Chapter 1

Introduction

In November 1895, Wilhelm Conrad Röntgen acquired the world's first X-ray photograph, now known as a *radiograph*, on a piece of paper painted with barium platinum cyanide [1]. The exposure to the new type of light, leading to the famous picture of the bones in the hand of Anna Bertha Ludwig, took *about 15 minutes*. Today, more than 125 years later, modern X-ray imaging technology can capture such radiographs with framerates up to hundreds or thousands Hertz, almost a million times faster in comparison to Röntgen's original measurement.

Computed Tomography (CT) may very well be the most influential invention originating from Röntgen's discovery. Also known as "computerized (axial) *tomography*" (*tomos*: slice, *graphein*: to write), CT enables non-intrusive diagnostics of human subjects, as well as inspection and observation in the experimental sciences. The principle of CT is to compute a digital representation of a 3D interior (a human or object) via a set of radiographs. This set of radiographs is obtained via scanning the object from different angles. In 1973, Hounsfield's iterative algorithm, delivered on a Minicomputer with 32 kilobyte of memory in the first commercial CT scanner called the EMI, took *about 5 minutes* to transform 28,800 data points into a 80×80 polaroid of a cross-sectional slice of the patient's brain [2]. Nowadays, more than 50 years later, computing a CT image from 1 billion data points typically takes anywhere from *a second to several minutes* on a personal computer, depending on the algorithm used and the size of the data.

After all these technological advances, one may get the impression that CT reached its full potential or that it does not need to be much faster. Many medical and research applications using CT permit a few minutes of waiting time while algorithms are executed, for example, between the CT scan of a patient and the doctor's diagnosis—or, similarly, between the scan of an object in a laboratory and a researcher's evaluation. In this thesis, however, we will see that CT is not nearly as fast as we would like it to be. One aspect is the unrealized potential that *interaction with the scan* holds. When CT reconstructions would be repeated during a scan, for example, a scan could be terminated precisely when enough information is collected—reducing laboratory occupation or limiting the radiation exposure to the subject. Another motivation is given by the recent surge of data-driven algorithms, for example in dynamic tomography, which require many consecutive CT reconstructions to be computed efficiently. Even though CT by itself is fast, *faster* CT could enable a new category of real-time and dynamic tomographic applications.

This dissertation uses the properties of sequential and dynamic data in X-ray applications to increase the speed of CT algorithms. It discusses a set-up for fast data acquisition, numerical implementations for faster algorithms, but also explores how, with artificial intelligence, data sets can improve fast X-ray Computed Tomography algorithms in practical, real-world applications. This chapter first introduces the preliminaries and background, and then formulates the research questions associated with the subsequent chapter-by-chapter publications.

1.1 Primer on Computed Tomography

Computed Tomography (CT) is a non-invasive technique to form a three-dimensional (3D) image of a subject or object using X-ray radiation. To suit different applications, there exist many variations of the technique and it is broadly researched within the fields of mathematics, computer science, physics, and applied sciences. In all its variations, the essence of the technique is to solve what is known as an *inverse problem*. To get an impression of what this means, assume an opaque *Object* that we would like to obtain a 3D image of, as well as access to an *X-ray Set-up* that can be used to obtain *Radiographs* of the object:

$$\text{Object} \xrightarrow{\text{X-ray Set-up}} \text{Radiographs}. \tag{1.1}$$

In CT, the goal is to reverse this relation, i.e., to find a digitalized 3D representation of the unknown object interior via its radiographs:

$$\text{Radiographs} \xrightarrow{\text{CT Algorithm}} \text{Object}. \tag{1.2}$$

To recover the *Object*, the *CT Algorithm* needs to model the geometry of the X-ray set-up, and take into account the X-ray physics that lead to the radiographs. The inverse problem of X-ray CT is known to be ill-posed, meaning that a solution may not exist, may not be unique, and may be sensitive to noisy data. Computed Tomography is a prototypical example of an inverse problem in the area of medical imaging, due to its long history, extensive use, and the comparatively simple X-ray physics.

Absorption contrast tomography X-rays are a frequency range of the electromagnetic spectrum (i.e., a type of light) with a wavelength shorter than ultraviolet light. X-rays that pass through a medium can interact with its atoms in several ways, such as absorption (the photo-electric effect), scattering, or phase shifts. During absorption and inelastic scattering, the energy of X-rays is transferred to the atoms in the material, and this reduces the intensity of the X-ray beam. This principle is the basis from which most Computed Tomography algorithms are derived. Interactions that are not modeled can lead to noise or artifacts in the algorithm's outcome. The reader is referred to the textbook by Buzug [3] for an in-depth treatment of the X-ray physics.

A CT algorithm works via the use of multiple *absorption images* or *projections*, which are in fact just radiographs processed in a way that the image intensity relates linearly to absorption inside the object (this is discussed as the measurement principle hereafter). Each absorption image must be taken from a different angle of the object. Absorption images are generally monochromatic (i.e., in grayscale color), since energy-integrating X-ray detectors are unable to distinguish between the individual energies of photons.

The digital representation that is the outcome of a CT algorithm, called a *reconstruction volume*, is a 3D image describing the local X-ray absorption inside the object—indeed, when the materials inside it have similar X-ray absorption characteristics, this results in a low contrast 3D volume. With sufficient time and without limitations on radiation exposure, hundreds to thousands of angular measurements may be required to resolve small-scale details inside the object.

The measurement principle As mentioned, tomographic algorithms do not work with raw detector radiographs, but instead require an initial preprocessing of the radiographs given by the detector. The radiographs, in an idealized case, are described by Beer-Lambert's law: For a monochromatic pencil beam [3], i.e., a narrow and collimated X-ray source, it describes the relation between the radiation intensity I^0 at the source, and its decay through an object with attenuation μ along a ray l_i as

$$I_i = I_i^0 \exp\left(-\int_{l_i} \mu(\eta)\,\mathrm{d}\eta\right), \tag{1.3}$$

where η denotes the position along the ray, $\mu(\eta)$ the attenuation coefficient at η, and I_i the remaining intensity measured at the detector pixel.

X-ray intensities are described with a discrete number of *detector counts*. These values represent an equivalent of the photons measured in the pixel [4]. Assuming that the attenuation of air is negligible, measurements taken in air under the same experimental conditions form a *flatfield*, and can be used to obtain I^0. With I^0, the *absorption image* or *projection* is defined as

$$\log\left(\frac{I^0}{I}\right) = \int_{l_i} \mu(\eta)\,\mathrm{d}\eta. \tag{1.4}$$

Note that the projection is linearly related to $\mu(\eta)$, which describes all attenuating objects along a ray l_i.

Cone-beam CT As an example, one of the most common configurations of CT is presented, namely with a cone-beam source, i.e., a three-dimensional divergent X-ray beam, and a flat-panel X-ray detector. The source and detector describe a full circular trajectory

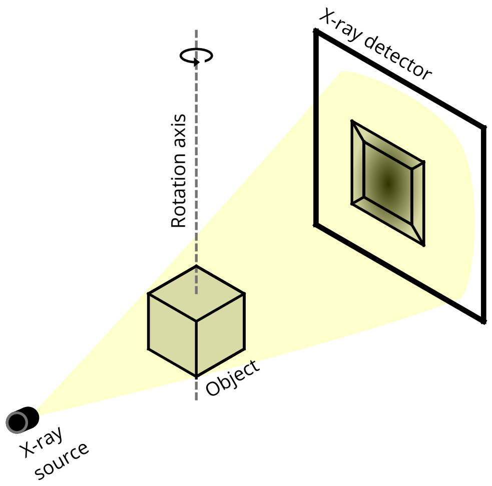


Figure 1.1: A CBCT set-up: Radiographic projections of the opaque interior of *Object* are acquired via illumination of the object with a divergent X-ray beam. The set-up or object is rotated around a central axis to obtain the radiographs from equidistant angles.

around the object to acquire an equiangular range of absorption images. This set-up geometry, which is the "forward problem" of equation 1.1, is termed circular *cone-beam CT* (CBCT) and visualized in figure 1.1.

The reader is referred to the textbooks of Hansen [4] and Kak & Slaney [5] for the mathematics and algorithms of tomography, in the following only a brief outline of the mathematics is presented. Mathematically, the inverse problem (equation 1.2) amounts to solving the linear equation

$$\mathbf{A}x = y. \tag{1.5}$$

for the unknown volume x. Here, y is a stacked vector, assembling the pixels from all absorption images and $\mathbf{A}$ denotes the linear X-ray operator. Each i-th row of $\mathbf{A}$ corresponds to a discretized line integral, running from the X-ray source to detector pixel associated with the index i. Most values in $\mathbf{A}$ are empty, as each line integral uses only a few voxels in x to compute the pixel i. The line integral and numerical implementation will be discussed later in the introduction (section 1.6).

Feldmann-Davis-Kress algorithm To solve equation 1.5, with $\mathbf{A}$ built using a cone-beam geometry, there exist many different CT algorithms. A classical example is the Feldkamp-Davis-Kress (FDK) algorithm [6] which was developed for cone-beam geometries and enjoys popularity due to its fast execution [5, 7]. The algorithm can be formulated as:

$$x^{\star}_{\mathrm{FDK}} := \mathbf{A}^T(y \circledast f), \tag{1.6}$$

with $x^{\star}_{\text{FDK}}$ denoting the FDK solution. The algorithm consists of just two steps: The first is to convolve the absorption images y with a high-pass filter f to reduce the low-frequency bias in the measurements. The second step executes the mathematical adjoint $\mathbf{A}^T$ of the forward X-ray operator. Since $\mathbf{A}$ contains integrals along the paths of the X-rays, $\mathbf{A}^T$ "smears out" the filtered absorption images along the original X-ray paths in the 3D space of the object.

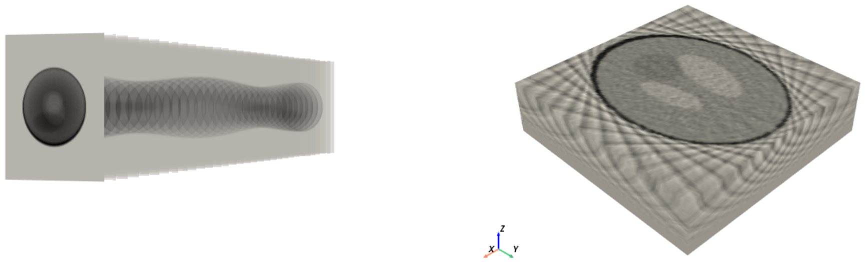


Figure 1.2: Reconstruction of the 3D *Shepp-Logan* phantom. On the left is a stack of 32 absorption images y, uniformly sampled from $[0, 2\pi)$ angles using the CBCT geometry of figure 1.1. The FDK reconstruction $x^{\star}_{\text{FDK}}$ with the Ram-Lak filter [5], on the right, is of low quality, due to the low number of angles.

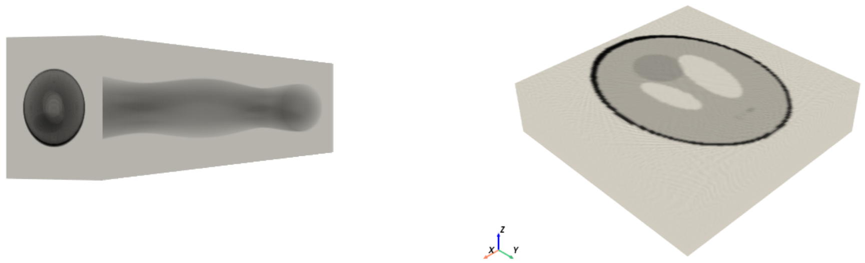


Figure 1.3: Reconstruction of the Shepp-Logan phantom of figure 1.2, now repeated using 256 absorption images. This removes the streaks and additionally recovers three small low-contrast ellipsoids in the bottom-right of the phantom.

Figure 1.2 gives an example of a numerically simulated CT reconstruction using a 3D phantom containing ellipsoid spheres of mixed intensities. The angular stack of simulated absorption images y is shown on the left. On the right, the 3D outcome of the FDK algorithm is displayed with a volume that is sliced through its center. This slice cuts three of the ellipsoids in the volume. However, because of the low number of absorption images, the FDK reconstruction also shows streaks throughout the volume. In figure 1.3, the simulation is repeated with 256 absorption images, demonstrating that the image quality is strongly dependent on the number of absorption images. In section 1.5, we will continue the discussion on sparse-view geometries.

Simultaneous Iterative Reconstruction Technique (SIRT) For later reference, we also present SIRT, an example of an iterative, algebraic reconstruction technique. Whereas the FDK algorithm is an analytical technique, *algebraic* reconstruction techniques are a class of CT algorithms that discretize x on a volumetric grid and solve equation (1.5) algebraically. Hounsfield's first iterative CT algorithm is an example of such technique. Compared to a direct method, such as the FDK algorithm, algebraic methods are better suited to sparse-angle or noisy data [4]. A method that is commonly used nowadays, is SIRT [8]. SIRT reformulates equation (1.5) as a constrained weighted least-squares minimization problem, i.e.,

$$x^{\star}_{\mathrm{SIRT}} = \operatorname*{arg\,min}_{x \in \mathcal{C}} \; \|\mathbf{A}x - y\|_R^2 \,, \tag{1.7}$$

where $x^{\star}$ is the least-squares solution, and $\mathcal{C}$ includes a set of constraints. An example would be the box constraints $0 \leq x \leq 1$, which restrict the reconstructed quantity to a percentage (Chapter 2). The norm $\|\cdot\|_R$ is weighted according to a diagonal matrix R, containing the reciprocals of the row sums of $\mathbf{A}$.

The optimization problem in equation (1.7) is solved iteratively using gradient descent, preconditioned with a diagonal matrix of column sums C. The algorithm proceeds with update steps

$$x^{(k+1)} = \Pi_{\mathcal{C}}\left(x^{(k)} + C\mathbf{A}^T R\left(\mathbf{A}x^{(k)} - y\right)\right), \tag{1.8}$$

where $x^{(k)}$ denotes the k-th iterate. $\Pi_{\mathcal{C}}$ denotes the orthogonal projection onto the constraint set, which effectively clips $x^{(k+1)}$ to $[0,1]$, and sets $x^{(k+1)}$ to zero in the masked area. To prevent fitting noise, and because of a model mismatch in Eq. (1.5), SIRT is usually stopped at a fixed number of iterations.

1.2 Applications

Projectional radiography X-ray CT is found useful for a plethora of medical, industrial and scientific applications. Before we explore these, it is good to mention that for certain applications, radiographs alone may already provide sufficient information. For example, in Fig. 1.3, the contrast in the radiographs already reveal the edges of the ellipses, which, under additional assumptions, can be used to determine their volume. For X-ray fluoroscopy, non-destructive testing or defect inspection, similar contrast on radiographs can also provide sufficient information for the task at hand. One example is the set-up of Chapter 2, wich is used simultaneously for projectional radiography of bubbles, to extract e.g., their gas holdup, and for CT, to compute the bubble shapes. At the same time, radiographs can also be used to assist with CT: Medical CT scanners can acquire "scout views" of the body, to help localize a scanning region inside of the patient. Also in laboratory set-ups, the angular radiographs are often visualized during the scan, for example to reassure that the object remains in view, that the set-up settings (calibration, source voltage) are suitable, and to track dynamics occurring in the object. Methods that work directly on radiographs, e.g., the denoising method that we introduce in Chapter 5, have the advantage that they can be extended to radiography applications.

Medical CT The traditional setting of CT is in hospitals, where it provides a fast and comparatively inexpensive imaging solution. Here the 3D reconstruction of a part of the human body can help with, e.g., the precise localization of a tumor or blood clot, or visualization of a bone fracture. To improve contrast, CT is sometimes used with iodinated contrast agents. However, it can also be used in conjunction with other modalities, such as MRI, which provides better contrast for soft tissues.

In medical CT scanners, a highly-streamlined two-step acquisition-reconstruction procedure obtains a single, three-dimensional computer representation of the human body. Medical CT scanners are equipped with a gantry, consisting of an opposing X-ray source and flat-panel detector, which enables capturing large cross-sectional parts of the body with a single orbit of the gantry. To guarantee safe and optimal use, manufacturers often enclose the software and hardware necessary for tomographic computations internally in the machine, and output the reconstruction as a slice-by-slice volume directly after the scan has proceeded.

Sequential CT In scientific and industrial applications, e.g., biology, life sciences, material sciences or engineering, a more fine-grained control over acquisition and reconstruction is needed. Experimental set-ups, such as the FleX-ray lab discussed in Chapter 3, allow laboratory technicians to tailor the scanning geometry on a per-experiment basis, and to retrieve the raw detector data after the acquisition. Rather than a single rotation, these devices can perform multiple continuous rotations, after which a *sequence of CT reconstructions* can be computed. To convey an impression, Figure 1.4 shows a few examples from application areas that use sequential reconstructions. Broadly speaking, sequential CT is common in the following areas:

- **Dynamic tomography**: A field of great interest in the scientific imaging is to recover the interior of a dynamic object or process, i.e., the tomographic data must be resolved in time. Prototypical examples are: the dynamics of multi-phase fluids (Chapter 2 and 3), dissolution of tablets, rising of dough, and combustion processes.

- **In-line tomography**: Here, a fixed measurement protocol is applied to a series of similar objects, for example, aligned on a conveyor belt. This scenario is more common in industrial applications, such as baggage screening, product inspection, or foreign object detection.

- **Explorative tomography**: Rather than dynamics that are inherent to the object, the scanning geometry is manipulated while the scan is ongoing. An example is to bring the source and object closer together, which increases the spatial resolution of the data and allows zooming into the sample. Another example is a *tiled scan*, stitching

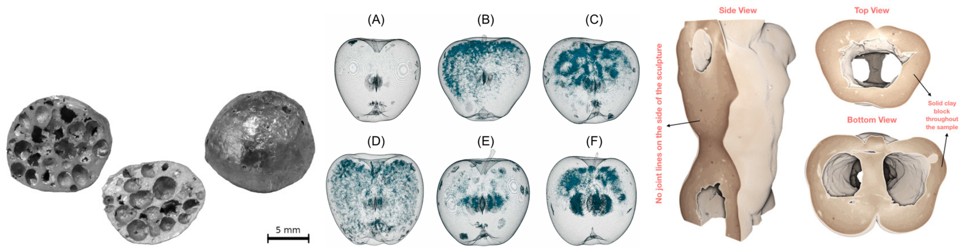

5 mm
(A)
(B)
(C)
(D)
(E)
(F)
Side View
No joint lines on the side of the sculpture
Top View
Solid clay block throughout the sample
Bottom View

Figure 1.4: Three examples from application areas using sequences of tomographic reconstructions. **Left**: observation of pore morphology under compressive loading, figure reproduced from Vopalensky *et al.* [10], published under a CC BY 4.0 license. **Middle**: detecting internal browning in apple tissue, figure reproduced from Wood & Schut *et al.* [11] published under a CC BY 4.0 license. **Right**: searching for finger prints and tool marks in a terracotta mini sculpture, figure reproduced from Coban *et al.* [9], published under a CC BY 4.0 license.

multiple high-resolution reconstructions together to assemble a single reconstruction. Explorative tomography is useful for cultural heritage research and preservation [9].

1.3 Tomographic pipelines

To facilitate all these applications, scientists and laboratories often use certain workflows to process the radiographs and reconstructions. Unlike medical CT, this does not take place inside the scanning device, but on the researcher's personal computer or on computation facilities, with a number of consecutive computational or procedural steps that we will call a *tomographic pipeline* [12]. For our purposes, the pipeline is defined such that it includes the acquisition, pre-processing, reconstruction, and visualization/analysis of the sequential data that passes through it. A few practical examples of pipelines are:

- **Spatio-temporal CT reconstruction**: A timeframe-by-timeframe CT reconstruction, with, e.g., one full rotation of the scanner per timeframe. In Chapter 2, we will see that this can result in a high-resolution "3D movie", and allows studying and segmenting bubbles in a multi-phase flow.

- **Real-time reconstruction**: A region-of-interest of the volume (i.e., not in fully-3D, but in a part of the volume), can be produced in milliseconds, thus at a high framerate, during the experiment using the RECAST3D software package. A scientist can interact with the RECAST3D graphical user-interface to reorient and manipulate multiple slices to quickly visualize certain parts of the object (Chapter 3).

- **On-the-fly reconstruction for machine learning**: A machine learning algorithm needs to see a large quantity of example reconstructions before it can be applied to unseen data. When reconstruction data sets are too large for storage in computer memory, a pipeline can generate these reconstructions on-the-fly from the data set of radiographs (Chapter 4).

The images passing through tomographic pipelines generally possess a high degree of consistency. In a slow-moving fluid, for instance, the acquired data is temporally continuous, and in in-line inspection images are a deformed or modified version of a target template. This requires that acquisition and reconstruction settings (e.g., radiation intensity, detector configuration, reconstruction algorithm), are fixed during the experiment.

On-line and real-time tomography Most tomographic pipelines are *off-line*, which means that radiographs are stored after scanning and that computational steps for CT are ran at a later point in time. In off-line pipelines, time constraints are usually dictated by the set-up or by the physics (e.g., enough photons must pass through the object, or the scanning must be fast enough to follow the physics). In *on-line* tomographic pipelines, the goal is to perform the computational steps concurrently with the scan. On-line pipelines can also be *real-time*, meaning that they aim to minimize the lag between the experimental physics and the visualization component at the beginning and end of the pipeline (Chapter 3). In scientific applications of tomography, real-time usually indicates a delay of milliseconds to seconds.

The pinnacle of real-time tomography would be to use the information gained from reconstructions in the pipeline as (automated) feedback during the scanning process, which is referred to as *steering*. Steering is particularly important when the success of the experiment depends on environmental parameters, for example, when certain physical phenomena must be reproduced under unstable conditions, e.g., with a precise temperature or chemical balance. Steering then allows a laboratory scientist or algorithm to validate or tune the state of the experiment, e.g., by raising the temperature. With steering implemented, obtaining high-quality reconstructions avoids the typical cycle of trial-and-error experiments, which can be costly and time-consuming in X-ray facilities with scarce resources.

1.4 Optimizing tomographic pipelines

X-rays are known for their suitability in fast 3D imaging techniques. Since X-rays are high-energy electromagnetic waves, they propagate at almost the speed of light through any medium. Therefore, the imaging speed with X-rays is often limited by the rate at which detector instruments can operate. Certainly not every imaging task can be optimally addressed with X-rays, for example, due to the radiation damage that X-rays inflicts on biological samples, or due to limited photon flux (treated in the next section). However, the fast underlying physics makes CT uniquely suitable for real-time and high-speed dynamic imaging problems. In comparison, the scanning speed of MRI (Magnetic Resonance Imaging) is constrained by the relaxation time of protons (milliseconds to seconds), and the scanning speed of ultrasound imaging is ultimately limited by the speed of sound in biological tissue (microseconds to milliseconds).

In the previous section, we explained the concept of tomographic pipelines, and discussed their different applications and settings. A closer look at the use cases hints at how fast we would like tomographic pipelines to be, given ideal circumstances. Fast would be “fast enough" when...

- ...a set-up can follow the concurrent physics with a sufficiently high temporal resolution (for dynamic CT in an off-line pipeline);
- ...a human or steering algorithm is able to respond timely in an ongoing experiment (for real-time CT);
- ...a sufficient amount of data can be generated not to throttle a neural network training process (in an on-the-fly data sampling pipeline).

To accomplish these goals, it is not unusual to make large concessions inside the tomographic pipeline. A slow-moving fluid may require a pipeline operating at 10 Hz framerate, human observation perhaps 30 Hz, and a neural network training tasks (section 1.7) several hundreds to thousands of samples per second. To achieve these requirements, several or all components in the tomographic pipeline must act closely together. For example, a set-up may have to rotate faster than what is ideal, or a reconstruction may need to be performed at a lower resolution, or a machine learning algorithm must work with a less-than-ideal number of examples.

In the next sections we will discuss how different pipeline components contribute to faster X-ray CT. In section 1.5 and 1.6, fast scanning and fast reconstruction techniques are discussed. In 1.7 and 1.8 self-supervised deep learning is treated.

1.5 The trade-offs inside fast X-ray imaging set-ups

There are two important qualities of X-ray set-ups to be considered for fast tomographic imaging. The first is the detector framerate: A higher framerate enables a higher temporal resolution, but simultaneously leads to noisier radiographs. The second is the rate of angular acquisition, i.e., the time that it takes for the set-up to acquire all scan angles. Faster angular acquisition can increase the temporal resolution of a dynamic CT reconstruction, but can reduce the quality of each individual timeframe. Figure 1.5 illustrates three different types of scientific X-ray imaging set-ups. We will first study these set-ups, and then further discuss the noise levels and number of angles per rotation.

X-ray set-ups The first set-up is the FleX-ray laboratory at Centrum Wiskunde & Informatica in Amsterdam. The laboratory features a micro-CT scanner with a PerkinElmer Dexela 1512NDT detector. This detector connects to a single cable, and is capped to 26 frames per second via its Camera Link interface (opposed to a slower gigabit GigE Vision interface). Its framerate can be increased to 86 Hz by reducing the resolution of the data via binning of 4×4 pixel regions. Assuming a safe rotation speed of one cycle per second, the set-up can then reconstruct an object once every half-rotation using information from 172 angles.

The facility in the upper right of Figure 1.5 shows a beamline, i.e., an experiment station, in the Canadian Light Source synchrotron. Synchrotron light sources are at the forefront of developing tomographic pipelines. These facilities use a (cyclic) particle accelerator and

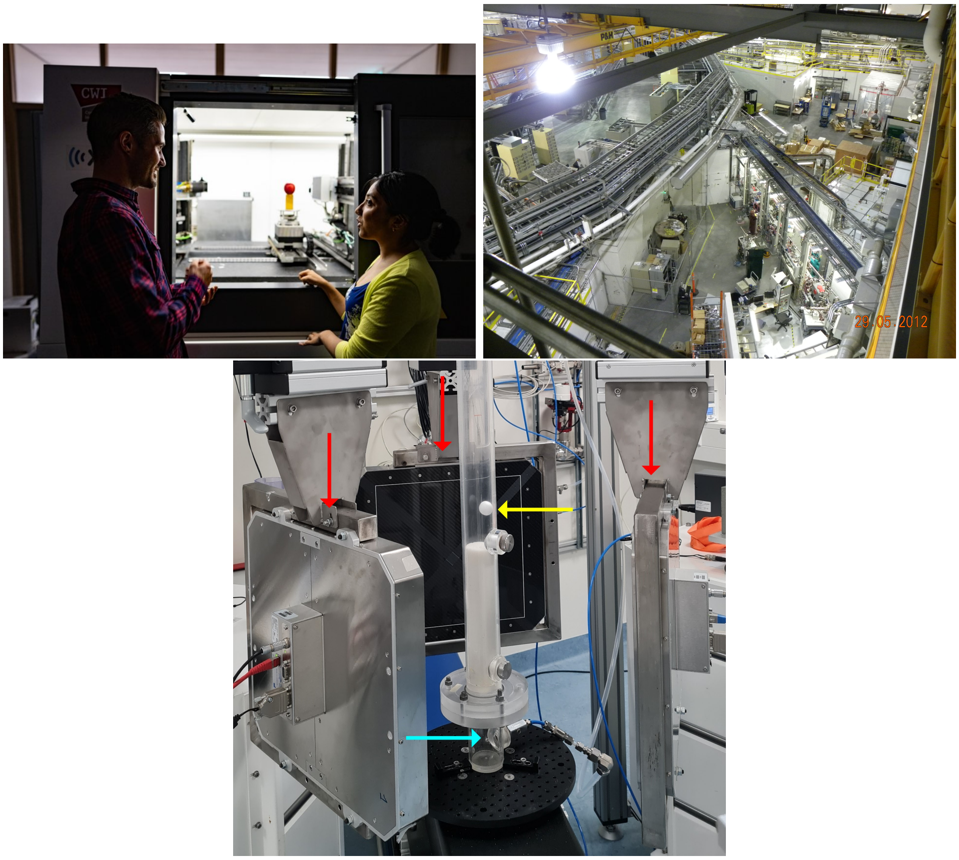


Figure 1.5: Upper left: Conebeam micro-CT scanner at the FleX-ray laboratory (CWI, Amsterdam, the Netherlands). **Upper right**: The SM beamline at the Canadian Light Source (CLS Research Office, Saskatoon, Canada), image provided under the CC BY-SA 2.0 license. **Bottom**: Multi-source multi-detector set-up (Delft University of Technology, Delft, the Netherlands).

storage ring to create brilliant, highly focused, monochromatic X-rays, thereby enabling a more precise investigation of samples than laboratory set-ups. Scientists from diverse fields of applied sciences, e.g., on materials and biological systems, frequently travel to synchrotrons to perform dynamic experiments. To help their users, synchrotrons conduct active research to store, process, and visualize data.

In some situations, the object dynamics can become too complicated or fast for the object to be adequately imaged using a system with a single source and detector. Modern medical devices, for instance, can perform a single rotation once every 0.25–0.30 seconds (3–4 Hz). This means that information on one particular angle of the object is only queried every 0.25–0.30 seconds, which can be too low of a temporal resolution to image certain dynamical processes. For these situations, systems with multiple sources and/or multiple detectors are considered, such as the system with three source-detector pairs in the bottom of figure 1.5. We will further explore the triplet source and detector set-up in Chapters 2 and 5.

Photon noise and electronic noise Similar to photographs taken under low-light conditions, radiographs require a sufficient number of photons to pass through the object and arrive at the X-ray detector. Thus, X-ray sources must emit a sufficient amount of photons, while detectors must register a sufficient amount of photons. Generating a high photon flux is a standing technical challenge. To see this, consider the CBCT set-up (figure 1.1). The X-ray tubes used in CBCT work by shooting electrons into a target material, such as tungsten. A small percentage of the electron energy is converted into X-rays, but the majority of energy dissipates into heat. Heat management is therefore an integral aspect of increasing the photon flux. Then again, to make use of higher photon flux, X-ray detectors must count and read-out the photons, which are converted to electric charge in the detector, at a faster pace. The detector design, such as flat-panel shape, sensor material, capacitance of the pixel sensors, and bandwidth of data buses like PCIe, all determine how fast the charge can be read from the sensor array.

Image quality depends on the number of photons captured and thus on the exposure time. In low-dose imaging, a grainy type of noise is observed known as *photon noise* or *shot noise*. This type of noise can be described by a Poisson probability distribution summarizing the X-ray photon generation in the X-ray source, the interactions with atoms in the sample, and the detection in the scintillator [3, 13, 14, 15]. A scintillator is a crystal, such as Caesium Iodine (CsI) or Gadolinium Oxysulfide (GadOx), consisting of high-atomic-number elements which increase the probability of photo-electric interaction. Detectors that use thick scintillator screens are able to capture more photons, but at the cost of stronger blurring of the measurement.

A second type of noise is due to the electronic components within the X-ray detector instrument. These noise characteristics depend mainly on the the type of photo-detectors used, for example, CMOS (Complementary Metal Oxide Semiconductor) active pixels. This type of noise is typically modeled as Gaussian. The combination of Poisson-Gaussian noise will be discussed more extensively in Chapter 5.

Sparse-view geometries The quality of a reconstruction depends on the spatial and angular sampling of the radiographs. For example, in Fourier-based reconstruction techniques, such as the FDK algorithm, the number of radiographs sampled in a circular trajectory determines until which frequency an object can be reconstructed [3]. For industrial applications (such as identifying knots in scanning of wood logs [16]), or scientific applications in synchrotrons [17], obtaining a sufficient number of angular samples often proves difficult.

Scientific applications use large-area flatpanels detectors that can cover larger parts of the to-be-scanned samples, but have a lower read-out speed compared to line detectors. As a result, detectors are often not fast enough to keep up with the dynamics of the object or could put constraints on the number of samples that can be scanned in high-throughput in-line tomography. The consequence is that the object or device must rotate faster and that fewer radiographs can be retrieved from a single rotation of the set-up. The reconstruction problem resulting from a sparse-angular range of data is known as *sparse-view CT*.

1.6 Faster reconstruction via tailored algorithms

Next to fast radiograph acquisition, tomographic pipelines require fast computational components to handle the extraordinarily large amounts of data that they pass. In the third set-up of figure 1.5, for example, a single timeframe of three detectors can contain the information to reconstruct the object on a 1500-by-1500-by-1500 spatial grid. With a framerate of 60 Hz and 64-bit numerical precision, the total size of all reconstructions after 20 minutes of experimentation would amount to 1,768 terabytes (1 terabyte is equivalent to 1,024 gigabytes or 1,048,576 megabytes). Efficient processing and reconstruction is then indispensable. This section discusses the role of GPU-accelerated tomographic projectors and their inclusion in software frameworks.

Tomographic projectors In Eq. 1.5, $\mathbf{A}\colon x \mapsto y$ (the X-ray transform) is often called a *forward projection* due to the geometric notion of the object casting a shadow on the detector. Similarly, its adjoint, $\mathbf{A}^T : y \mapsto x$, is called a *backprojection*. Several discretization strategies can be considered to construct $\mathbf{A}$ and $\mathbf{A}^T$ [4, 18]. Each row of $\mathbf{A}$ (which corresponds to a column of $\mathbf{A}^T$), discretizes a single line integral from Eq. 1.3. That is, it contains the interpolation weights for the integral

$$[\mathcal{A}x]_{u,v,\psi} = \int_{-\infty}^{\infty} x\left(s_\psi + (d_{\psi,u,v} - s_\psi)t\right)\, \mathrm{d}t, \tag{1.9}$$

which describes the straight line from a point source s_ψ to a detector pixel midpoint $d_{\psi,u,v}$ through the volume x.

A common choice for estimating the line integral (Eq. 1.9) is the Joseph kernel [4, 19]. In this *ray-driven* approach, each integration point takes a trilinearly interpolated value from neighboring points on the voxel grid. Importantly, the line is modeled such that it precisely arrives at a detector pixel's midpoint, and hence no sinogram interpolation is needed. Conversely, during a *voxel-driven* backprojection, a bilinear interpolation at each angle of the sinogram sums up to the voxel's value [20]. In this case, all lines of backprojection go precisely through the voxel's center, and now, interpolation in the volume is avoided. The reader is referred to [4] for interpolation formulae.

Parallel implementation on GPUs Most numerical implementations of projectors nowadays rely on hardware acceleration using graphical processing units (GPUs) (Chapter 4). GPUs consist of a large number of computational cores, and are therefore generally more efficient for large-scale parallel operations than a computer's central processing unit (CPU). The most common platform for writing GPU-accelerated code today is Nvidia CUDA. Listing 1.1 demonstrates a purposefully-simplified piece of CUDA code that performs backprojection for a circular fan-beam geometry. The fan-beam geometry is equivalent to a two-dimensional cone-beam geometry in the horizontal plane.

Listing 1.1: A simplified CUDA kernel for fan-beam backprojection used for demonstration on github.com/adriaangraas/astra-kernelkit. This function is launched in parallel with one thread per voxel (X, Y) in the volume.

```
__global__ void backproject(
   float * volume, float * sinogram, float * angles,
   int nr_angles, int nr_voxels_x, int nr_voxels_y, int nr_pixels,
   float src_obj_dist, float obj_det_dist
) {
   // X, Y are the voxel's center coordinates of this thread
   unsigned x = blockIdx.x * blockDim.x + threadIdx.x;
   unsigned y = blockIdx.y * blockDim.y + threadIdx.y;
   float X = float(x) - nr_voxels_x / 2.0 + 0.5;
   float Y = float(y) - nr_voxels_y / 2.0 + 0.5;

   // if this thread number does not map to a voxel, just ignore it
   if (x >= nr_voxels_x || y >= nr_voxels_y) return;

   for (int psi = 0; psi < nr_angles; ++psi) {
      // project the voxel (X, Y) onto a coordinate U on the 1D detector
      float U = forward_project(
         X, Y, angles[psi], src_obj_dist, obj_det_dist);
      // linearly interpolate the value at U from sinogram gridpoints
      volume[y * nr_voxels_x + x]
         += linearly_interpolate(sinogram, psi, nr_pixels, U);
   }
}
```

In this CUDA function, termed a *kernel* in the jargon of CUDA computing, the arrays `float * volume` and `float * sinogram` are passed as pointer arguments. This means no memory copies are made, and implies that the values reside in global GPU memory. The parameters of the call, furthermore, contain the geometry description of the 2D reconstruction. The kernel is launched in parallel, using a block of *threads* organized by pairs of 2D indices.

Each thread (`x`, `y`) maps to a single voxel coordinate (`X`, `Y`), and computes and writes its outcome to `volume`. In the function, a loop accumulates the contribution of each angle, by first projecting the voxel onto the detector at the specified angle, and then bilinearly interpolation using the nearest pixel values. CUDA kernels thus enable a matrix-free approach of computing the volume from the sinogram, i.e., the backprojection matrix $\mathbf{A}^T$ was never explicitly formed in memory.

The combination of a ray-driven forward projector and voxel-driven backprojector has advantages for an implementation on GPUs. All threads (discussed in Chapter 4) are independent of each other, which avoids potential race conditions, i.e., when two threads would write to the same memory simultaneously [7]. However, due to the difference in forward and backward lines, the projectors are not each others exact transpose (this is called *unmatched*). Unmatched projectors lead to nonconvergence in iterative algorithms, due to a nonsymmetry of the iteration matrix [21]. In the presence of noise, this does not always pose a problem [22].

Software implementations The precise implementation of algorithms in tomographic pipelines can play a critical role in their efficiency. Common software packages used by researchers are the ASTRA Toolbox [23], TIGRE [24], or packages that use these frameworks as a back-end, such as TomoPy [25], Tomosipo [26] or the MATLAB Spot toolbox. Via several software layers (e.g., Python, MATLAB, Cython, C++), these packages call CUDA kernels similar to Listing 1.1 to implement algorithms such as SIRT and FBP efficiently.

For sequential tomographic reconstructions in pipelines, standard packages can quickly become inefficient, as they are geared towards making single reconstructions. To see this, compare Listing 1.2 with Listing 1.3, which are pseudo-codes for a repeated filtered-backprojection reconstruction from a sliding window of projections. The left listing treats each reconstruction individually: re-uploading measurement data, recomputing the same geometry, and repeating a filtering step with the Ram-Lak filter unnecessarily. The implementation in the right listing instead removes transfers and geometry computations from the sequential loop, thereby reducing bandwidth and computations.

Listing 1.2: Naive FBP implementation for a sliding projection window.

```
for t in [0, 1, ..., T - 360]:
   # repeat reconstruction
   geom = compute_geometry(
     src, det, angles)
   y = upload_to_GPU(  # allocate
     [data[t], ..., data[t+360]])
   y = convolution(y, ram_lak)
   x = backproject(y, geom)
   delete y, x  # deallocate
```

Listing 1.3: Incremental FBP implementation, recycling memory and computations.

```
geom = compute_geometry(
   src, det, angles)
y = upload_to_GPU(  # allocate once
   [data[0], ..., data[T-1]])

for t in [0, 1, ..., T - 360]:
   # update a single angle
   y[t % 360] = convolution(
      y[t], ram_lak)
   x = backproject(y, geom)

delete y, x  # deallocate
```

While the example above is purposefully simplified, many current software packages do not permit the flexibility of the second form, Listing 1.3, which requires a more fine-grained control over, e.g., GPU memory. The reason is often that an implementation is hidden by language barriers in the software. Sometimes, users of tomographic packages find inventive ways to use existing software more efficiently, for example, by reformulating a stack of 2D reconstructions as if the data were acquired from a single 3D geometry, using the z-dimension for the stack. In Chapter 4 we will further discuss how software packages can become more flexible by eliminating the barrier between the high-level "prototyping language" and the low-level "efficient language".

Tuning reconstruction problems In tomographic CUDA kernels, such as Listing 1.1, there are often several free parameters: the number of angles handled by one parallel thread, the distribution of voxels over threads, as well as certain numerical choices such as the axis ordering of the geometry, interpolation method, and types of memory to store data and intermediate computations. Oftentimes, sensible defaults already yield good performance, however, for unconventional reconstruction problems the standard choices can be significantly less efficient.

Kernel tuning aims at optimizing free parameters in kernels. Tuned algorithms can achieve better run-times, reduce energy consumption [27, 28], or utilize less resources, in particular, GPU memory and computation. In high-throughput applications, such as in-line CT scanning, a kernel can be tuned toward a fixed measurement protocol and dedicated GPU architecture. In these situations, even a slight improvement can lead to significant energy savings over the equipment's lifetime.

1.7 Introduction into Deep Learning

For inverse problems in imaging, it would be no exaggeration to call our last decade the *decade of deep learning*. Deep learning is a form of Machine Learning (ML) or Artificial Intelligence (AI) using neural networks consisting of many consecutive operations (*layers*). It brought a new perspective on algorithm design, showing that existing algorithms can be replaced or improved by "learned" alternatives, which means that these algorithms have free parameters that are initialized based on prior information obtained from (large) data sets of examples.

In tomography, deep learning-based algorithms are currently considered the state-of-the-art [29, 30, 31, 32]. Moreover, —and this is an often under-emphasized advantage— neural networks can be much faster than traditional algorithms, especially when they are executed on GPUs. Compared to traditional algorithms, neural networks can perform certain image enhancement tasks in milliseconds to seconds, rather than in minutes or hours. This allows supersampling, artifact removal, segmentation, or denoising [33, 34, 35] to be executed at high framerates in tomography pipelines, especially in synchrotron and laboratory environments. The disadvantage of deep learning, however, is that these algorithms need to be trained before usage, which can take days to weeks to complete.

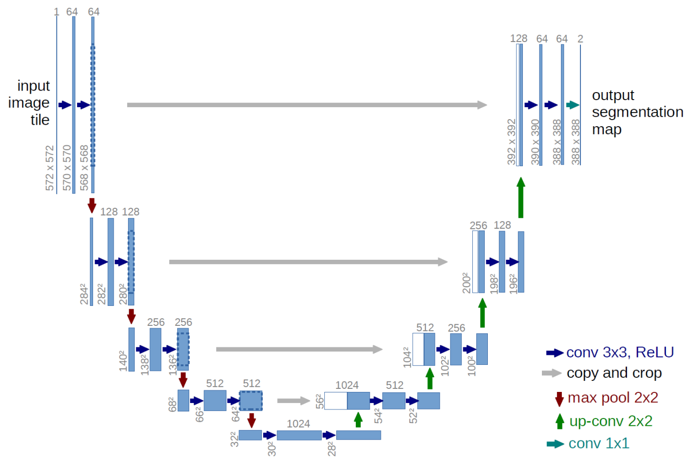


Figure 1.6: The U-Net architecture [38] is a DNN consisting of an encoding-decoding architecture using many convolutional layers, shaped in a “U”. The implementation illustrated here relies on learnable “max pooling” and “up-convolution” operators to decrease and increase the internal image resolution.

Next, the three pillars of deep learning are explained: *representation*, *optimization* and *generalization*.

Representation Designing a deep neural network (DNN) architecture that is well suited to the task at hand, is the area of *representation*. A design concerns the number of layers and channels, choices of operators, and connections in the network. For imaging tasks, architectures often consist of convolutional layers and non-linear activation functions [36, 37]. One prominent example is the U-Net (figure 1.6), a neural network that downsamples and upsamples the internal representation of the image, in order to detect image features on multiple scales.

The goal of a neural network architecture f is to approximate an unknown mapping $f^\dagger : \mathcal{U} \to \mathcal{V}$, where $\mathcal{U}$ and $\mathcal{V}$ denote image manifolds in case of an image-to-image network. For image denoising, for example, the aim of f would be to approximate the perfect denoiser $f^\dagger$. In this case, the ideal but unknown $f^\dagger$ would map each noisy input image from $\mathcal{U}$ to a denoised counterpart in $\mathcal{V}$. Another example of an image-to-image neural network would be one that removes sparse-angle artifacts from reconstructions (cf. figure 1.2 and figure 1.3).

The approximation f of $f^\dagger$ is accomplished by a parametrization, which is denoted by $f \equiv f_\theta$, with $\theta \in \Theta$ being the free parameters. Those parameters can be, for example, the filters of convolution operators or the matrix entries of linear operators.

Optimization During *optimization*, or *training*, the parameters θ are obtained using a data set of input-target examples. Let $\mathbf{u}_i \in \mathcal{U}$ denote an input—for example, a 2D or 3D noisy reconstruction—and $\mathbf{v}_i \in \mathcal{V}$ denote a target. In *supervised learning*, the target must be a ground truth, i.e., $\mathbf{v}_i \approx f^\dagger(\mathbf{u}_i)$. In the case of supervised denoising, for example, the input could be a noisy reconstruction, and the target must then be a noise-free counterpart.

Given a task-specific *loss function* $\ell(\mathbf{u}, \mathbf{v})$ that measures the misfit between two images, training can be formulated as the optimization of the so-called *empirical risk* $R_\mathcal{D}$ for parameters θ, data set $\mathcal{D}$, architecture f_θ and loss ℓ:

$$\min_{\theta \in \Theta} \quad R_\mathcal{D}[f_\theta] := \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{u}_i, \mathbf{v}_i) \in \mathcal{D}} \ell\left(f_\theta(\mathbf{u}_i), \mathbf{v}_i\right). \tag{1.10}$$

Generalization The performance of a trained network on unseen data is called *generalization.* To quantify generalization, image pairs from $\mathcal{D}$ are considered as random samples from a latent *training distribution*, a probability distribution $\mathbb{P}$ over $\mathcal{U} \times \mathcal{V}$. The *expected risk* describes the true but unknown performance of f_θ over $\mathbb{P}$,

$$R_\mathbb{P}[f_\theta] := \mathbb{E}_{(\mathbf{u},\mathbf{v})\sim\mathbb{P}}\left[\ell(f_\theta(\mathbf{u}), \mathbf{v})\right], \tag{1.11}$$

whereas the empirical risk is limited to a finite sample of it. The *generalization gap*, $|R_\mathcal{D} - R_\mathbb{P}|$, describes the distance between the two risks, and a network is said to generalize well when this gap is small. Since the expected risk cannot be computed, $R_\mathbb{P}$ is commonly approximated using a *hold-out data set* $\mathcal{D}'$ also sampled from $\mathbb{P}$ (also often referred to as the *test data set*). One then computes $|R_\mathcal{D} - R_{\mathcal{D}'}|$ for an approximation of the generalization gap.

1.8 Self-supervised Learning: Scan faster, repair later

Learned algorithms have the potential to remove noise and sparse-angle artifacts that were introduced by fast scanning procedures (section 1.5), and thereby allow to image with higher temporal resolution or throughput. In Chapters 3 and 4 we will discuss several challenges that these algorithms bring, such as a sufficient amount of training time or integrated tomographic projectors. Their largest challenge, however, is typically the availability of ground truth training data. Low-noise or artifact-free images are difficult to acquire in real-world applications, for example, because of limitations of the equipment, laboratory time, or during scanning of fast dynamic processes. At the same time, learned tomographic algorithms cannot rely on the large training data sets that are publicly available on the internet. These data sets do not generalize well, as experimental tomographic data consists of unique image features.

To overcome these difficulties, *self-supervised learning* replaces the ground truth targets $\mathbf{v}_i$ in the data set by noisy surrogates that are easier to obtain in practice. Even though $\mathbf{v}_i \not\approx f^\dagger(\mathbf{u}_i)$, training with surrogates can still yield a network f_θ that approximates the sought $f^\dagger$ nonetheless. In the remainder of this section we will give examples of self-supervised denoising.

The *Noise2Noise* strategy: assume paired images The *Noise2Noise* [39] (N2N) strategy is one of the most influential works published on self-supervised denoising. N2N requires a training data set $\mathcal{D}$ consisting of pairs $(\mathbf{u}_i', \mathbf{u}_i'')$, where $\mathbf{u}_i'$ and $\mathbf{u}_i''$ are two images of the same object but with different i.i.d. (independent and identically distributed) realizations of the latent noise distribution. Similar to the supervised loss (equation 1.10), the goal is to minimize

$$\min_{\theta \in \Theta} \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{u}_i', \mathbf{u}_i'') \in \mathcal{D}} \ell\left(f_\theta(\mathbf{u}_i'), \mathbf{u}_i''\right). \tag{1.12}$$

Equation 1.12 thus simply takes a noisy target $\mathbf{u}''$ instead of the ground truth $\mathbf{v}$. Since the noise is not correlated between inputs and targets, f_θ can at best reproduce noiseless image features that are consistent between inputs and targets. Those image features are encoded as convolutional filters in θ, and are learned due to the similarity of patterns in the data set.

The main limitation of Noise2Noise is that it requires paired acquisition, i.e., two noisy images are required for the network to be trained. The next two examples show two more self-supervised denoising strategies that are derived from Noise2Noise and bypass this requirement.

The *Noise2Inverse* strategy: splitting the angular projections Noise2Noise is brought to the tomographic domain by *Noise2Inverse* [40] (N2I). In its simplest form, it chooses for $\mathbf{u}'$ an FBP reconstruction from projection angles with even indices and for $\mathbf{u}''$ an FBP reconstruction from odd indices. Assuming that the detector noise is temporally uncorrelated, $\mathbf{u}'$ and $\mathbf{u}''$ result in closely similar objects, but with statistically independent noise. It thereby avoids acquisition with two noisy projections for each angle.

In Chapter 3 we will discuss a real-time pipeline built with Noise2Inverse:

$$\boxed{\text{FleX-ray set-up} \rightarrow \text{Region-of-interest FBP} \rightarrow \textit{Noise2Inverse}} \tag{1.13}$$

Here, N2I is executed as a post-processing neural network to denoise filtered-backprojection reconstructions. To keep the pipeline fast, training the reconstruction algorithm is executed on patches, i.e., small regions-of-interest, meaning that the size of the reconstruction is restricted.

The *Noise2Self* strategy: splitting by pixels Blind-spot denoisers are another type of self-supervised methods that aim to overcome the problem of paired noisy measurements. The method is not restricted to the tomographic domain but can be applied to any data set of images as long as assumptions on the noise are satisfied. Blind-spot denoisers are based on the idea that the image features within a single image have spatially-extended structures, whereas pixel-wise noise is statistically independent. The seminal works are Noise2Self [41] and Noise2Void [42], to which there have been several extensions and improvements [43, 44, 45]. A formalization of their principle is termed $\mathcal{J}$-invariance [41], which describes the statistical independence of a function (in our case, a neural network) between subsets of its input and its output. In practice, it is enforced by masking or randomization of pixels in the input. Definition 1.8.1 gives our tailored redefinition of $\mathcal{J}$-invariance for images.

Definition 1.8.1 ($\mathcal{J}$-invariance)**.** *Let $\mathcal{J}$ be a partition of the pixel indices $\{0, \ldots, N_u N_v\}$ into non-overlapping grids. Let $J \in \mathcal{J}$ be one grid, and denote with $(\cdot)_J$ its values. An $f_\theta \colon \mathbb{R}^{N_u N_v} \to \mathbb{R}^{N_u N_v}$ is called $\mathcal{J}$-*invariant *if $\mathbf{u}_J$ and $(f_\theta(\mathbf{u}))_J$ are statistically independent for all $J \in \mathcal{J}$.*

To understand why $\mathcal{J}$-invariance is effective, recall that in a self-supervised method the ground truth target $\mathbf{v}$ is replaced by a noisy realization. Blind-spot denoisers choose for this realization the input $\mathbf{u}$ again and constrain f_θ to be a $\mathcal{J}$-invariant function. The expansion of a self-supervised mean-squared error loss, with $\mathbf{u} = \mathbf{v} + (\mathbf{u} - \mathbf{v})$ decomposed in ground truth and noise, gives:

$$\mathbb{E} \left\| f_\theta(\mathbf{u}) - \mathbf{u} \right\|_2^2 = \mathbb{E} \| f_\theta(\mathbf{u}) - \mathbf{v} \|_2^2 + \mathbb{E} \left\| \mathbf{u} - \mathbf{v} \right\|_2^2 - 2\mathbb{E} \left\langle f_\theta(\mathbf{u}) - \mathbf{v}, \mathbf{u} - \mathbf{v} \right\rangle, \tag{1.14}$$

which consists of the supervised loss, but adds two more terms: A variance term and a negative cross-product. The variance term does not depend on f_θ, and therefore does not play a direct role during training. The cross-product describes the correlation between noise in the input (i.e., the right-hand term, $\mathbf{u} - \mathbf{v}$) and noise in the output (i.e., the left-hand term, $f_\theta(\mathbf{u}) - \mathbf{v}$). In a $\mathcal{J}$-invariant network, by definition 1.8.1, $f_\theta(\mathbf{u}) - \mathbf{v}$ and $\mathbf{u} - \mathbf{v}$ do not correlate, and the cross-term therefore vanishes, regardless of the choice of network parameters θ. Therefore, optimization of a blind-spot network resembles supervised optimization. Note, however, that constraining f_θ to be a $\mathcal{J}$-invariant network will typically lower its ability to represent $f^\dagger$, e.g., it may perform less well at denoising compared to an unconstrained network trained in a supervised way.

In Chapter 5, we will consider the tomographic pipeline

$$\boxed{\text{TU Delft set-up} \to \textit{Noise2Self} \to \text{SIRT reconstruction.}} \tag{1.15}$$

In comparison to the previous pipeline, the learned component appears in the middle, and removes noise *before* data enters the reconstruction algorithm.

1.9 Research Questions

In this dissertation, several aspects of fast CT have come together in four research questions, each associated with a chapter and a journal publication. Although unintentional, the chronology of the publications roughly outline that of a "fast tomographic pipeline": The first work is on the topic of fast acquisition, the second on real-time imaging pipelines, the third on fast reconstruction software, and the last paper on self-supervised deep learning.

The first paper publication is a collaboration with Delft University of Technology and in particular with Evert Wagner and Luis Portela from the Transport Phenomena group. It describes an X-ray set-up dedicated to imaging the fast dynamics of fluidized beds with a tailored reconstruction method. The project entailed several challenges with real-world data, such as the set-up calibration, and the development of a robust X-ray measurement principle. The collaboration also provided a data set that was used in the third and fourth research projects.

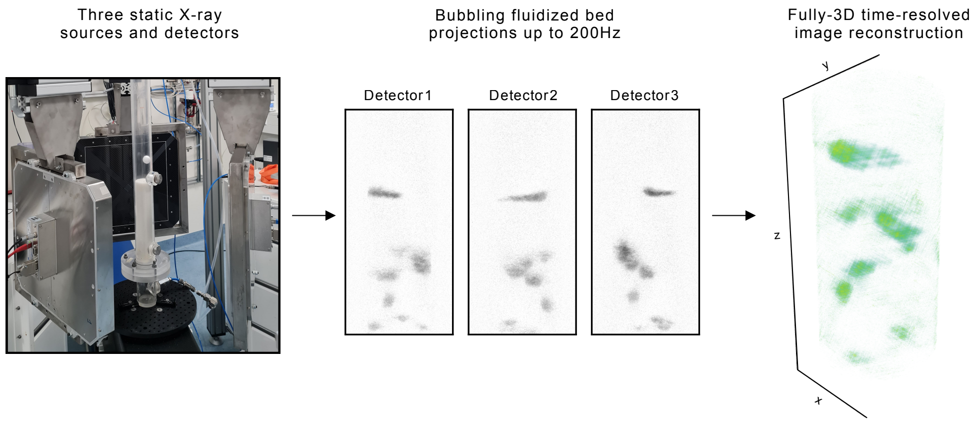
The second project discusses *just-in-time deep learning*, a learning technique for real-time tomographic pipelines. The original motivation for the work stemmed from the RECAST3D software project, developed by Jan-Willem Buurlage and other members of the Computational Imaging group at CWI. RECAST3D facilitates real-time reconstruction by providing a graphical user interface around FDK/FBP reconstruction restricted to cross-sectional slices. Since high-resolution real-time 3D reconstruction is not yet feasible, the software aims to reconstruct only what the user needs to see, effectively reducing the 3D problem to a few 2D problems. The existing pipeline opened the opportunity to explore real-time learning as well as a real-time reconstruct-and-denoise set-up. The data used to demonstrate the concept was acquired in the FleX-ray laboratory at CWI by Felix Lucka and Sophia Bethany Coban.

The third project concerns a software project that enables faster reconstruction for unconventional geometries or 4D reconstruction problems. The software is a continuation of the first two projects. The original reconstruction algorithm used by TU Delft required a CPU operation inside of a GPU algorithm, resulting in excessive algorithm runtimes. The second project then required thousands of reconstructions per second for the purpose of on-the-fly training of neural networks. Both of these scenarios could not be optimally addressed with the reconstruction software that was present at the time. The paper was written together with Willem Jan Palenstijn and Ben van Werkhoven from Leiden University.

The focus of the last research question and paper publication was on denoising, an important topic within the first and second project. In tomographic imaging of fast dynamic processes, radiographs can degrade severely due to noise. Although several techniques exist to remove noise in image space (after reconstruction), denoising radiographs has particular advantages, especially with sparse or ultra-sparse geometries. Hence, a self-supervised deep-learned denoising method based on Noise2Self was investigated that could work entirely without ground truth or noisy training targets.

Can gas-solids fluidized beds be imaged with Computed Tomography, without sacrificing a spatial or temporal dimension? (Chapter 2)

A fluidized bed is a dynamic gas-particle mixture that has fluid-like characteristics. Due to its fast dynamics, it can only be imaged with a set-up consisting of multiple non-rotating sources and detectors. Current imaging methods either use a high-speed slice-based (2D) reconstruction, on a single height of the bed, or a temporally-averaged 3D reconstruction, over the bed's full height. In this chapter, we present a method that is both fully-3D and time-resolved. It uses a set-up with three source-detector pairs, a marker-based calibration procedure, a tailored preprocessing technique, and a constrained SIRT reconstruction. In comparison to existing techniques and other modalities, the X-ray technique enables investigation of the high-velocity morphological changes and interactions of bubbles in large-scale laboratory simulations of fluidized beds.


Three static X-ray sources and detectors
Bubbling fluidized bed projections up to 200Hz
Fully-3D time-resolved image reconstruction
Detector1
Detector2
Detector3
y
z
x

How to overcome the issue of neural network generalization in real-time tomography? (Chapter 3)

Many synchrotron tomography beamlines and X-ray laboratories are moving toward real-time visualization and experimental steering using tomographic imaging pipelines. Deep learning has an incredible potential for these pipelines, e.g., as denoising or segmentation add-ons, as it can provide high framerates and is typically more accurate than traditional methods. However, due to distributional changes in the data during experiments, e.g., due to user interactions, experimental modifications, and set-up reconfigurations, there is generally no reliable training data available, and therefore networks do not generalize well. We propose to train small CNNs concurrently with the ongoing experiment by intercepting the reconstructions that are pushed through the pipeline. In our experiments from the FleX-ray laboratory, we show that a denoising network generalizes better than a pre-trained CNN on pipeline data of dissolving tablets with RECAST3D.

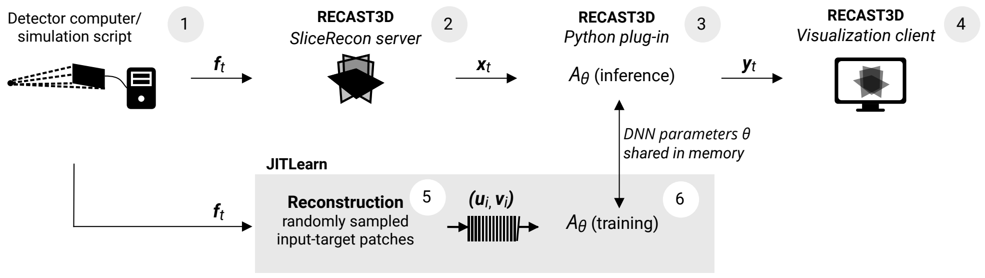


Figure 1.7: Just-in-time learning using a U-Net for image denoising. **Labels 1 to 4**: The pipeline reconstructs slices $\mathbf{x}_t$ from a stream of projections $\mathbf{f}_t$, and sends these to the visualization client. **Labels 5 and 6**: The neural network process intercepts the slices via a plug-in at (3). In a concurrent process, the U-Net architecture A_θ is trained on reconstructions using a capacity queue.

How to make the high-performance ASTRA Toolbox projectors easily modifiable for Python end-users? (Chapter 4)

Tomographic algorithms often use matrix-free implementations of the X-ray operators **A** and $\mathbf{A}^T$ (equation 1.5) called *forward* and *backprojectors*. In ASTRA Toolbox and Tomosipo, they are optimized for a generic geometry and data size, which can lead to suboptimal performance in neural networks, with very small and high-volume data, or in dynamic reconstructions. Modifying or tuning them towards a specific use case, however, is not straightforward due to several programming layers of abstractions between the user's programming language (Python/MATLAB) and the graphics processing unit (GPU). In this article, we use CuPy, a Python library resembling NumPy and SciPy, to compile GPU projectors during the Python script. We then show improved performance for problems with non-standard geometries and data sizes.

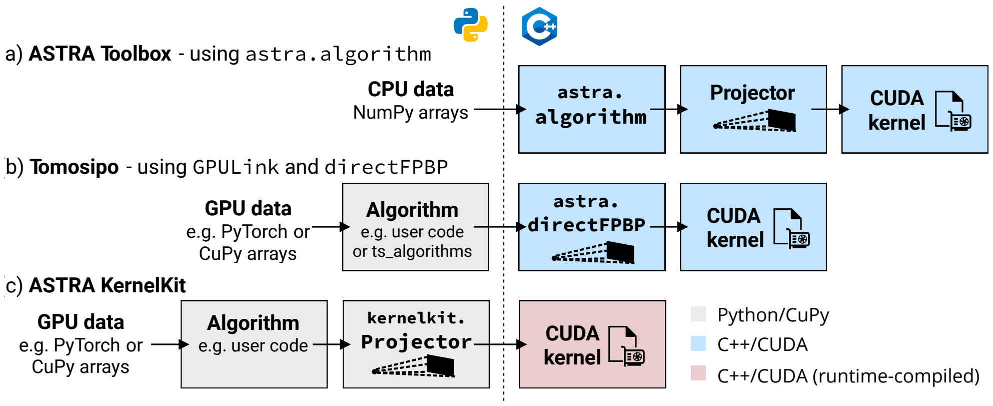


Figure 1.8: Software overview of ASTRA Toolbox, Tomosipo [26], and our package, ASTRA KernelKit, illustrating where its components are located with respect to the Python-C++ language barrier. (a) and (b) show two methods of accessing a projector in ASTRA Toolbox. In comparison, ASTRA KernelKit uses a Python-based projector and runtime compilation of the CUDA kernel.

How to denoise radiographs using neural networks, without ground truth example data? (Chapter 5)

Recently, a new class of self-supervised denoising strategies has emerged for data sets consisting of *unpaired* noisy images, i.e., without clean ground truths or even secondary noisy realizations. Despite their large potential for X-ray imaging, where dynamical set-ups often acquire long sequences of single, noisy, X-ray radiographs, these methods cannot be applied straightforwardly. The main problem is that X-ray scintillator detectors cause a spatial correlation of the noise, preventing denoisers in this class from distinguishing clean image features (which correlate spatially) from noise (which does not correlate spatially). We show that, because the point-response function of scintillator blur is highly uniform, the noise correlation can be reverted by a deconvolution. This allows self-supervised denoising of the preprocessed radiographs while leveraging image features from large experimental data sets.

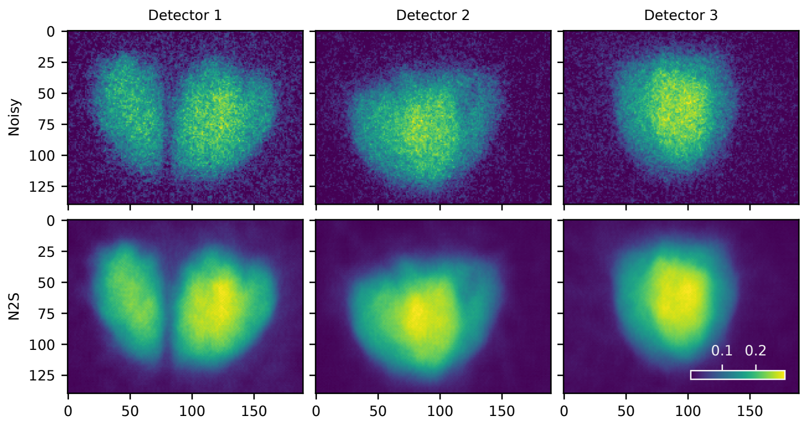


Figure 1.9: An application of the trained self-supervised denoiser Noise2Self (N2S) to a sample of a large data set of noisy fluidized-beds radiographs without ground truths.