components and connect them by a flexible and user-friendly high-level algorithmic part was already presented in subsection 5.1.3 on page 48. This thesis concentrates mainly on statistical iterative reconstruction (SIR), which is considered the most general and powerful approach, as well as the two analytical approaches, the FBP for parallel-beam and the FDK for cone-beam geometries. Other methods like SART, for example, can be implemented likewise.

Identifying the core components

Having a look into appendix A on page 110 on the functions different solvers require for SIR, the following common building blocks can be identified (with abbreviations):

FP the product of the system matrix $\boldsymbol{A}$ with a vector,

BP the product of $\boldsymbol{A}^{\mathsf{T}}$ with a vector, and

REG the different voxel-wise operations for the Gibbs neighborhood prior with exchangeable potential function and optionally a sum over all voxel elements.

The first two operations are also known as forward (FP) and back projection (BP) switching from image to projection space and vice versa. They must be able to work on a subset of the projections for the ordered-subset solvers described above. Since projecting a subset is the same operation as projecting a smaller complete CT data set, this requirement is usually fulfilled for any implementation.

Analytical reconstruction requires the same back projection for parallel-beam geometry (FBP) optionally extended with the Feldkamp weights w_{BP} for cone-beam geometry (FDK) as seen in section 3.2 on page 21. In addition, there is

FF the Fourier filter step

optionally with the according Feldkamp weights w_{F}.

Choosing the right hardware

As seen in subsection 5.1.5 on page 49, the most suitable hardware setup for CT reconstruction is a single-server system holding CPUs and GPUs. There are two deciding questions for each of the four core components above. The first is which hardware offers the best performance independent of the other components. This first question depends on whether the degree of parallelism is high enough to benefit from GPU acceleration and if the computational effort justifies the expensive data transfer to the GPU devices. The second point can also mean if the computation takes long enough to hide the transfers sufficiently. The second deciding question is if there are core components that can run in parallel on different devices. If that is the case, it might be more beneficial to take the less-performant hardware before leaving it idle.

Concerning the first question, the GPU turned out to be perfectly suited for FP and BP as many previous implementations proved [73, 74, 75] over the last decade. Only very recently a SIMD CPU approach was able to reach comparable performance [76], which was not shown for big data sets. The REG operates voxel-wise and is thus also expected to allow a degree of parallelism that is high enough to benefit from GPU acceleration. But it has a small computational effort compared to the amount of data involved. For each voxel, a function is evaluated on each of its neighbors. If using GPUs is beneficial at all, the data transfers are very likely to dominate the run time. The FF depends mostly on a fast implementation of the Fourier transform and its inverse. Also here, the GPU offers a better performance, especially for big data sizes [77, 78].

Concerning the second question, many possibilities can be excluded in advance. Everywhere FP and BP occur together, one depends on the result of the other. An example is the gradient of the data term where the BP takes the result of the FP. The FF and BP for analytical approaches also need to be applied in sequence. The only operation left is REG, computing the regularization term and its derivations. It could possibly run parallel to the according evaluation of the data term. Reasonably, operations with higher effort have greater priority in the choice of hardware. In this case, this is clearly FP and BP.

In conclusion, the hardware for the FF, FP and BP can be chosen independently. The best-suited choice for all three of them is a multi-GPU implementation. For the REG it is not yet clear, which option is better, neither for a stand-alone operation, nor for its operation together with the data term, where it might be beneficial to run it parallel on the idle CPUs. Therefore, both have to be implemented, a multi-GPU version and a multi-threading, possibly vector–accelerated CPU version.

The platform used for development was a *Linux 64-bit* system, all the C/C++ source code is however equipped with platform-independent *CMake* files. CMake is an open-source cross-platform tool to build software [79].

Data representation

The functions $\mu\left(x,y,z\right)$ and $p\left(\varphi,u,v\right)$ have to be discretized in two ways in order to use them in a computer. First concerning the underlying 3-D grid of coordinates and second concerning the values. The discrete versions will be denoted with square brackets.

We decided to store $\mu\left[x,y,z\right]$, i. e. the vector $\boldsymbol{\mu}$, and $p\left[\varphi,u,v\right]$, i. e. the vector $\boldsymbol{p}$, in 3-D Python Numpy arrays. This choice enables a much more intuitive access to the data as suggested by the one-dimensional vector notation commonly used to describe iterative reconstruction (like in section 3.3). Nevertheless, the underlying data array is enforced to be linear with C-contiguous order. This restriction enables direct access from our C/C++ modules without expensive resorting. In concrete terms, that means for $\boldsymbol{\mu}$ that z is the slowest axis (first index) followed by y and x, which is the fastest (last index). The corresponding order for $\boldsymbol{p}$ is $\mathrm{\varphi}$, v, and u from slowest to fastest. Array alignment to multiples of at least $16\,\mathrm{Bytes}$ is usually guaranteed from the compiler for any modern build in order to take advantage of vectorization.

For simplicity and the means of maximum performance, we decided for cuboid image voxels and (optionally slanted) rectangular detector pixels. That means that the discrete functions represented by the data arrays are piecewise constant and have discontinuous jumps at the voxel or pixel borders. Integer positions on the underlying 3-D grid are always identified with the center of a voxel or pixel. The sampling is given by the detector in measurements or has to be defined manually in simulations.

The data type for storing the values has to be chosen with care. It has a great impact on the performance of the implementation. Integer data types would be well-suited to represent the output of many detectors but can be ruled out with a look on the functions that have to be applied inside the reconstruction. Their lack of decimal places does not work well with the continuous value range of the weights inside the system matrix $\boldsymbol{A}$, for example, or the exponential function and the logarithm in Lambert-Beer's law and its inverse.

The remaining options are floating-point numbers with different accuracy. Higher accuracy results in less numerical errors, but lower accuracy saves memory and, in particular, expensive data transfer times. We will see later that the last point is very deciding for the performance of all operations, including the core operations defined as the most compute-intensive ones. Therefore, the accuracy is best chosen as low as possible to still cover the data range, but also match the required accuracy of the operations in order to avoid expensive type conversions.

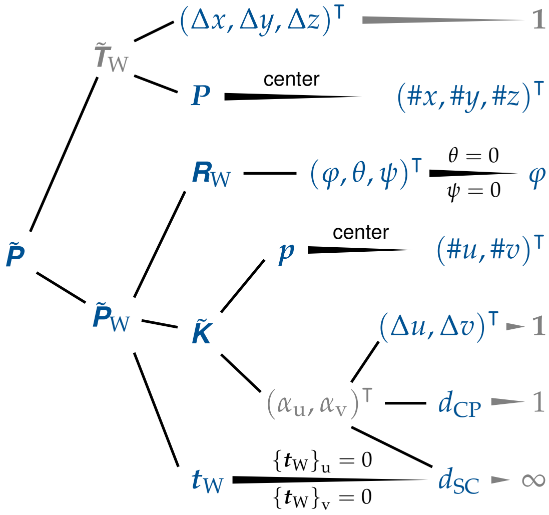
Most detectors have an accuracy of only $12\,\mathrm{Bit}$. The limiting factor is thus the numerical error in the long sums as shown in subsection 5.1.4 on page 48. It suggests float32 to be on the save side for sums over single dimensions of typical tomographic volumes, like in the four core components. Float16 is exactly on the brink of the required accuracy. The value ranges would have to be normalized carefully in order to avoid the inaccurate boarders of the floating-point range. It is however questionable whether the error from the multitude of subsequent operations in SIR would stay small enough to converge and return quantitative results. Moreover, only the GPUs have native float16 support. CPUs require extra type conversion.

Summarizing, we decided for float32 with optional (experimental) support for float16. The few really big sums that go over whole volumes are implemented in float64 with type conversion. De facto, they only play a role for computing the value $\mathcal{L}_{\mathrm{G}}(\boldsymbol{\mu})$ of the cost function or its denominator $\operatorname{denom} \mathcal{L}_{\mathrm{G}}(\boldsymbol{\mu})$. The popular OS-OGM solver introduced in subsection 3.3.4 on page 28 requires neither.

Geometry representation

The choice of the axes in the image volume and projections volume was shown in figure 12 on page 34 in the section explaining the projection matrices for CT. It does not play a role for the performance of any core part, except for the FF requiring u to be the fastest dimension in order to carry out the Fourier transform on the detector rows. For some techniques that will be presented in section 6.3 on page 64, it would be inconvenient if y was the fastest axis. Therefore, the choice was such that in the unrotated case the orientations of the last two dimensions, x, y and u, v, respectively are the same. The central ray from the source to the detector then traverses parallel to z, which is a common notation in x-ray physics.

The whole geometry is organized in a Python class holding hierarchically arranged objects for all the quantities required to compute the projection matrices $\tilde{\boldsymbol{P}}$. Each higher-level quantity can either be set manually, tries to compute its value from lower-level quantities or falls back to a reasonable default value. For example, this structure allows to either specify custom rotation matrices $\boldsymbol{R}$, or the Euler angles $\boldsymbol{\varphi}$ from which the rotation matrices $\mathbf{R}_{\boldsymbol{\varphi}}$ are automatically computed. Similar for the principal point $\boldsymbol{p}$ that can either be given or automatically be computed as the center of the detector. Figure 17 sketches the whole hierarchy. Parallel-beam support is triggered by setting d_{SC} to infinity. The class is created to return the whole set of projection matrices



$(\Delta x, \Delta y, \Delta z)^\mathsf{T}$
1
$\tilde{\boldsymbol{T}}_\mathrm{W}$
$\boldsymbol{P}$
center
$(\#x, \#y, \#z)^\mathsf{T}$
$\boldsymbol{R}_\mathrm{W}$
$(\varphi, \theta, \psi)^\mathsf{T}$
$\theta = 0$
$\psi = 0$
φ
$\tilde{\boldsymbol{P}}$
$\boldsymbol{p}$
center
$(\#u, \#v)^\mathsf{T}$
$\tilde{\boldsymbol{P}}_\mathrm{W}$
$\tilde{\boldsymbol{K}}$
$(\Delta u, \Delta v)^\mathsf{T}$
1
$(\alpha_\mathrm{u}, \alpha_\mathrm{v})^\mathsf{T}$
d_CP
1
$\boldsymbol{t}_\mathrm{W}$
$\{\boldsymbol{t}_\mathrm{W}\}_\mathrm{u} = 0$
$\{\boldsymbol{t}_\mathrm{W}\}_\mathrm{v} = 0$
d_SC
∞

Figure 17 The hierarchy inside the Geometry class. Each quantity (blue) is a property of the class and can either be set manually or computed from lower-level quantities, which are shown in the hierarchy from left to right. All variable names were defined in section 4.2.
The most high-level quantity is the projection matrix $\tilde{\boldsymbol{P}}$. The most low-level ones are the minimum parameters required for parallel-beam geometry and, in gray, corresponding default values. Asymmetric connectors indicate that the quantity at the thicker end is able to hold more information if set directly than if computed from the quantity on the thinner end. The assumptions for the simplification made are indicated above or below. Grayed out quantities, besides the default values, are intermediate results that are not separate properties of the class.

for all views at once. Each parameter can be set per view or once for the whole set.

6.2. The kernels

The kernel is the function defining the computation on a GPU. This section describes the main kernels of the four core components defined in the previous section. All of are deliberately based on simple concepts to reach the maximum performance. As API we decided for OpenCL because it has no general restriction to certain hardware.

6.2.1. The back projector (BP)

The discrete function to be implemented in the BP can be derived from algorithm 2 on page 25 keeping in mind not only its definition in the FBP or FDK but also as the operation $\boldsymbol{A}^\mathsf{T} \cdot \boldsymbol{p}$ in SIR:

$$\mu[x,y,z] = \sum_{\varphi=0}^{\#\varphi-1} w_\mathrm{BP}(\varphi,x,z)\; p[\varphi,u,v] = \left\{ \sum_{j\in[\boldsymbol{p}]} a_{ij}\, p_j \right\}_i$$

where j is the linear index of the vector $\boldsymbol{p}$ pointing to element $p[\varphi,u,v]$ and i the linear index of $\boldsymbol{\mu}$ pointing to $\mu[x,y,z]$. The two definitions suggest two different views on the BP. The first is to collect and sum up all values $p[\varphi,u,v]$ that can be found on the intersection of the ray from the source with the detector going through the image point $(x,y,z)^\mathsf{T}$ and optionally applying the weighs $w_\mathrm{BP}(\varphi,x,z)$. The second view is to carry out a huge matrix product, where the weights a_{ij} relate every element in $\boldsymbol{\mu}$ to every element in $\boldsymbol{p}$. The matrix $\boldsymbol{A}$ has a size of $[\boldsymbol{\mu}] \times [\boldsymbol{p}]$. This would be $500\,\mathrm{EiB}$ of float32 data for the example presented in figure 13 on page 37. Therefore, this definition would only be suited for very small problems, but still be inefficient on a computer, because $\boldsymbol{A}$ is very sparse.

The first definition leaves two options that are depicted in figure 18. First, there is the voxel-driven approach. For each voxel, the corresponding position on each view is computed for example by $\tilde{x} = \tilde{\boldsymbol{P}} \cdot \tilde{X}$. The corresponding values are collected and summed up. Second, there is the ray-driven approach. It starts a ray $\boldsymbol{X}(\xi)$ from each pixel position $\boldsymbol{x}$ in each view and dumps its value at each voxel along the ray path inside the volume. On a GPU, the most natural choice is to assign a thread to each voxel for the first and to each pixel for the second technique. In that case, the great advantage of the first technique is that data races are inherently prevented. Each position in the image volume is uniquely assigned

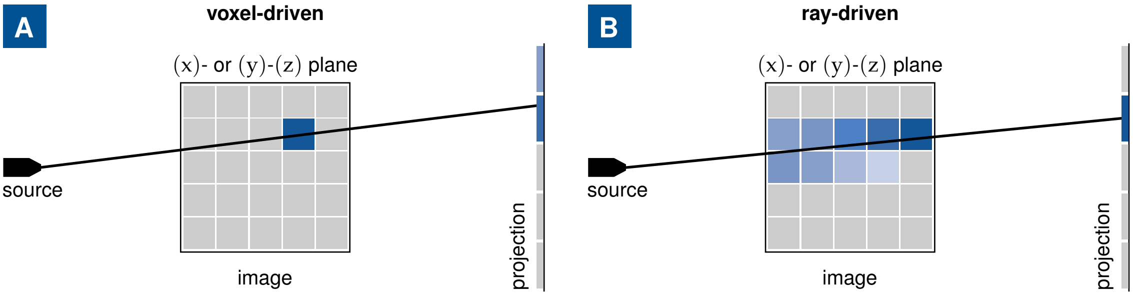


Figure 18 Two basic types of tomographic projectors.
The schematic shows two different implementations of the discrete sum that has to be computed inside a forward or back projector. Starting from a voxel in the image and laying its footprint over the pixels of the projections is called the voxel-driven approach (A). Starting from a pixel on the projection and following its footprint along a ray through the image is called the ray-driven approach (B). There are different models for the shape of the footprints defining the relative contribution of the involved pixels or voxels. The simplest model is linear interpolation. Different shades of blue indicate the corresponding weights in the drawing.

to a single thread. A ray-driven approach would have to guarantee that two threads do not write to the same voxel at the same time. That is the reason why our implementation uses a voxel-driven BP.

One deciding question still remaining is how to move from the integral over a continuous function to a discrete sum over square pixels. Different advanced models were suggested to solve that problem [80, 81, 82, 83, for example]. We decided for the still simplest and fastest solution, namely bilinear interpolation by the texture hardware. Unfortunately, it is known for several limitations [84]. The most prominent one is the limited sampling. For example, reconstructing with a large cone angle or from a fine pixel grid into a coarse voxel volume gives wrong results. The extend of the voxel footprint on the detector is not considered. It can cover several pixels in those two cases. But, for the moderate cone angles common in most lab-based or industrial applications, it still delivers reasonable results and impresses with its performance. If necessary, that part of the kernel can easily be exchanged with a more accurate model.

Summarizing, the BP kernel works on one voxel with position X. It loops over all views computing the position $\tilde{x} = \tilde{\boldsymbol{P}} \cdot \tilde{X}$ and summing up the corresponding interpolated values. The views $p\left[\varphi, u, v\right]$ are held as OpenCL images to take advantage of the intrinsic bilinear hardware interpolation feature, the volume $\mu\left[x, y, z\right]$ is held in global memory, which can be accessed linearly as required and the matrices $\tilde{p}$ are held in constant memory. Support for Feldkamp weights can be added as shown in subsection 4.3.4.

6.2.2. The forward projector (FP)

The forward projector is required in iterative reconstruction or for simulations. It was described by the operation $\boldsymbol{A} \cdot \mu$. As the FP is the transposed operation to the BP, the simplest implementation would take the BP and replace all image writing by reading operations and all projection reading by writing operations. However, this simple approach would result in lots of data races on a parallel computing device, like the GPU. This problem can be circumvented by taking the ray-driven approach for the forward projector where each pixel can uniquely be assigned to a single thread.

Again, we decided for the simplest approach, namely Joseph's method [49]. It applies a bilinear interpolation within each slice of voxels perpendicular to the main direction ν along the ray path. On the one hand, the simplicity of this approach, like before, guarantees the highest performance possible. On the other hand, there is a drawback. The FP is not matched with the BP. That means, that through the different way data is interpolated the weights in the FP and the BP differ implying that the one is not the exact transpose of the other. This can lead to worse convergence for some iterative approaches. But, again, this part of the kernel can easily be replaced with a more accurate model if necessary.

Following algorithm 6 on page 35, the kernel first computes the reference point $\boldsymbol{X}_0$, second the direction vector $\boldsymbol{U}_0$ normalized to its component along the main direction ν, and third, the initial ray parameter ξ_{min}. The required position of the source S and main direction ν are precomputed once for each view and provided together with the projection matrices $\tilde{\boldsymbol{P}}$ and their pseudo-inverse $\tilde{\boldsymbol{P}}^{+}$. Collecting the values then starts at $\boldsymbol{X}_{\mathrm{min}} = \boldsymbol{X}_0 + \xi_{\mathrm{min}}\,\boldsymbol{U}_0$, the intersection of the ray with the first slice of voxels. The kernel continues stepping through all the $\#\nu$ slices by adding $\boldsymbol{U}_0$ to the current ray position in each loop iteration. The total sum is normalized by the intersection length of the ray with a layer of voxels, which is just the length of $\boldsymbol{U}_0$.

An image object is used for the volume $\mu\,[x, y, z]$ in order to take advantage from the fast hardware interpolation and cache. The views $p\,[\varphi, u, v]$ is held in global memory. Aligned access can be guaranteed by a corresponding thread order. The projection matrices and all precomputed quantities are held in constant memory. A compiler macro flag optionally modifies the kernel for parallel-beam support according to subsection 4.2.4 on page 36.

6.2.3. The regularization (REG)

The Gibbs neighborhood regularization and its derivations are required for SIR and were implemented on the GPU and in C for multi-threading on CPUs. They all have in common the sum over the neighborhood of a voxel and a function operating on the gradient of the that voxel to its neighbors. Some additionally require a sum over the whole volume in the end. Both implementations realize that general structure. Compiler macros then allow to choose the actual derivation, a penalty function and the floating-point precision. Shortcuts were implemented for cases that do not involve the current guess, like $\operatorname{diag}\operatorname{Hess}\,\mathrm{R}_{\mathrm{G}}\,(\boldsymbol{\mu})$ for the Huber penalty. For the CPU, all possible combinations are precompiled in a C library.

The penalty functions implemented are the Huber penalty $\Psi^{\gamma}_{\mathrm{H}}\,(t)$ described in algorithm 4 on page 27, a simple quadratic penalty $\Psi^{\gamma}_{\mathrm{Q}}\,(t) = 1/2\,t^2$ and the total-variation [85] penalty $\Psi^{\gamma}_{\mathrm{TV}}\,(t) = 1/2\,|t|$. We implemented all derivations described in appendix A. In addition it is convenient to compute popular combinations of functions as one operation to save unnecessary data transfers.

The investigated neighborhood includes all 26 direct and diagonal neighbors. For voxels at the boarders, the gradients towards outside the volume, are always zero. That equals a clamped edge ensuring that no intensity can leave the image. In contrast, a free edge turned out to hinder convergence in some cases. It can be realized by mirroring the values for neighbors lying outside at the border to the inside. For the GPU, images automatically provide that feature. The CPU implementation uses an array to store the relative positions to the neighbors which is modified at the borders.

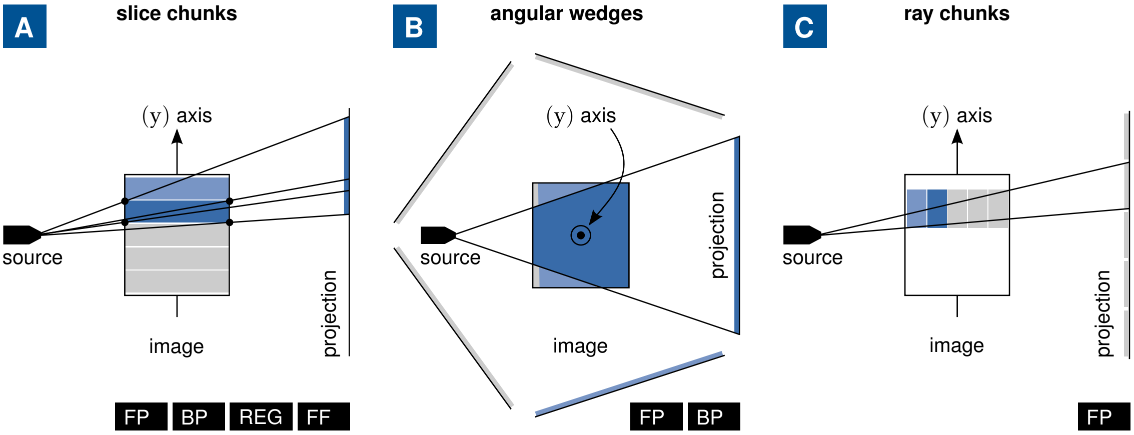


Figure 19 Data splitting for cone-beam CT computations on GPUs.
We applied three data splitting mechanisms to exceed the limited data capacities of GPU devices. The first one, slice-chunk splitting (A), is applied for all four core components. It creates chunks perpendicular to the y axis of the volume, or the v axis of the projection. These chunks are completely independent, but costly for FP and BP. Angular wedges (B) means limiting the number of views for FP and BP that are processed at once. It involves no overhead and gives extra speedup if sorted by the main direction ν. Ray chunks (C) additionally provide a way to process only parts of the image volume in the FP which becomes necessary for large cone angles. The image is split along the main direction of the ray.

Another important feature are manual weights for the three spatial dimensions. They have to be set if the voxels are defined non-uniform and do not have the same spacing in each direction.

The data required can be described as maximal two input arrays, $\mu\,[x,y,z]$ and $d\,[x,y,z]$, two output arrays, for example the gradient together with the curvature and one output scalar like the value or the denominator. On the GPU, the two input arrays are held in images to take advantage of the automatic boarder treatment and the local cache, which is perfectly suited for operating on the local neighborhood. The output arrays can be held in global memory. As one thread operates on one voxel, aligned access can be realized.

6.2.4. The Fourier filter (FF)

The FF as required for the FBP and FDK consists of a Fast Fourier Transform (FFT), the filter and the inverse transform. A very detailed description of the implementation can be found in the master thesis of L. Hehn [51].

We decided to take the open-source *clFFT* implementation [86] for the FFT and its inverse. It provides a very fast implementation that matches well with the rest of our OpenCL framework. As one-dimensional FFTs can only be carried out along an array axis, analytical reconstruction is only possible if the v axis of the detector is quasi parallel to the axis of rotation.

6.3. Handling big data

A major restriction for GPU computing is the limited memory available on those devices and the limited transfer rates from the host memory. Memory is also not shared across several GPUs. The performance of any GPU software implementation that has to handle many gigabyes of data is highly dependent on good ways to split the work with few overhead. The performance and scalability of our implementation is mainly based on the considerations presented in this section. Figure 19 shows the three different approaches we used [75, 87]. The challenge for splitting a problem is to create independent subtasks with ideally no computational or data overlap.

6.3.1. Slice chunks

Slice chunks are probably the most efficient way to split the CT reconstruction problem so that each chunk looks like a smaller CT problem inside the big one. Efficient thereby means that no data point has to be visited twice and the fraction of data that has to be loaded twice can be kept small. It is the only technique of the three presented that provides completely independent tasks. As crosstalk between several GPU devices is very expensive, it is the one that enables to distribute the work to multiple GPU devices.

The exact working principle is to split up the volume on which the thread indices are defined along the y or v axis. For all four core components this is the output volume, namely the projections $p\left[\varphi, u, v\right]$ for the FP, the filtered projections for the FF, the image volume $\mu\left[x, y, z\right]$ for the BP and the volume containing the computed derivation for REG. The chunk size is defined as a multiple of the workgroup size in that direction in order to leave no thread idle. We use the rectangular data copy functions provided by OpenCL to cut the required data from the 3-D arrays. That explains, why neither y, nor v should be the fastest axis, because copying continuous parts is always faster than copying a lot of single values.

Calculating the required in input data for a slice chunk is straight forward for the FF and the REG. The FF works independently on all stripes in the u direction. Therefore, it is possible to divide the volume into any number of chunks without overhead. In principle, those chunks even would not require full slices in v. The REG operates in the local neighborhood of one voxel. A slice chunk therefore requires one voxel overhead above the chunk volume and one below for the input. The fraction of overlapping data is thus small, and gets smaller with bigger slice chunks.

Retrieving the required input for a slice chunk in the FP and BP is more complicated. We suggested an extra single-GPU kernel for that task [75]. For the BP, it takes the positions of the eight corners of the chunk volume, projects them onto each view and finds the minimum and maximum slice in v by a reduction algorithm. This process is illustrated in figure 19A simplified for a single view. The black dots in the image volume mark the corners investigated. The blue area on the detector is the region that corresponds to the blue slice chunk investigated. That way, it takes only about a millisecond to find the corresponding regions for all slice chunks. The FP uses an equally fast analog mechanism. Here, the four corners of the slice chunk region in each view are backprojected into the volume by computing the corresponding ray. The minimum and maximum image slice in y is then determined by computing all entry and exit points of that rays followed by a reduction algorithm.

Figure 19A reveals also the limitations of slice-chunk splitting. First, the region required from the input volume can be quite large, depending on the how far the slice chunk is away from the central ray. Second, one can imagine that these regions can cover up to the whole projections volume if the y axis of the image is strongly rotated against the v axis of the projections or even upside-down. Third, slice chunks cannot be applied at all if y is the main direction. The last problem as well as the upside-down projections can be corrected as shown in subsection 4.3.2 on page 39. Strongly rotated projections remain a problem. However, only very special use cases require such a geometry.

The overlap due to the cone beam can be minimized by keeping the number of slice chunks low and prefer other methods for further splitting. It is also important to keep in mind that due to the divergent cone, the different slice chunks can involve very different amounts of data. These last two issues can be compensated very well by intelligent scheduling as discussed later.

In summary, slice chunks are the best way to obtain entirely independent subtasks, can substantially increase the possible y and v extent, but have to be used modestly for FP and BP, because of their overhead.

6.3.2. Angular wedges

Angular wedges are generated by splitting up the projection data along the φ axis of the according slice-chunk regions. The result are artificial angular subsets containing a preferably continuous set of views. This kind of splitting is the working horse for limiting the amount of data on each single GPU and therefore explains why φ is preferably the slowest axis.

The great advantage of angular wedges is that they produce no copy or compute overhead. If the current image chunk stays in memory, many angular wedges can subsequently be pushed or pulled from the device. The restriction that the image chunk has to remain explains why angular wedges cannot be used to split data over multiple GPUs.

Another side effect is that the angular wedges can be chosen such that one wedge only contains views of equal main direction ν. That prevents additional branching in the kernel resulting in a speedup of more than two.

Concluding, angular wedges are able to make the implementation work for an arbitrary number of views with no additional overhead. They provide a way to limit the data size of the projection views independently from the cone-beam geometry. This is especially helpful for the BP, where the projection slice chunks can be very large.

6.3.3. Ray chunks

Ray chunks were introduced for the FP, because the large image slice chunks occurring for large cone angles remain unaffected by the angular-wedge splitting. Thereby the image volume is split along the main propagation axis ν. Also this techniques comes with no additional overhead and is restricted to single GPUs. The ray chunks can dynamically be pushed to the GPU while the current angular wedge remains in the GPU memory. As ray chunks are a division in ν, they only work with sorted angular wedges that do not contain views of different main propagation directions.

Summarizing, ray chunks enable arbitrary extend in x and z for the FP, i. e. the two main propagation directions allowed. Analog to the angular wedges, they are independent from the cone-beam geometry and allow to limit the large data size of the image slice chunks in the FP.

Concluding for all three splitting techniques, the data sizes possible in the FP theoretically remains only limited in y for very large cone angles and in u. The BP theoretically remains limited in v for very large cone angles and in x, z and u. FF has only a theoretical limit in x and REG additionally in z. De facto, the limit for the FF is given by the maximum size allowed in the clFFT, which is probably huge. For the other three, they are defined by the maximum possible size of 3-D image objects, which are $4096\,\mathrm{vx}$ in each direction in current hardware.

6.3.4. Automatic chunk scheduling

The dimensions for CT datasets can be very different in size and ratio. Sizes typically reach from 128 voxels in x, y and z for small simulations up to three or four thousand in high-resolution measurements. Often $\#y$ is reduced to only a few slices in order to quickly sweep reconstruction parameters, or z is adapted to the actual sample thickness resulting in unequal aspect ratios. The dimensions along u and v are mostly in the same range as the corresponding dimensions in the image. But, the number of views can vary a lot, reaching from a couple of dozens for SIR reconstructions with subsets to many thousands for systems with a continuously rotating gantry or helical trajectory. All of these options would require different settings for splitting up the data sets into the different types of chunks in FP and BP.

In order to reach a good performance without the need for finding the optimum parameters in each case, we introduce an automatic scheduler. It reads out the total amount of global and constant memory so that the chunk sizes are guaranteed to stay within the given limits.

Before discussing the actual splitting it is important to explain how the GPU memory shall be used in general. The different chunks allow the GPU to dynamically push and pull data while computing. That means that while the GPU computes on the data of one chunk, the results of the previous can be fetched and the data for the next one pushed. Therefore, the data of two full chunks have to fit onto the device at once. Those to spaces are used alternatingly for computing and exchanging data. The biggest components per chunk are the corresponding part of the image and the projections. That makes four big arrays in total. Most current GPU devices have a maximum allocation size for arrays or images in global memory that is one quarter of the total space. That matches exactly the two arrays holding the image and two holding the projection data.

Slice chunks are the only ones that allow completely distinct tasks for multi-GPU computing but also the only ones producing overhead in some cases. There can be bigger and smaller chunks because the overhead depends on how far away a chunk is from the central ray. The scheduler takes care for these facts by splitting up the output volume into a number of equal chunks. It works differently for FP and BP.

In the BP, the size of a slice chunk is always a multiple of the memory footprint of a workgroup in y, except for the last chunk. The number of slice chunks is chosen such that the image chunks do not exceed the maximum allocation size and that there is at least one per device. Limiting the projection chunks to the maximum allowed space is left to the angular-wedge splitting.

In the FP, the number of slice chunks is at maximum twice the number of GPU devices. The devices are then assigned alternatingly a chunk from the top and the center of the volume. After a device finished its task it is assigned a new chunk in the same way. That procedure ensures first, that most devices have one chunk with bigger and one with smaller overhead and second, that chunks with bigger run in parallel to chunks with smaller overhead. This second effect is very important to avoid peaks at the bus of the main memory.

The number of angular wedges is first estimated such that the according projection data does not exceed the maximum allocation size in global memory and the data required for the geometry does not exceed constant memory. Second and only for the FP, it is ensured that there are enough angular wedges so that each wedge can exclusively be filled with views of the same main direction. Third, the views are distributed equally and as subsequently as possible among the available wedges. The last step assumes that the views are stored in an angular subsequent order. If that is not the case, the GPU-internal automatic image cache works significantly less efficient.

The number of ray chunks in the FP is chosen according the maximum memory available.

6.4. Remarks on the OpenCL framework

This section contains some important technical remarks on how we used the OpenCL framework that is required for our GPU components.

6.4.1. Providing the kernel sources at runtime

As GPU kernels are compiled to binary programs on runtime, the kernel sources have to be stored in a way the program can access it. Two possible options are separate OpenCL source files at a fixed location or hard-coded string literals inside the C++ library. On the one hand, it is inconvenient to store hard-coded absolute paths to source code files in a shared library, because their final location must already be known at compile time and cannot be altered afterwards.

On the other hand, storing the source code as hard-coded string literals is hard to maintain by hand. Therefore, a small program was written that finds all OpenCL source files at compile time and hard codes them as string literals into a C++ header file that is included into the library. Furthermore a common header file is automatically prepended containing all the common macros and definitions. CMake, the build tool we use, even allows to fully automate this process by defining a so-called *generator*. This can be any program autogenerating code for the actual project.

6.4.2. The environment class

The OpenCL framework defines several structures that contain information about the computing environment and programs involved. In our implementation, each GPU device is assigned to a separate compute context containing each two queues. The first is for data transfers and the second for kernel executions so that those operations can run in parallel.

All these properties are pooled in a C++ class, called the environment class. An instance of that class is stored in a static variable inside the library. That makes it possible to reuse the same instance for all GPU operations within a program. It is initialized automatically the first time it is required and has to be released manually after the last GPU operation.

If the library is used in a Python context, this task can be carried out by the delete handler of a Python class instance that is generated at the time, the library is imported. At the latest when the Python shell is closed, also the GPU environment is released.

The compiled OpenCL kernel binaries for all GPU devices are also stored inside that environment class. If a certain kernel binary is required, a class method checks if there is already a compiled version available for the set of compiler macros given. That mechanism prevents the code from recompiling the same binaries for recurring operations as required for iterative reconstruction.

6.4.3. Profiling

Profiling is very important for code optimization. There is a great tool for the CUDA technology of Nvidia, namely the Nvidia visual profiler that allows to see the runtimes of all operations in a time line, displays the actual data throughput and identifies the bottlenecks of the program. Unfortunately, they stopped support for OpenCL some years ago.

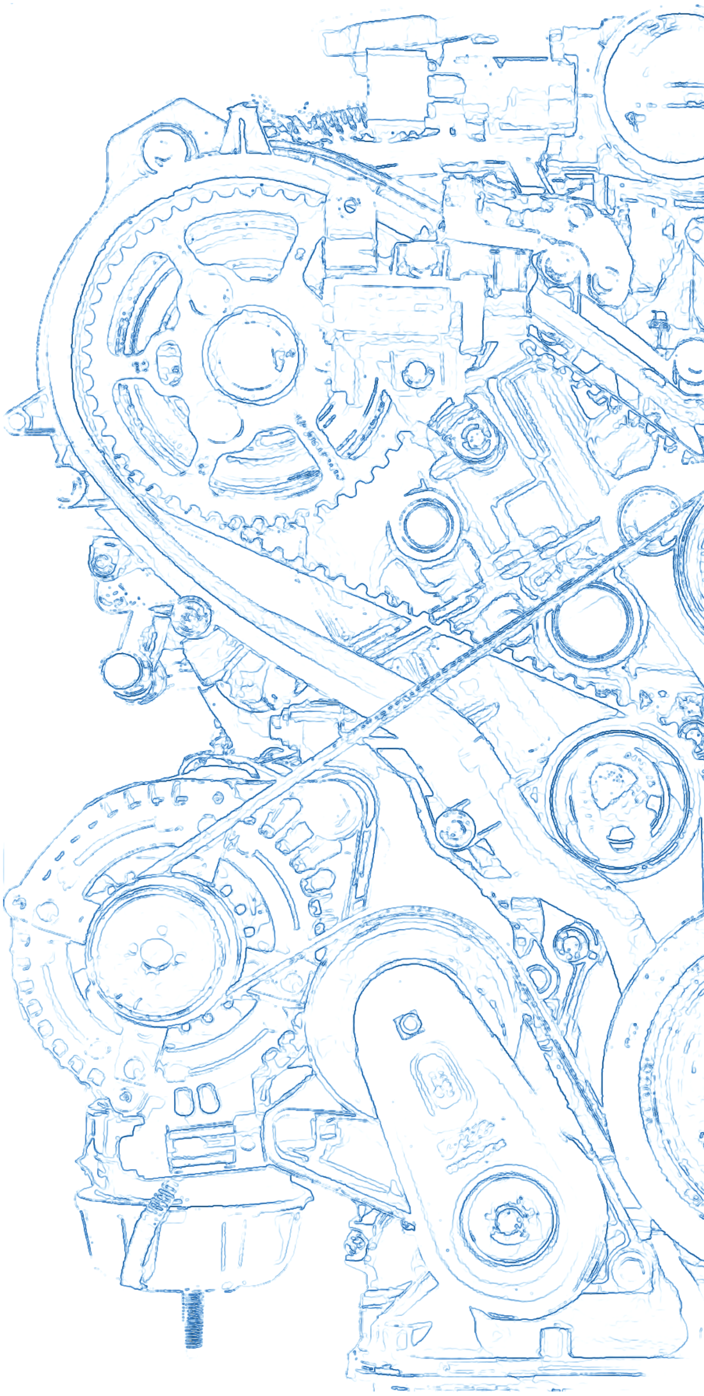
Another option is to set the OpenCL-intrinsic profiling flag allowing to record the timings of all the operations in a queue and write them into a file. A Python program is used to visualize the results. All GPU profilings shown in this thesis were generated this way.

Closure

The main building blocks we identified from the mathematical equotions for analytical and iterative CT reconstruction, namely FP, BP, FF, and REG are by far the four most compute-intensive operations. Fortunately, they occur excactly in the same form in many of the required functions. The GPU turned out to be the most suited hardware for all operations. Only for REG, also multithreding on CPUs might be a good alternative. Float32 was identified as the most suited datatype, apart from the seldom sums over the whole volume. We introduced the kernels and three ways for splitting up the data so that it fits on the limited memory recources of GPU devices and multi-GPU computing is possible. The automatic scheduler allows to handle virtually all possible data sizes and ratios at a good performance. At the end, we showed an efficient workflow for the OpenCL framework.

7. The reconstruction framework

Our collaborations with leading manufacturers of industrial and dental CT devices revealed two main obstacles for a broad application of iterative reconstruction. The first is usability, the second reconstruction time. In this chapter, we introduce a reconstruction framework to investigate these two problems. The FDK is considered to be the most powerful analytical and SIR the currently most powerful and general iterative approach. Within our scientific environment, active algorithm development and application on a daily routine play equally important roles but sometimes impose requirements that are hard to reconcile. The first section introduces the aims and visions of the framework and our concepts how to meet the given needs. The second section gives a short overview of the already implemented methods.



Picture: Engine. 🅭

7.1. Visions and their realization

The main vision for the framework started in 2010 was to make *SIR as easy as FDK* but with all its superior properties. The broad spectrum of x-ray CT applications investigated at the chair allowed a twofold development concept. On the one hand, implementing and developing many different algorithms in a small group of people concentrating mainly on software, on the other hand, constant and immediate testing and feedback of many different application experts.

The main challenges to realize this two-fold concept were the different requirements of both groups. This section describes the ways we tried to keep the framework both, constantly evolving but easy-to-use and versatile but fast.

A second vision, *plug and play*, was the goal that all developments are compatible with the rest of the framework if that makes physically sense.

7.1.1. High- and low-level APIs

Chapter 6 showed that a good strategy to combine high performance with ease of use and rapid development is to use high-level together with low-level programming languages. All performance-critical parts required for CT reconstruction could be identified as a few core components and were implemented in C/C++ and OpenCL.

The remaining parts of the code, for example the definitions of the cost functions and solvers as well as the user API, are written in a high-level language. Definitions do not depend on computing performance but can change rapidly. Therefore, a more abstract, high-level language is suited well. We decided to use Python, a modern, widespread and multi-platform, optionally interactive scripting language that is well-suited for object-oriented programming and provides a variety of predefined modules for data processing, analysis and visualization. Therefore, the tomographic reconstruction can be well integrated into other data processing.

Similar to different languages, a high- and a low-level API can be combined to provide good usability for common applications and allow sophisticated fine-tuning for developers or experienced users. To meet the low-level requirements, the Python classes and functions are written as general as possible, but equipped with default values in order to obtain good usability for less experienced users.

These default parameters, for example the choice of the solver for SIR, can vary for different problems and also change if new methods were implemented. The greatest advantage of that dual concept is that any user can obtain a reasonable reconstruction for most applications without knowing anything about the underlying technique. Only very little understanding is enough to reach further improvements. Another advantage is that automatically the most state-of-the-art techniques are in use. If old results have to be reproduced and the default options have changed in between, the old default parameters can be looked up in the version history.

7.1.2. Towards automatic parameter finding

One of the most crucial but challenging parameters to find is the strength λ of the regularization in SIR. For the cost function defined in algorithm 3 on page 26 of the theory part, it can vary over many orders of magnitude for data sets of different size, value range or sample structure. A weak regularization results in a noisy result, a strong regularization in smeared edges, lost structures or patchy clusters.

On the one hand, there can be no absolute measure defining image quality, because the required properties depend on the application and sometimes on the preferences of human observers. [88] On the other hand, the range of reasonable parameters is definitely limited. Also, choosing a certain set of parameters should result in a similar image impression for different data sets.

One way to meet these requirements for λ is a reasonable normalization of the cost function. We suggest the following modification:

$$\mathcal{L}_{\mathrm{G}}^{(\mathrm{FDK})}(\boldsymbol{\mu}) = \frac{D(\boldsymbol{\mu})}{D(\boldsymbol{\mu}_{\mathrm{FDK}})} + \lambda^{(\mathrm{FDK})} \frac{R(\boldsymbol{\mu})}{R(\boldsymbol{\mu}_{\mathrm{FDK}})}$$

where $D(\,\boldsymbol{.}\,)$ denotes the data term and $\boldsymbol{\mu}_{\mathrm{FDK}} = \mathrm{FDK}(\boldsymbol{p})$. This way, the absolute value of the cost function can be interpreted as relative improvement compared to the result of the FDK. It automatically becomes independent of the data size, the value range and, to some degree, also the structure of the measured object. The same properties translate likewise to the parameter $\lambda^{(\mathrm{FDK})}$. Reasonable values are usually found in the small range between 0.1 and 1.

Drawbacks are that edge-preserving regularization terms like the Huber penalty are very sensitive to the streaking artifacts that often occur in FDK reconstructions. Another drawback is that computing μ_{FDK} is a rather expensive operation, but at least a lot less expensive than finding the correct λ over many orders of magnitude. Both problems can be improved by using a very smooth FDK filter kernel, which is usually also a very good initial guess $\mu^{(0)}$ and thus can be used for both purposes. Unfortunately, the assumption that $\lambda^{(\mathrm{FDK})}$ is independent from the data size and sample features holds only roughly. Often parameters are optimized on a subvolume because of performance reasons and then transferred to the whole data set. This approach works well together with the suggested normalization, but for some use cases the parameter of the full and the reduced data set do not match exactly.

Another parameter that has to be guessed is the transition parameter γ for the Huber regularization. It describes the transition from the noise level to real features. An easy way to guess it is by estimating the noise level σ_{FDK} of of a (subvolume) FDK, e. g. from the histogram. As all Gibbs neighborhood priors work on differences between voxels, the according parameter results in $\gamma^{(\mathrm{FDK})} = \sqrt{2}\,\sigma_{\mathrm{FDK}}$.

Alternative, more advanced methods estimate the noise level of the intensity views and propagate it into the image domain to aim for uniform noise or resolution [89, 90], apply a control loop to reach a certain noise level [91] or optimize regularization for a certain task [92, 93]. These techniques are not yet implemented but would definitely be a great benefit for the framework. The chosen method impresses mainly by its simplicity and generality.

7.1.3. Modular structure

An important concept to keep code well-structured and easy to maintain despite of rapid development is a modular structure. It is the realization of our second vision, *plug and play*. That means that single parts can easily be exchanged without touching the rest of the framework. This way, it is possible to improve or replace individual modules independently and regard the rest of the framework as black box. Furthermore, anyone can take advantage of any development by others and all developments can be combined.

The FDK is just a straight-forward single Python function that generates the filter kernel and executes the multi-GPU FF and BP core components. SIR , in contrast, has a lot of different options and new methods develop rapidly. Therefore, we identified several basic modules that can be implemented independently and that cover the various choices:

- the data term,
- the regularization, and
- the solver.

Each module is realized as a set of exchangeable classes providing several methods. Data term and regularization classes can compute their contribution to the cost function for the current guess and all its different derivations required for various solvers. The solver classes contain the minimizer and a common method for displaying the current status and plotting intermediate results. A possible future extension could be a separate module for the different types of normalizations.

Furthermore, there are two static components, namely the cost function, combining and scheduling an arbitrary number of data and regularization terms, and the reconstructor class that brings together all the class instances involved with the various reconstruction parameters. Furthermore, it provides methods for common functions like forward and back projection or profiling. There are several front ends possible that communicate with the reconstructor. Currently there is a Python function similar to the FDK function.

7.2. An overview

One of the greatest advantages of SIR is its potential to model many aspects of the physical environment inside the reconstruction. There are several applications of this feature that can be implemented into the data or regularization term.

7.2.1. Data terms

Starting from the very basic Gaussian or Poisson noise model introduced in section 3.3, there are many sophisticated data terms for various problems. The first common application is task-based reconstruction. That means that tomographic reconstruction is combined with pre- or post-processing steps that are required to obtain the final quantity. Especially data terms that incorporate more than one signal can add a lot of extra value.

A very simple approach to reconstruct the contributions of two single materials from a scan at two distinct energies is by using the data term

$$D_{2\,\mathrm{mat.}}\left(\{\boldsymbol{\rho}\}_m\right) = \frac{1}{2}\sum_k \left\| \boldsymbol{A}\cdot\left(\sum_m \mu_{k,m}\,\boldsymbol{\rho}_m\right) - \boldsymbol{p}_k \right\|_{w_k}^2$$

where k is the index over the two energies, m the index over the two materials and $\mu_{k,m}$ the absorption coefficient of material m at energy k.

A very successful multi-signal approach is the combination of tomographic reconstruction with the retrieval of the attenuation, phase and darkfield signal for grating-based phase-constrast aquisitions. The forward model for this direct intensity-based approach

$$\bar{\boldsymbol{I}}_{\mathrm{IB}}\left(\boldsymbol{\mu},\boldsymbol{\epsilon},\boldsymbol{\delta}\right) = \boldsymbol{I}_0\,\mathrm{e}^{\boldsymbol{A}\cdot\boldsymbol{\mu}}\left(\boldsymbol{1}+\boldsymbol{V}_0\,\mathrm{e}^{\boldsymbol{A}\cdot\boldsymbol{\epsilon}}\cos\left(\boldsymbol{\Phi}_0+\boldsymbol{A}\cdot\boldsymbol{\delta}\right)\right)$$

can for example be used in a Poisson data model. It takes into account the three reconstructions for the attenuation $\boldsymbol{\mu}$, the darkfield signal $\boldsymbol{\epsilon}$ and the phase $\boldsymbol{\delta}$. $\boldsymbol{V}_0$ is the local visibility map and $\mathbf{\Phi}_0$ the local phase offset. The approach allows to omit the technically challenging and time-consuming phase stepping measurement and to directly consider the statistical properties of the measurement. [94, 95]

The second common application of advanced data models is artifact reduction. If the physical reason behind an artifact is understood, it can be included into the model. All imperfections that do not impose a general limitation on tomographic reconstruction, can thus be eliminated.

One very useful application modeling the polychromatic spectrum of laboratory setups to eliminate beamhardening artifacts. This can be done by using the forward model

$$\bar{\boldsymbol{I}}_{\text{polychrom.}}(\boldsymbol{\mu}) = \boldsymbol{I}_0 \sum_k b_k \exp\left(-\Phi_k \boldsymbol{A} \cdot \phi(\boldsymbol{\mu}) - \Theta_k \boldsymbol{A} \cdot \theta(\boldsymbol{\mu})\right)$$

which can also be used in a Poison data model. It reconstructs a virtually monochromatic result $\boldsymbol{\mu}$ using the parameters Φ_k and Θ_k to model the relative influence of the photo and the Compton effect at the given energy k. The functions ϕ and θ approximate the fraction of photoelectric effect and Compton scatter in the guess μ. The coefficients b_k contain the fraction of the measured spectrum that lies within energy bin k. A great advantage of this model compared to other polychromatic approaches is that the expensive operation $\boldsymbol{A}$ has to be evaluated only twice for the whole forward model. [96, 97]

7.2.2. Regularization terms

The prime task of the regularization term is to favour realistic noise realizations. Therefore we implemented several smoothness constraints including the Huber regularization, introduced in 4 on page 27. Other penalties for the Gibbs neighborhood prior are for example the quadratic penalty $\Psi_{\text{Q}}(t) = {}^1\!/\!_2\, t^2$ and the total variation penalty $\Psi_{\text{TV}}(t) = |t|$. Optionally, also the gradient between neighboring voxels which is fed into the penalty function, can be replaced by the Laplacian [98].

But, the regularization also allows to add priror knowledge. We introduce the *fixed* regularization

$$\text{R}_{\text{F}}(\boldsymbol{\mu}) = \frac{1}{2}\, \|\boldsymbol{\mu} - \boldsymbol{\mu}_{\text{F}}\|^2_{\boldsymbol{W}_{\text{F}}}$$

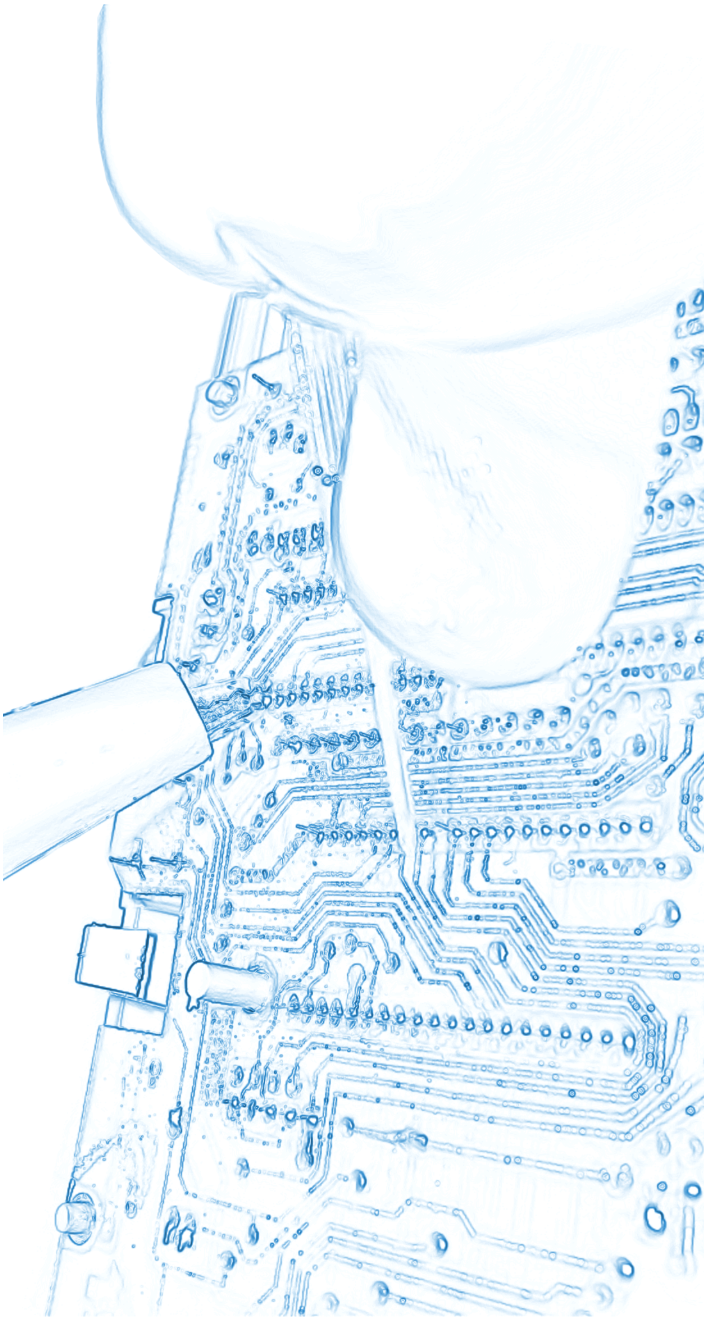
for that task. It allows to enforce a fixed guideline $\boldsymbol{\mu}_{\text{F}}$ within Gaussian noise deviations. The local weights $\boldsymbol{W}_{\text{F}}$ allow to restrict the operation to a certain region in the image. This way, also shape support can be given. Modifications of that regularization compromise a variable guideline $\boldsymbol{\mu}_{\text{F}} := \boldsymbol{F}(\boldsymbol{\mu})$ where $\boldsymbol{F}$ can be any function, in particualar any image filter. This technique was for example used to try dictionary denoising as regularization [99]. Unfortunately, the cost function changes each time $\boldsymbol{F}$ is evaluated, which is a problem for some advanced solvers. Another possible choice is $\boldsymbol{\mu}_{\text{F}} := \boldsymbol{W}_{\text{F}} \cdot \boldsymbol{\mu} / \|\boldsymbol{W}_{\text{F}}\|_1$ where $\|\,\boldsymbol{\cdot}\,\|_1$ denotes the sum over all weights. That allows to favor a uniform region where the weights are positive.

Closure

Designing a reconstruction framework comes with several challenges if employed in an environment where both, algorithm development for a broad variety of use cases and routine application of those algorithms are important. First, we introduced different levels of complexity, not only in the languages used, but also in the APIs in order to keep the framework versatile, but still easy to use. Second, we tried to find automatic settings for all parameters that give reasonable results for most applications. Of course, manually tuned settings are still best. Third, the modular structure not only keeps the code well organized, but allows to combine many independent techniques and makes it easily to add new features. The modules implemented by various people trying to solve very specific and different problems and the broad use of the software at the various laboratory setups within the chair show that these three concepts turned out to be good choices.

8. Integral optimization

After discussing ways to improve the usability in the previous chapter, this chapter discusses the second main obstacle for a broader application of SIR: its performance. We found that major advances in performance can only be reached by an integral optimization of the whole process. In our optimization we aim for a certain task, namely iterative reconstruction for absorption micro-CT, which was not done before in reasonable time because of the vast amount of data. We start with choosing the best-suiting modules and identifying the basic components from our choice. The first set of optimizations deals with the synergy of those components, the second with the optimization of the components itself and the third with optimizing the solver.



Picture: Detailwork on a circuit board. ©

8.1. The right components and their synergy

The aim of our optimization is to make the great advantages of iterative reconstruction available for high-resolution CT. Current micro- or nano-CT devices are often equipped $(2\,\mathrm{k})^2$ detectors. A tomographic data set requries 3217 views according to the Nyquist criterion as discussed in subsection 3.2.2 on page 22. The corresponding tomogram has a size of $(2\,\mathrm{k})^3$ voxels. This is $[p] = 50\,\mathrm{GiB}$ of measured and $[\mu] = 32\,\mathrm{GiB}$ of reconstruction data in float32 precision. The system matrix $\boldsymbol{A}$ in SIR would consume $[p] \times [\mu] = 402\,\mathrm{EiB}$ if computed at once. These large sizes make iterative reconstruction very challenging. Our approach is therefore taking the fastest components still reasonable.

8.1.1. Cost function and solver

We decided for a very simple cost function and fast solver in order to keep the computational effort as low as possible. The simplest cost function from the ones presented in the previous chapter is the one with the Gaussian noise model (algorithm 3 on page 26) and a regularizer that operates in the local neighborhood. Assuming the measurements are not in the ultra-low-dose regime, the Gaussian noise model is well-suited [97, pp. 61-66]. As regularization, the Huber regularization (algorithm 4 on page 27) turned out to be a convenient choice because of its edge-preserving properties.

The solver chosen is the OS-OGM, the fastest-converging choice currently available. A simple implementation is shown in algorithm 9.

Algorithm 9 The OS-OGM 2 [45, p. 101] implemented in Python.

```python
def _update_t(t, is_last_it):
    f = 8. if is_last_it else 4.
    return (1. + (1. + f * t**2)**.5) * .5


def os_ogm2(mu_, n_it, n_sub):
    # initialization
    t     = 1.
    cmom_ = mu_.copy()
    curv_ = curv()

    for i in range(n_it):
        m      = i % n_sub
        t_old  = t
        t      = _update_t(t, i == n_it - 1)

        # momentum update
        q_     = grad_sub(mu_, m) / curv_
        cmom_ -= 2. * t_old * q_

        # image update
        mu_ = (1. - 1. / t) * (mu_ - q_) +
              (1. / t) * cmom_

    return mu_
```

Where `mu_` holds the image μ, `n_it` the number of iterations, `n_sub` the number of subsets M, `cmom_` the cumulative momentum z, `grad_sub` the function $\operatorname{grad} \mathcal{L}_{G,m}(\mu) = M\,\boldsymbol{A}_m^{\mathsf{T}} \cdot (\boldsymbol{A}_m \cdot \mu - p_m)|_{w_m} + \lambda \operatorname{grad} R(\mu)$ operating on subset m and `curv` the function $\operatorname{curv} \mathcal{L}_{G}$ for a precomputable curvature. Tailing underscores indicate arrays.

8.1.2. The Solver-specific basic operations

A closer look at the definition of OS-OGM allows to identify some more basic operations.

Only once:

- Ⓐ a FP and
- Ⓑ a BP for the precomputed curvature,
- Ⓒ (optionally) computing the line integrals (FF),
- Ⓓ (optionally) a FF, and
- Ⓔ (optionally) a BP for an FDK initial guess.

For each iteration:

- ⓐ the REG gradient,
- ⓑ choosing and copying the views that belong to the current subset,
- ⓒ a FP and

ⓓ a BP for computing the gradient of the data term, and

ⓔ updating the reconstruction volume and the cumulative momentum.

At the beginning, the precomputable curvature is generated (operations Ⓐ and Ⓑ). According to the derivations of the costfunction in appendix A, it depends only on the scan geometry and array dimensions, but not on the actual guess $\boldsymbol{\mu}$. It can be computed during data acquisition or loading. The curvature for the Huber regularizer is constant and therefore simply added to the curvature of the data term, which is not an extra step in the operations defined above.

For absorption tomography, the measured intensities $\boldsymbol{I}$ have to be converted to the line integrals $\boldsymbol{p}$ by applying the negative logarithm. This step is carried out with an additional feature of the FF implementation on multi-GPUs (operation Ⓒ).

A blurry FDK turned out to be a good initial guess (operations Ⓓ and Ⓔ). The blurriness can be generated with a modified Fourier filter in FF.

Choosing the views $\boldsymbol{p}_m$ for the current subset m in bit-reversal order (operation ⓑ) involves a copy process, because the data has to be stored in a contiguous C-order for the FP and BP.

The difference of the forward model (operation ⓒ) to the measured data $\boldsymbol{p}_m$ in the (subset) gradient function $\operatorname{grad} \mathcal{L}_{G,m}(\boldsymbol{\mu})$ can be computed in-place within the BP (operation ⓓ), if the weights w_m are $\mathbf{1}$ for simplicity. In-place functions are important to save memory.

The update of the current guess $\boldsymbol{\mu}$ and the cumulative momentum z (operation ⓔ) can also be carried out in-place. Updating the scalar variable t is not considered as an extra operation.

8.1.3. Handling the small operations

The two smaller operations, namely copying views into the right order (operation ⓑ) and the update (operation ⓔ) are expected to be limited by the memory transfer independent of the hardware. The small computational work requires at most single add or multiply operations. We preferred CPUs over GPUs for those tasks.

First, GPU computing always involves some overhead for compiling kernels and scheduling. Second, the theoretical data throughput to the four GPU devices is still slower than the one of the DDR4 RAM bus. The fastest transfers possible are over *Direct memory access* (DMA), which is a technique allowing to transfer data directly from RAM over PCIe to the four GPU devices. Using PCIe 3.0 16x, the resulting total theoretical throughput is $62\,\mathrm{GiB\,s^{-1}}$ for all four devices. This transfer rate is still slower than the theoretical throughput of the DDR4 RAM bus to the CPUs which is $68\,\mathrm{GiB\,s^{-1}}$ over the four available lanes.

The implementation was carried out with a multi-threaded for loop using OpenMP. According to the *gcc* C compiler, the loop could additionally be fully vectorized with SIMD operations (compiler flag *-fopt-info*).

8.1.4. REG on multi-CPU or -GPU

The last remaining choice is the hardware for REG (operation ⓐ), which we had to leave open to the actual application in chapter 6 about the building blocks.

As the gradient of the data term and the regularization can be computed independently from each other, the first idea is to use the idle CPUs for the regularization while the data term is handled on the GPUs. We showed that it is even possible to synchronize both such that they take equally long to exploit the

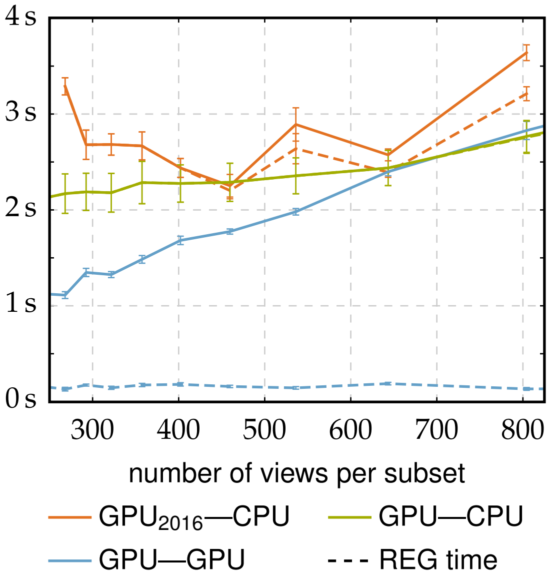


Figure 20 Mean runtime for the gradient on heterogeneous hardware.
The figure shows the mean total runtime and the standard deviation for computing the gradient for several iterations of SIR over different numbers of views per subset. We compare three different implementations represented by the three colors. The time required for the regularization is indicated by a dashed line in the corresponding color. The data labels indicate the hardware used for computing the contribution of the data term (FP and BP) and the regularization (REG) to the total gradient. If REG runs on CPUs, it is executed parallel to FP and BP which always run on GPUs. We compare an older, more data-hungry implementation of the projectors [87] (orange) with the recent one (green) and a serial all-GPU solution (blue). Details about the comparison can be found in the main text. The high performance of the GPU REG implementation makes the all-GPU solution the fastest one.

hardware best [87]. However, we saw some strange behavior that is depicted in figure 20. The runtime of the regularization became dependent on the number of views per subset (orange line). While the runtime of the data term is expected to show roughly a linear dependence of the number of views, the regularization term stays exactly the same operation with the same computational effort. The dependence on the number of views shows a mutual dependence of both processes which was assumed to be caused by the common memory bus. This assumption was also supported by the GPU profilings of the projectors. They showed the usually high transfer rates alternatingly only for one GPU at a time. In conclusion, the memory throughput was not high enough to supply both processes.

An upgrade of the projectors was able to decrease the high demand of data by decreasing the number of slice chunks to a minimum as described in section 6.3 above. The result is a lower dependence of REG on the number of views but the big variations in the total runtime and the slight slant of the REG curve (green line) still indicate some mutual blocking. Furthermore, the total time is dominated by the regularization term for less than about 700 views per subset, whereas it is expected to be by far the less demanding operation. In summary, the idea to run the two parts of the gradient in parallel on GPUs and CPUs did not work reasonably. In contrast, the mutually blocking memory transfers prevented the speedup usually gained by more subsets in the orderd-subset solvers.

The sequential all-GPU implementation, in contrast, shows a significant decrease in runtime for less views. Comparing the green (dashed) line and the blue dashed line shows the difference in performance of REG on multi-CPUs and GPUs, which is obviously more beneficial than using both architectures [100].

8.2. Component-wise optimization

A great benefit a sequential all-GPU solution is that improving one component directly influences the whole iterative reconstruction. Especially for FP and BP choosing the GPU-intrinsic kernel parameters make a great difference. Those critical parameters are the kernel dimensions and the walk parameter k. Many effects depend on the choice of those parameters, especially how good the automatic image cache works, coalesced memory access and concurrent data exchange via PCIe. As REG and FF are both already transfer limited, speeding up the kernel runtime would not be beneficial. Therefore, we restrict ourselves to FP and BP only.

A common measure for the performance of tomographic projectors are GUPS (gigaupdates per second) [101]. It is defined as

$$\mathrm{GUPS} = \frac{\#x\,\#y\,\#z\,\#\varphi}{t}\,10^{-9}$$

including the volume measures, the number of views and the execution time t. We always evaluate the GPUS per device and thus multiply the runtime with the number of devices.

The hardware used for the tests was a single compute server holding four *NVIDIA GeForce GTX Titan X* GPU devices. Each provides 3072 threads with a clock frequency of $1.0\,\mathrm{GHz}$ and holds $12\,\mathrm{GiB}$ of graphics memory. Each test run consisted of one warmup run and five subsequent runs with some idle time in between to cool down the device. The performance displayed is always the best of those six in order to rule out single performance drops.

8.2.1. The workgroup shape

The smallest organization unit of threads is called a workgroup as explained in section 5.2 on page 49. We chose a workgroup size of 512, which is usually the greatest divider of the total number of threads available on different devices. The indices assigned to the threads can have up to three dimensions which goes well with the indices of the three-dimensional volumes in tomographic reconstruction. Therefore, also the total workgroup size can be distributed over those three dimensions. The kernel shape has a great effect on the memory access pattern and can result in good or bad caching.

First, it can make a great difference which of the three kernel dimensions is assigned to which dimension in memory. Second, if the best of the six possible configurations is found, the actual dimensions have to be found. Unfortunately, there are much more possibilities. To prevent unnecessary if-branches in the kernel, the dimensions of the output volumes on the GPU are extended to be a multiple of the kernel dimensions. The third optimization is the number of consecutive voxels in the y-direction that is handled per thread. We called it the kernel *walk parameter* k. It was first introduced for the *Nvidia Kepler* GPU architecture [102].

The fastest order of work group dimensions turned out to be $\#U$, $\#V$, $\#A$ running along the dimensions u, v, φ for the FP and $\#Y$, $\#Z$, $\#X$ running along the dimensions y, z, x for the BP. Figure 21 shows the performance for all possible shape settings with the restriction that the total size is 512. The triangular subplots show the plane

$$\begin{aligned} \#U\,\#V\,\#A &= 512 \quad |\log_2 \\ \log_2(\#U) + \log_2(\#V) + \log_2(\#A) &= 9 \end{aligned}$$

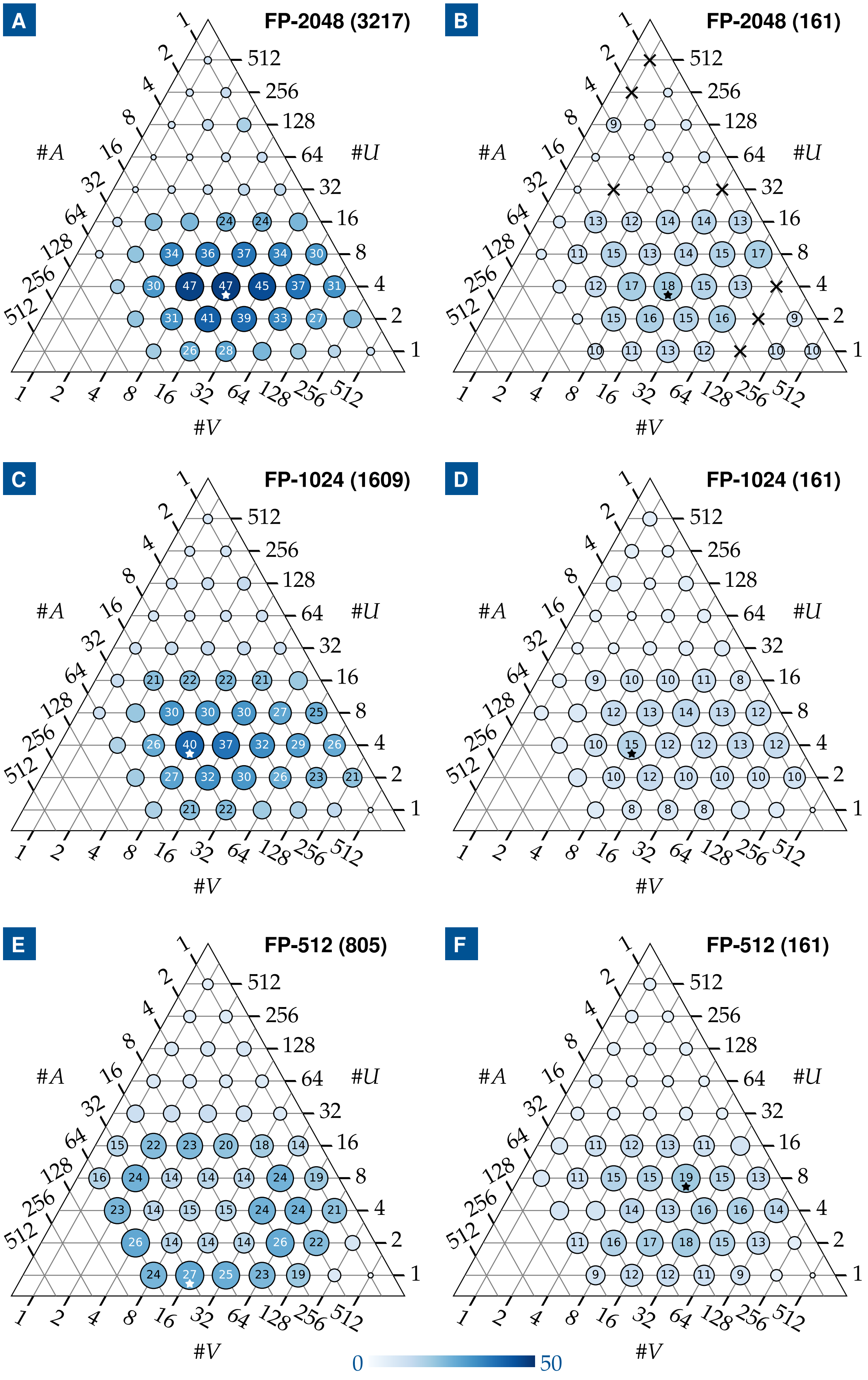
for the FP and analog for the BP.

Fortunately, the best settings (marked with the little star) are mostly very close for different data sizes. The best common settings are

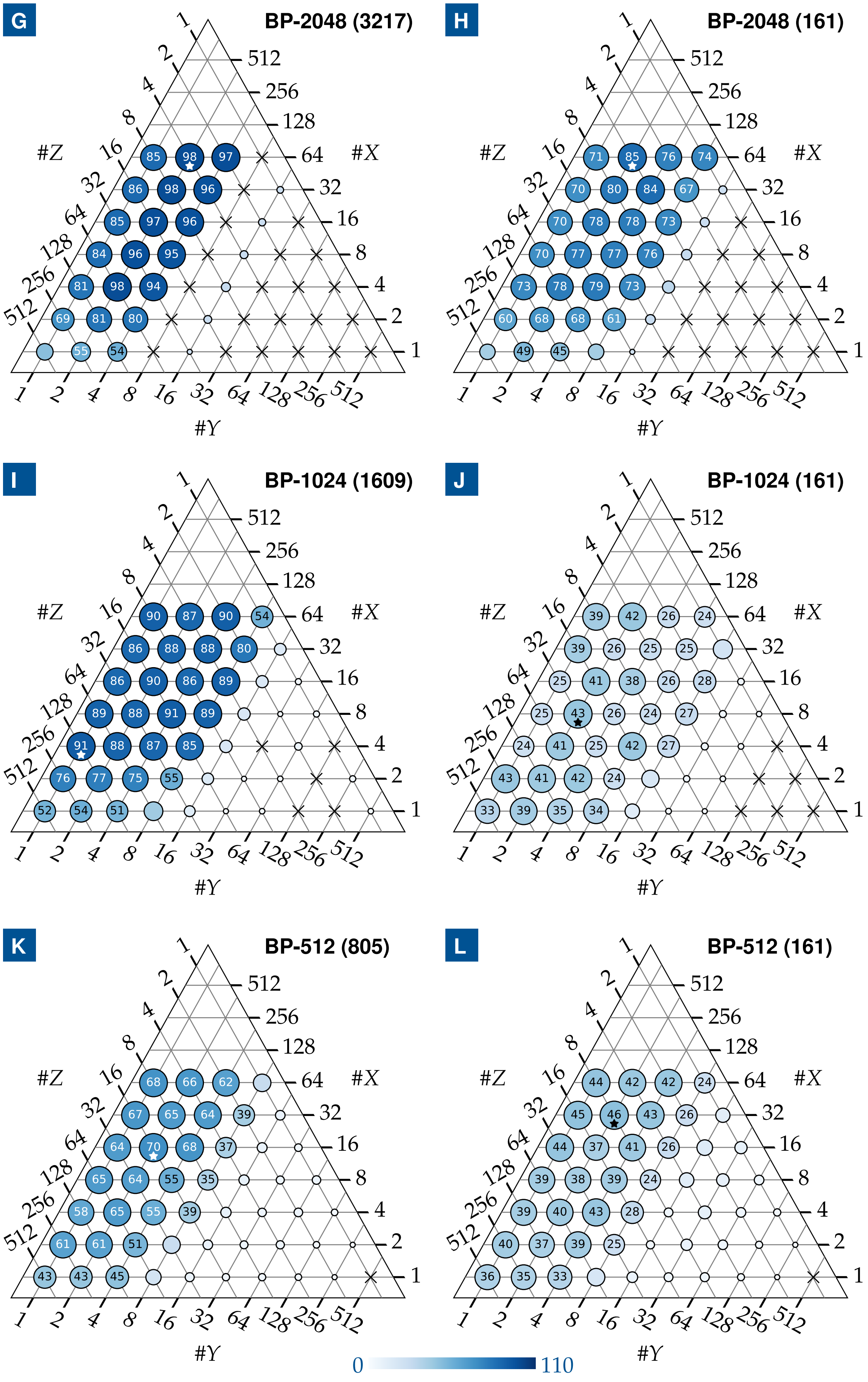
$$\begin{aligned} \#U\text{-}\#V\text{-}\#A &= 4\text{-}16\text{-}8 && \text{for the FP} \\ \#X\text{-}\#Y\text{-}\#Z &= 64\text{-}2\text{-}4 && \text{for the BP.} \end{aligned}$$

We considered the performance for big data more important than for small data, where the runtime is short anyways.

The big differences in performance can be traced back to several reasons. First, the memory access patterns are different for dif-



(Figure 21) Performance of FP and BP over the workgroup shape. Full caption on page 86.



(Figure 21) Performance of FP and BP over the workgroup shape. Full caption on the following page.

◄ Figure 21 Performance of FP and BP over the workgroup shape.
The plots on the previous two pages show the performance of the FP and BP in GPUS. Each data point is the maximum of six subsequent runs. Six triangular subplots (A–F and G–L) always show the results for different data sizes, which are encoded in the label at the upper right of each triangle. After the type of operation, it shows the side length of the cubic test volume ($\#x$, $\#y$, $\#z$), which equals the side length of the quadratic virtual detector ($\#u$, $\#v$). The number of views involved ($\#\varphi$) is given in round brackets and was chosen according to the Nyquist criterion (left column; A, C, E, G, I, K) or a number typical for ordered-subset solvers (right column; B, D, F, H, J, L). All tests used the same cone-beam with an opening angle of 19.4°. The walk parameter in the BP was set to $k = 32$.
Each data point is represented by a circle. Small numbers inside show the performance in GUPS per device. The size of the circles relates the results within a triangle (biggest is best), the color within all other triangles of the same operation (colorbar at the bottom). A little star marks the best setting per triangle. Crossed-out data points mark positions where the automatic splitting does not succeed and results in memory overflow. The driver of our GPUs does not allow workgroup dimension with $\#A > 64$ for the FP and $\#X > 64$ for the BP.

ferent workgroup shapes, especially the number of image cache hits. Second, for the BP, the results are shifted towards smaller $\#Y$ because of the kernel walk parameter $k = 32$ which multiplies the kernel footprint in memory along the y dimension. Third, our memory splitting techniques produce overhead for the data transfers over the PCIe bus for some configurations. Appendix D shows the corresponding results if the transfer times are not considered for computing the performance. Comparing figure 21 and figure 29 in the appendix, the maximum performance possible with and without transfers differs the more the less data is considered. In the 2048-cubed case with the full set of views (plots A and G), it is almost equal. That means, that transfers are hidden very efficiently for big data sets. For smaller data sets, it could be more beneficial to split up the data into more chunks to hide more transfers.

8.2.2. Walk parameter

The other parameter still left for optimization is the walk parameter k. The performance for different settings are shown in figure 22. To rule out that the performance is higher for a slightly different choice of the workgroup shape together with a certain k, not only the best (star markers) choice has been investigated, but also some others that resulted in a good performance before.

The series over different walk parameters suggests either 32 or 64 as best common choice. Despite of the slightly smaller performance, we decided for 32 in order to improve the slice chunk splitting for smaller data sets in y. As the slice chunk size is computed as $k\#Y = 64$, a data set can take advantage of all four GPUs only if $\#y \geq k\#Y\#d = 256$ where $\#d$ is the number of devices. For $k = 64$ this lower limit would already be 512. Notably, our k is a lot higher then $k = 4$ which was suggested by the time the walk parameter was first introduced [102]. The reason for the difference is very probably the greater caching capabilities of recent GPUs. This setting was also applied for the workgroup test above.

8.3. Optimizing the minimizer

Having chosen the optimum components, their synergy and intrinsic parameters the last item to tune in the reconstruction chain is the solver. Apart from the number of iterations, which depends on the data set, there is another parameter: the number of subsets. It determines the number of views investigated per iteration. Less views result in less time per iteration, but too few might result in a worse convergence rate. The question treated in this section is how many subsets result in the fastest convergence per time.

Two series of measurements are required for the answer. The first is a series of SIR reconstructions on the same problem using different numbers of subsets. In order to guaranty convergence, we use full cycles only and append one iteration that uses all views. That choice makes the number of iterations a multiple of the number of subsets plus one and thus different for each run. Two quantities have to be recorded for each setting. The first is the reconstruction time and the second the state of convergence.

The second measurement is necessary to relate these two quantities to one another. As the state of convergence does not change linearly with time, the exact relationship has to be measured. The measurement was carried out with help of a SIR where the current guess was recorded after each iteration. We used a very conservative choice of ten subsets for 3217 views compared the original paper of the very similar OS-MOM solver suggesting 24 subsets for 2934 views [44] which are even distributed over several turns of a helical scan.

The last element missing to carry out the subset optimization is a scalar measure for evaluating the state of convergence. We decided for one sophisticated measure, the

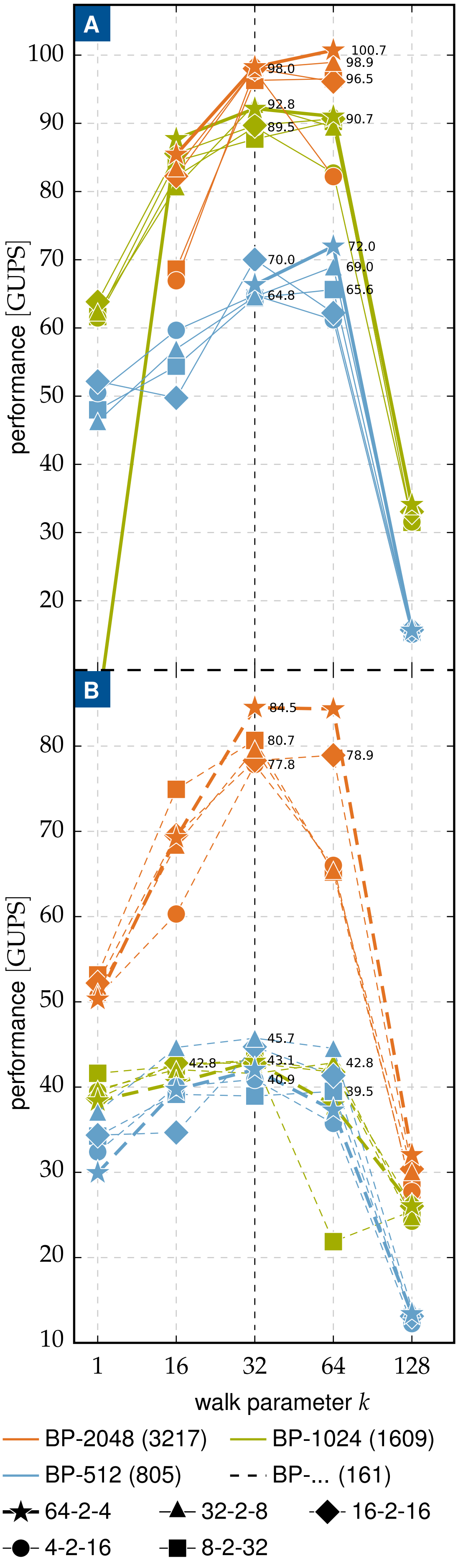


Figure 22 Performance of the BP over the walk parameter. ▶
The plot shows the performance (in GUPS) over the walk parameter k in the BP. It consists of many data series differing in data size (color) and workgroup shape (marker shape) which are explained in the legend at the bottom. The notation for the data sizes is the same as in figure 21. The three numbers in sequence used to label the workgroup shape stand for the kernel dimensions $\#X$-$\#Y$-$\#Z$. Similar to the columns used to visualize the performance over the workgroup shape, the upper row here shows data sets with a full angular range (A) and the lower row (B) a set of corresponding typical subset versions. The best setting for each series is labeled with the exact performance value (some overlapping labels were removed). The setting that was considered as the best overall setting is marked with a vertical dotted line.

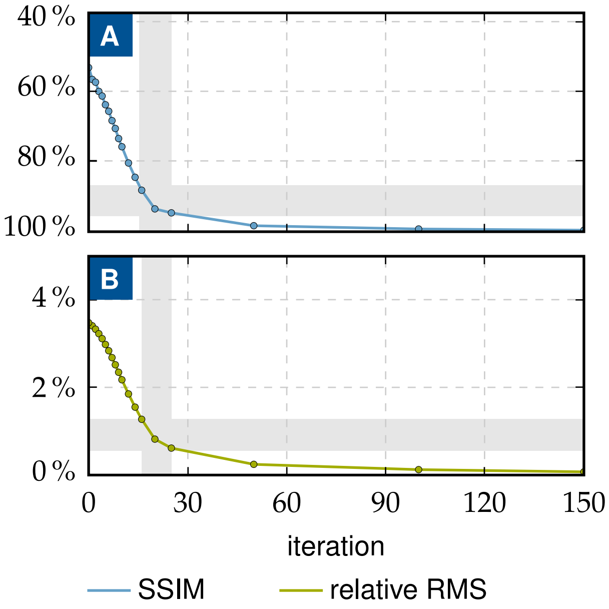


Figure 23 A reference for the convergence of SIR. The state of convergence over the iterations of a reference SIR is required to evaluate the final convergence of each SIR from the subset optimization series. Each data point shows the comparison of the reconstructed tomogram after the given number of iterations with the final reconstruction after 200 iterations. The upper plot (A) uses the structural similarity index (SSIM) as measure, the lower plot (B) uses the root mean square (RMS) normalized to the value range of the final reconstruction. A horizontal gray bar in the background labels the range of values covered by the results from the subset test series. A vertical bar marks the corresponding range of reference iterations to which those values map according to the measured curve.

structural similarity index (SSIM) [103, 104], and one classic measure, the relative root mean square (RMS), between the current guess and the converged state. The reconstruction was considered as fully converged after 200 iterations which is justified by the little changes happening with respect to these two measures long time before this high number of iterations is reached.

The data set used for the evaluation is a micro-CT measurement of a real object, which will be described in detail in the next chapter. In order to save compute time, the innermost 64 slices were considered significant enough to compute the required measures.

Figure 23 shows the resulting state of convergence over the iterations. The data points were recorded only at the given points. The last point of measurement taken after the full 200 iterations is not shown, because it was used as the fully converged reference.

Connecting the data points with piecewise linear function allows to map the final state of convergence from each reconstruction in the subset series to the corresponding number of iterations the reference SIR would have required. For example, the final reconstruction from the test with 17 subsets resulted in a SSIM of $90.1\,\%$. According to the recorded curve, this value is equivalent to 17.3 iterations of the reference SIR. Dividing the time each subset test took by its corresponding number of reference iterations gives the time per reference iteration, which is a good measure for the performance of each setting.

As a reference iteration is not a very intuitive measure and further depends on the arbitrary choice of the reference SIR, this performance measure can be scaled by the number of reference iterations, the most performant setting reached. The result is the theoretical time each subset test would have taken to reach the same state of convergence as the best setting. Figure 24 displays this quantity for all subset tests we carried out.

The decrease in time until the optimum can be explained by the smaller number of views that have to be processed for each subset. A comparison with the high number of subsets used in the original publication of the OS-MOM [44] suggests that the increase afterwards is presumably not due to a decreased solver convergence for more subsets. Rather, the runtime for FP and BP decreases slower as they turn more and more from a compute-limited regime into a transfer-limited regime.

Apart from a look at the GPU profilings, which are not shown here, this argumentation is strengthened by comparing figure 21 B and H

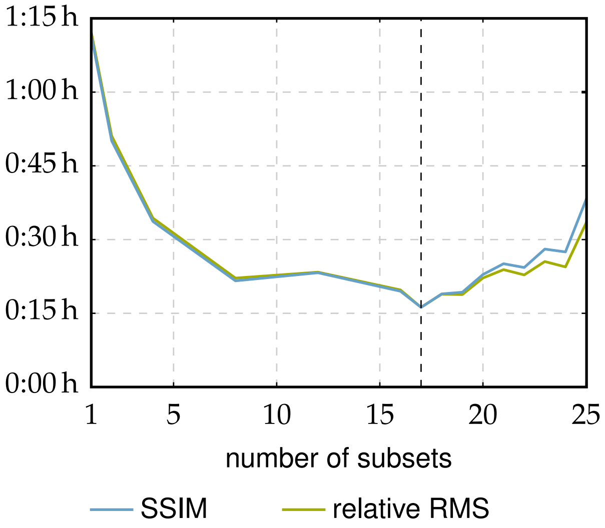


Figure 24 Runtime for different numbers of subsets in SIR until an equal state of convergence.
A series of SIR test runs carried out with different numbers of subsets reveals the fastest converging setting for our implementation. The curve shows the time the different reconstructions required normalized to the state of convergence of the fastest setting. Two different measures were acquired to rate the state of convergence resulting in slightly different normalizations. Details about the normalization are shown in figure 23. The clear optimum can be found at 17 subsets which is highlighted by the dashed line. It required 16 min for the test. The worst setting (one subset) took almost five times longer.

with figure 29 B and H in the appendix. The plots show that the performance of the subset FP and BP are significantly faster if transfers are not considered in the performance evaluation. This behavior is much less dominant for the corresponding operations on the full set of views, which are shown in subfigures A and G, respectively. The subset configuration of 161 views, which is shown in the figure, equals exactly the configuration required for the 20 subset test here.

Conclusion

This chapter showed that, independent of the performance of the implementation, the choice of the right parts and their synergy, the choice of the GPU-internal parameters and the choice of the right number of subsets together can make a difference of more then two orders of magnitude in performance. Conceptionally, we saw that data and regularization term for the gradient are best computed in sequence, both on GPUs, and that the best number of subsets depends on the transfer and compute performance of FP and BP and thus not only on the implementation, but also on the hardware.

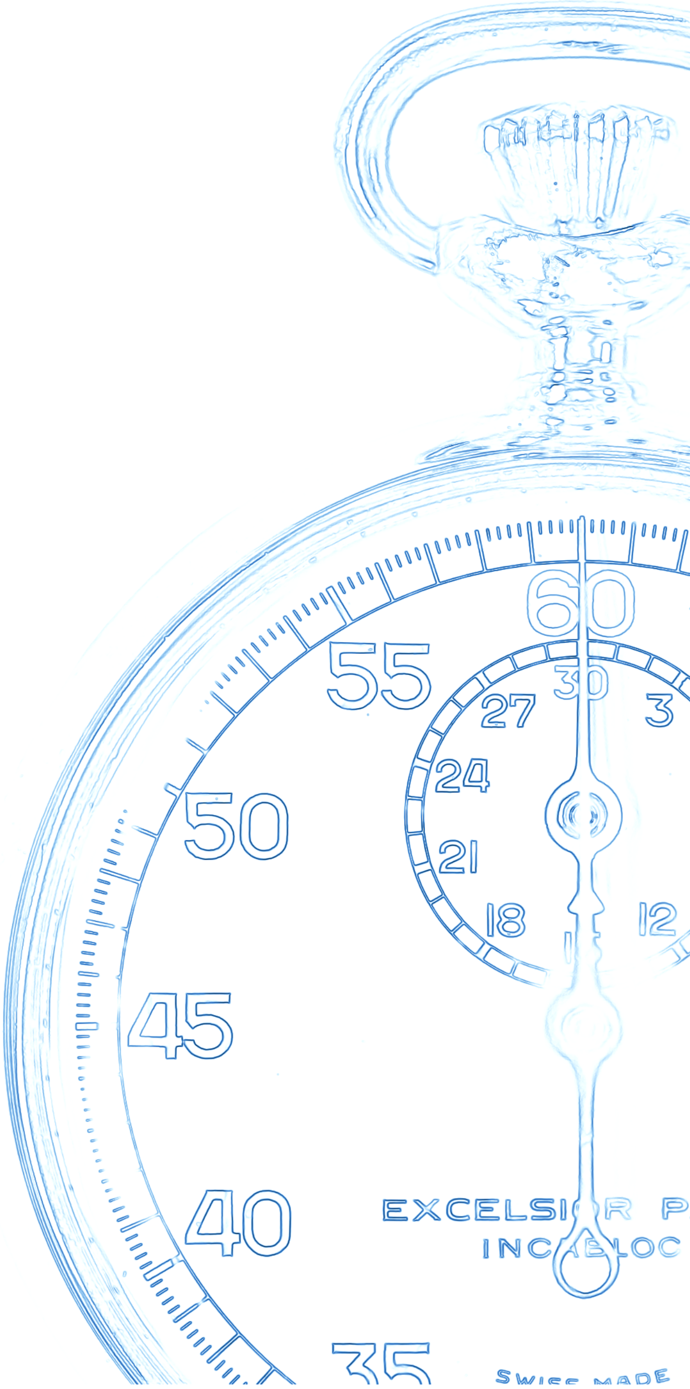
Part III

Results

Having spent three chapters on implementation and optimization, it is finally time to see the fruit of the labor. A real sample investigated in a high-resolution micro-CT is used to evaluate if our suggested methods are suited to make the benefits of iterative reconstruction available for high-resolution CT. Deciding questions for its convenience are if it can keep up with the measurement process, how long it takes until the result is ready and if there is still room for fine tuning or for more sophisticated models in the future. The last chapter of this thesis (chapter 9) is dedicated to these questions.

9. Real-time SIR for micro-CT

Many applications on moderate data sizes have shown the great benefits of SIR for image quality compared to analytical methods. This fact holds especially for sparse angular sampling and noisy data. The last chapter of this thesis evaluates the benefits of SIR for high-resolution CT in a scientific or industrial environment. It will turn out that reconstruction time is a deciding factor. But before we are able to benchmark the algorithm, some considerations about the actual task and the constraints of that environment are necessary. Further, as the runtime goes quasi linearly with the number of iterations and the image quality improves with the number of iterations until convergence, the actual task for the benchmark has to be clearly defined.



Picture: An old-school stopwatch. 🅭

9.1. Constraints and potential

The most common environments for micro and nano CT devices are industry and science. Typical fields of application are error analysis, material research and structure analysis. Its high investment and maintenance costs in addition to the long acquisition times of typically several hours per sample often require those devices to be constantly in use in order to be profitable. Long acquisition times also prevent micro CT from their near- or even in-line application. The profound insights possible with CT would be a great asset for inspection, but the state-of-the-art technique is too slow to be integrated into most established processes. These circumstances form the setting for new reconstruction techniques. They impose some constraints, but also give a lot of room for improvement.

9.1.1. Time

Heading more towards near- or in-line application, time is a crucial issue. On the one hand, a time-critical environment poses a great constraint on the reconstruction algorithm. It has to work *real-time*. Using the following very common definition, that means

> “pertaining to the processing of data by a computer in connection with another process outside the computer according to time requirements imposed by the outside process.” [105]

For CT, the reconstruction algorithm has to be faster than the acquisition time in order not to limit the throughput of a device.

On the other hand, we see a great potential for SIR to make micro CT more profitable and push it a big step closer to the production line for several reasons.

First, SIR is less picky on the acquisition parameters than the state-of-the-art approaches. Both of the two most deciding factors for the total acquisition time can be reduced: the number of views which determine the angular sampling [106], and the exposure time which determines the noise level as will be shown afterwards.

Second, SIR has the possibility to lay the emphasis of the reconstruction to a certain image quality feature. This asset is very valuable in the application, because different problems have different needs. As an example, a subsequent material segmentation or surface detection profits rather from a better SNR, whereas a metrology application measuring distances or absolute positions would prefer more well-defined edges. The reconstruction parameter allows to tune the same measurement for the optimum result regarding the required quantity. Thus, a fast low-quality measurement reconstructed with SIR may yield similar or better results for a certain application than a high-quality measurement reconstructed with an analytical approach.

The benchmark section below will measure the reconstruction time and evaluate the total speedup that is possible by SIR for a predefined task.

9.1.2. Image quality

There is no general definition of a good or bad image. The measure for quality always depends on the application and still there are different choices of metrics that allow to put image quality into a number.

For medical CT, several task-based approaches were suggested to evaluate the quality of a tomogram [88, e. g.]. However, what counts in the end, is how certain a radiologist can diagnose diseases from the result.

This property can hardly be put into a number or related to a certain image quality metric.

In micro- and nano-CT, the situation is fortunately clearer for most applications. The images are often evaluated by software that has well-defined requirements or the manual evaluation concentrates on a few very well-defined aspects.

Here are some of the most common image quality metrics:

SSIM the structural similarity index is a modern measure to compare the similarity of two images, especially of an image to a reference image. Common image errors like noise, distortions or compression artifacts result in a smaller similarity than changes considered less serious by human perception like differences in brightness, shifting or rescaling. It considers the mean values, standard deviations and covariances in local areas of both images. The values returned are in the range $[0, 1]$, where 1 is reserved for identical images. [103, 104]

RMS the root-mean-square error can also be used to compare an image μ to a reference image μ_{ref}:

$$\mathrm{RMS} = \frac{1}{\sqrt{N}} \left\| \mu - \mu_{\mathrm{ref}} \right\|_2$$

where $N = \|\mu\|_0$ is the number of pixel or voxel elements in μ. It returns a quantitative measure for the noise level if a noisy image is compared against a noise-free reference. Feature mismatches can also be detected. As the absolute value is usually not meaningful without knowing the value range of the two images, we use the relative RMS in this thesis, which is computed by normalizing the RMS to the value range of the reference.

histogram entropy is defined as

$$S_{\mathrm{E}} = - \sum_{b \in [0, \#b[} \mathrm{H}_{\mu}(b) \log_2 \left(\mathrm{H}_{\mu}(b) \right)$$

where $\mathrm{H}_{\mu}(b)$ is the histogram of μ containing $\#b$ bins [107, e. g.]. It is a measure for the sharpness of the histogram peaks which becomes worse with increasing noise or unsharpness, for example.

SNR the signal-to-noise ratio can be measured by selecting two regions in the image. One in the background for measuring its mean value $\bar{\mu}_{\mathrm{BG}}$ and one on a homogeneous region on the signal for measuring the signal strength $\bar{\mu}_{\mathrm{FG}}$ and the standard deviation σ_{FG}. The SNR is then obtained by

$$\mathrm{SNR} = \frac{\bar{\mu}_{\mathrm{FG}} - \bar{\mu}_{\mathrm{BG}}}{\sigma_{\mathrm{FG}}}.$$

It is a good measure to access the noise level.

edge FWHM is the measure we use to characterize edges. First, we select a rectangular area containing only a straight edge. Second, the direction of the edge is determined by a linear fit. Third, we use a parallel-beam forward projector to project the rectangular area along the edge direction. Fourth, the full-width at half-maximum (FWHM) of a Gaussian fit on the derivative of the resulting line profile is taken as width of the edge. This rather complicated procedure allows to evaluate edges not only at a very local spot, but also considers whether an edge is fringed.

None of these measures can be regarded as the one quality criterion to characterize an image and still, visual inspection is sometimes the only meaningful method. But these measures can be very helpful if quality has to be put in a number, especially for comparing quality. The amount of improvement possible depends strongly on the actual application, including for example the complexity of the structure of the measured sample.

SIR is able to outperform conventional methods either by using the same measurement and reaching higher image quality or by relaxing the high demands on the acquisition and reaching the equal image quality. A detailed quality investigation will be shown for the benchmark test sample below.

9.2. Benchmark and profilings

The benchmark test presented in this section is meant to serve as an exemplary, a prove-of-principal example of the value added to micro- or nano-CT by introducing SIR. It was designed with the pre-defined goal to reduce the exposure time by a factor of six and thus make the acquisition faster. The number of views was intentionally not decreased so that the reconstruction time depends only of the choice of algorithms and their implementation, but not in the amount of data involved.

9.2.1. Sample and setup

As test sample we have chosen a ball pen. Of course, it is not directly relevant to industry or science, which is also not important for our benchmark, but it has several convincing features. First, it is a sample composed mainly of one material containing many clear edges as many industrial samples do. Several straight edges make it easy to apply edge quality metrics. Small impurities in the material allow to test the capability of SIR to preserve small features, which sometimes get lost, if the regularization is not configured well. Second, there is no material involved that would require extra beam-hardening correction, which is a problem for many applications, but not meant to be solved in our benchmark. Third, it fits into the field of view of the detector so that special local tomography treatment is not required.

The micro-CT device used is a *Zeiss Versa XRM 500*. It was used in the full 2k mode, meaning that each projection view has a size of $(2048\,\mathrm{px})^2$ and the corresponding reconstruction volume a size of $(2048\,\mathrm{vx})^3$. The opening angle is $19.4\,°$ in fan and cone direction and the resulting voxel size is $(6.83\,\mu\mathrm{m})^3$. In order to fulfill the Nyquist criterion, 3217 views were taken.

There were two measurements taken. A reference measurement at $30\,\mathrm{s}$ exposure time per view ($29{:}32\,\mathrm{h}$ in total) resulting in a decent image quality with a usual FDK reconstruction and the benchmark measurement at $5\,\mathrm{s}$ exposure time per view ($6{:}45\,\mathrm{h}$ in total). It is important to note, that the reference is not meant to be seen as gold standard, but rather defining the quality level we want to obtain.

We decided for the simple Gaussian data term and a Huber regularization as presented before in chapter 8. The reconstruction computer was equipped with the maximum performance available at that time. It holds

- four *NVIDIA GeForce GTX Titan X* (Maxwell) GPU devices each providing 3072 threads with a clock frequency of $1.0\,\mathrm{GHz}$ and holding $12\,\mathrm{GiB}$ of graphics memory,
- two *Intel Xeon E5-2667 v3* CPUs with a clock frequency of $3.2\,\mathrm{GHz}$ supporting AVX 2 vectorization and 16 threads in total (Hyperthreading deactivated), and
- $512\,\mathrm{GiB}$ DDR4 RAM operating at $2133\,\mathrm{MHz}$ organized in four channels and 16 banks.

9.2.2. Finding the reconstruction settings

Before the actual benchmark can be performed, the required reconstruction parameters and number of iterations have to be determined. They have to result in the fastest reconstruction possible that reaches the same image quality as the reference measurement.

The transition parameter γ of the Huber regularization has been guessed from the width of the signal peak in the histogram of the FDK reconstruction. Its strength λ was determined empirically by applying the FDK normalization proposed in chapter 7 and running several subsequent subvolume reconstructions until the best setting was reached, evaluated by visual comparison.

The number of iterations has been chosen from a look on the behavior of different image quality measures over the number of iterations up to 200, which was considered to be the fully converged state. Figure 25 suggests a value of 17 plus the final iteration on all views suggested in subsection 3.3.4. Notably, it was possible to choose a multiple of the optimum-performance setting found in the previous chapter.

	hist. entropy	SNR	edge FWHM
FDK/5 sec	42 %	0.9	2.28 ± 0.16
SIR/5 sec	36 %	3.5	3.43 ± 0.04
FDK/30 sec	37 %	2.5	3.38 ± 0.22

Table 2 Absolute quality measures reached for the performance benchmark.
The table shows the quality reached in the FDK (top) and for the choice of SIR parameters that are used in the performance benchmark (middle) compared to the 30 sec-exposure high-statistics reference (bottom). SIR exceeds the edge quality of the reference only by a little, but its histogram entropy and SNR are quite superior.

All metric plots were oriented such that better values result in a lower data point. The only metric which does not improve already for a small number of iterations is the edge FWHM. Two reasons speak for such a behavior, both going back to the edge-preserving Huber prior. The first is the fact that Huber assumes a linear penalty for edges which does not smoothen edges at all, but only with reduced strength. This claim is supported by the low error in the Gaussian fit in table 2 indicating a close to Gaussian smoothed edge. The second is that a reduced edge smoothening does not mean a more correct edge. Huber-smoothened edges often become sharp but straight edges tend to become slightly jagged. Our way of evaluating the edge quality along a small straight segment reflects that effect in the quality plot. Figure 26 C and D show the areas where the edge quality and the SNR were measured inside the registered reconstructions.

The plot also shows that the chosen number of iterations is not close to the converged state. Rather it was chosen to reproduce the quality of the reference. Fortunately the small region where SNR and histogram entropy are good enough already and the edge still not affected too much contains one of the settings for optimum performance making the final choice easy. Table 2 shows the quality values reached for the chosen setting.

9.2.3. Visual quality assessment

Optimizing the reconstruction parameters required some representative metrics that allow to express quality in single numbers. But still, we consider the visual quality inspection still very meaningful for rating whether the quality of our chosen SIR parameters is high enough for a valid speed comparison.

Figure 26 shows representative slices of the different measurements and reconstruction techniques and discusses their quality. The low-statistics SIR reconstruction shows the same level of detail as the high-statistics reference. It shall be noted that the coarser noise

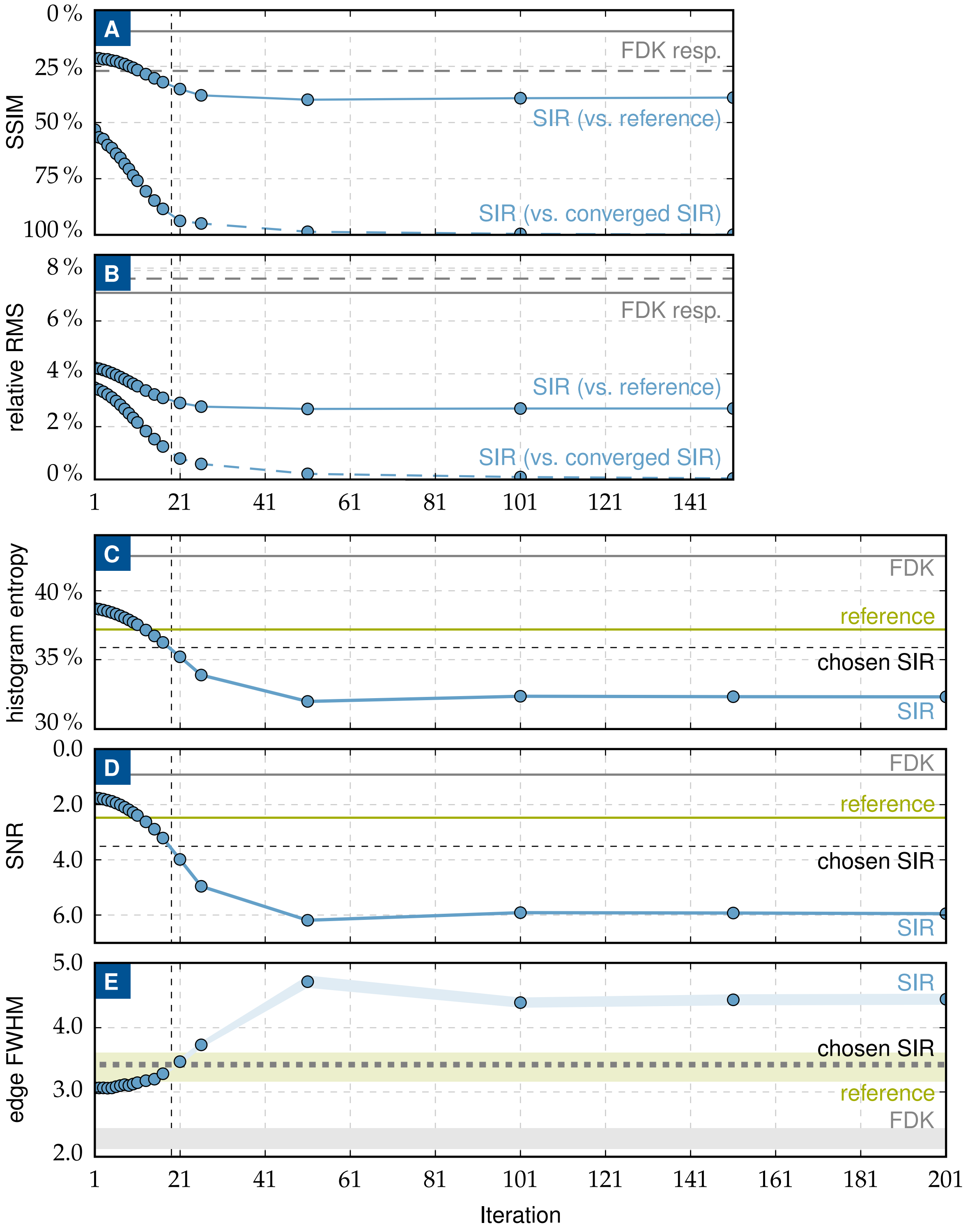
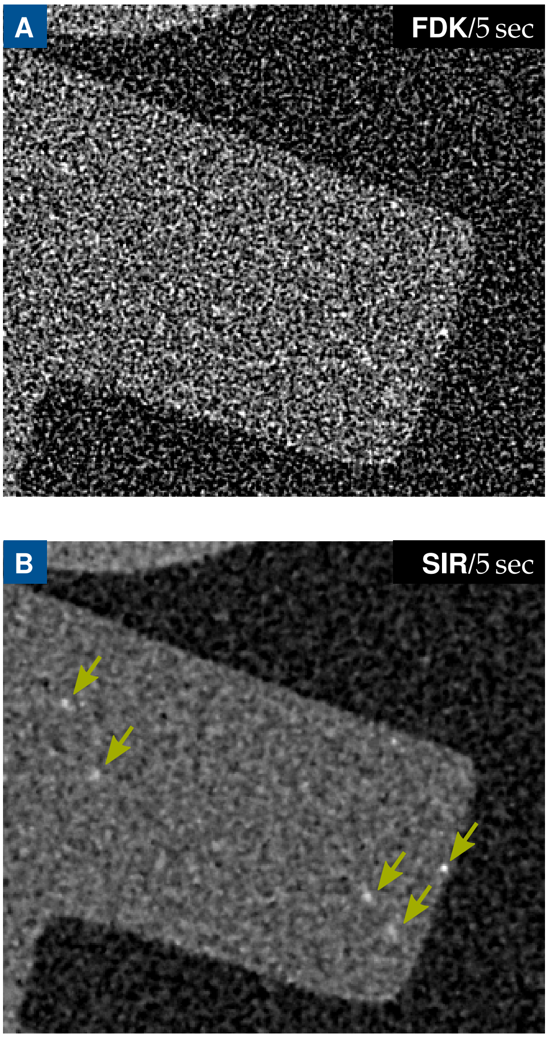


Figure 25 Several quality measures over the iterations.
These global image quality measures were recorded to compare our low-statistics SIR benchmark (in blue) reconstruction to the high-statistics FDK reference (in green) and choose the minimum number of iterations required to reach equal quality. Linear interpolation was applied between the actual points of measurement (blue dots). The quality level of the low-statistics FDK is indicated by a gray line.
The two relative measures (A, B) are each evaluated once compared to the high-statistics FDK (solid line) and once to the fully converged SIR (at 200 iterations, dashed line). In the absolute measures (C–E) a dotted line indicates the quality level of the chosen SIR. For the edge FWHM (E), confidence areas are shown instead of line plots. SSIM (A) and SNR (D) are shown with inverted vertical axis so that better quality results in a lower curve. Notably, the edge quality first decreases with more iterations due to the Huber regularization. We chose a number of 18 iterations for the benchmark (dotted vertical line). At this point, the edge quality (E) is quasi the same as in the reference and the noise properties represented by the histogram entropy (C) and the SNR (D) are well improved.



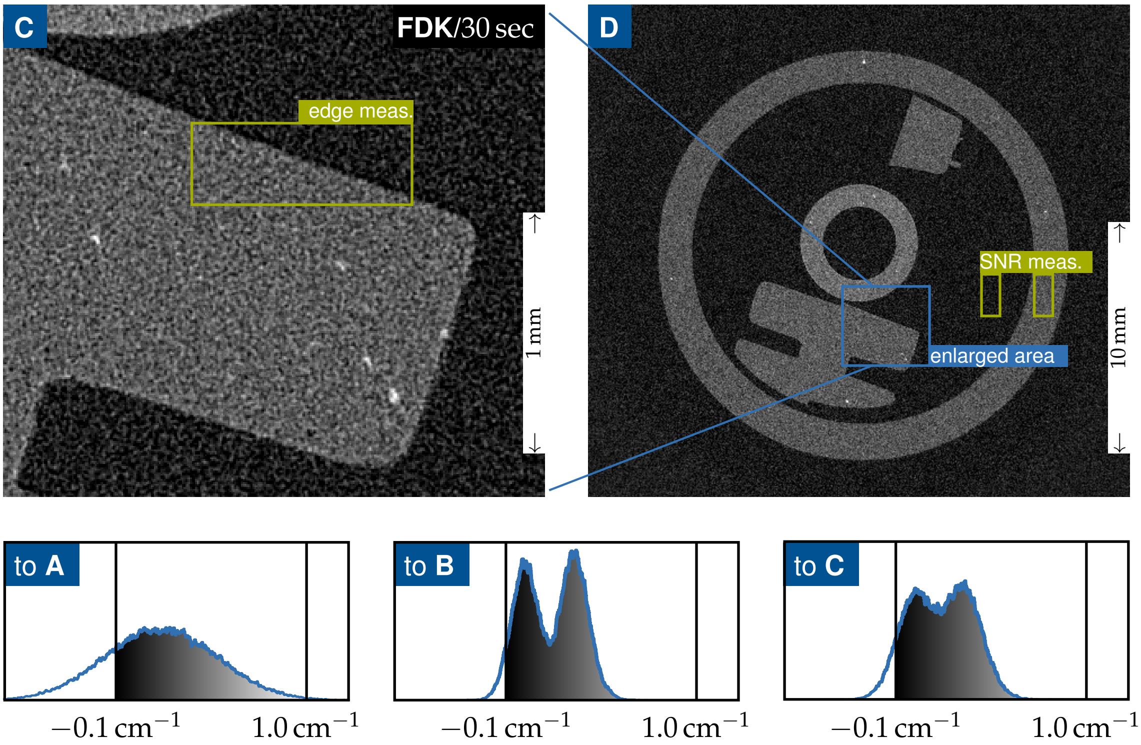


Figure 26 Benchmark quality reference: A real sample. The figure shows from top to bottom the results of different combinations of exposure time and reconstruction algorithm applied on a ball-pen sample. A low-statistics measurement with $5\,\mathrm{sec}$ exposure time was reconstructed with FDK (A) and proposed SIR (B). A high-statistics FDK (C) reconstructed from $30\,\mathrm{sec}$ exposure time per view serves as quality reference for evaluating the benefit of SIR. Each view is an enlarged area inside coregistered reconstructions. The whole slice is shown below the caption (D). A blue rectangle marks the enlarged area. Green rectangles in C and D mark the regions taken for different image quality measures that were discussed in figure 25. The bottom row shows the histograms belonging to figures A–C. Black vertical lines mark the gray-scale window shown. Scalebars can be found at the right edge of figure C and D.

The low-statistics FDK shows the noisy appearance typically resulting from a fast measurement in combination with an analytic reconstruction. In contrast to SIR, the histogram does not allow to separate plastic and air. Five green arrows in B mark impurities in the material, which got lost in the FDK but can be revealed through SIR. The resulting contrast is similar to the one in the reference. Comparing SIR to the reference, the different noise texture makes it hard to compare their quality. The finer noise of the reference is more pleasing to the eye, but does not reveal more detail. A comparison of the histograms even suggests clearly improved signal separation for SIR.

Concluding, SIR can reconstruct all important details visible in the reference and separate them very well from the noise. Its coarser noise pattern is first unfamiliar, but on a second look does not harm the image quality with respect to contrast, sharpness or level of detail. Together with the previously evaluated quality metrics, this result is the basis for a fair benchmark test below.

pattern easily tricks the eye as it generates an impression of roughness and unsharpness, which is not reflected in the quantitative noise and edge measurements above. As a reason, we assume that the eye is much more efficient in filtering high-frequency than low-frequency noise. The reduced but coarser noise of SIR thus looks quasi as severe as the noise in the reference and somehow less pleasing. But, for a machine trying material separation for example, this subjective impression does not play a role.

In summary, we assume the quality of SIR at least as good as the reference. This finding is the basis for a fair benchmark below. The quantitative measurements investigated are all in favor of SIR. The SNR is even $40\,\%$ better. Also small details can successfully be recovered by SIR as the comparison of exemplary image slices showed. The higher SNR helps to overcome the visual impression of more roughness generated by the coarser noise texture.

9.2.4. The actual gain of SIR for micro-CT

The final outcome of the different investigations carried out in this thesis is illustrated in figure 27. Three profiling plots show the performance of SIR in 2k microCT. The whole reconstruction has finished $23{:}37\,\mathrm{min}$ after data acquisition is complete and always stays below the $256\,\mathrm{GB}$ memory mark. That means that it is over 17 times faster than the acquisition and the maximum memory occupied is only three times the amount of the input and output volumes. That means that 2k SIR is definitely feasible in real time on a single workstation enabling a future four times higher efficiency of microCT devices. Compared to currently possible improvements in hardware, especially a higher source or detector efficiency, this is a vast improvement at very low cost.

The profiling shows that our implementation is almost GPU-only. The remaining small and memory-limited operations on CPU allow a very cost-efficient choice of CPU devices. Further, our implementation is able to benefit from the current intense GPU developments. By the time of the end of this thesis, in 2017, the latest hardware available was already claimed to be up to three times faster than the hardware used for the benchmark [108]. This rapid performance boost has for sure a great impact on our overall performance.

A look on the already very good ratio of acquisition and reconstruction time suggests a lot of room left for further development. Possible options are

- applying more of the inexpensive subset iterations, e. g. until full convergence for higher quality or less optimized and more general-purpose use of SIR,
- using sparsely sampled data further increasing performance linearly without overhead,
- including more sophisticated data models incorporating Poisson noise for low statistics [17], polychromatic spectra [96] or source and detector blur [18],
- optimizing the acquisition together with the reconstruction for a dedicated task or prior measurement [92] to gain maximum performance in near- or in-line applications.

Closure

The greatest benefits of SIR for microCT can be found in the vast improvement of either throughput or image quality and its open de-

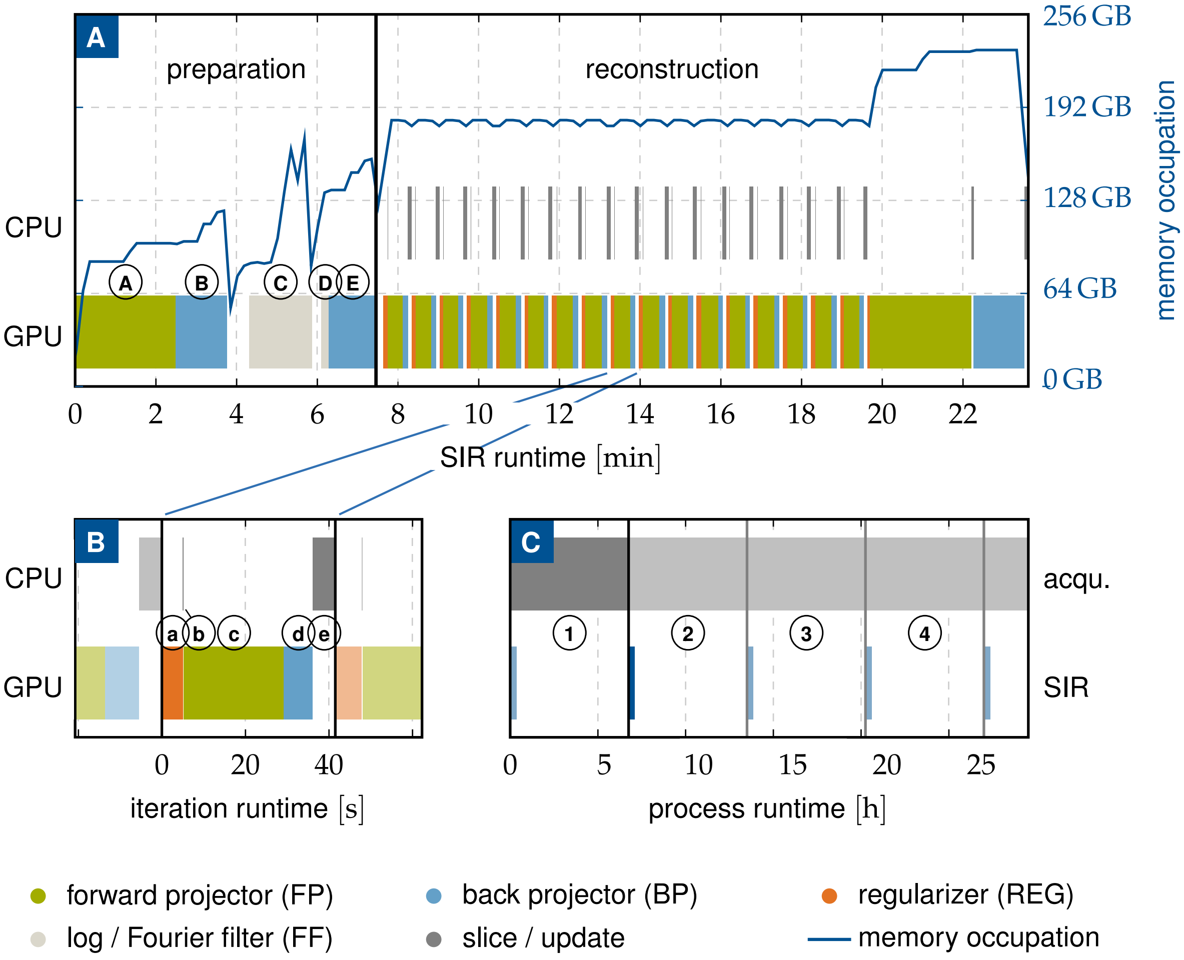


Figure 27 Profiling SIR. This figure visualizes the required resources of the proposed iterative reconstruction approach over time and how much benefit can be gained compared to the standard procedure using FDK. The underlying data is a fully-sampled 2k micoCT dataset.

The horizontal bars in the upper plot (A) show all important processes on CPU and GPU for the whole SIR. A blue line indicates the memory occupied during these processes. The total time sums up to 23:37 min. It can be divided into a preparation phase (7:27 min) and a reconstruction phase (16:10 min). Preparation includes computing the curvature (Ⓐ and Ⓑ), the line integrals (Ⓒ) and the FDK for the first guess (Ⓓ and Ⓔ). The projection data is loaded in the background while the data-independent curvature is computed. The reconstruction includes 17 equal iterations on subsets and one final iteration on all views.

One of the iterations is exemplarily shown enlarged on the lower left (B). The total time is 41.5 s. It includes computing the gradient of the regularizer (ⓐ) and the data term (ⓑ, ⓒ and ⓓ) as well as the momentum and image update (ⓔ). The circled labels in subfigures A and B match the nomenclature of the solver-specific basic operations introduced in subsection 8.1.2 on page 80. A legend for the colors can be found at the bottom of the figure.

Finally, the benefit of SIR over FDK for our showcase is illustrated on the lower right (C). It shows several consecutive acquisitions of the same type as our test measurement (upper bar) with subsequent interlaced iterative reconstruction (lower bar). The throughput of measured samples is clearly limited by the acquisition process (6:45 h each) making SIR a real-time approach. The total width of the plot illustrates the time a single investigation of the same quality takes using conventional FDK reconstruction (29:32 h). Small circled numbers enumerate the samples that can be measured in that time using SIR. The total increase of throughput is 4.28.

sign for including physical models that are often cause to severe image artifacts. In our benchmark, we replaced a long 2k high-resolution measurement and FDK reconstruction with a shorter measurement and subsequent SIR keeping the image quality as similar as possible by several different measures. We were able to gain more than a factor of four in speedup with a lot of room still open for further improvement and future developments.

Outtake

We consider it fair to not only show the “good” results in a scientific work but also some impressions of what was considered “bad” or “false” results. If a scientist passes a negative verdict on a result, the question for the reason of the failure has to be answered. Otherwise it would be unclear if our results tell something about nature or if they are just our presumptions and expectations reflected in another form. Apart from that, some of the images are almost artistic and nice to look at. The misfits can be found in figure 28.

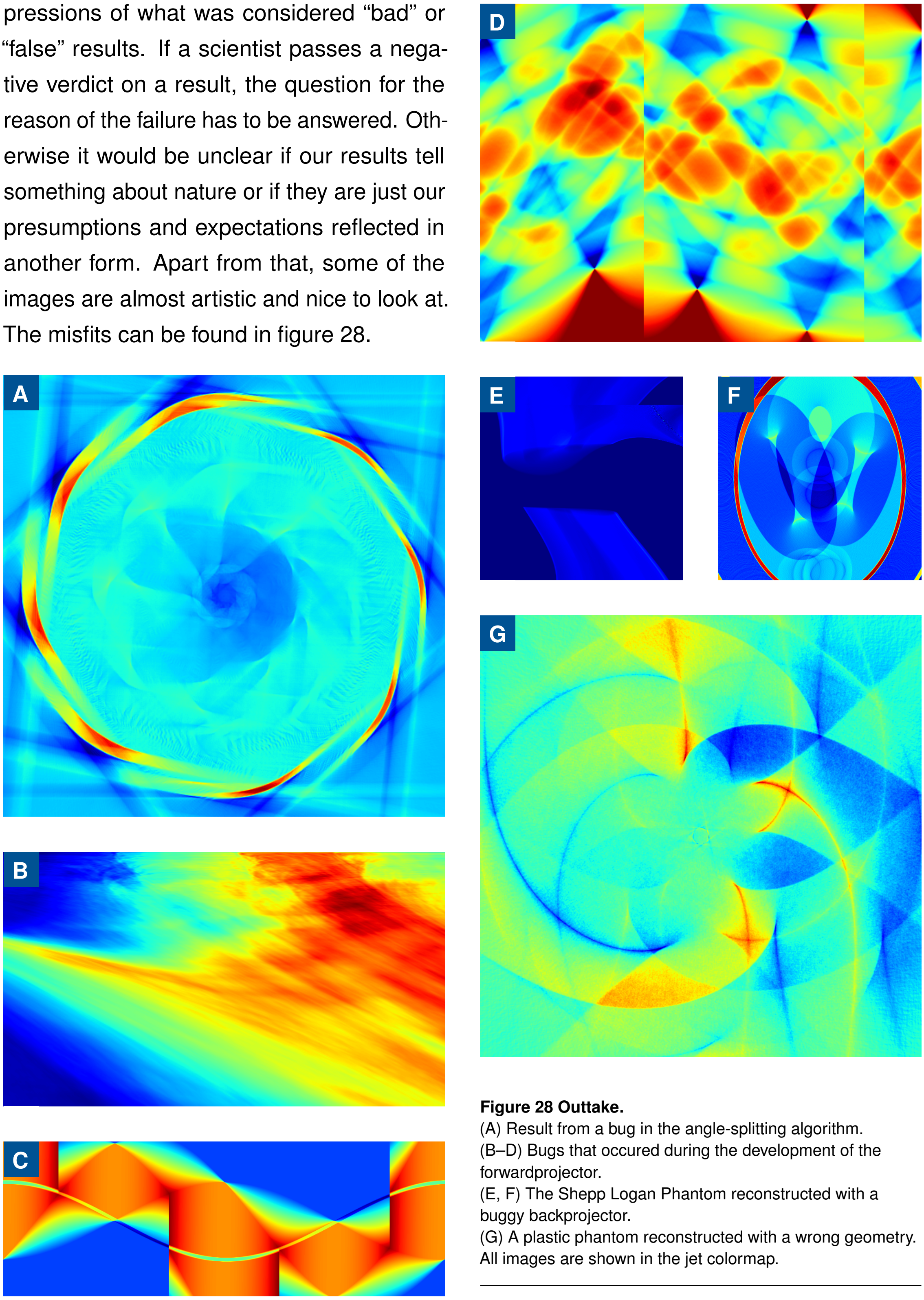


Figure 28 Outtake.
(A) Result from a bug in the angle-splitting algorithm.
(B–D) Bugs that occured during the development of the forwardprojector.
(E, F) The Shepp Logan Phantom reconstructed with a buggy backprojector.
(G) A plastic phantom reconstructed with a wrong geometry.
All images are shown in the jet colormap.

Appendix

A.

Derivations of the cost function

Most solvers require derivatives of the cost function as stated in subsection 3.3.4 on page 28. Algorithm 10 summarizes the most common ones.

Each derivation is calculated for

- the cost function with a Gaussian data term as defined in algorithm 3 on page 26,
- the generic Gibbs neighborhood prior as defined in algorithm 4 on page 27 assuming a symmetric potential $\Psi(t) = \Psi(-t)$, i. e. $\Psi'(t) = -\Psi'(-t)$,
- the Huber, Quadratic and TV penalty functions, where the Huber penalty was defined first in algorithm 4, and
- the Neighborhood prior on the Laplacian assuming equally spaced voxels separately in each spatial direction (i. e. $\Delta_{i,p} = \Delta_{i,n}$).

The gradient

The gradient is required in almost every solver, because it gives the direction to the minimum. It is the vector $\{\partial_k \mathcal{L}\}_k$ for $k \in [\boldsymbol{\mu}]$ holding the first derivative of the cost function in every direction μ_k. The factor $1/2$ in the gradient of the Gibbs prior vanishes considering that each gradient between two neighbors is computed from either side.

Deriving the Laplacian formulation additionally requires considering the opposite neighbor. In contrast to the gradient-based neighborhood prior, some next neighbors are required to compute the gradient or the subsequent second-order derivatives.

The curvature

The curvature is the second derivative of a paraboloid surrogate function around the cost function. It is used by the SPS algorithm and variants like the OS-SPS or OS-OGM described in subsection 3.3.4. There are different possible definitions of the curvature depending on the exactness of the surrogate. The most popular one is independent from $\boldsymbol{\mu}$. Therefore, it can be precomputed once for the whole optimization.

For a very general regularizer $R(\boldsymbol{\mu}) = \sum_r \Psi_r\left([\boldsymbol{C} \cdot \boldsymbol{\mu}]_r\right)$, it writes

$$\operatorname{curv} R = |\boldsymbol{C}|^\mathsf{T} \cdot \left(\operatorname{diag}\left\{\Psi_r''(0)\right\}_r\right) \cdot |\boldsymbol{C}| \cdot \mathbf{1}$$

where $|\boldsymbol{C}|$ denotes the element-wise absolute of matrix $\boldsymbol{C}$ [44, eq. 7, p. 169].

The Hessian diagonal

The Hessian matrix $\operatorname{Hess} \mathcal{L} = \{\partial_k \partial_l \mathcal{L}\}_{k,l}$ for $k, l \in [\boldsymbol{\mu}]$ would be the full second derivative of the cost function. It is huge for usual tomographic problems and cannot be stored in memory.

One way, to take advantage of it is by computing only is diagonal. This approach is for example used in some preconditioners. A preconditioner $\boldsymbol{T}$ is a matrix that can be multiplied to an ill-posed linear problem

$$\boldsymbol{T} \cdot \boldsymbol{A} \cdot \mu = \boldsymbol{T} \cdot p$$

like tomographic reconstruction. It leads to the same solution as the original problem, but is supposed to improve the speed of convergence for the solver.

Algorithm 10 Derivatives of the cost function

Common derivatives of the SIR cost function with Gaussian noise model are

$$
\begin{aligned}
\mathcal{L}_{\mathrm{G}}(\boldsymbol{\mu}) &= \frac{1}{2}\, \|\boldsymbol{A}\cdot\boldsymbol{\mu}-\boldsymbol{p}\|_w^2 + \lambda\, R(\boldsymbol{\mu}) \\
\operatorname{grad}\, \mathcal{L}_{\mathrm{G}}(\boldsymbol{\mu}) &= \boldsymbol{A}^{\mathsf{T}}\cdot\left.(\boldsymbol{A}\cdot\boldsymbol{\mu}-\boldsymbol{p})\right|_w + \lambda\, \operatorname{grad}\, R(\boldsymbol{\mu}) \\
\operatorname{curv}\, \mathcal{L}_{\mathrm{G}} \quad &= \boldsymbol{A}^{\mathsf{T}}\cdot\left.(\boldsymbol{A}\cdot\mathbf{1})\right|_w + \lambda\, \operatorname{curv}\, R \\
\operatorname{diag}\operatorname{Hess}\, \mathcal{L}_{\mathrm{G}}(\boldsymbol{\mu}) &= \left.\boldsymbol{A}^{\mathsf{T}}\right|^2\cdot\left.\mathbf{1}\right|_w + \lambda\, \operatorname{diag}\operatorname{Hess}\, R(\boldsymbol{\mu}) \\
\operatorname{Hess}\mathrm{d}\, \mathcal{L}_{\mathrm{G}}(\boldsymbol{\mu}) &= \boldsymbol{A}^{\mathsf{T}}\cdot\left.(\boldsymbol{A}\cdot\boldsymbol{d})\right|_w + \lambda\, \operatorname{Hess}\mathrm{d}\, R(\boldsymbol{\mu}) \\
\operatorname{denom}\, \mathcal{L}_{\mathrm{G}}(\boldsymbol{\mu}) &= \|\boldsymbol{A}\cdot\boldsymbol{d}\|_w^2 + \lambda\, \operatorname{denom}\, R(\boldsymbol{\mu})
\end{aligned}
$$

with the according derivations of the Gibbs neighborhood prior

$$
\begin{aligned}
\mathrm{R}_{\mathrm{G}}(\boldsymbol{\mu}) &= \frac{1}{2}\sum_{i\in[\boldsymbol{\mu}]}\sum_{n\in\mathcal{N}_i}\frac{1}{\Delta_{i,n}}\,\Psi\left(\partial_n\,\mu_i\right) \qquad \text{with} \quad \partial_n\,\mu_i = \frac{\mu_i-\mu_n}{\Delta_{i,n}} \\
\operatorname{grad}\, \mathrm{R}_{\mathrm{G}}(\boldsymbol{\mu}) &= \left\{\sum_{n\in\mathcal{N}_k}\frac{1}{{\Delta_{k,n}}^2}\,\Psi'\left(\partial_n\,\mu_k\right)\right\}_k \qquad \text{for} \quad k\in[\boldsymbol{\mu}] \\
\operatorname{curv}\, \mathrm{R}_{\mathrm{G}} \quad &= \left\{2\sum_{n\in\mathcal{N}_k}\frac{1}{{\Delta_{k,n}}^3}\,\Psi''(0)\right\}_k \\
\operatorname{diag}\operatorname{Hess}\, \mathrm{R}_{\mathrm{G}}(\boldsymbol{\mu}) &= \left\{\sum_{n\in\mathcal{N}_k}\frac{1}{{\Delta_{k,n}}^3}\,\Psi''\left(\partial_n\,\mu_k\right)\right\}_k \\
\operatorname{Hess}\mathrm{d}\, \mathrm{R}_{\mathrm{G}}(\boldsymbol{\mu}) &= \left\{\sum_{n\in\mathcal{N}_k}\frac{1}{{\Delta_{k,n}}^3}\,\Psi''\left(\partial_n\,\mu_k\right)\,\left(d_k-d_n\right)\right\}_k \\
\operatorname{denom}\, \mathrm{R}_{\mathrm{G}}(\boldsymbol{\mu}) &= \sum_{i\in[\boldsymbol{\mu}]}\sum_{n\in\mathcal{N}_i}\frac{1}{{\Delta_{i,n}}^3}\,\Psi''\left(\partial_n\,\mu_k\right)\,\left(d_i-d_n\right)^2 .
\end{aligned}
$$

(continued on the next page)

Hessian times vector and the denominator

The Hessian matrix provides very useful information about the minimum of a convex function. Another way to exploit the Hessian despite of its size, is by directly multiplying it with a vector. Some solvers use the function $(\operatorname{Hess}\,\mathcal{L})\cdot\boldsymbol{d}$ computing the Hessian times a given search direction $\boldsymbol{d}$. Some others rely on a scalar called denominator, which is defined as $2\,\partial_\alpha^2\,\mathcal{L}\left.(\boldsymbol{\mu}+\alpha\boldsymbol{d})\right|_{\alpha=0}$. It is for example used for computing the step size in some NLCG variants.

(Algorithm 10) Derivatives of the cost function (continued from the previous page)

The corresponding derivatives of the neighborhood prior on the Laplacian read

$$
\begin{aligned}
\mathrm{R_G}(\boldsymbol{\mu}) &= \frac{1}{2} \sum_{i\in[\boldsymbol{\mu}]} \sum_{n\in\mathcal{N}_i} \frac{1}{\Delta_{i,n}} \Psi\left(\partial_n^2 \mu_i\right) \qquad \text{with} \quad \partial_n^2 \mu_i = \frac{\mu_p - 2\,\mu_i + \mu_n}{{\Delta_{i,n}}^2} \\
\operatorname{grad} \mathrm{R_G}(\boldsymbol{\mu}) &= \left\{ \sum_{n\in\mathcal{N}_k} \frac{1}{{\Delta_{k,n}}^3} \left(\Psi'\left(\partial_{nn}^2 \mu_n\right) - \Psi'\left(\partial_n^2 \mu_k\right)\right) \right\}_k \qquad \text{for} \quad k \in [\boldsymbol{\mu}] \\
\operatorname{curv} \mathrm{R_G} &= \left\{ 4 \sum_{n\in\mathcal{N}_k} \frac{1}{{\Delta_{k,n}}^5} \Psi''(0) \right\}_k \\
\operatorname{diag}\operatorname{Hess} \mathrm{R_G}(\boldsymbol{\mu}) &= \left\{ \sum_{n\in\mathcal{N}_k} \frac{1}{{\Delta_{k,n}}^5} \left(2\,\Psi''\left(\partial_n^2 \mu_k\right) + \Psi''\left(\partial_{nn}^2 \mu_n\right)\right) \right\}_k \\
\operatorname{Hess} \mathrm{d}\, \mathrm{R_G}(\boldsymbol{\mu}) &= \left\{ \sum_{n\in\mathcal{N}_k} \frac{1}{{\Delta_{k,n}}^5} \left(2\,\Psi''\left(\partial_n^2 \mu_k\right) \left(d_n - d_k\right) + \Psi''\left(\partial_{nn}^2 \mu_n\right) \left(d_k - 2\,d_n + d_{nn}\right)\right) \right\}_k \\
\operatorname{denom} \mathrm{R_G}(\boldsymbol{\mu}) &= \sum_{i\in[\boldsymbol{\mu}]} \sum_{n\in\mathcal{N}_i} \frac{1}{{\Delta_{i,n}}^5} \Psi''\left(\partial_n^2 \mu_i\right) \left(d_p - 2\,d_n + d_n\right)^2
\end{aligned}
$$

with, for example, either the Huber penalty function

$$
\begin{aligned}
\Psi_{\mathrm{H}}^{\gamma}(t) &= \begin{cases} 1/2\,\gamma^{-1}\,t^2 & \text{for } |t| < \gamma \\ |t| - \gamma/2 & \text{else} \end{cases} \\
{\Psi_{\mathrm{H}}^{\gamma}}'(t) &= \begin{cases} \gamma^{-1}\,t & \text{for } |t| < \gamma \\ \operatorname{sign}(t) & \text{else} \end{cases} \\
{\Psi_{\mathrm{H}}^{\gamma}}''(t) &= \begin{cases} \gamma^{-1} & \text{for } |t| < \gamma \\ 0 & \text{else,} \end{cases}
\end{aligned}
$$

the quadratic

$$
\begin{aligned}
\Psi_{\mathrm{Q}}^{\gamma}(t) &= 1/2\,t^2 \\
{\Psi_{\mathrm{Q}}^{\gamma}}'(t) &= t \\
{\Psi_{\mathrm{Q}}^{\gamma}}''(t) &= 1,
\end{aligned}
$$

or TV penalty

$$
\begin{aligned}
\Psi_{\mathrm{TV}}^{\gamma}(t) &= |t| \\
{\Psi_{\mathrm{TV}}^{\gamma}}'(t) &= \operatorname{sign}(t) \\
{\Psi_{\mathrm{TV}}^{\gamma}}''(t) &= 0.
\end{aligned}
$$

Notation: $\|\,\boldsymbol{.}\,\|_w^2$ is the weighted square norm, $\mathbf{1}$ a vector where all entries are 1, $\boldsymbol{.}\,|_w$ a component-wise weight, $\boldsymbol{.}\,|^2$ a component-wise square, $\mathcal{N}_{\boldsymbol{.}}$ the spatial 3-D neighborhood of a voxel, and $\Delta_{\boldsymbol{.},\boldsymbol{.}}$ the spatial distance between two voxels.

Indices: If n (for *next*) denotes a neighbor in the neighborhood of a voxel, then p (for *previous*) is the neighbor opposing n and nn (for *next neighbor*) the voxel behind n.

B.

A possible definition of the Euler angles

This appendix shows a possible definition of the Euler angles required in section 4.2 on page 33 and how the rotation matrix $\mathbf{R}_{\boldsymbol{\varphi}}$ is derived from them. In the definition below, the three Euler angles $\boldsymbol{\varphi} = (\varphi, \theta, \psi)^{\mathsf{T}}$ successively describe the rotation of the detector plane and the opposing source S around the axes y, $-\mathrm{z}'$, and y'. Each dash indicates a previous rotation of the axis itself. The origin of the coordinate system is assumed in the center of rotation C shown in figure 12 on page 34.

The corresponding rotation matrix reads

$$
\begin{aligned}
\mathbf{R}_{\boldsymbol{\varphi}} &= \mathbf{R}_{\psi} \cdot \mathbf{R}_{\theta} \cdot \mathbf{R}_{\varphi} \\
&= \begin{pmatrix} \mathrm{c}_{\psi} & & \mathrm{s}_{\psi} \\ & 1 & \\ -\mathrm{s}_{\psi} & & \mathrm{c}_{\psi} \end{pmatrix} \cdot \begin{pmatrix} \mathrm{c}_{\theta} & -\mathrm{s}_{\theta} & \\ \mathrm{s}_{\theta} & \mathrm{c}_{\theta} & \\ & & 1 \end{pmatrix} \cdot \begin{pmatrix} \mathrm{c}_{\varphi} & & \mathrm{s}_{\varphi} \\ & 1 & \\ -\mathrm{s}_{\varphi} & & \mathrm{c}_{\varphi} \end{pmatrix} \\
&= \begin{pmatrix} -\mathrm{s}_{\psi}\,\mathrm{s}_{\varphi} + \mathrm{c}_{\theta}\,\mathrm{c}_{\psi}\,\mathrm{c}_{\varphi} & -\mathrm{s}_{\theta}\,\mathrm{c}_{\psi} & \mathrm{s}_{\psi}\,\mathrm{c}_{\varphi} + \mathrm{c}_{\theta}\,\mathrm{c}_{\psi}\,\mathrm{s}_{\varphi} \\ \mathrm{s}_{\theta}\,\mathrm{c}_{\varphi} & \mathrm{c}_{\theta} & \mathrm{s}_{\theta}\,\mathrm{s}_{\varphi} \\ -\mathrm{c}_{\psi}\,\mathrm{s}_{\varphi} - \mathrm{c}_{\theta}\,\mathrm{s}_{\psi}\,\mathrm{c}_{\varphi} & \mathrm{s}_{\theta}\,\mathrm{s}_{\psi} & \mathrm{c}_{\psi}\,\mathrm{c}_{\varphi} - \mathrm{c}_{\theta}\,\mathrm{s}_{\psi}\,\mathrm{s}_{\varphi} \end{pmatrix}
\end{aligned}
$$

where $\mathrm{s}_{\bullet} = \sin(\bullet)$ and $\mathrm{c}_{\bullet} = \cos(\bullet)$.

C.

Computing the parallel-beam direction vector

For a parallel-beam geometry as described in subsection 4.2.4 on page 36, the projection direction $\boldsymbol{D}$ can be derived from an affine projection matrix $\tilde{\boldsymbol{P}}_\infty$ by solving

$$\boldsymbol{M}_{2\times 3}\cdot\boldsymbol{D}\overset{!}{=}0$$

$$\Leftrightarrow\quad\begin{pmatrix} p_{\mathrm{u,x}} & p_{\mathrm{u,y}} & p_{\mathrm{u,z}} \\ p_{\mathrm{v,x}} & p_{\mathrm{v,y}} & p_{\mathrm{v,z}} \end{pmatrix}\cdot\begin{pmatrix} d_{\mathrm{x}} \\ d_{\mathrm{y}} \\ d_{\mathrm{z}} \end{pmatrix}\overset{!}{=}0$$

where $\boldsymbol{M}_{2\times 3}$ is the upper left 2×3 submatrix of $\tilde{\boldsymbol{P}}_\infty$ [47, p. 173].

This is an underdetermined linear system of two equations and three unknowns. But, as the vector $\boldsymbol{D}$ only describes a direction, its length can be variable. We found a solution by assuming that one component of $\boldsymbol{D}$ is 1 without loss of generality.

Assuming first that this component is $d_{\mathrm{z}}=1$, the equations simplify to

$$\begin{aligned} p_{\mathrm{u,x}}\,d_{\mathrm{x}}+p_{\mathrm{u,y}}\,d_{\mathrm{y}}+p_{\mathrm{u,z}}&=0 \\ p_{\mathrm{v,x}}\,d_{\mathrm{x}}+p_{\mathrm{v,y}}\,d_{\mathrm{y}}+p_{\mathrm{v,z}}&=0 \end{aligned}$$

or

$$\boldsymbol{M}_{\mathrm{xy}}\cdot\begin{pmatrix} d_{\mathrm{x}} \\ d_{\mathrm{y}} \end{pmatrix}=\begin{pmatrix} -p_{\mathrm{u,z}} \\ -p_{\mathrm{v,z}} \end{pmatrix}$$

where the matrix $\boldsymbol{M}_{\mathrm{xy}}$ holds the columns x and y of $\boldsymbol{M}_{2\times 3}$. The inverse of an arbitrary 2×2 matrix $\boldsymbol{A}$ is given by

$$\boldsymbol{A}^{-1}=\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1}=\frac{1}{\det\boldsymbol{A}}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

with

$$\det\boldsymbol{A}=a\,d-c\,b.$$

It follows that

$$\begin{aligned} \begin{pmatrix} d_{\mathrm{x}} \\ d_{\mathrm{y}} \end{pmatrix}&=\boldsymbol{M}_{\mathrm{xy}}^{-1}\cdot\begin{pmatrix} -p_{\mathrm{u,z}} \\ -p_{\mathrm{v,z}} \end{pmatrix} \\ &=\frac{1}{\det\boldsymbol{M}_{\mathrm{xy}}}\begin{pmatrix} p_{\mathrm{v,y}} & -p_{\mathrm{u,y}} \\ -p_{\mathrm{v,x}} & p_{\mathrm{u,x}} \end{pmatrix}\cdot\begin{pmatrix} -p_{\mathrm{u,z}} \\ -p_{\mathrm{v,z}} \end{pmatrix} \\ &=\frac{1}{\det\boldsymbol{M}_{\mathrm{xy}}}\begin{pmatrix} -p_{\mathrm{v,y}}\,p_{\mathrm{u,z}}+p_{\mathrm{u,y}}\,p_{\mathrm{v,z}} \\ p_{\mathrm{v,x}}\,p_{\mathrm{u,z}}-p_{\mathrm{u,x}}\,p_{\mathrm{v,z}} \end{pmatrix} \\ &=\frac{1}{\det\boldsymbol{M}_{\mathrm{xy}}}\begin{pmatrix} \det\boldsymbol{M}_{\mathrm{yz}} \\ -\det\boldsymbol{M}_{\mathrm{xz}} \end{pmatrix} \end{aligned}$$

where $\boldsymbol{M}_{\mathrm{yz}}$ holds the columns y and z of $\boldsymbol{M}_{2\times 3}$ and $\boldsymbol{M}_{\mathrm{xz}}$ the columns x and z, respectively.

Similar expressions can be derived if one of the other components of $\boldsymbol{D}$ is assumed to be 1:

$$\begin{aligned} \begin{pmatrix} d_{\mathrm{y}} \\ d_{\mathrm{z}} \end{pmatrix}&=\frac{1}{\det\boldsymbol{M}_{\mathrm{yz}}}\begin{pmatrix} -\det\boldsymbol{M}_{\mathrm{xz}} \\ \det\boldsymbol{M}_{\mathrm{xy}} \end{pmatrix}\quad\text{for } d_{\mathrm{x}}=1 \\ \begin{pmatrix} d_{\mathrm{x}} \\ d_{\mathrm{z}} \end{pmatrix}&=\frac{1}{\det\boldsymbol{M}_{\mathrm{xz}}}\begin{pmatrix} -\det\boldsymbol{M}_{\mathrm{yz}} \\ \det\boldsymbol{M}_{\mathrm{xy}} \end{pmatrix}\quad\text{for } d_{\mathrm{y}}=1 \end{aligned}$$

In summary, the whole solution can be computed as

$$\boldsymbol{D}=\begin{cases} \left(\delta_{\mathrm{yz}},-\delta_{\mathrm{xz}},\delta_{\mathrm{xy}}\right)^{\mathsf{T}}/\delta_{\mathrm{xy}} & \text{if } \delta_{\mathrm{xy}}\neq 0 \\ \left(\delta_{\mathrm{yz}},-\delta_{\mathrm{xz}},\delta_{\mathrm{xy}}\right)^{\mathsf{T}}/\delta_{\mathrm{yz}} & \text{if } \delta_{\mathrm{yz}}\neq 0 \\ \left(-\delta_{\mathrm{yz}},\delta_{\mathrm{xz}},\delta_{\mathrm{xy}}\right)^{\mathsf{T}}/\delta_{\mathrm{xz}} & \text{else} \end{cases}$$

with

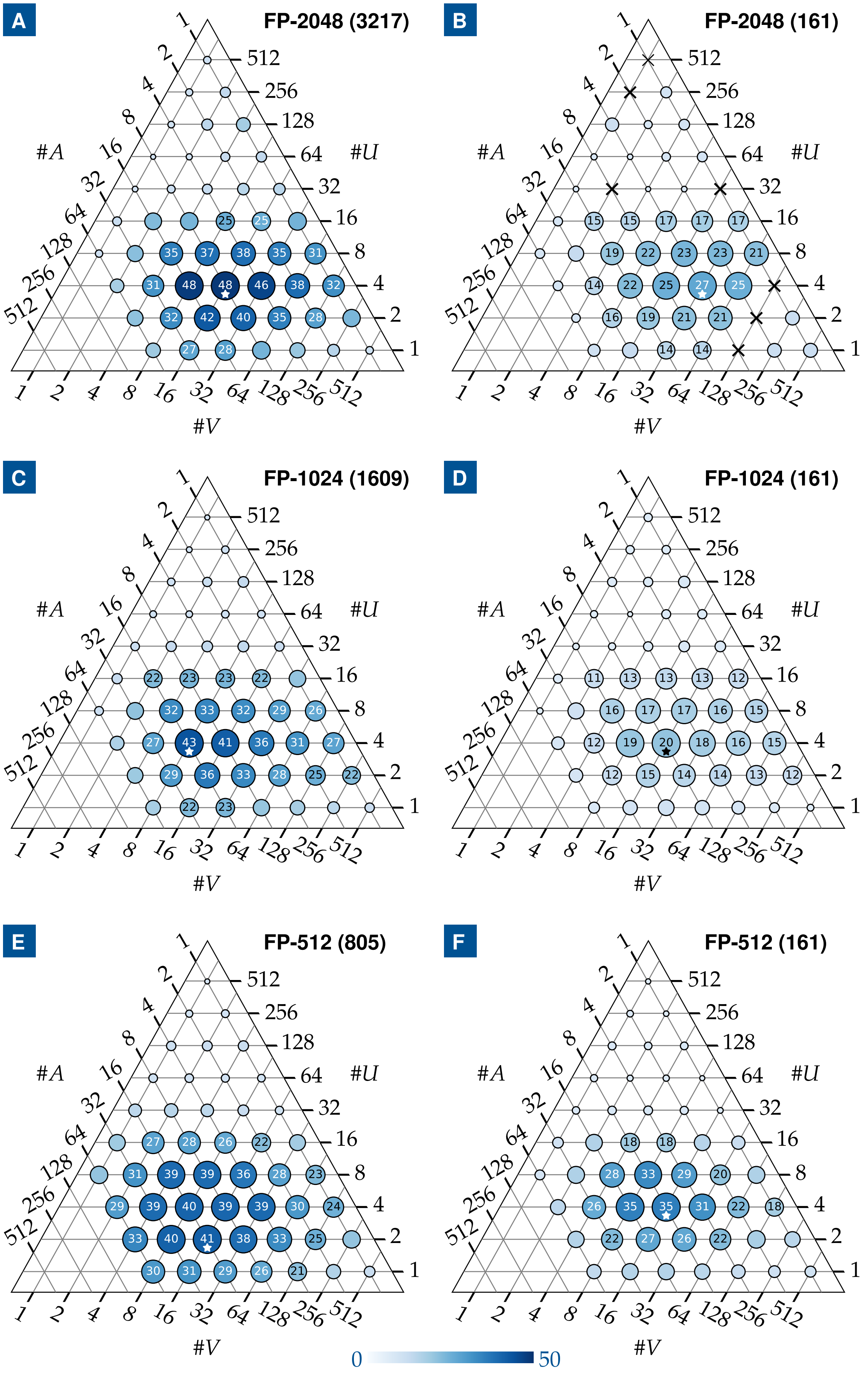
$$\begin{aligned} \delta_{\mathrm{xy}}&=\det\boldsymbol{M}_{\mathrm{xy}}=p_{\mathrm{u,x}}\,p_{\mathrm{v,y}}-p_{\mathrm{v,x}}\,p_{\mathrm{u,y}} \\ \delta_{\mathrm{yz}}&=\det\boldsymbol{M}_{\mathrm{yz}}=p_{\mathrm{u,y}}\,p_{\mathrm{v,z}}-p_{\mathrm{v,y}}\,p_{\mathrm{u,z}} \\ \delta_{\mathrm{xz}}&=\det\boldsymbol{M}_{\mathrm{xz}}=p_{\mathrm{u,x}}\,p_{\mathrm{v,z}}-p_{\mathrm{v,x}}\,p_{\mathrm{u,z}}. \end{aligned}$$

D.

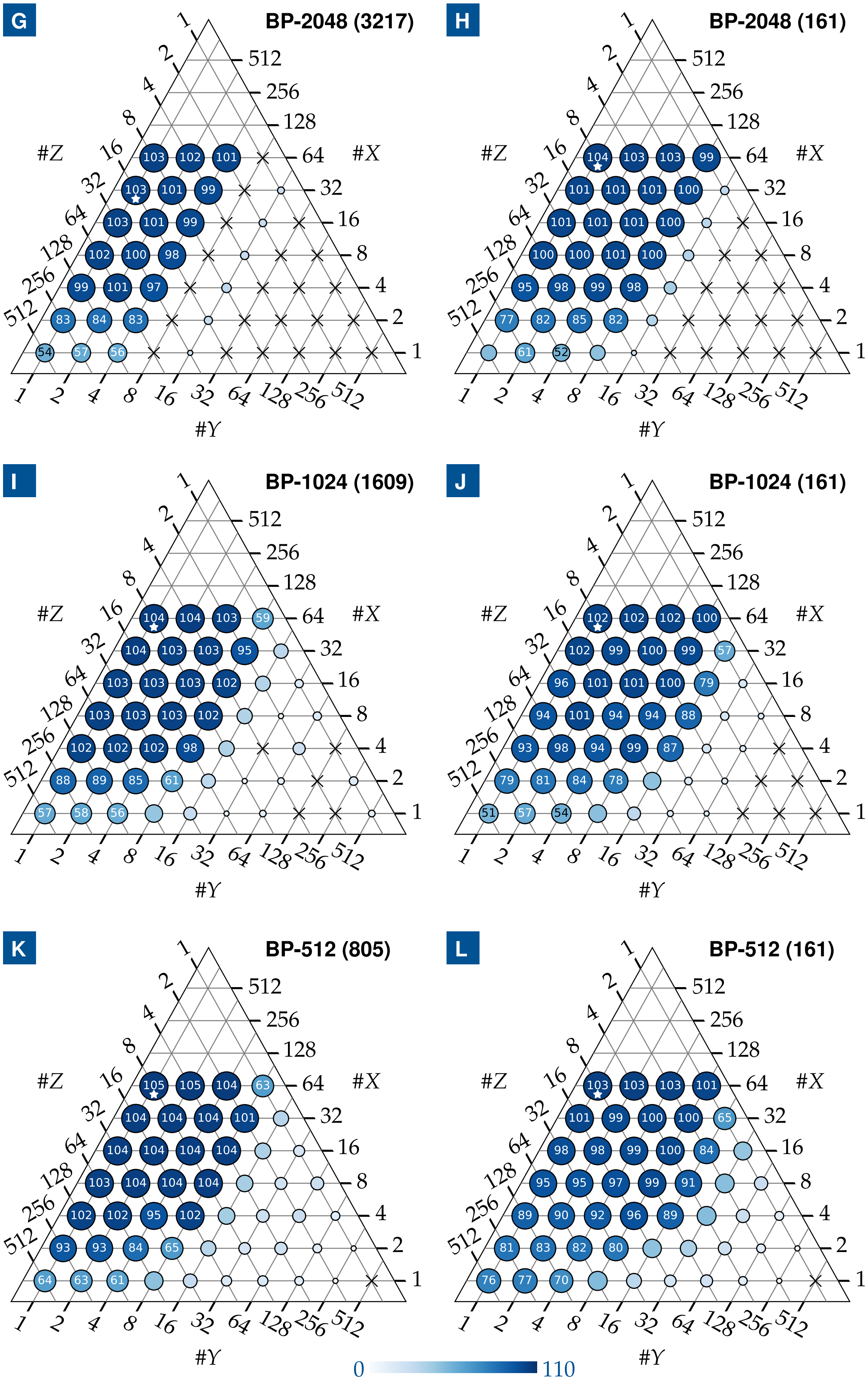
Optimizing the workgroup size without considering transfers

Subsection 8.2.1 on page 83 showed how to optimize the workgroup size of the FP and BP OpenCL kernel for maximum performance. Figure 29 displays an analog analysis to figure 21 on page 84 but without considering the times required for transfers from the host memory to the GPU devices. Comparing the two figures allows to evaluate how much the transfers limit performance for the different settings. Details can be found in the referenced subsection.

Figure 29 Performance of over the workgroup shape (FP, BP without transfers). ►
The plots on the next two pages show the performance of FP and BP over the workgroup size analog to figure 21 on page 84 but without considering the transfer times. The arrangement of subfigures, the labels, size and color codes are all equivalent.
In contrast to figure 21, the performance values differ only little for the different data sizes (columns of the plot) leading to the conclusion that transfers can be hidden more efficiently by bigger data sizes. The GUPS values are highest around a common maximum at $\#U$-$\#V$-$\#A = 4$-16-8 for the FP. The BP seems to prefer high $\#X$ and low $\#Y$. From the comparison with the results in the main document we conclude that the transfer overhead influences the position of the maximum only little.



(Figure 29) Performance over the workgroup shape (FP, BP without transfers). Full caption on the previous page.



(Figure 29) Performance over the workgroup shape (FP, BP without transfers). Full caption on page 115.

Acknowledgments

This thesis was written with the great support of many people. I want to acknowledge, in particular

- my doctorate supervisor Prof. Dr. Franz Pfeiffer for his scientific and team-leading skills I was allowed to benefit from and the freedom and opportunities he gave for my research,
- the mates from my office for all the great discussions and friendships, and
- all the others from the tomography teams lead by PD. Dr. Peter Noël and PD. Dr. Tobias Lasser,
- my colleagues Lorenz Hehn, Sebastian Allner, Korbinian Mechlem and Wolfgang Noichl and two other friends, Felix Will and Alexandra Göldner, for proofreading,
- Dr. Martin Dierolf and Wolfgang Noichl for a lot of things I was allowed to learn about IT and the great teamwork,
- Dr. Michael Epple, Dr. Marian Willner and Dr. Astrid Velroyen for putting much effort in building up our little company during the last two years,
- all the rest of the great team at the chair for a lot of fun and cake,
- Prof. Dr. Pierre Thibault and Dr. Dieter Hahn for the initial codebase,
- my flat mates, the team of the Lighthouse, my church, and the Cambridge Scholar Network, for great personal support and the fruitful discussions about science and faith,
- my dear mom and dad for all their effort and love, and
- my beloved wife Hanna.

Picture credits

The title image of chapter 1 on page 3 was taken from Pixabay (provided by `PublicDomainPictures` ↗) and is available under the Creative Commons license `CC 0` ↗. 🅭🄍

Figure 1 on page 4 was kindly provided by the artist Judith Ganter. It is published at Rannenberg & Friends.

The figure shows a subregion of the original illustration after small image enhancements.

Figure 2 on page 5 was taken from Wikipedia (name: `Coat of Arms of Oxford University` ↗) and is available under public domain. 🅮

The title image of chapter 2 on page 9 was taken from Wikimedia Commons (name: `Roentgen2.jpg` ↗) and is available under public domain. 🅮

The original picture was modified with artistic image filters.

Figure 3 on page 10 was taken from Wikipedia (name: `X-ray by Wilhelm Röntgen of Albert von Kölliker's hand - 18960123-02.jpg` ↗) and is available under public domain. 🅮

The title image of chapter 3 on page 17 was taken from Wikipedia (name `RIMG0279.JPG` ↗) and is available under the Creative Commons license `CC BY-SA 3.0` ↗. 🅭🅯🄎

The original picture was modified with artistic image filters.

Figure 4 on page 19 was taken from the book of Als-Nielsen et al. [22, fig. 1.1, p. 2]. The explicit license of *John Wiley and Sons* to reproduce the figure in this thesis (electronic and print) has the number `4000220630663`.

Colors and labels were slightly modified.

Figure 5 on page 20 is reproduced from Hubbell et al. [26, fig. 1, p. 1024], with the permission of AIP Publishing (DOI: 10.1063/1.555629 ↗).

Small modifications were made to improve the readability and highlight the relevant energy region.

The title image of chapter 4 on page 31 was taken from mediaTUM with friendly permission (created by `Thorsten Naeser` ↗, Technical University of Munich).

The original picture was modified with artistic image filters.

Figure 10 on page 32 has a photograph in the background which was taken from Pixabay (provided by `Fotoworkshop4You` ↗) and is available under the Creative Commons license `CC 0` ↗. 🅭🄍

Figure 11 on page 33 was taken from the book of Hartley et al. [47, fig. 2.1, p. 29] with friendly permission of the authors.

Colors and labels were slightly modified.

The title image of chapter 5 on page 45 was taken from Flickr (provided by `GBPublic_PR`) and is available under the Creative Commons license `CC BY 2.0`.
The original picture was modified with artistic image filters.

Figures 14 and 15 on page 47 were taken from the website of Karl Rupp (posts `40 Years of Microprocessor Trend Data` [52] and `CPU, GPU and MIC Hardware Characteristics over Time` [56]) and are available under the Creative Commons license `CC BY 4.0`.
Colors, layout and scaling were adapted for this thesis.

The title image of chapter 6 on page 57 was taken from the website of the Boise State University with friendly permission (created by the `LEGO MSE LAB`).
The original picture was modified with artistic image filters.

The title image of chapter 7 on page 71 was taken from Wikipedia (name `A drivebelt nightmare` by `Brian Snelson`) and is available under the Creative Commons license `CC BY 2.0`.
The original picture was modified with artistic image filters.

The title image of chapter 8 on page 79 was taken from Wikipedia (name `Bending.jpg` by `Holotone`) and is available under the Creative Commons license `CC BY-SA 2.5`.
The original picture was modified with artistic image filters.

The title image of chapter 9 on page 95 was taken from Wikimedia Commons (name: `Stopwatch A.jpg` by `Wouterhagens`) and is available under the Creative Commons license `CC BY-SA 3.0`.
The original picture was modified with artistic image filters.

List of publications

Peer-reviewed articles

Co-authored

1. S. Allner, A. Gustschin, A. Fehringer, P. B. Noël, and F. Pfeiffer. “Metric-guided regularization parameter selection for statistical iterative reconstruction in computed tomography”. Submitted.
2. L. Birnbacher, M. Viermetz, W. Noichl, S. Allner, A. Fehringer, M. Marschner, M. von Teuffenbach, M. Willner, K. Achterhold, P. B. Noël, T. Koehler, J. Herzen, and F. Pfeiffer. “Tilted grating phase-contrast computed tomography using statistical iterative reconstruction”. In: *Scientific Reports* 8.6608 (Apr. 2018).
3. M. Müller, I. de Sena Oliveira, S. Allner, S. Ferstl, P. Bidola, K. Mechlem, A. Fehringer, L. Hehn, M. Dierolf, K. Achterhold, B. Gleich, J. U. Hammel, H. Jahn, G. Mayer, and F. Pfeiffer. “Myoanatomy of the velvet worm leg revealed by laboratory-based nanofocus X-ray source tomography”. In: *PNAS* 114.47 (Nov. 2017), pages 12378–12383.
4. M. von Teuffenbach, T. Koehler, A. Fehringer, M. Viermetz, B. Brendel, J. Herzen, R. Proksa, E. J. Rummeny, F. Pfeiffer, and P. B. Noël. “Grating-based phase-contrast and dark-field computed tomography: a single-shot method”. In: *Scientific Reports* 7.7476 (Aug. 2017).
5. K. Mei, F. K. Kopp, R. Bippus, T. Koehler, B. J. Schwaiger, A. S. Gersing, A. Fehringer, A. Sauter, D. Münzel, F. Pfeiffer, E. J. Rummeny, J. S. Kirschke, P. B. Noël, and T. Baum. “Is multidetector CT-based bone mineral density and quantitative bone microstructure assessment at the spine still feasible using ultra-low tube current and sparse sampling?” In: *European Radiology* 27.12 (Dec. 2017), pages 5261–5271.
6. P. Bidola, K. S. Morgan, M. Willner, A. Fehringer, S. Allner, F. Prade, F. Pfeiffer, and K. Achterhold. “Application of sensitive, high-resolution imaging at a commercial lab-based X-ray micro-CT system using propagation-based phase retrieval”. In: *Journal of Microscopy* 266.2 (Jan. 2017), pages 211–220.
7. S. Ehn, T. Sellerer, K. Mechlem, A. Fehringer, M. Epple, J. Herzen, F. Pfeiffer, and P. B. Noël. “Basis material decomposition in spectral CT using a semi-empirical, polychromatic adaption of the Beer-Lambert model”. In: *Physics in Medicine and Biology* 62.1 (Dec. 2016), N1.
8. M. Marschner, L. Birnbacher, K. Mechlem, W. Noichl, A. Fehringer, M. Willner, K. Scherer, J. Herzen, P. B. Noël, and F. Pfeiffer. “Two-shot X-ray dark-field imaging”. In: *Optics Express* 24.23 (Nov. 2016), pages 27032–27045.
9. C. Jud, F. Schaff, I. Zanette, J. Wolf, A. Fehringer, and F. Pfeiffer. “Dentinal tubules revealed with X-ray tensor to-

mography". In: *Dental Materials* 32.9 (July 2016), pages 1189 –1195.

10. S. Allner, T. Koehler, A. Fehringer, L. Birnbacher, M. Willner, F. Pfeiffer, and P. B. Noël. "Bilateral filtering using the full noise covariance matrix applied to x-ray phase-contrast computed tomography". In: *Physics in Medicine and Biology* 61.10 (Apr. 2016), page 3867.
11. M. Marschner, M. Willner, G. Potdevin, A. Fehringer, P. B. Noël, F. Pfeiffer, and J. Herzen. "Helical X-ray phase-contrast computed tomography without phase stepping". In: *Scientific Reports* 6.23953 (Apr. 2016).
12. S. Ehn, M. Epple, A. Fehringer, D. Pennicard, H. Graafsma, P. Noël, and F. Pfeiffer. "X-ray deconvolution microscopy". In: *Biomed. Opt. Express* 7.4 (Apr. 2016), pages 1227–1239.
13. A. Velroyen, A. Yaroshenko, D. Hahn, A. Fehringer, A. Tapfer, M. Müller, P. B. Noël, B. Pauwels, A. Sasov, A. Ö. Yildirim, O. Eickelberg, K. Hellbach, S. D. Auweter, F. G. Meinel, M. F. Reiser, M. Bech, and F. Pfeiffer. "Grating-based X-ray Dark-field Computed Tomography of Living Mice". In: *EBioMedicine* (Aug. 2015).
14. J. Vogel, F. Schaff, A. Fehringer, C. Jud, M. Wieczorek, F. Pfeiffer, and T. Lasser. "Constrained X-ray tensor tomography reconstruction". In: *Optics Express* 23.12 (June 2015), pages 15134–15151.
15. D. Hahn, P. Thibault, A. Fehringer, M. Bech, T. Koehler, F. Pfeiffer, and P. B. Noël. "Statistical iterative reconstruction algorithm for X-ray phase-contrast CT". In: *Scientific reports* 5.10452 (June 2015).
16. R. A. Nasirudin, K. Mei, P. Panchev, A. Fehringer, F. Pfeiffer, E. J. Rummeny, M. Fiebich, and P. B. Noël. "Reduction of Metal Artifact in Single Photon-Counting Computed Tomography by Spectral-Driven Iterative Reconstruction Technique". In: *PLoS ONE* 10.5 (May 2015), e0124831.
17. M. Stockmar, M. Hubert, M. Dierolf, B. Enders, R. Clare, S. Allner, A. Fehringer, I. Zanette, J. Villanova, J. Laurencin, P. Cloetens, F. Pfeiffer, and P. Thibault. "X-ray nanotomography using near-field ptychography". In: *Optics Express* 23.10 (May 2015), pages 12720–12731.
18. K. Burger, T. Koehler, M. Chabior, S. Allner, M. Marschner, A. Fehringer, M. Willner, F. Pfeiffer, and P. B. Noël. "Regularized iterative integration combined with non-linear diffusion filtering for phase-contrast x-ray computed tomography". In: *Optics Express* 22.26 (Dec. 2014), pages 32107–32118.

Conference proceedings

First-authored

19. A. Fehringer, K. Mechlem, M. Epple, S. Allner, F. Lorenz Hehn Pfeiffer, and P. B. Noël. "Ultra-fast cone-beam SIR on 2k-cubed data". In: *International Conference on Image Formation in X-Ray Computed Tomography*. Volume 4. July 2016, pages 129–132.
20. A. Fehringer, B. Brendel, D. Hahn, P. B. Noël, F. Pfeiffer, and T. Koehler. "Performance evaluation of OS-SPS and CG for differential phase-contrast

X-ray tomography”. In: *International Conference on Image Formation in X-Ray Computed Tomography*. Volume 3. June 2014, pages 198–202.

21. A. Fehringer, T. Lasser, I. Zanette, P. B. Noël, and F. Pfeiffer. “A versatile tomographic forward- and backprojection approach on Multi-GPUs”. In: *SPIE Medical Imaging*. Volume 9034. International Society for Optics and Photonics. Mar. 2014, 90344F–1–90344F–7.

Co-authored

22. S. Allner, K. Mechlem, N. Gustschin, A. Fehringer, F. Pfeiffer, and P. B. Noël. “Virtual equivalent thickness calibration for X-ray computed tomography”. In: *Workshop on High Performance Image Reconstruction (Fully 3 D)*. Volume 14. June 2017, pages 547–550.
23. B. J. Schwaiger, K. Mei, F. K. Kopp, R. Bippus, T. Koehler, A. S. Gersing, A. Fehringer, F. Pfeiffer, E. J. Rummeny, J. S. Kirschke, P. B. Noël, and T. Baum. “Dose reduction in MDCT-based bone mineral density and microstructure assessment: effects of low-dose simulation and sparse sampling”. In: *Insights into Imaging*. Volume 8. European Society of Radiology. Mar. 2017.
24. K. Mei, F. K. Kopp, A. Fehringer, F. Pfeiffer, E. J. Rummeny, J. S. Kirschke, P. B. Noël, and T. Baum. “Effects of Sparse Sampling in combination with Iterative Reconstruction on Quantitative Bone Microstructure Assessment”. In: *SPIE Medical Imaging*. Volume 10132. International Society for Optics and Photonics. Feb. 2017, pages 1013244–1013244–4.
25. M. Viermetz, L. Birnbacher, A. Fehringer, M. Willner, P. B. Noël, F. Pfeiffer, and J. Herzen. “High Resolution Laboratory Grating-Based X-Ray Phase-Contrast CT”. In: *SPIE Medical Imaging*. Volume 10132. International Society for Optics and Photonics. Feb. 2017, 101325K–101325K–6.
26. M. von Teuffenbach, B. Brendel, A. Fehringer, P. B. Noël, F. Pfeiffer, and T. Koehler. “Iterative Reconstruction of Sliding Window PCCT”. In: *International Workshop on X-ray and Neutron Phase Imaging with Gratings (XNPIG)*. Volume 3. International Society for Optics and Photonics. Sept. 2015, page 58.
27. M. Marschner, L. Birnbacher, M. Willner, A. Fehringer, J. Herzen, P. B. Noël, and F. Pfeiffer. “Enabling low dose scans in x-ray phase-contrast CT”. In: *International Workshop on X-ray and Neutron Phase Imaging with Gratings (XNPIG)*. Volume 3. International Society for Optics and Photonics. Sept. 2015, page 57.
28. C. Jud, F. Schaff, I. Zanette, J. Wolf, A. Fehringer, and F. Pfeiffer. “Dentinal tubules revealed with X-ray Tensor Tomography”. In: *International Workshop on X-ray and Neutron Phase Imaging with Gratings (XNPIG)*. Volume 3. International Society for Optics and Photonics. Sept. 2015, page 31.
29. F. K. Kopp, R. A. Nasirudin, K. Mei, A. Fehringer, F. Pfeiffer, E. J. Rummeny, and P. B. Noël. “ROI-Aufbereitung für iterative Rekonstruktionsalgorithmen in der Computertomographie”. In: *Jahrestagung der*

Deutschen Gesellschaft für Medizinische Physik (DGMP). Volume 46. Sept. 2015, pages 453–455.

30. S. Allner, A. Fehringer, J. Schock, F. Pfeiffer, and P. B. Noël. “Dual-band projection alignment applied to X-ray microscopy”. In: *International Conference on Image Formation in X-Ray Computed Tomography*. Volume 4. July 2016, pages 515–518.
31. M. von Teuffenbach, B. Brendel, A. Fehringer, P. B. Noël, F. Pfeiffer, and T. Koehler. “Iterative Reconstruction of Grating-based PCCT Without Phase-Stepping”. In: *International Conference on Image Formation in X-Ray Computed Tomography*. Volume 4. July 2016, pages 367–370.
32. M. Marschner, L. Birnbacher, M. Willner, M. Chabior, A. Fehringer, J. Herzen, P. B. Noël, and F. Pfeiffer. “Redefining the lower statistical limit in x-ray phase-contrast imaging”. In: *SPIE Medical Imaging*. Volume 9412. International Society for Optics and Photonics. Mar. 2015, pages 94120M–94120M–6.
33. B. Brendel, M. von Teuffenbach, A. Fehringer, P. B. Noël, F. Pfeiffer, and T. Koehler. “Intensity-Based Iterative Reconstruction for Differential Phase-Contrast Imaging with Reconstruction Parameter Estimation”. In: *Workshop on High Performance Image Reconstruction (Fully 3 D)*. Volume 13. June 2015, pages 713–716.
34. K. Mei, A. Valentinitsch, F. K. Kopp, A. Fehringer, F. Pfeiffer, E. J. Rummeny, J. S. Bauer, and P. B. Noël. “Iterative CT Image Reconstruction using 3D Dictionary Learning”. In: *Workshop on High Performance Image Reconstruction (Fully 3 D)*. Volume 13. June 2015, pages 130–133.
35. S. Allner, A. Fehringer, A. Velroyen, F. Pfeiffer, and P. B. Noël. “Statistical iterative reconstruction for ultra high-resolution x-ray tomography from undersampled data”. In: *Workshop on High Performance Image Reconstruction (Fully 3 D)*. Volume 13. June 2015, pages 614–617.
36. M. Marschner, L. Birnbacher, M. Willner, M. Chabior, A. Fehringer, J. Herzen, P. B. Noël, and F. Pfeiffer. “Redefining the lower statistical limit in x-ray phase-contrast computed imaging”. In: *SPIE Medical Imaging*. Volume 9412. International Society for Optics and Photonics. Mar. 2015, pages 94120M–94120M–6.
37. S. Allner, A. Velroyen, A. Fehringer, F. Pfeiffer, and P. B. Noël. “Statistical iterative reconstruction for multi-contrast x-ray micro-tomography”. In: *SPIE Medical Imaging*. Volume 9412. International Society for Optics and Photonics. Mar. 2015, 94123O–94123O–6.
38. F. K. Kopp, R. A. Nasirudin, K. Mei, A. Fehringer, F. Pfeiffer, E. J. Rummeny, and P. B. Noël. “Region of interest processing for iterative reconstruction in x-ray computed tomography”. In: *SPIE Medical Imaging*. Volume 9412. International Society for Optics and Photonics. Mar. 2015, 94122E–94122E–7.
39. L. Birnbacher, M. Viermetz, M. Marschner, S. Allner, A. Fehringer, M. Willner, F. Pfeiffer, and J. Herzen. “Tilted grating laboratory phase-

contrast computed tomography using statistical iterative reconstruction". In: *International Symposium on Biomedical Applications of X-Ray Phase Contrast Imaging (IMXP)*. Volume 5. Jan. 2015.

40. S. Allner, T. Koehler, A. Fehringer, M. Willner, F. Pfeiffer, and P. B. Noël. "Bilateral filtering for x-ray phase-contrast imaging". In: *International Conference on Image Formation in X-Ray Computed Tomography*. Volume 3. June 2014, pages 388–392.
41. M. Marschner, M. Willner, A. Fehringer, J. Herzen, P. B. Noël, and F. Pfeiffer. "Continuous helical X-ray phase-contrast computed tomography without phase-stepping". In: *International Symposium on Biomedical Applications of X-Ray Phase Contrast Imaging (IMXP)*. Volume 4. Jan. 2014.
42. M. Marschner, D. Hahn, M. Willner, G. Fior, A. Fehringer, M. Chabior, P. B. Noël, and F. Pfeiffer. "Towards Lower Dose - Statistical Iterative Reconstruction for Phase-Contrast Computed Tomography". In: *International Workshop on X-ray and Neutron Phase Imaging with Gratings (XNPIG)*. Volume 2. Jan. 2014, page 39.
43. D. Hahn, P. Thibault, A. Fehringer, M. Bech, P. B. Noël, and F. Pfeiffer. "Bone artifact reduction in differential phase-contrast CT". In: *Workshop on High Performance Image Reconstruction (Fully 3 D)*. Volume 12. June 2013, pages 416–419.
44. S. Allner, A. Fehringer, D. Hahn, T. Lasser, I. Zanette, P. B. Noël, and F. Pfeiffer. "Improving Ptychographic Imaging using Iterative Tomographic Algorithms". In: *International workshop on the state and the future of ptychography*. Volume 1. May 2013.

Oral presentations

45. A. Fehringer, K. Mechlem, M. Epple, S. Allner, P. B. Noël, and F. Pfeiffer. *Gamer graphics boards for science - how to make iterative tomographic reconstruction get a move on*. At the IMETUM seminar of the Technical University Munich. Apr. 2016.
46. A. Fehringer, S. Allner, P. B. Noël, and F. Pfeiffer. *pyCT - an introduction to iterative tomographic reconstruction*. Within the cooperation with Carl Zeiss X-ray Microscopy at the Technical University Munich. Jan. 2016.
47. A. Fehringer, K. Mechlem, M. Epple, S. Allner, P. B. Noël, and F. Pfeiffer. *OpenCL and its application in iterative tomographic reconstruction*. At the seminar of the theoretical physics division of the Werner-Heisenberg-Institut, Munich. Nov. 2015.
48. A. Fehringer, S. Allner, P. B. Noël, and F. Pfeiffer. *The pyCT reconstruction framework*. At the Development Center for X-ray Technology (EZRT) of the Fraunhofer Institute IIS, Fürth. July 2015.
49. A. Velroyen, A. Fehringer, P. B. Noël, and F. Pfeiffer. *Research and Reconstruction @ Chair for Biomedical Physics, TUM*. Within the cooperation with Planmeca, Helsinki, Finnland. May 2015.

50. A. Fehringer, P. B. Noël, and F. Pfeiffer. *A versatile tomographic forward and back projection approach on multi-GPUs*. At the TUM Reconstruction Workshop, Technical University Munich. June 2014.
51. A. Fehringer, P. B. Noël, and F. Pfeiffer. *A versatile tomographic forward and back projection approach on multi-GPUs*. At the Interdisciplinary Cluster Workshop on GPUs, TUM Excellence Center, Technical University Munich. Apr. 2014.

Theses

52. A. Fehringer. “Advanced Algorithms for Ptychographic X-Ray Computed Tomography”. Diploma thesis. Technical University Munich, Nov. 2011.

Accompanied

53. L. Hehn. “High-Performance Algorithms for Improved Reconstruction of X-Ray Computed Tomography”. Master thesis. Technical University Munich, Oct. 2015.
54. J. Schwarz. “Dual-Energy Micro-CT”. Master thesis. Technical University Munich, Oct. 2014.
55. S. Allner. “Local tomography alignment and bilateral filter postprocessing for phase-contrast imaging”. Master thesis. Technical University Munich, Nov. 2013.
56. M. Reis. “GPU-accelerated Fourier Filtering for Filtered Backprojection in Computed Tomography”. Bachelor thesis. Technical University Munich, Aug. 2013.

Awards

The idea to spin off parts of this work into a startup in 2014 and a poster presentation in 2013 were honored with an award. We thank the responsible committees.

57. A. Velroyen, A. Fehringer, P. B. Noël, and F. Pfeiffer. *TUM IdeaAward 2014*. Presented by the Technical University of Munich together with the Zeidler-Forschungs-Stiftung and the UnternehmerTUM. Feb. 2014.
58. S. Allner and A. Fehringer. *Best Poster Award*. Presented by the committee of the international workshop on the state and the future of ptychography. May 2013.

Bibliography

[1] The Apostle Matthew. “Gospel of Matthew”. In: The Bible, New International Version, 2011. Chapter 19:21-26 (cited on page 4).

[2] M. Born. *The Born Einstein Letters*. Translated by Irene Born. The context was not religious but a discussion about the non-deterministic elements in quantum mechanics. Macmillan Press, 1971, page 149 (cited on page 4).

[3] The Apostle John. “Gospel of John”. In: The Bible, New International Version, 2011. Chapter 3:16 (cited on page 5).

[4] The Apostle Paul. “Letter to the Ephesians”. In: The Bible, New International Version, 2011. Chapter 2:8-9 (cited on page 5).

[5] The Apostle John. “John’s Gospel”. In: The Bible, New International Version, 2011. Chapter 8:12.31-32 (cited on page 5).

[6] Bundesministerium für Bildung und Forschung. “Luther hat ein völlig neues Bildungsethos geschaffen”. In: *bmbf.de* (2017) (cited on page 6).

[7] R. Stark. *For the Glory of God: How Monotheism Led to Reformations, Science, Witch-Hunts, and the End of Slavery*. Princeton University Press, 2004 (cited on page 6).

[8] The Apostle John. “John’s Gospel”. In: The Bible, New International Version, 2011. Chapter 1:14 (cited on page 6).

[9] R. A. Carhart and A. Cenian. “Implications of proven limits on scientific knowledge: Gödel’s proof, quantum uncertainty, chaos theory and specified complexity of information theory”. In: *Bulletin de la Société des Sciences et des Lettres de Łódź* 59 (2009), pages 7–18 (cited on page 6).

[10] W. C. Röntgen. “Über eine neue Art von Strahlen”. In: *Annalen der Physik* 300.1 (1898), pages 1–11 (cited on page 9).

[11] G. N. Hounsfield. “Computed Medical Imaging”. In: *Journal of Computer Assisted Tomography* 4.5 (1979). Nobel Lecture, Dec 8, 1979, pages 665–674 (cited on page 10).

[12] J. C. Elliott and S. D. Dover. “X-ray microtomography”. In: *Journal of Microscopy* 126.2 (1982), pages 211–213 (cited on page 10).

[13] R. Gordon, R. Bender, and G. T. Herman. “Algebraic Reconstruction Techniques (ART) for three-dimensional electron microscopy and X-ray photography”. In: *Journal of Theoretical Biology* 29.3 (1970), pages 471–481 (cited on pages 10, 26).

[14] R. A. Brooks and G. Di Chiro. “Theory of Image Reconstruction in Computed Tomography”. In: *Radiology* 117.3 (1975), pages 561–572 (cited on pages 10, 26).

[15] P. B. Noël, B. Renger, M. Fiebich, D. Münzel, A. A. Fingerle, E. J. Rummeny, and M. Dobritz. “Does Iterative Reconstruction Lower CT Radiation Dose: Evaluation of 15,000 Examinations”. In: *PLoS ONE* 8.11 (Nov. 2013), e81141 (cited on page 10).

[16] M. Kachelrieß. *CT-Technik*. Talk. Deutsches Krebsforschungszentrum (DKFZ) Heidelberg, 2014 (cited on page 10).

[17] J. A. Fessler. “Statistical image reconstruction methods for transmission tomography”. In: *Handbook of Medical Imaging, Volume 2: Medical Image Processing and Analysis*. Edited by M. Sonka and J. M. Fitzpatrick. Bellingham: SPIE, 2000, pages 1–70 (cited on pages 10, 26, 102).

[18] S. I. Tilley, J. H. Siewerdsen, and J. W. Stayman. “Model-based iterative reconstruction for flat-panel cone-beam CT with focal spot blur, detector blur, and correlated noise”. In: *Physics in Medicine & Biology* 61.1 (2016), page 296 (cited on pages 10, 102).

[19] L. Ritschl, S. Sawall, M. Knaup, A. Hess, and M. Kachelrieß. “Iterative 4D cardiac micro-CT image reconstruction using an adaptive spatio-temporal sparsity prior”. In: *Physics in Medicine and Biology* 57.6 (2012), page 1517 (cited on page 10).

[20] Y. Long and J. A. Fessler. “Multi-Material Decomposition Using Statistical Image Reconstruction for Spectral CT”. In: *IEEE Transactions on Medical Imaging* 33.8 (Aug. 2014), pages 1614–1626 (cited on page 10).

[21] A. Löve, M. L. Olsson, R. Siemund, F. Stålhammar, I. M. Björkman-Burtscher, and M. Söderberg. “Six iterative reconstruction algorithms in brain CT: a phantom study on image quality at different radiation dose levels”. In: *The British Journal of Radiology* 86.1031 (2013), page 20130388 (cited on page 10).

[22] J. Als-Nielsen and D. McMorrow. *Elements of Modern X-ray Physics*. Volume 2. John Wiley & Sons, Inc., 2011 (cited on pages 18, 19, 120).

[23] O. Hemberg, M. Otendal, and H. M. Hertz. “Liquid-metal-jet anode electron-impact x-ray source”. In: *Applied Physics Letters* 83.7 (2003), pages 1483–1485 (cited on page 18).

[24] R. J. Loewen. “A Compact Light Source: Design and Technical Feasibility Study of a Laser-Electron Storage Ring X-Ray Source”. SLAC-Report-632. PhD thesis. Stanford University, Stanford CA, USA, 2003 (cited on page 19).

[25] E. Eggl, M. Dierolf, K. Achterhold, C. Jud, B. Günther, E. Braig, B. Gleich, and F. Pfeiffer. “The Munich Compact Light Source: initial performance measures”. In: *Journal of Synchrotron Radiation* 23.5 (Sept. 2016), pages 1137–1142 (cited on page 19).

[26] J. H. Hubbell, H. A. Gimm, and I. Øverbø. “Pair, Triplet, and Total Atomic Cross Sections (and Mass Attenuation Coefficients) for 1 MeV - 100 GeV Photons in Elements Z=1 to 100”. In: *Journal of Physical and Chemical Reference Data* 9.4 (1980), pages 1023–1148 (cited on pages 20, 120).

[27] A. Momose. “Phase-sensitive imaging and phase tomography using X-ray interferometers”. In: *Opt. Express* 11.19 (2003), pages 2303–2314 (cited on page 20).

[28] F. Pfeiffer, T. Weitkamp, O. Bunk, and C. David. “Phase retrieval and differential phase-contrast imaging with low-brilliance X-ray sources”. In: *Na-*

ture physics 2.4 (2006), pages 258–261 (cited on page 20).

[29] P. Willmott. “An Introduction to Synchrotron Radiation : Techniques and Applications”. In: John Wiley & Sons, Inc., 2011. Chapter Detectors, pages 113–127 (cited on page 21).

[30] S. M. Gruner. “X-ray imaging detectors”. In: *Physics Today* 65.12 (Dec. 2012), pages 29–34 (cited on page 21).

[31] E. Roessl and R. Proksa. “K-edge imaging in x-ray computed tomography using multi-bin photon counting detectors”. In: *Physics in Medicine and Biology* 52.15 (July 2007), page 4679 (cited on page 21).

[32] A. C. Kak and M. Slaney. *Principles of computerized tomographic imaging*. Classics in applied mathematics. IEEE Press, 1988 (cited on pages 21, 22, 24, 25, 34).

[33] H. Turbell. “Cone-Beam Reconstruction Using Filtered Backprojection”. PhD thesis. Linköping University, Sweden, 2001 (cited on pages 21, 22, 24, 25).

[34] J. K. A. Radon. “Über die Bestimmung von Funktionen durch ihre Integralwerte längs gewisser Mannigfaltigkeiten”. In: *Berichte über die Verhandlungen der Königlich-Sächsischen Akademie der Wissenschaften zu Leipzig, Mathematisch-Physische Klasse*. Volume 69. Teubner, 1917, pages 262–277 (cited on page 22).

[35] H. K. Tuy. “An Inversion Formula for Cone-Beam Reconstruction”. In: *SIAM Journal on Applied Mathematics* 43.3 (1983), pages 546–552 (cited on page 24).

[36] L. A. Feldkamp, L. C. Davis, and J. W. Kress. “Practical cone-beam algorithm”. In: *J. Opt. Soc. Am. A* 1.6 (June 1984), pages 612–619 (cited on pages 24, 25).

[37] H. Scherl, M. Kowarschik, H. G. Hofmann, B. Keck, and J. Hornegger. “Evaluation of state-of-the-art hardware architectures for fast cone-beam CT reconstruction”. In: *Parallel Computing* 38.3 (2012), pages 111 –124 (cited on pages 24, 35).

[38] D. L. Parker. “Optimal short scan convolution reconstruction for fan beam CT”. In: *Medical Physics* 9.2 (1982), pages 254–257 (cited on page 25).

[39] B. De Man and J. A. Fessler. “Statistical iterative reconstruction for x-ray computed tomography”. In: *Biomedical Mathematics: Promising Directions in Imaging, Therapy Planning and Inverse Problems*. Edited by Y. Censor, M. Jiang, and G. Wang. Madison, WI: Medical Physics Publishing, 2000. Chapter 7 (cited on page 26).

[40] J. Nuyts, B. De Man, J. A. Fessler, W. Zbijewski, and F. J. Beekman. “Modelling the physics in the iterative reconstruction for transmission computed tomography”. In: *Physics in Medicine and Biology* 58.12 (2013), R63 (cited on page 27).

[41] J. R. Shewchuk. *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*. Technical report. Carnegie Mellon University, Pittsburgh PA, USA, 1994 (cited on page 28).

[42] H. Erdoğan. “Statistical Image Reconstruction Algorithms Using Paraboloidal Surrogates for PET Transmission Scans”. PhD thesis. University of Michigan, Ann Arbor MI, USA, 1999 (cited on pages 28, 29).

[43] H. Erdoğan and J. A. Fessler. “Ordered subsets algorithms for transmission tomography”. In: *Physics in Medicine and Biology* 44.11 (1999), page 2835 (cited on pages 28, 29).

[44] D. Kim, S. Ramani, and J. A. Fessler. “Combining ordered subsets and momentum for accelerated X-ray CT image reconstruction”. In: *IEEE transactions on medical imaging* 34.1 (2015), pages 167–178 (cited on pages 29, 87, 88, 110).

[45] D. Kim and J. A. Fessler. “Optimized first-order methods for smooth convex minimization”. In: *Mathematical Programming* 159.1 (2016), pages 81–107 (cited on pages 29, 80).

[46] A. F. Möbius. “Ueber die Zusammensetzung gerader Linien und eine daraus entspringende neue Begründungsweise des barycentrischen Calculs”. In: *Crelle’s Journal* 28 (1844), pages 1–9 (cited on page 31).

[47] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Second. Cambridge University Press, 2004 (cited on pages 32, 33, 36, 41, 114, 120).

[48] R. R. Galigekere, K. Wiesent, and D. W. Holdsworth. “Cone-beam reprojection using projection-matrices”. In: *IEEE Transactions on Medical Imaging* 22.10 (2003), pages 1202–1214 (cited on page 33).

[49] P. M. Joseph. “An Improved Algorithm for Reprojecting Rays through Pixel Images”. In: *IEEE Transactions on Medical Imaging* 1.3 (Nov. 1982), pages 192–196 (cited on pages 35, 62).

[50] A. Malecki, G. Potdevin, T. Biernath, E. Eggl, K. Willer, T. Lasser, J. Maisenbacher, J. Gibmeier, A. Wanner, and F. Pfeiffer. “X-ray tensor tomography”. In: *EPL* 105 (2014), page 38002 (cited on page 39).

[51] L. Hehn. “High-Performance Algorithms for Improved Reconstruction of X-Ray Computed Tomography”. Master thesis. Technical University Munich, Oct. 2015 (cited on pages 42, 64).

[52] K. Rupp. *40 Years of Microprocessor Trend Data*. Post on website. June 2015 (cited on pages 46, 121).

[53] S. P. E. C. (SPEC). *CINT2006 (Integer Component of SPEC CPU2006)*. Aug. 2006 (cited on page 46).

[54] G. M. Amdahl. “Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities”. In: *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*. AFIPS ’67 (Spring). New York, NY, USA: ACM, 1967, pages 483–485 (cited on page 46).

[55] J. L. Gustafson. “Reevaluating Amdahl’s Law”. In: *Commun. ACM* 31.5 (May 1988), pages 532–533 (cited on page 46).

[56] K. Rupp. *CPU, GPU and MIC Hardware Characteristics over Time*. Post on website. Aug. 2016 (cited on pages 47, 121).

[57] The Open MPI Team. *Open Message Passing Interface (Open MPI)*. Open-source software. Since 2004 (cited on page 47).

[58] University of California, Berkeley. *Berkeley Open Infrastructure for Network Computing (BOINC)*. Open-source software. Since 2002 (cited on page 47).

[59] The OpenMP Architecture Review Board. *Open Multi-Processing (OpenMP)*. Specification. Since 1997 (cited on page 47).

[60] IEEE and The Open Group for Unix. *Portable Operating System Interface (POSIX)*. Specification. Norm ISO/IEC/IEEE 9945. Since 1985 (cited on page 47).

[61] The Khronos Group. *Open Computing Language (OpenCL)*. Specification. Since 2008 (cited on pages 47, 49).

[62] Nvidia. *CUDA*. Free software. Since 2007 (cited on page 47).

[63] M. Buxton. *Haswell New Instruction Descriptions Now Available!* Technical report. Intel, June 2011 (cited on page 48).

[64] *ctypes – A foreign function library for Python*. Python Software Foundation. Delaware, USA (cited on page 48).

[65] *Python/C API Reference Manual*. Python Software Foundation. Delaware, USA (cited on page 48).

[66] *The OpenCL Specification, Version 2.0*. Rev. 29. The Khronos Group. Beaverton OR, USA, July 2015 (cited on page 49).

[67] *OpenCL 2.0 Reference Card*. Rev. 1115. The Khronos Group. Beaverton OR, USA, July 2015 (cited on page 49).

[68] *AMD APP SDK OpenCL User Guide*. Rev. 1.0. Advanced Micro Devices (AMD). Sunnyvale CA, USA, Aug. 2015 (cited on page 49).

[69] *OpenCL Programming Guide for the CUDA Architecture*. 4.1. Obsolate, but a good introduction. Nvidia. Santa Clara CA, USA, Mar. 2012 (cited on page 49).

[70] *CUDA C Programming Guide*. 8.0. Nvidia. Santa Clara CA, USA, Sept. 2016 (cited on pages 49, 50).

[71] Nvidia. *GeForce GTX TITAN X*. Specification. Mar. 2015 (cited on page 50).

[72] *AMD APP SDK OpenCL Optimization Guide*. Rev. 1.0. Advanced Micro Devices (AMD). Sunnyvale CA, USA, Aug. 2015 (cited on page 50).

[73] B. Cabral, N. Cam, and J. Foran. “Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware”. In: *Proceedings of the 1994 Symposium on Volume Visualization*. VVS ’94. Tysons Corner, Virginia, USA: ACM, 1994, pages 91–98 (cited on page 58).

[74] F. Xu and K. Mueller. “Accelerating popular tomographic reconstruction algorithms on commodity PC graphics hardware”. In: *IEEE Transactions on Nuclear Science* 52.3 (June 2005), pages 654–663 (cited on page 58).

[75] A. Fehringer, T. Lasser, I. Zanette, P. B. Noël, and F. Pfeiffer. “A versatile tomographic forward- and back-projection approach on multi-GPUs”. In: *Proceedings of SPIE* 9034

(2014), 90344F–90344F–7 (cited on pages 58, 64, 65).

[76] R. Sampson, M. G. McGaffin, T. F. Wenisch, and J. A. Fessler. “Investigating Multi-threaded SIMD for Helical CT Reconstruction on a CPU”. In: *International Conference on Image Formation in X-Ray Computed Tomography*. Volume 4. July 2016, pages 275–278 (cited on page 58).

[77] C. Cullinan, C. Wyant, and T. Frattesi. *Computing Performance Benchmarks among CPU, GPU, and FPGA*. Technical report. Sponsored by MathWorks. Worcester Polytech Institute, 2012 (cited on page 59).

[78] M. Reis. “GPU-accelerated Fourier Filtering for Filtered Backprojection in Computed Tomography”. Bachelor thesis. Technical University Munich, Aug. 2013 (cited on page 59).

[79] T. Lovelace. “CMake: The Cross Platform Build System”. In: *Linux Magazine* (June 2006) (cited on page 59).

[80] B. De Man and S. Basu. “Distance-driven projection and backprojection in three dimensions”. In: *Physics in Medicine and Biology* 49.11 (2004), page 2463 (cited on page 62).

[81] Y. Long, J. A. Fessler, and J. M. Balter. “3D Forward and Back-Projection for X-Ray CT Using Separable Footprints”. In: *IEEE Transactions on Medical Imaging* 29.11 (Nov. 2010), pages 1839–1850 (cited on page 62).

[82] R.-D. Bippus, T. Koehler, F. Bergner, B. Brendel, E. Hansis, and R. Proksa. “Projector and Backprojector for Iterative CT Reconstruction with Blobs using CUDA”. In: *Workshop on High Performance Image Reconstruction (Fully 3D)*. Volume 3. July 2011, pages 68–71 (cited on page 62).

[83] S. Ha, H. Li, and K. Mueller. “Efficient Area-Based Ray Integration Using Summed Area Tables and Regression Models”. In: *International Conference on Image Formation in X-Ray Computed Tomography*. Volume 4. July 2016, pages 507–510 (cited on page 62).

[84] A. Fehringer. “Advanced Algorithms for Ptychographic X-Ray Computed Tomography”. Diploma thesis. Technical University Munich, Nov. 2011 (cited on page 62).

[85] V. Y. Panin, G. L. Zeng, and G. T. Gullberg. “Total variation regulated EM algorithm”. In: *IEEE Nuclear Science Symposium Conference Record*. Volume 3. 1998, pages 1562–1566 (cited on page 63).

[86] Advanced Micro Devices (AMD). *clFFT*. Open-source software. Since 2013 (cited on page 64).

[87] A. Fehringer, K. Mechlem, M. Epple, S. Allner, L. Hehn, F. Pfeiffer, and P. B. Noël. “Ultra-fast cone-beam SIR on 2k-cubed data”. In: *International Conference on Image Formation in X-Ray Computed Tomography*. Volume 4. July 2016, pages 129–132 (cited on pages 64, 82).

[88] J. Y. Vaishnav, W. C. Jung, L. M. Popescu, R. Zeng, and K. J. Myers. “Objective assessment of image quality and dose reduction in CT iterative reconstruction”. In: *Medical Physics* 41.7 (2014) (cited on pages 73, 96).

[89] B. Brendel and T. Koehler. “Penalty weighting for statistical iterative CT reconstruction”. In: *IEEE Nuclear Sci-*

ence Symposuim Medical Imaging Conference. Volume 17. Oct. 2010, pages 3475–3478 (cited on page 73).

[90] J. H. Cho and J. A. Fessler. “Regularization Designs for Uniform Spatial Resolution and Noise Properties in Statistical Image Reconstruction for 3-D X-ray CT”. In: *IEEE Transactions on Medical Imaging* 34.2 (Feb. 2015), pages 678–689 (cited on page 73).

[91] F. Bergner, B. Brendel, P. Noel, M. Dobritz, and T. Koehler. “Robust Automated Regularization Factor Selection for Statistical Reconstructions”. In: *International Conference on Image Formation in X-Ray Computed Tomography*. Volume 2. 2012, pages 267–270 (cited on page 73).

[92] H. Dang, J. H. Siewerdsen, and J. W. Stayman. “Prospective regularization design in prior-image-based reconstruction”. In: *Physics in medicine and biology* 60.24 (2015), pages 9515–9536 (cited on pages 73, 102).

[93] H. Dang, J. W. Stayman, J. Xu, A. Sisniega, W. Zbijewski, X. Wang, D. H. Foos, N. Aygun, V. E. Koliatsos, and J. H. Siewerdsen. “Task-Based Regularization Design for Detection of Intracranial Hemorrhage in Cone-Beam CT”. In: *International Conference on Image Formation in X-Ray Computed Tomography*. Volume 4. July 2016, pages 557–560 (cited on page 73).

[94] B. Brendel, M. von Teuffenbach, A. Fehringer, P. B. Noël, F. Pfeiffer, and T. Koehler. “Intensity-Based Iterative Reconstruction for Differential Phase-Contrast Imaging with Reconstruction Parameter Estimation”. In: *Workshop on High Performance Image Reconstruction (Fully 3 D)*. Volume 13. June 2015, pages 713–716 (cited on page 75).

[95] B. Brendel, M. von Teuffenbach, P. B. Noël, F. Pfeiffer, and T. Koehler. “Penalized maximum likelihood reconstruction for x-ray differential phase-contrast tomography”. In: *Medical Physics* 43.1 (2016), pages 188–194 (cited on page 75).

[96] B. De Man, J. Nuyts, P. Dupont, G. Marchal, and P. Suetens. “An iterative maximum-likelihood polychromatic algorithm for CT”. In: *IEEE Transactions on Medical Imaging* 20.10 (Oct. 2001), pages 999–1008 (cited on pages 75, 102).

[97] K. Mechlem. “Advanced Statistical Iterative Reconstruction for X-ray Computed Tomography”. Master thesis. Technical University Munich, Dec. 2015 (cited on pages 75, 80).

[98] P. Charbonnier, L. Blanc-Feraud, G. Aubert, and M. Barlaud. “Two deterministic half-quadratic regularization algorithms for computed imaging”. In: *Proceedings of 1st International Conference on Image Processing*. Volume 2. Nov. 1994, 168–172 vol.2 (cited on page 75).

[99] Q. Xu, H. Yu, X. Mou, L. Zhang, J. Hsieh, and G. Wang. “Low-Dose X-ray CT Reconstruction via Dictionary Learning”. In: *IEEE Transactions on Medical Imaging* 31.9 (Sept. 2012), pages 1682–1697 (cited on page 75).

[100] A. Fehringer, K. Mechlem, M. Epple, S. Allner, L. Hehn, F. Pfeiffer, and P. B. Noël. “Real-Time Iterative Reconstruction on 2k-Cubed X-Ray

Micro-CT Images". Submitted (cited on page 82).

[101] I. Goddard, A. Berman, O. Bockenbach, F. Lauginiger, S. Schuberth, and S. Thieret. "Evolution of computer technology for fast cone beam backprojection". In: *SPIE Computational Imaging*. Volume 6498. International Society for Optics and Photonics. 2007, 64980R–64980R–8 (cited on page 83).

[102] T. Zinßer and B. Keck. "Systematic Performance Optimization of Cone-Beam Back-Projection on the Kepler Architecture". In: *Proceedings of the Fully 3D*. Volume 12. 2013, pages 225–228 (cited on pages 83, 86).

[103] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. "Image quality assessment: from error visibility to structural similarity". In: *IEEE Transactions on Image Processing* 13.4 (2004), pages 600–612 (cited on pages 88, 97).

[104] Z. Wang and A. C. Bovik. "Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures". In: *IEEE Signal Processing Magazine* 26.1 (2009), pages 98–117 (cited on pages 88, 97).

[105] *ISO/IEC 2382:2015: Information technology – Vocabulary*. Standard. Geneva, CH: International Organization for Standardization, May 2015 (cited on page 96).

[106] S. Allner, A. Fehringer, A. Velroyen, F. Pfeiffer, and P. B. Noël. "Statistical iterative reconstruction for ultra high-resolution x-ray tomography from undersampled data". In: *Workshop on High Performance Image Reconstruction (Fully 3D)*. Volume 13. June 2015, pages 614–617 (cited on page 96).

[107] W. K. Pratt. "Image Feature Extraction". In: *Digital Image Processing*. John Wiley & Sons, Inc., 2007. Chapter 16, page 539 (cited on page 97).

[108] Nvidia. *NVIDIA TITAN Xp*. Product website. Apr. 2017 (cited on page 102).