Universiteit Leiden

FleX-DT: Design and Auto Calibration of a Digital Twin for the FleX-ray CT System

Jyothis Gireesan Mini (s3777103)

Supervisors:
Prof.dr. Joost Batenburg & Prof.dr. Tristan van Leeuwen &
Dr. Alexander Skorikov

Masters Thesis– Artificial Intelligence

Leiden Institute of Advanced Computer Science (LIACS)
liacs.leidenuniv.nl

May 3, 2026

Abstract

Computed Tomography (CT) is a non-destructive technique for 3D digital interior reconstruction of an object using X-ray projections taken at different angles. Modular X-ray systems allow flexible hardware configurations that expand experimental and scientific research; however, this also makes these machines costly to use and subject to limited access. Thus, a geometrically accurate digital twin serves as a virtual testbed, enabling experimentation, trajectory testing, and scan simulations at minimal cost and without risk.

This work investigates the design, implementation, and calibration of a geometrically accurate, interactive, and configurable **Digital Twin** for the FleX-ray machine, a modular cone-beam computed tomography (CBCT) system at Centrum Wiskunde & Informatica (CWI) in the Netherlands, developed using Unity. The digital twin is also integrated with ASTRA Toolbox by mapping Unity 3D geometry into a 12-parameter cone-beam projection model that describes the source, object, and detector geometry and computes real-time X-ray projections of the object for visualization. The Digital Twin is further improved by applying a calibration technique to align it closely with the physical geometry. An iterative Gauss-Newton nonlinear least-squares method is explored to calibrate the digital twin's geometry, using a cuboid assembled from LEGO bricks with embedded metal markers as the calibration phantom, by minimizing the reprojection error between measured and predicted bead positions.

The calibration method was evaluated using simulated phantoms and validated using a physical phantom. Sensitivity analysis was performed to investigate the influence of the number of embedded beads, angular sampling frequency, and spatial spread of the beads within the phantom, which are necessary for reliable calibration, and how spatial spread significantly affects convergence robustness. The findings demonstrate that spatial bead distribution and angular coverage are critical for parameter distinguishability and convergence robustness.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Approach . . . 2
- 1.3 Research Questions . . . 2
- 1.4 Contributions . . . 2
- 1.5 Thesis Structure . . . 3

2 System Context and Technical Foundations **4**
- 2.1 Fundamentals of X-ray Computed Tomography . . . 4
- 2.2 Cone-Beam CT . . . 5
- 2.3 FleX-ray System Overview . . . 6
- 2.4 ASTRA Toolbox Overview . . . 8
- 2.5 Unity Engine Overview . . . 10

3 Related Work **13**
- 3.1 Digital Twin Systems . . . 13
- 3.2 Geometry Calibration in Cone-Beam CT . . . 14
- 3.3 Optimization Methods for Geometric Calibration . . . 16

4 Design and Implementation of the Digital Twin **20**
- 4.1 System Architecture . . . 20
- 4.2 Coordinate System Definition and Scaling . . . 21
- 4.3 Component Modeling and Hierarchy . . . 22
- 4.4 Physics Engine Assumptions and Trade-offs . . . 24
- 4.5 Kinematic Modeling of Mechanical Components . . . 26
- 4.6 System Interactivity and External Connectivity . . . 27
- 4.7 Script Parsing and Automated Command Execution . . . 29
- 4.8 Integration with ASTRA: The Geometry Export Pipeline . . . 31
- 4.9 Projection Simulation and Communication Workflow . . . 32
- 4.10 Validation of the Digital Twin Geometry . . . 33

5 Mathematical Formulation of the Calibration Problem **34**
- 5.1 Calibration Objective . . . 34
- 5.2 Phantom Design . . . 34
- 5.3 Geometry Parameter Vector . . . 35
- 5.4 Forward Projection Model and Residual Formulation . . . 37
- 5.5 Jacobian Construction . . . 38
- 5.6 Optimization Method . . . 38
- 5.7 Observability and Identifiability . . . 39
- 5.8 Extension: Detector Lateral Calibration . . . 40

6 Experimentation **41**
- 6.1 Simulated Experimental Setup . . . 41
- 6.2 Experiment 1: Effect of Bead Count and Projection Angles on Calibration Accuracy 42

6.3 Experiment 2: Effect of Phantom Geometry on Identifiability . . . 44
6.4 Experiment 3: Effect of LM Damping on Convergence . . . 45
6.5 Experiment 4: Calibration using Real FleX-ray Data . . . 46
6.6 Additional Experimentation: Detector Lateral Calibration using Multi-Geometry Acquisition . . . 48

7 Discussion **50**
7.1 Unity Digital Twin . . . 50
7.2 Interpretation of Calibration Performance . . . 50
7.3 Observability Limits . . . 51
7.4 Conditioning of the Jacobian . . . 52
7.5 Practical Implications for FleX-ray . . . 52
7.6 Limitations of the Proposed Approach . . . 53
7.7 Future Work . . . 53

8 Conclusion **54**
8.1 Key Findings . . . 54
8.2 Implications for Future Digital Twin Systems . . . 55
8.3 Final Remarks . . . 55

References **59**

A Appendix A: Experimentation Results **60**
A.1 Experiment 1 — Effect of Bead Count . . . 60
A.2 Experiment 2 — Effect of Phantom Geometry . . . 61
A.3 Experiment 3 — Effect of LM Damping . . . 62
A.4 Experiment 4 — Real FleX-ray Calibration . . . 63
A.5 Experiment 4 — Real FleX-ray Calibration Across All N . . . 64

1 Introduction

The discovery of X-rays by Wilhelm Conrad Röntgen in 1895 marked a fundamental turning point in imaging science. Over the last century, this has become indispensable in fields such as medical science, industrial inspection, material analysis, security screening, and more, becoming one of the most widely used non-destructive imaging techniques.

Computed Tomography (CT) is the principle in which a 3D digital interior reconstruction is computed from multiple X-ray projections at different angles. This is necessary and helpful for distinguishing internal structures. Modern CT systems employ different computational methods to solve this inverse problem, transforming attenuated projection measurements into volumetric reconstructions [8]. Advances in detector technology, computing power, and reconstruction methods have led to the widespread adoption of cone-beam CT systems. This drives research to make CT faster, more robust, and more reliable. In cone-beam CT, even small deviations in source or detector position propagate into the reconstruction as geometric artifacts, degrading image quality and quantitative accuracy. Thus, accurate reconstruction critically depends on precise knowledge of imaging geometry, making geometric modeling and calibration essential components of modern CT systems.

1.1 Motivation

Modern cone-beam CT systems, such as the FleX-ray scanner at Centrum Wiskunde & Informatica (CWI), are made modular and reconfigurable. This flexibility allows us to test different geometric scanning scenarios, which can be especially helpful for experimentation and furthering scientific research. Physical experimentation with CT systems is costly and time-consuming; any error in experimental setup can lead to lost time and money. Pushing the limits is part of scientific research; non-standard configurations increase the risk of geometric misconfiguration or mechanical collision, which can trigger the machine's safety guards, halting the experiment or causing it to operate inaccurately.

Virtual experimentation environments are an attractive alternative. A **digital twin** can enable scanning configuration, trajectory testing, and validation of geometric assumptions without requiring physical access to the machine. It can also run extensive simulated experiments, iterating over multiple lifecycles. For this, the digital twin needs to be geometrically consistent with the physical machine. The machine's modularity allows the source, detector, and sample to move independently. This can lead to small variations in position, causing inconsistencies that propagate into projection artifacts. A digital counterpart of the machine also has the same geometric inconsistencies if not properly calibrated. To overcome this, a calibration technique needs to be incorporated into the digital twin.

Thus, the central challenges are twofold. First, a geometrically consistent, configurable, and interactive digital twin of the FleX-ray system must be constructed using a viable and configurable 3D engine such as Unity. This needs to be architected in such a way that it is accessible to researchers through both user interface and programmatic APIs. It can then be coupled with a forward projection library such as the ASTRA Toolbox[26] to simulate the projections for the geometry of the digital twin. Secondly, a digital twin's virtual geometry must closely match the physical machine's current state, achieved through a calibration step. A non-linear geometric calibration problem is formulated to solve this using projection measurements by minimizing the

reprojection error, while analyzing the identifiability and stability of the estimated parameters. The projections from the FleX-ray machines can be compared with the ones generated using the forward projection library to achieve this.

1.2 Approach

Calibration is performed using a phantom, a physical reference object with known geometry that has metallic markers or beads embedded inside it. The bead positions detected in the projections are then used to formulate a nonlinear least-squares problem that minimizes the reprojection error between measured and predicted bead positions to estimate geometric offsets. This optimization is solved iteratively using the Gauss-Newton method, which is an iterative optimization algorithm for nonlinear least-squares problems that approximates the Hessian using the Jacobian matrix, enabling efficient parameter updates without requiring second-order derivatives. Levenberg-Marquardt extends Gauss-Newton by adding a damping term to the normal equations, interpolating between Gauss-Newton steps and gradient descent steps to improve convergence stability when the initial estimate is far from the solution.

1.3 Research Questions

This Thesis investigates the following research questions:

- ***RQ1*** - What geometric assumptions and design decisions are required to develop a consistent and interactive digital twin of the modular FleX-ray system using the Unity game engine?
- ***RQ2*** - Which geometric parameters of the cone-beam system need to be estimated for calibration, and can they be uniquely identified from bead projection measurements?
- ***RQ3*** - How does the spatial distribution of embedded beads in a calibration phantom affect parameter identifiability and estimation uncertainty during calibration?
- ***RQ4*** - What are the trade-offs and minimum conditions regarding the number of beads and projection angles required to achieve stable calibration?
- ***RQ5*** - How sensitive is the convergence of the Gauss-Newton optimization to initial estimation errors in the geometric parameters, and under what conditions does Levenberg-Marquardt regularization improve robustness?

1.4 Contributions

The main contributions of this thesis include:

1. A geometrically consistent and interactive digital twin of the FleX-ray system, implemented in Unity with kinematic motor modeling, collision detection, and a real-time geometry export pipeline to the ASTRA Toolbox.
2. A nonlinear least-squares calibration framework using a low-cost LEGO-bead phantom, solved via Gauss-Newton with Levenberg-Marquardt regularization, formulated to estimate object-stage geometric offsets while treating source and detector parameters separately to avoid parameter coupling.

3. An analysis of geometric parameter identifiability showing that non-coplanar, spatially distributed bead configurations across all three axes are necessary for stable and accurate calibration, and that coplanar or axis-aligned arrangements lead to ill-conditioned Jacobians.

4. A sensitivity analysis establishing that a minimum of three beads is required for calibration, with fewer beads requiring significantly more projection angles for stable convergence, and that local minima can be detected via the final cost value and resolved by increasing angular sampling.

1.5 Thesis Structure

Chapter 2 introduces the foundations of X-ray computed tomography and describes the technical context of the FleX-ray system, ASTRA toolbox, and Unity engine. Chapter 3 reviews related work on digital twins and geometric calibration methods. Chapter 4 presents the design and implementation of the digital twin. Chapter 5 formulates the geometric calibration problem mathematically. Chapter 6 evaluates the proposed framework experimentally using simulated and real data. Chapter 7 discusses the implications and limitations of the results. Finally, Chapter 8 concludes the Thesis.

2 System Context and Technical Foundations

2.1 Fundamentals of X-ray Computed Tomography

Tomography comes from Greek *tomos*, meaning a section or a slice, and *graphos* to describe. Computed tomography (CT), therefore, is a method in which the inner slices of an object are obtained by calculating the intensity changes of the X-ray as it passes through the object, rather than physically slicing it. The X-ray shadow produced by an object at a single angle is called a *projection*. Today, full 3D images can be obtained with enough projections and a compatible X-ray machine. CT scans are most commonly used for medical and biological imaging, nondestructive inspection and testing, materials science, and security screening.

X-rays are a form of electromagnetic radiation with a frequency ranging from 30 petahertz to 30 exahertz, higher than ultraviolet radiation but lower than gamma rays. This characteristic allows these rays to pass through an opaque object, becoming attenuated as they pass through different materials. The X-rays, when passed through an object, interact with the atoms in several ways, such as absorption (the photoelectric effect), scattering, or phase shifts. During absorption and inelastic scattering, the energy of X-rays is transferred to atoms in the material, thereby reducing the intensity of the X-ray beam. This principle forms the basis from which most CT reconstruction methods are derived. Interactions that are not modeled, such as beam hardening, scatter, or detector noise, can introduce artifacts into the reconstructed volume.

$$\text{Object} \xrightarrow{\text{CT Setup}} \text{Radiographs.} \tag{2.1}$$

The energy of an X-ray determines its *penetration length*, which is a measure of how deep the electromagnetic radiation can penetrate into material. X-rays can be classified in to three based on their energy, soft($\leq 10keV$) which hardly pass through any object, hard ($\geq 250keV$) which passes without any interaction, and medium (in between). The appropriate energy range, therefore, depends on the material composition and thickness of the object under examination: energies must be high enough to penetrate the object, yet low enough that sufficient attenuation contrast is produced. In industrial imaging, X-rays ranging from 50 keV to 250 keV are frequently used.

Computed Tomography (CT) works by combining multiple projections acquired from different angles around the object to recover its internal structure. The resulting digital representation is called a reconstruction volume, a 3D image that describes local X-ray absorption within the object.

$$\text{Radiographs} \xrightarrow{\text{Reconstruction}} \text{Object.} \tag{2.2}$$

Most reconstruction methods do not operate directly on raw detector measurements, but instead require pre-processed projection data. In an idealised case, each radiograph is described by the Beer–Lambert law. For a ray i connecting the X-ray source to detector pixel i, the measured intensity decays through the object according to

$$I_i = I_i^0 \exp\left(-\int_{l_i} \mu(\boldsymbol{\eta})\, d\boldsymbol{\eta}\right), \tag{2.3}$$

where I_i^0 is the incident intensity at the source, I_i is the intensity measured at detector pixel i, l_i is the straight-line path of ray i through the object, $\boldsymbol{\eta}$ denotes position along that path, and $\mu(\boldsymbol{\eta})$ is the local X-ray attenuation coefficient. In a cone-beam geometry, one such ray exists for each

detector pixel, and a full projection consists of all rays cast simultaneously from a point source to the detector plane at a given acquisition angle.

X-ray intensities are recorded as discrete *detector counts*, representing the number of photons measured at each pixel [8]. Assuming that the attenuation of air is negligible, measurements taken in air under the same experimental conditions form a *flatfield*, which provides an estimate of I_i^0. Taking the negative logarithm of the intensity ratio yields the *absorption image* or *projection*:

$$\log\left(\frac{I_i^0}{I_i}\right) = \int_{l_i} \mu(\boldsymbol{\eta})\, d\boldsymbol{\eta}. \tag{2.4}$$

The projection is linearly related to the attenuation coefficient $\mu(\boldsymbol{\eta})$ integrated along ray l_i, which accounts for all attenuating material the ray passes through.

The Beer–Lambert model assumes a monochromatic source, but in practice, most X-ray sources emit a polychromatic spectrum. Since $\mu(\boldsymbol{\eta})$ is energy-dependent, lower-energy photons are attenuated more strongly than higher-energy ones, so the effective attenuation decreases as the beam traverses dense material. This phenomenon, known as *beam hardening*, violates the linear relationship in Equation 2.4 and can introduce cupping artifacts and streaks in the reconstructed volume [11]. A related effect, *photon starvation*, occurs when highly attenuating objects absorb nearly all incident photons along certain rays, producing detector readings dominated by noise rather than signal. Neither effect is captured by the idealized model and represents a source of artifact in practical CT acquisitions.

2.2 Cone-Beam CT

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 around the object to acquire an equiangular range of absorption images. This setup geometry, which is the "forward problem" of equation 2.1, is termed circular cone-beam CT (CBCT) and visualized in figure 2.1.

The textbooks by Hansen [8] and Kak & Slaney [11] cover the mathematics and methods of tomographic reconstruction in detail. Mathematically, the inverse problem in Equation 2.2 amounts to solving a linear system of the form:

$$\boldsymbol{W}\boldsymbol{v} = \boldsymbol{p} \tag{2.5}$$

for the unknown volume $\boldsymbol{v}$. Here, $\boldsymbol{p}$ is a stacked vector, assembling the pixels from all absorption images, and W denotes the linear X-ray operator. Each i-th row of W corresponds to a discretized line integral, running from the X-ray source to the detector pixel associated with the index i. Most values in W are empty, as each line integral uses only a few voxels in $\boldsymbol{v}$ to compute the pixel i.

The system equation 2.5 is in general overdetermined when the number of detector measurements exceeds the number of voxels, and underdetermined in limited-angle or sparse-view scenarios where few projections are acquired. In practice, even when formally overdetermined, the system is poorly conditioned due to the sparse structure of W and measurement noise in $\boldsymbol{p}$. Direct inversion is therefore infeasible, and reconstruction is instead posed as a least-squares problem $\min_{\boldsymbol{v}} \|W\boldsymbol{v} - \boldsymbol{p}\|_2^2$, typically solved using iterative algorithms such as FBP, SIRT, or CGLS [8].

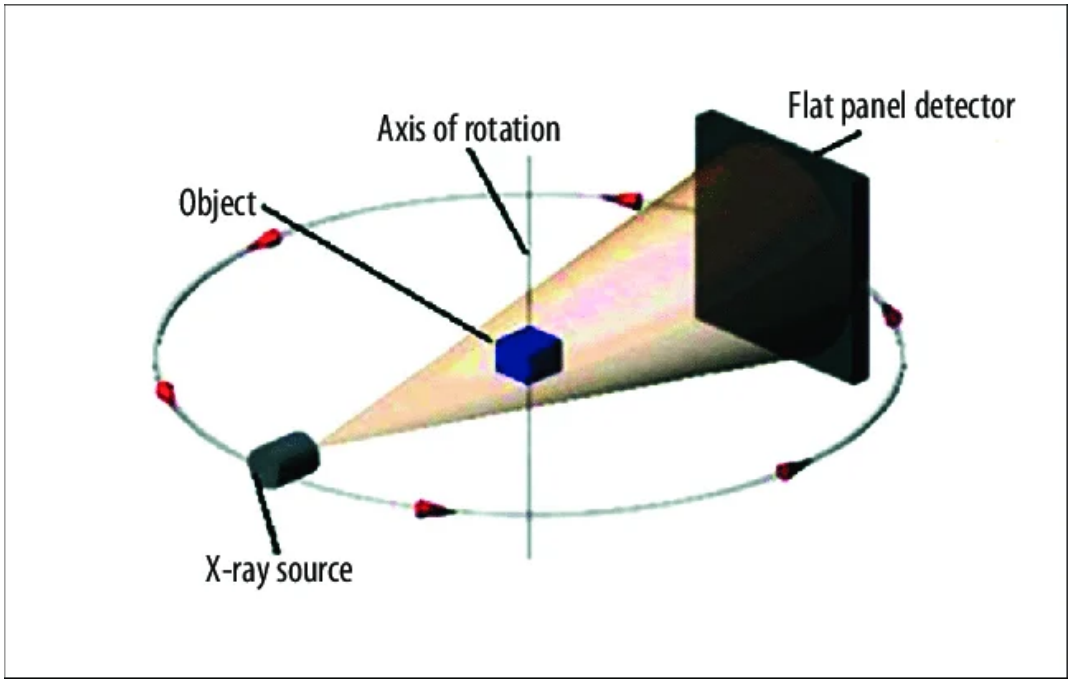


Figure 2.1: A CBCT setup: Radiographic projections of the opaque interior of the object are acquired via illumination of the object with a divergent X-ray beam. The setup or object is rotated around a central axis to obtain the radiographs from equidistant angles.[23]

2.3 FleX-ray System Overview

The FleX-ray machine is a custom-built, highly flexible X-ray Computed Tomography (CT) scanner developed by XRE nv (https://xre.be/) and located in the FleX-ray Lab at the Centrum Wiskunde & Informatica (CWI) in Amsterdam, the Netherlands, used for scientific research and experimentation.

The scanner consists of a cone-beam microfocus X-ray point source (limited to 90 kV and 90 W) that projects polychromatic X-rays onto a flat-panel CMOS detector with a CsI(Tl) scintillator (Dexella 1512NDT, [4]), comprising 1536×1944 pixels of $74.8\,\mu m^2$ each with 14-bit depth. A motorized rotation stage is positioned between the source and detector, upon which the sample is mounted. All three components are mounted on translation stages, which allow them to move independently from one another. A schematic view of the setup with the description of possible movements is shown in Fig. 2.2. [22]

Acquila is the user interface to manage the main features of the FleX-ray scanner: motor movements, x-ray power and voltage, sample position, and so on. Moreover, it also allows streaming what the flat panel detects while scanning is in progress. The motors can be programmed for the full duration of a scan, or can be interrupted to take on new commands from the operator during data acquisition. The movement of the motors is controlled using the parameters listed in 2.1. In addition to programmable motors, we can also change tube or camera settings during an experiment for better-suited data acquisition around an object or to follow an unconventional experimental plan. The sample stage offers a continuous rotation, and the data are discretized by the exposure rate of the camera at that time, and are then stored as unsigned 16-bit integer TIFF files in the given directory.

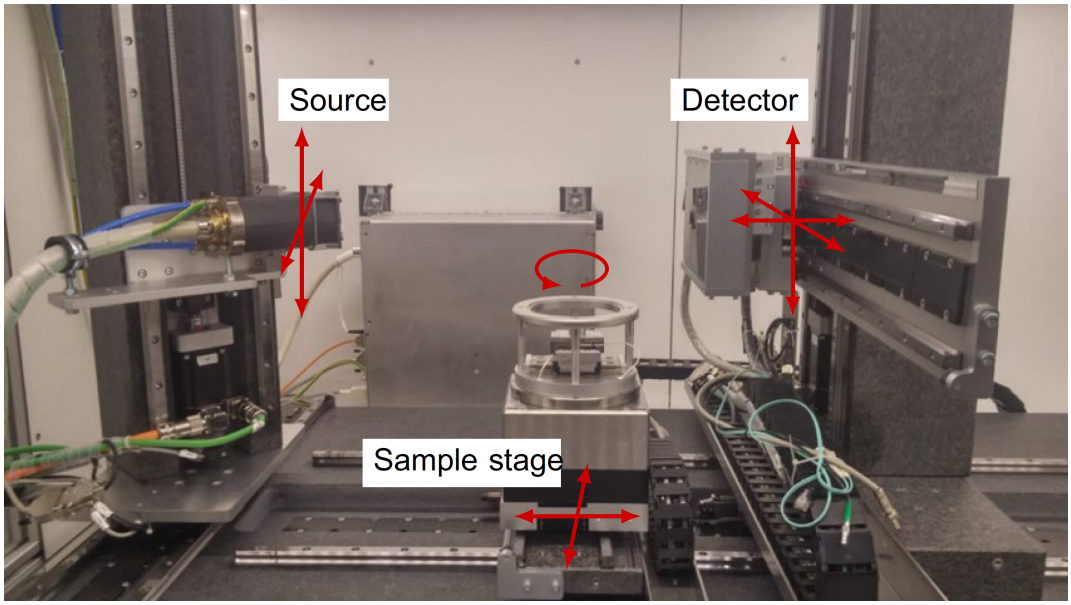


Figure 2.2: FleX-ray Lab: the X-ray cone-beam tomography setup used for the data acquisition. The arrows indicate the degrees of freedom, from [22].

Table 2.1: Overview of scanner motion parameters, including source, detector, and sample stage movements with their corresponding directions and operational limits.

Parameter	**Description**	**Limits**
ver tube	Moves the source up (+) and down (-).	0 (lower), 379 (upper)
ver det	Moves the detector up (+) and down (-).	4 (lower), 350 (upper)
ver obj	Moves the source and the detector together up (-) and down (+) to align their height with the sample stage.	–
tra tube	Moves the source left (+, toward scanner door) and right (-, toward scanner bottom).	419 (right), -359 (left)
tra det	Moves the detector left (-, toward scanner door) and right (+, toward scanner bottom).	404 (right), -459 (left)
tra obj	Moves the sample stage left (-, toward scanner door) and right (+, toward scanner bottom).	194 (right), -234 (left)
mag det	Moves the detector back (+, toward the wall) and forth (-, toward the source).	1059 (back), 100/133 (forth)
mag obj	Moves the sample stage back (+, toward the wall) and forth (-, toward the source).	996 (back), 70 (forth)
rot obj	Rotates the sample stage.	–

2.3.1 Geometry and nomenclature

The FleX-ray scanner hardware comprises one fully flexible tube (source), a detector, and a rotation stage. This is defined by a set of parameters that encode all component positions and orientations, reference coordinate conventions, and information for a tilted stage or detector. It also standardizes

the x-, y-, and z-directions and rotation conventions.

- **x**: magnification axis (source-detector line) (positive towards detector)
- **y**: horizontal axis, orthogonal to magnification (positive away from scanner door)
- **z**: vertical axis, orthogonal to magnification (positive upwards)

In FleX-ray, the origin (0,0,0) is the lowest vertical position of the source in the center of the traversal(y) axis. StaticGeometry is built from the current motor position readings, SOD, and SDD. Laboratory CT systems have a natural zooming capability, since the X-rays emanate from a point source and are projected onto a linear (fan-beam) or planar (cone-beam) detector. Therefore, moving the object closer to the source magnifies the projected image on the detector. This is characterized by the magnification factor $\beta = \frac{SDD}{SOD}$. Moving the object closer to the source increases the magnification factor, which decreases the minimal voxel size given by $\text{vox\_sz} = \text{det\_pix\_sz} * \frac{SDD}{SOD}$. As the object is moved closer to the source, the projections of the object may become truncated. Consequently, only a part of the object, the region of interest (ROI), is consistently projected onto the detector at every angle. This causes truncation artifacts in the reconstruction.

2.4 ASTRA Toolbox Overview

The All Scale Tomographic Reconstruction Antwerp (ASTRA) Toolbox is an open-source, GPU-accelerated software library for forward projection and 3D image reconstruction in tomography, developed jointly at the University of Antwerp and the Centrum Wiskunde & Informatica (CWI) in Amsterdam [21, 26, 27]. It was designed to address a fundamental limitation of conventional tomography software: most existing packages restrict users to a fixed acquisition geometry and a small, predetermined set of reconstruction algorithms. Both restrictions are addressed by providing flexible, high-performance building blocks that can be combined freely to construct advanced reconstruction pipelines and to simulate projection data for arbitrary scanning configurations [21].

At its core, the toolbox exposes two GPU-accelerated operators: the *forward projector* (FP), which computes the line-integral projections of a volume through a specified geometry, and the *backprojector* (BP), its adjoint. These operators are implemented in NVIDIA CUDA and achieve single-iteration runtimes of well under one second for typical 512^3-voxel cone-beam datasets [21]. Built on top of these primitives, the toolbox ships with ready-to-use implementations of the most common reconstruction algorithms—Filtered Backprojection (FBP), the Feldkamp–Davis–Kress (FDK) cone-beam reconstruction, the Simultaneous Iterative Reconstruction Technique (SIRT), the Simultaneous Algebraic Reconstruction Technique (SART), Conjugate Gradients Least Squares (CGLS), and Expectation Maximisation (EM)—all of which run on the GPU without modification to the user's script [26, 27].

Internally, ASTRA is layered to provide direct access to CUDA kernel implementations of the forward projection (FP) and back projection (BP) operators [21, 27]. By calling these operators directly from Python (e.g., via `astra.experimental.direct_fp`), the overhead of algorithm configuration and input/output buffer allocations associated with full reconstruction pipelines is bypassed. This low-level access enables the forward-projection evaluations used during the iterative calibration loop to complete in milliseconds per evaluation, meeting the computational requirements of the interactive digital twin. Both MATLAB and a Python interface are provided, allowing ASTRA to be embedded seamlessly into existing scientific computing workflows.

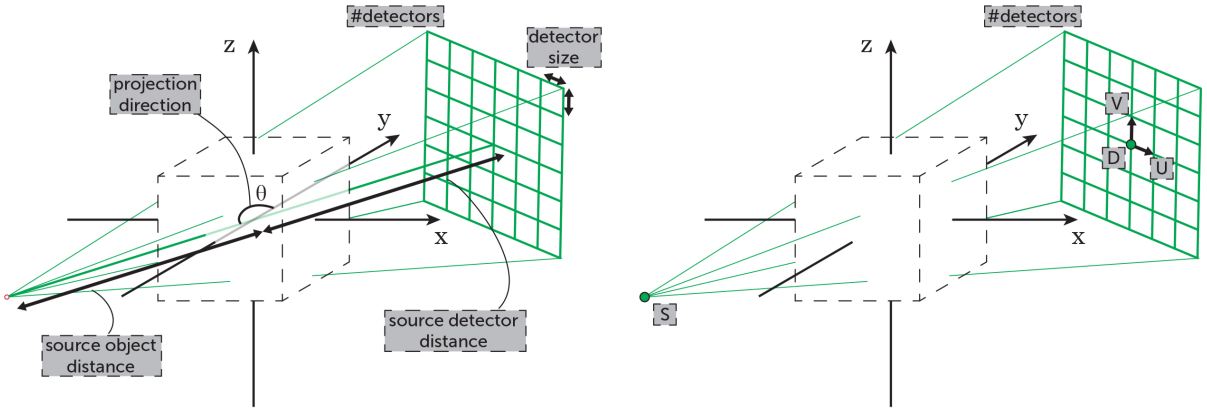
2.4.1 Geometry

ASTRA operates within a standard mathematical **right-handed coordinate system** (typically assuming Z as the vertical axis). This formulation maps directly to the physical FleX-ray parameters defined in Section 2.3.1. A distinguishing feature of ASTRA is its fully *vector-parametrised* projection geometry. Rather than restricting the user to a small set of named scanning modes, each projection angle in a 3D cone-beam setup is described by four 3D vectors. The geometry of projection is described by 12 floating point numbers: source position $[s_x, s_y, s_z]$, detector center x position $[d_x, d_y, d_z]$, detector 3D basis vector from pixel $(0, 0)$ to pixel $(1, 0)$ - horizontal basis $\vec{u}$ - parametrized as u_x, u_y, u_z, and detector 3D basis vector from pixel $(0, 0)$ to pixel $(0, 1)$ - vertical basis $\vec{v}$ - parametrized by $[x, y, z]$ as v_x, v_y, v_z. This parametrization is illustrated in Fig. 2.3. By changing these four vectors per projection, the same toolbox code transparently handles single-axis, dual-axis, and multi-axis tilt series; cone-beam, fan-beam, and parallel-beam geometries; and arbitrary misalignments such as detector tilt, lateral offset, or a non-centred rotation axis—all with sub-pixel accuracy and without any interpolation on the raw data [26, 27]

Mathematically, the ASTRA cone-beam geometry (`cone_vec`) for a single projection is instantiated by concatenating these four vectors into a 12-element row vector:

$$\boldsymbol{\theta}_i = [s_x, s_y, s_z, d_x, d_y, d_z, u_x, u_y, u_z, v_x, v_y, v_z] \in \mathbb{R}^{12} \tag{2.6}$$

A complete CT acquisition of N projections is thereby defined by an $N \times 12$ matrix.



(a) default (b) vector-based

Figure 2.3: Parametrization of the cone-beam geometry: Each projection is described by 12 floating point numbers: $(s_x, s_y, s_z, d_x, d_y, d_z, u_x, u_y, u_z, v_x, v_y, v_z)$, from [26]

While the `cone_vec` geometry allows for arbitrary 3D positioning, ASTRA's underlying mathematical framework uses a standard right-handed Cartesian coordinate system. The typical convention of ASTRA 3D geometries defines the origin $(0, 0, 0)$ at the center of the reconstructed volume, the z-axis represents the vertical axis of rotation, the y-axis aligns with the primary X-ray beam direction (pointing from source to detector), and the x-axis is orthogonal in the horizontal plane. This axis convention differs from the physical FleX-ray geometry defined in Section 2.3.1, in which the source is at the origin. $(0, 0, 0)$

2.4.2 FleX-ray Mapping

For **forward projection**, ASTRA's vector-based cone-beam geometry allows every candidate FleX-ray configuration—defined by the source-to-detector distance (SDD), the source-to-object distance (SOD), the detector orientation, and the rotation-axis offset—to be instantiated as an ASTRA projection geometry with a few lines of Python code. Evaluating each projection, $\boldsymbol{W}(\boldsymbol{\theta})\,\boldsymbol{v}$ for a given volume $\boldsymbol{v}$ and geometry $\boldsymbol{\theta}$, then takes milliseconds on a GPU, producing synthetic radiographs that are geometrically consistent with those the physical scanner would acquire.

For **auto-calibration**, the same forward projector is embedded inside an optimization loop. Given a phantom volume $\boldsymbol{v}$ (derived from a CAD model or a reference scan) and a set of real scanner measurements $\boldsymbol{p}$, the calibration procedure minimizes

$$\min_{\boldsymbol{\theta}} \; \|\boldsymbol{W}(\boldsymbol{\theta})\,\boldsymbol{v} - \boldsymbol{p}\|_2^2\,, \tag{2.7}$$

Here $\boldsymbol{p}$ denotes the stacked vector of observed bead positions detected across all projections, playing the same role as the measurement vector in equation 2.5. The vector $\boldsymbol{\theta}$ collects the geometric parameters to be recovered, including source position, detector position, and orientation. Because ASTRA can reconfigure a geometry object from a new vector set in microseconds, the optimization loop can evaluate hundreds of candidate geometries per second, making iterative calibration computationally feasible in an interactive laboratory setting.

The ASTRA Toolbox acts as the core computational engine of the digital twin: its flexible geometry representation bridges the abstract mathematical description of the FleX-ray's scanning geometry to fast, GPU-accelerated projection operations that drive both the simulation and the calibration components of the system.

2.5 Unity Engine Overview

2.5.1 3D Engine Overview and Use Cases

Unity is a cross-platform real-time 3D development engine first released in 2005 with the stated goal of democratising interactive content creation. Although its roots lie in video-game development, the engine has since been adopted across a broad range of industries—including film, automotive, architecture, engineering, and defence—wherever interactive real-time 3D visualisation or simulation is required.

The engine exposes its functionality through a primary scripting interface written in C# and built on the .NET runtime, giving developers access to a high-level API that spans rendering, physics, audio, animation, and networking. This combination of an accessible scripting layer and a capable rendering backend has made Unity a popular choice for scientific visualisation and digital-twin applications, where large simulation datasets must be explored interactively in real time [28, 29].

2.5.2 World Coordinate Conventions, Axis Orientation, Transform Hierarchy, and Scaling

Axis orientation. Unity uses a *left-handed*, Y-up, Z-forward Cartesian coordinate system. Concretely:

- The **X axis** points to the right.

- The **Y axis** points upward.
- The **Z axis** points forward (into the screen in the default camera orientation).

In the Scene View these axes are colour-coded red, green, and blue, respectively . The left-handedness of the system differs from the right-handed convention used in most mathematical and physical literature as well as in software packages such as Blender, a fact that must be accounted for whenever geometry or coordinate data are imported from external sources.

Transform hierarchy. Every object in a Unity scene is called a *GameObject*, and every GameObject carries a `Transform` component that encodes its position, rotation, and scale. Transforms may be nested: when a GameObject is designated the parent of another, the child inherits the parent's transformation, so that moving or rotating the parent moves or rotates the entire subtree. Positions and orientations can be expressed either in *world space* (relative to the scene origin) or in *local space* (relative to the parent's transform); the scripting API exposes both through the `transform.position`/`transform.localPosition` and the corresponding rotation properties.

Scaling and unit conventions. By default, Unity defines **one world unit as one metre**, a convention that is shared with the built-in physics engine . Deviating from this convention—for example, by applying non-uniform scale factors high up in the hierarchy—can produce counter-intuitive results in physics simulation and should be avoided where possible. Scale can be set independently along each axis (`transform.localScale`), allowing non-uniform stretching of meshes. Still, non-uniform scaling of parent objects propagates to all children, potentially distorting the shapes of other connected components in Unity.

2.5.3 Components

Unity follows a *component-based* architecture: the behaviour and appearance of a GameObject is determined entirely by the components attached to it rather than by inheritance from a fixed class hierarchy. A camera, a directional light, and a rigid body are all simply components that can be added to or removed from any GameObject at design time or at runtime via the scripting API.

The most fundamental component is the `Transform`, which is mandatory and cannot be removed. Additional components commonly used in simulation contexts include:

- **MeshFilter** and **MeshRenderer** — supply geometry and material properties for visual representation.
- **Collider** — defines the shape used for collision detection (box, sphere, capsule, mesh, etc.).
- **Rigidbody** — enables physics simulation on the owning GameObject (see Section 2.5.4).
- **MonoBehaviour** subclasses — user-defined C# scripts that respond to engine lifecycle events (`Start`, `Update`, `FixedUpdate`, ...) and implement custom logic.

2.5.4 Physics

Unity's 3D physics simulation is powered by the NVIDIA *PhysX* engine [3]. Physics behaviour is activated on a GameObject by attaching a `Rigidbody` component, which exposes properties such as mass, drag, angular drag, and gravity scale, and which causes the physics solver to apply forces, detect collisions, and resolve contacts each fixed time step [25].

Collision geometry is provided by *Collider* components, which are independent of the visual mesh. Mesh Colliders provide exact shape matching but are computationally expensive; primitive colliders (box, sphere, capsule) are preferred for performance-critical objects [25]. Objects without a `Rigidbody`—so-called *static colliders*—do not move under physics forces and are optimised accordingly by the engine; applying `Transform` changes to them at runtime is discouraged because it forces an expensive rebuild of the internal acceleration structures [25].

Joints connect two `Rigidbody` instances (or a `Rigidbody` to a fixed point in world space) and constrain their relative motion. Available joint types include the Fixed Joint, Hinge Joint, Spring Joint, Character Joint, and Configurable Joint, the last of which allows an arbitrary combination of linear and angular degrees of freedom to be locked, free, or spring-constrained [25].

Physics updates run on a fixed time step (`Time.fixedDeltaTime`, default 0.02 s), decoupled from the rendering frame rate. The engine assumes SI units; in particular, the correspondence between one world unit and one metre is critical for obtaining physically plausible simulation results.

3 Related Work

3.1 Digital Twin Systems

3.1.1 Definition and Core Principles

Digital twin is a technology that accomplishes the mapping of the real world to the digital world and realizes the interaction between them in real time. In its most widely accepted definition, a digital twin is a computational model that maintains a continuously updated virtual representation of a physical asset, system, or process, synchronizing with its real-world counterpart through sensor data and capable of predicting, diagnosing, or controlling the physical system's behavior [31, 1]. This allows the technology to overcome the constraints of real-world environmental factors and enables users to familiarize themselves with or explore the machine without concerns about safety requirements or real-world constraints.

Systematic reviews of digital twin technology identify two foundational technical requirements: (i) a *high-fidelity virtual model* that accurately captures the geometry, physics, and behavior of the physical asset, and (ii) *real-time or near-real-time synchronization* between the physical and virtual domains. A multi-domain model that combines geometric, kinematic, and signal-processing models is essential for complex equipment in which different physical phenomena evolve on different timescales [31].

The digital twin developed in this Thesis follows a modular architecture: a Unity-based digital twin, an independent Python ASTRA server, and a Calibration module.

3.1.2 Applications in Industrial and Imaging Systems

Digital twins have been deployed across a wide range of industrial and scientific domains, demonstrating the value of maintaining a live virtual replica alongside the physical system.

Radiology and CT equipment. Aghamiri *et al.* [1] survey digital twin applications in radiology and identify real-time bidirectional data synchronisation as the key differentiator between a passive simulation tool and an operational twin. They report applications of CT and MRI digital twins for remote monitoring, predictive maintenance, and detector calibration, noting that continuous comparison of measured performance metrics with the virtual model enables detection and correction of anomalies before they propagate to clinical measurements.

Manufacturing and robotics. In the manufacturing domain, digital twins of CNC machine tools [13], production lines [12], and robotic assembly systems [15] have shown that pre-commissioning simulation reduces setup time and calibration errors, and that continuous comparison between virtual predictions and physical measurements enables early fault detection. These systems share a common pattern with the present Thesis: a physics-based simulation engine is kept in synchronisation with a physical system, and it is able to identify errors in the scanning script and provide safety checks.

Robotic CT systems. Twin robotic CT (RoboCT) systems, in which both X-ray source and detector are mounted on industrial robots, represent an extreme case of geometric variability: the acquisition geometry changes at every projection and is subject to millimetre-scale positioning uncertainty. Schielein *et al.* [10] identify projection-wise geometry estimation as the principal open

challenge in this setting, and Handke *et al.* [5] address it through an online calibration method that uses spheres of unknown position attached to the specimen, demonstrating robustness to positioning errors without dedicated calibration scans. The reconfigurable geometry of the FleX-ray system motivates a similar approach, where the geometric model must be updated reliably after any external change to the physical setup.

This Thesis presents the combination of arbitrary reconfigurable geometry and tight integration with an iterative auto-calibration loop for the calibration of FleX-ray digital twin. By using the ASTRA Toolbox's vector-based cone-beam projector, the simulation environment developed in this thesis can represent *any* FleX-ray geometric configuration, evaluate synthetic projections, and embed this evaluation directly inside a geometry optimisation loop—capabilities that are not available off-the-shelf in any existing CT simulation framework.

3.2 Geometry Calibration in Cone-Beam CT

Accurate geometric calibration is fundamental for high-resolution Cone-Beam CT. Small misalignments in source, detector, or rotation axis propagate through the inverse problem 2.5, resulting in blurring, streaking, or artifacts in the reconstructed volume. Calibration methods, such as self-calibration and phantom-based methods, have been developed to calibrate the geometries of physical CT systems.

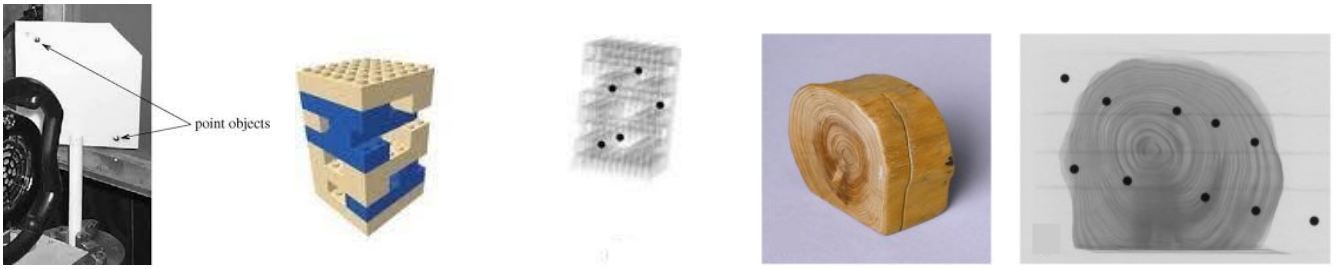
A digital twin must also perfectly replicate the physical scanner's configuration, and its virtual geometry must also be synchronized with the physical machine. Inspirations from established physical calibration techniques can help us formulate a calibration pipeline for the digital twin.

3.2.1 Self-Calibration vs. Phantom-Based Methods

Self-calibration methods attempt to estimate geometric parameters directly from the projection data of the unknown sample being scanned. These approaches iteratively adjust geometry parameters to maximize image-quality metrics (such as edge sharpness or entropy) in the reconstructed 3D volume.

While self-calibration is attractive because it removes the need for a dedicated calibration object, it was rejected for the digital twin for one fundamental reason: **the requirement of a prior digital model**. The digital twin's calibration loop operates by matching simulated forward projections against real physical measurements. To compute these simulated projections accurately, the twin requires an exact 3D digital replica (such as a CAD model) of the object being scanned. In practical operational scenarios, the internal structure of the sample is exactly what is unknown, making object-based calibration impossible. Furthermore, traditional self-calibration algorithms that attempt to bypass this by iteratively computing a full 3D reconstruction at every optimization step are computationally prohibitive for interactive, real-time synchronization.

Instead, a **phantom-based method** was chosen. By scanning a dedicated physical calibration phantom, we can construct an exact, matching 3D digital counterpart in the Unity environment. Calibration is then reduced to tracking high-contrast markers in the 2D projection space. This entirely bypasses the need for an *a priori* model of an unknown sample and avoids expensive 3D reconstructions, making it fast, deterministic, and ideal for feeding rapid parameter updates to a digital twin.



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

Figure 3.1: Representative phantom-based calibration approaches from the literature. **(a)** Noo et al. [19]: point objects on a flat plate used to trace elliptical trajectories for analytical geometry recovery. **(b)** Nguyen et al. [17]: low-cost LEGO phantom with embedded steel beads (right: CT cross-section showing staggered bead positions) for numerical calibration of a modular cone-beam CT system. **(c)** Bossema et al. [2]: museum wood sample augmented with steel bead markers (left: photograph, right: radiograph) for ad-hoc numerical calibration of a flexible 2D X-ray setup.

3.2.2 Analytical vs. Numerical Phantom Calibration

Once a phantom-based approach was selected, the next design choice was between analytical and numerical calibration.

Analytical approaches derive closed-form mathematical solutions by analyzing the trajectories of point markers. For instance, Noo et al. [19] demonstrated that scanner geometry could be calibrated by identifying the parameters of the ellipses traced by point markers during a standard circular cone-beam scan (see Fig. 3.1a). Building upon this, Yang et al. [30] developed a comprehensive analytic method to calculate geometric parameters directly from the spatial relationships of projected markers. However, analytical solutions were rejected for this work because they are inherently rigid. They typically assume ideal circular trajectories and fixed rotation axes. The FleX-ray scanner features independent, highly reconfigurable translation stages for the source, detector, and object. Analytical closed-form equations easily break down when applied to these multi-axis, arbitrary trajectories.

Therefore, a **numerical calibration approach** was selected. Numerical methods formulate calibration as a nonlinear optimization problem, iteratively updating the geometric parameters to minimize the difference between observed and simulated marker positions (the reprojection error). This approach is highly favored for flexible systems because it is geometry-agnostic. This choice is directly inspired by recent literature on modular systems. Nguyen et al. [17] developed a robust numerical calibration procedure specifically for a modular cone-beam X-ray CT system (see Fig. 3.1b), proving that expensive, highly-machined commercial phantoms are unnecessary if the numerical optimization framework is robust. Similarly, Bossema et al. [2], even though they use a self-calibration method, demonstrated the advantages of numerical optimization when reconstructing 3D volumes in flexible 2D X-ray setups such as the ones in museums (see Fig. 3.1c), where standard analytical assumptions completely fail. By adopting a numerical optimization framework based on ASTRA's vector-based geometry, the digital twin can seamlessly accommodate arbitrary geometric configurations of the FleX-ray scanner.

3.2.3 Phantom Design Principles in Literature

With numerical frameworks established as the standard for flexible systems, the physical design of the calibration phantom must be considered. The literature highlights three primary design principles for robust numerical calibration: marker shape, spatial distribution, and material accessibility.

Marker Shape and Localization. The accuracy of numerical calibration is strictly bounded by the precision of 2D marker extraction. The consensus in the literature strongly favors spherical metal beads over wires or complex geometric shapes [30, 17]. A projection of a 3D sphere can be approximated as a 2D circle or ellipse onto the detector regardless of the viewing angle. This allows for accurate identification of the markers in the projections, which can then be used for the calibration procedure.

Spatial Distribution of Markers. The spatial arrangement of embedded beads is a critical parameter for reliable tracking. If markers are placed on identical horizontal planes, their projected sinusoidal trajectories will intersect on the 2D detector during a rotational scan, leading to severe tracking ambiguities and data association errors. To guarantee unambiguous marker tracking across all projection angles, calibration literature routinely employs staggered, helical, or spiral 3D distributions [30]. By spatially staggering the beads along the vertical axis, their projected trajectories remain parallel and distinct, preventing overlap. The thesis also explores the reliability of the calibration for different spatial distribution patterns.

Accessibility and Custom Phantoms. Historically, geometric calibration required expensive, highly-machined commercial phantoms. However, recent works on modular systems advocate for accessible, custom-built alternatives. For instance, Nguyen et al. [17] successfully utilized a low-cost custom phantom by leveraging standard, precision-manufactured components. Their work demonstrated that as long as the numerical optimization framework is robust and the structural base provides reliable geometric rigidity, highly accurate calibration can be achieved without commercial phantoms. This precedent in the literature directly motivates the use of low-cost, precision-molded components (such as LEGO bricks) for the development of customized digital twin calibration objects.

3.3 Optimization Methods for Geometric Calibration

Once the geometric calibration task is formulated using a physical phantom, the problem reduces to finding the optimal set of system parameters that align the virtual projections with the physical measurements by minimizing the emphreprojection error—the squared Euclidean distance between the actual 2D coordinates of the markers detected in the physical radiographs and the predicted 2D coordinates generated by the mathematical forward model [17]. Because the relationship between the 3D geometric parameters (source position, detector orientation) and the 2D projection coordinates involves trigonometric projections and perspective division, the system is inherently nonlinear [9]. Consequently, the calibration must be solved using iterative numerical optimization.

3.3.1 Nonlinear Least Squares Formulation

The standard approach in computer vision and CT calibration is to frame the task as a Nonlinear Least Squares (NLS) optimization [24]. The objective is to minimize the *reprojection error*.

To achieve stability, the calibration problem is heavily overdetermined by stacking the residuals. A single projection of K beads yields $2K$ residual equations (for the u and v detector coordinates). By acquiring projections across N rotation angles, a large system of $2KN$ stacked residuals is constructed. This massive redundancy is crucial for averaging out inaccuracies in the marker detection process and improving the observability of highly coupled geometric parameters [30].

The residual vector for a single bead k in projection n is defined as the difference between the predicted and observed detector coordinates:

$$\boldsymbol{r}_{n,k}(\boldsymbol{\theta}) = \begin{bmatrix} x_{n,k}(\boldsymbol{\theta}) - \hat{x}_{n,k} \\ y_{n,k}(\boldsymbol{\theta}) - \hat{y}_{n,k} \end{bmatrix} \in \mathbb{R}^2 \tag{3.1}$$

Stacking all residuals across N projections and K beads gives the full residual vector $\boldsymbol{r}(\boldsymbol{\theta}) \in \mathbb{R}^M$, where $M = 2KN$. The calibration problem is then formulated as:

$$\min_{\boldsymbol{\theta}} \frac{1}{2} \|\boldsymbol{r}(\boldsymbol{\theta})\|_2^2 \tag{3.2}$$

3.3.2 Gauss-Newton Method

To solve the NLS problem, gradient-based descent methods are typically employed. The *Gauss-Newton (GN) method* is a widely used algorithm specifically tailored for least-squares problems [18]. Unlike standard Newton's method, which requires the computation of the dense and computationally expensive Hessian matrix (second derivatives), Gauss-Newton approximates the Hessian using only the first-order Jacobian matrix $\boldsymbol{J}$.

When the initial parameter guess is close to the true solution, and the residuals are small, the Gauss-Newton method exhibits rapid, quadratic convergence [18]. This makes it highly attractive for calibrating a digital twin, where the mechanical limits of the FleX-ray scanner provide a reasonable initial estimate. However, if the Jacobian matrix $\boldsymbol{J}$ becomes rank-deficient or ill-conditioned—often caused by an insufficient number of beads or limited angular coverage—the matrix $\boldsymbol{J}^T\boldsymbol{J}$ becomes non-invertible, leading to massive, unstable parameter updates and algorithmic divergence [24].

The Gauss-Newton method solves equation 3.2 iteratively. At each iteration t, the nonlinear residual is linearized using a first-order Taylor expansion:

$$\boldsymbol{r}(\boldsymbol{\theta}_t + \delta\boldsymbol{\theta}) \approx \boldsymbol{r}(\boldsymbol{\theta}_t) + \boldsymbol{J}(\boldsymbol{\theta}_t)\, \delta\boldsymbol{\theta} \tag{3.3}$$

where the Jacobian matrix is defined as:

$$\boldsymbol{J}(\boldsymbol{\theta}) = \frac{\partial \boldsymbol{r}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \in \mathbb{R}^{M \times P} \tag{3.4}$$

with P the number of geometric parameters. Substituting the linearization into the objective yields the normal equations:

$$\left(\boldsymbol{J}^T\boldsymbol{J}\right) \delta\boldsymbol{\theta} = -\boldsymbol{J}^T\boldsymbol{r} \tag{3.5}$$

The parameter vector is then updated as $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \delta\boldsymbol{\theta}$.

3.3.3 Levenberg-Marquardt Extension

The Gauss-Newton method can fail to converge when the Jacobian $\boldsymbol{J}$ is ill-conditioned, or the initialization is far from the solution. To address this instability of the pure Gauss-Newton method and to improve robustness, the *Levenberg-Marquardt (LM)* algorithm is universally considered the industry standard for geometric calibration and bundle adjustment [16, 24, 18]. The LM method introduces a non-negative damping parameter λ to the normal equations which acts as a regulirizer: $\left(\boldsymbol{J}^T\boldsymbol{J} + \lambda\boldsymbol{I}\right)\delta\boldsymbol{\theta} = -\boldsymbol{J}^T\boldsymbol{r}$.

Damping Strategies and Stability. This formulation acts as a hybrid approach. When λ is large, the matrix becomes diagonally dominant, and the step mimics a stable but slow gradient descent. When λ is small, the step approaches the rapid Gauss-Newton update. By dynamically adjusting λ based on whether the reprojection error increases or decreases after a step, the LM algorithm guarantees monotonic descent and robust convergence even when the initial guess is poor or the Jacobian is temporarily ill-conditioned [16]. Because modular CT systems frequently exhibit highly coupled parameters (e.g., source translations compensating for detector translations), the damping matrix is often scaled by the diagonal of $\boldsymbol{J}^T\boldsymbol{J}$ to standardize the parameter updates, a technique crucial for stable convergence in highly variable geometric spaces [17].

Algorithm 1 Gauss-Newton with Levenberg-Marquardt regularization

Require: Initial parameters $\boldsymbol{\theta}_0$, damping $\lambda > 0$, tolerance $\eta > 0$
Ensure: Calibrated parameters $\boldsymbol{\theta}^*$

$t \leftarrow 0$
repeat
  Compute residual $\boldsymbol{r}_t \leftarrow \boldsymbol{r}(\boldsymbol{\theta}_t)$
  Compute Jacobian $\boldsymbol{J}_t \leftarrow \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}_t}$ via finite differences
  Solve: $\left(\boldsymbol{J}_t^T\boldsymbol{J}_t + \lambda I\right)\delta\boldsymbol{\theta} = -\boldsymbol{J}_t^T\boldsymbol{r}_t$
  Compute trial cost: $c' \leftarrow \|\boldsymbol{r}(\boldsymbol{\theta}_t + \delta\boldsymbol{\theta})\|_2^2$
  if $c' < \|\boldsymbol{r}_t\|_2^2$ **then**
    $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \delta\boldsymbol{\theta}$
    Decrease λ
  else
    $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t$ ▷ Reject step
    Increase λ
  end if
  $t \leftarrow t + 1$
until $\|\delta\boldsymbol{\theta}\| < \eta$ **or** maximum iterations reached
return $\boldsymbol{\theta}_t$

The damping parameter λ is updated adaptively: if a step reduces the cost $\|\boldsymbol{r}(\boldsymbol{\theta})\|_2^2$, the step is accepted and λ is decreased to approach Gauss-Newton behaviour; if the cost increases, the step is rejected and λ is increased to fall back toward gradient descent. This guarantees monotonic cost reduction across iterations.

3.3.4 Advanced Global Optimization Methods

While gradient-based methods like LM are exceptionally fast, they can still become trapped in local minima if the objective function is highly non-convex. To overcome this, recent literature has explored derivative-free, evolutionary algorithms for CT calibration.

A prominent example is the *Covariance Matrix Adaptation Evolution Strategy (CMA-ES)* [7]. CMA-ES is a stochastic optimization method that iteratively samples a population of candidate geometry configurations from a multivariate normal distribution. By updating the covariance matrix of the distribution based on the most successful candidates, the algorithm effectively learns the complex, coupled geometric landscape without ever computing a Jacobian [6]. While CMA-ES provides unparalleled robustness against local minima and noisy objective functions—making it ideal for fully automated, unsupervised "blind" calibration, it requires thousands of forward projection evaluations per generation. This immense computational overhead makes evolutionary algorithms significantly slower than Levenberg-Marquardt.

4 Design and Implementation of the Digital Twin

The development of the digital twin for the FleX-ray system requires a high-fidelity virtual environment that mirrors the physical scanner's geometry, kinematic constraints, and operational logic. This chapter details the software architecture, the geometric transformations, and design decisions required to synchronize disparate coordinate systems and the integration of the Unity engine with the ASTRA Toolbox. To ensure the simulator is both a valid research tool and an interactive environment, the following requirements were established:

- **Geometric Fidelity:** The twin must replicate the degrees of freedom (DOFs) of the physical scanner, ensuring that virtual projections match physical radiographs within a predefined tolerance.
- **Kinematic Constraints:** Virtual components must respect the physical travel limits of the motorized stages to prevent the simulation of impossible scanning trajectories.
- **Safety Constraints:** The virtual components must behave such that the movements are physically possible and do not collide with each other.
- **Real-time Interactivity:** The system must provide a responsive interface where geometric changes can be made and trigger instantaneous ASTRA forward-projections for visual validation.
- **API Interactivity:** The system must provide an API interface to run scans, change motor positions, do calibrations, etc., so that users can interact with it directly from their project.

4.1 System Architecture

The digital twin is implemented using a **decoupled, client-server architecture**. This approach is necessitated by the technical incompatibility between Unity's C#-based environment and the ASTRA Toolbox's Python/C++ requirements. By separating the high-frequency visualization tasks from the computationally intensive tomographic math, the system maintains the responsiveness required for an interactive research tool.

4.1.1 Layered Software Components

The architecture is organized into three distinct layers as shown in 4.1 can operate in parallel to provide a synchronized simulation experience:

1. **Visualization and Kinematics Layer (Unity Frontend):** This layer manages the 3D rendering, user interface, and mechanical hierarchy. It enforces the physical constraints of the FleX-ray components using the joint-based kinematic modeling described in Section 4.4.
2. **Communication Middleware (TCP Socket Layer):** To facilitate cross-platform data exchange, a dedicated `SocketServer` (within Unity) and a corresponding Python client are used. This layer handles the serialization of 12-parameter geometry vectors and the transmission of simulated X-ray image data.

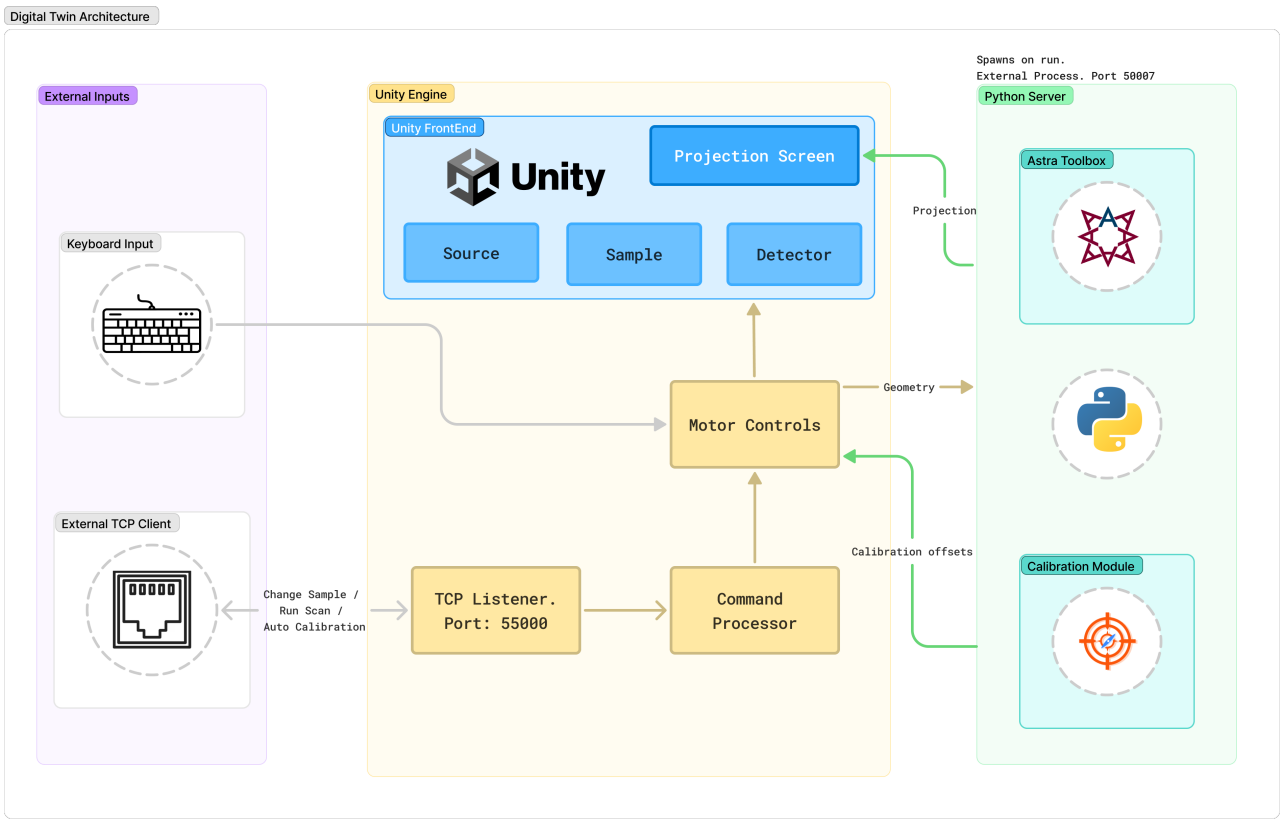


Figure 4.1: Overall system architecture of the Digital Twin System

3. **Tomographic Computation Backend (Python/ASTRA):** A standalone Python process serves as the "mathematical engine". It hosts the ASTRA Toolbox, manages the voxelized volumes of scanned objects, and executes the Gauss-Newton optimization scripts.

4.1.2 Event-Driven Synchronization Loop

Even though the software components are decoupled, the 3D animation, the projection and events need to work without interfering with each other. To achieve this, the system utilizes an **event-driven reactive loop** rather than the fixed-rate update tied to the frame rate of Unity. This minimizes network overhead and ensures that the computational backend is only engaged when the system state changes whenever a virtual motor stage moves (via manual input or an automated script).

4.2 Coordinate System Definition and Scaling

To maintain geometric consistency between the digital twin and the physical FleX-ray lab, a rigorous mapping of spatial units and axes is required. In the Unity environment, the coordinate system is defined as a left-handed Cartesian frame where:

- **X-axis (Lateral):** Represents the traversal movement of the source, sample, and detector. This axis is equivalent to the Y-axis (traversal axis) in the ASTRA/FleX-ray geometry

representation.

- **Y-axis (Vertical):** Represents the elevation of the source and detector towers. Equivalent to Z-axis (Vertical Axis).
- **Z-axis (Longitudinal):** Represents the magnification axis, controlling the Source-to-Object Distance (SOD) and Source-to-Detector Distance (SDD). Equivalent to the X-axis (Magnification Axis).

We adopt the axis representation defined by the Unity coordinate system for the remainder of this thesis, with the ***x*, *y*, and *z*** axes corresponding to **traversal, vertical, and magnification**, respectively.

4.3 Component Modeling and Hierarchy

To mirror the modularity of the physical system, the machine is modeled using three primary kinematic chains:

Source Stage: The source assembly consists of a micro-focus X-ray emitter (*Source*) mounted to a vertical *SourceTower*. This tower provides vertical translation along the Unity Y-axis. The tower itself is connected to a floor-mounted railing system (*SourceRail*), allowing for lateral traversal along the X-axis. Consequently, the Source Stage possesses two independent translational degrees of freedom, both the traversal and vertical motion of the *Source*.

Sample Stage: The Sample Stage is designed to handle object rotation, traversal, and magnification. There is no vertical movement of the sample stage, and any additional height of the *Sample* is set through the Sample mount. It consists of a circular *SampleHolder* providing a full 360° continuous rotation. This base is mounted upon a *SampleBase*, which is nested within a *SampleRail*. The assembly allows for lateral traversal along the X-axis, while the entire platform is mounted on primary railings along the Z-axis. This longitudinal movement is critical for defining the Source-to-Object Distance (SOD) and the resulting geometric magnification factor.

Detector Stage: The Detector Stage mirrors the elevation and traversal capabilities of the source but resides on the primary longitudinal rails shared with the sample stage. The flat-panel detector is mounted on a *DetectorTraversalStage* equipped with an independent railing for lateral (X-axis) movement. This platform is integrated into a *DetectorTower* providing vertical (Y-axis) elevation. By moving the entire tower assembly along the Z-axis railings, the system can dynamically adjust the Source-to-Detector Distance (SDD) independently of the sample position.

In Unity, these components are grouped together and connected with Unity Joints for easier control. This ensures that when a base platform moves, all its connected railings, towers, and emitters move with it automatically, preserving their relative offsets. For example, the *SampleHolder* is connected with *SampleBase*, which is a child of the *SampleRail*. This mirrors the physical-mechanical stages of the FleX-ray scanner, where moving the longitudinal magnification rail (Z) shifts the entire sample assembly without disturbing the lateral traversal (X) or the rotational state.

The 3D meshes representing the FleX-ray components in Unity were sourced from a Blender project originally created for the doctoral thesis of Maximilian Kiss [14]. The models were exported from Blender as FBX files, including their associated materials, and imported into the Unity scene. To align the models with Unity's left-handed Y-up coordinate system and the 1:1 millimeter-scale convention established in Section 4.3.1, each mesh underwent an axis rotation and uniform rescaling.

Minor material adjustments were applied to ensure compatibility with Unity's rendering pipeline while preserving the visual fidelity of the physical FleX-ray components.

Table 4.1: Geometric specifications and kinematic roles of the digital twin components.

Category	Component	Dimensions (mm)	Axis of Motion
Base	Main Platform	$2000 \times 50 \times 2000$	Fixed (Origin)
	Mag Rail (2)	$20 \times 40 \times 1500$	Fixed (Z-dir)
Source	Source Platform	$300 \times 50 \times 300$	Fixed to Platform
	Source Tower	$100 \times 1000 \times 100$	Traversal (X)
	X-ray Source	$10 \times 10 \times 100$	Vertical (Y)
Sample	Sample Rail	$400 \times 50 \times 400$	Magnification (Z)
	Sample Base	$200 \times 40 \times 200$	Traversal (X)
	Sample Holder	$150 \times 20 \times 150$	Rotation (360°)
	Sample	$X \times Y \times Z$	Fixed to Sample Holder
Detector	Detector Platform	$400 \times 50 \times 400$	Magnification (Z)
	Detector Tower	$100 \times 1000 \times 100$	Fixed to Platform
	Detector Traversal Stage	$145 \times 114 \times 10$	Vertical (Y)
	Detector	$145 \times 114 \times 10$	Traversal (X)

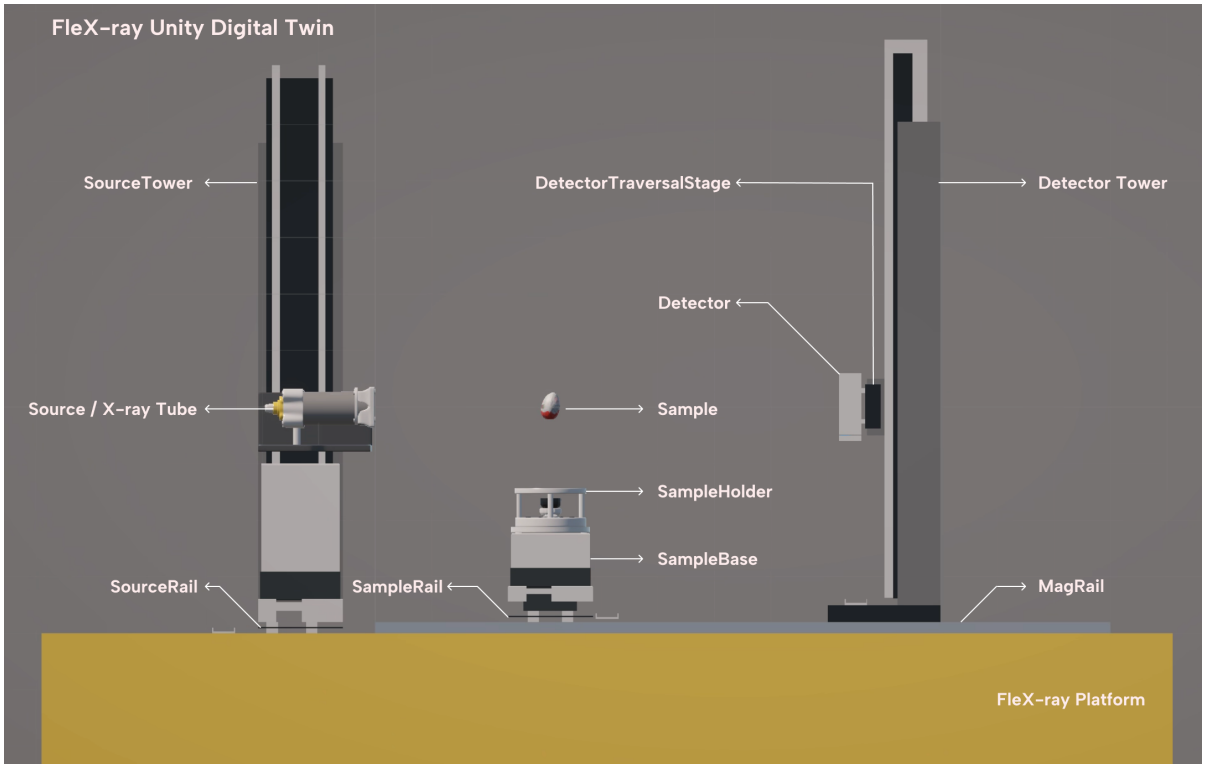


Figure 4.2: Isometric view of the Unity-based digital twin representing a cone-beam CT setup, including the X-ray source stage, rotating sample stage, and detector stage. A Kinder Joy egg mesh is used as a proxy object for testing the acquisition geometry.

4.3.1 Unit Scaling

A 1:1 scale ratio is established between the virtual world and the physical environment, where **one Unity world unit is equivalent to one millimeter (mm)** in the real world. This scale was chosen to maintain direct compatibility with the FleX-ray machine, which defines distances in millimeters. All 3D meshes and collision volumes are scaled according to their physical specifications. The primary components and their corresponding virtual dimensions are summarized in Table 4.1.

The origin of the Unity scene $(0, 0, 0)$ is set to the lowest central point of the ***Source*** or the ***X-ray tube***, keeping with the FleX-ray machine. This ensures that all x, y, z values of all the components correspond to traversal, vertical, and magnification values in FleX-ray, respectively.

4.4 Physics Engine Assumptions and Trade-offs

The digital twin utilizes Unity's integrated physics engine, **NVIDIA PhysX** [20], to manage the mechanical interactions between the FleX-ray components. While PhysX is primarily optimized for real-time gaming applications where visual plausibility is prioritized over absolute physical accuracy, it provides a robust framework for enforcing kinematic constraints and collision safety in a virtual laboratory environment. The following assumptions and trade-offs were made to balance simulation fidelity with computational performance.

4.4.1 Simplifications in Joint and Motor Modeling

The digital twin utilizes a simplified kinematic model that ignores secondary physical effects such as friction, motor torque, and thermal expansion.

- **Joint Constraints:** Rather than simulating the complex internal gears and belt-drive systems of the FleX-ray stages, we utilize **Configurable Joints** to lock all but the intended axes of motion.
- **Rationalization:** For the purpose of geometric calibration, the *absolute spatial positioning* of the focal spot and detector is the only relevant variable. Simulating the complex dynamics of motor torque or friction would add significant computational overhead without improving the accuracy of the tomographic forward model.

4.4.2 Kinematic vs. Dynamic Rigidbody Modeling

To maintain strict control over the imaging geometry, all moving components—including the Source, Sample, and Detector stages—are modeled as **non-kinematic Rigidbodies** with gravity disabled (`useGravity = false`). We assume the physical FleX-ray scanner is a perfectly rigid system where structural components do not undergo measurable elastic deformation or "sag" under their own weight. This is reliable because motors in a physical system are tightly coupled to the railing, so they exhibit accurate movement during operation.

4.4.3 Kinematic Actuation and Motor Constraints

To replicate the mechanical behavior of the FleX-ray scanner, the digital twin utilizes Unity's physics-based **Joint system** to define the spatial relationships and movement limits between

components. By using joints instead of simple transform parenting, the system can strictly enforce the physical Degrees of Freedom (DOF) of each motorized stage.

Fixed Constraints. Components that are rigidly attached to one another without an active degree of freedom—such as the floor-mounted railings attached to the main platform, or the towers attached to their respective traversal stages—are connected using **Fixed Joints**. These joints eliminate all relative translational and rotational motion between the bodies, ensuring that the structural integrity of the virtual assembly is maintained during high-speed trajectory simulations.

Motorized Motion and Configurable Joints. For components that undergo active motor movement, **Configurable Joints** are employed. These are the most flexible joint type in Unity, allowing for the explicit restriction or liberation of individual axes.

- **Linear Translation:** For the traversal (X), elevation (Y), and magnification (Z) stages, the joints are configured to be "Locked" in all rotational axes and two translational axes, while remaining "Limited" or "Free" along the axis of motor travel.
- **Rotational Motion:** The *SampleBase* utilizes a joint configuration where all translational axes are locked, and only the vertical angular DOF (Yaw) is permitted, mirroring the continuous 360° rotation of the physical sample stage.

Travel Limits and Software Clamping. A key requirement for a high-fidelity digital twin is the enforcement of physical travel ranges. Within the Configurable Joints, **Linear Limits** are defined to match the hardware specifications of the FleX-ray scanner (e.g., the limits listed in Table 2.1). This prevents the virtual components from "ghosting" or exceeding the physical length of the railings, ensuring that any trajectory designed within the simulator is physically executable in the real laboratory.

4.4.4 Collision Handling and Safety Buffers

A primary requirement for a high-fidelity digital twin is the prevention of "mesh ghosting", where virtual components pass through one another. It is also required so that the objects does not collide with each other, fulfilling the safety constraint. We use **Discrete Collision Detection** combined with **Sweep-Testing** for this.

- **Mechanism:** Before executing any linear translation, the system performs a `SweepTest` in the direction of travel. This "look-ahead" mechanism checks for potential intersections between the moving stage and other laboratory components (e.g., the source colliding with the sample holder). If a collision is imminent within the current frame's velocity vector, the motion is immediately halted, and a `blockedCounter` is incremented. If the path remains blocked after a maximum number of attempts, the motorized move is aborted to prevent virtual hardware damage and inconsistent geometric states.
- **Physical Justification:** This mirrors the physical *Emergency Stop* (E-Stop) and hardware limit switch protocols of the FleX-ray lab. By enforcing these constraints in the digital twin, researchers can safely design and test non-standard trajectories without risking hardware damage in the physical facility.

4.5 Kinematic Modeling of Mechanical Components

To achieve high-fidelity synchronization between the digital twin and the physical FleX-ray lab, the components are modeled as dynamic bodies that follow physical laws of motion rather than static geometric transforms. By implementing custom controllers (`MovableControl` and `Sample`), the digital twin replicates the acceleration, velocity, and deceleration profiles of industrial stepper motors.

4.5.1 Linear Axis Modeling and Velocity Profiles

Each translational stage (Source, Sample, and Detector) is governed by a trapezoidal velocity profile. This ensures that the digital twin mimics the ramp-up and ramp-down behavior of the physical rails. The motion logic, implemented in the `MovableControl` class, is defined by three distinct phases:

- **Acceleration Phase:** The component increases its linear velocity at a constant rate (`acceleration`) until reaching the `maxVelocity` (e.g., $50\,\mathrm{mm\,s^{-1}}$).
- **Cruise Phase:** The component maintains a steady-state velocity, ensuring a stable transition between coordinates.
- **Deceleration Phase:** When the component enters a predefined `decelerationDistance` from its target, the velocity is reduced linearly.

To prevent numerical oscillation at the target coordinate, an `arrivalThreshold` (e.g., 1.0 mm) is employed. Once within this threshold, the component is strictly clamped to the target position, ensuring the sub-millimetre geometric precision required for ASTRA projections.

4.5.2 Rotation Stage Modeling

The sample rotation is modeled as a specialized kinematic chain that extends beyond standard 0°–360° orientations. To support complex CT scanning protocols that involve multiple full rotations, the `Sample` class tracks an **accumulated rotation angle** (supporting a range of ±7200°).

- **Absolute Targeting:** Rotation commands are sent as absolute target angles. The controller calculates the shortest angular path while accounting for the current accumulated displacement.
- **Normalized Deceleration:** While the internal logic tracks the total accumulated degrees, the deceleration logic utilizes normalized angle differences (−180° to +180°) to ensure smooth arrival at the final imaging angle.

4.5.3 Discrete vs. Continuous Motion Modes

The digital twin handles two distinct operational modes to accommodate different tomographic workflows:

1. **Discrete (Step-and-Shoot):** During manual interaction or keyboard-driven movement, the `astraClient` is notified of geometric changes during every `FixedUpdate`. The respective projection is then projected on the overlay, allowing researchers to study dynamic magnification effects and FOV adjustments interactively.

2. **Continuous (Real-Time):** In this mode, the system completes a full move to a target coordinate, verifies arrival. This is the primary mode for high-fidelity scanning and geometric calibration. In this mode, the Astra projections are updated based on the motor movement and Unity framerate.

4.6 System Interactivity and External Connectivity

Interactivity is the defining characteristic that distinguishes a digital twin from a static 3D model. The FleX-ray simulator implements a dual-layer control architecture, providing both manual local interaction for researchers and a remote API layer for automated experimentation and AI-driven workflows.

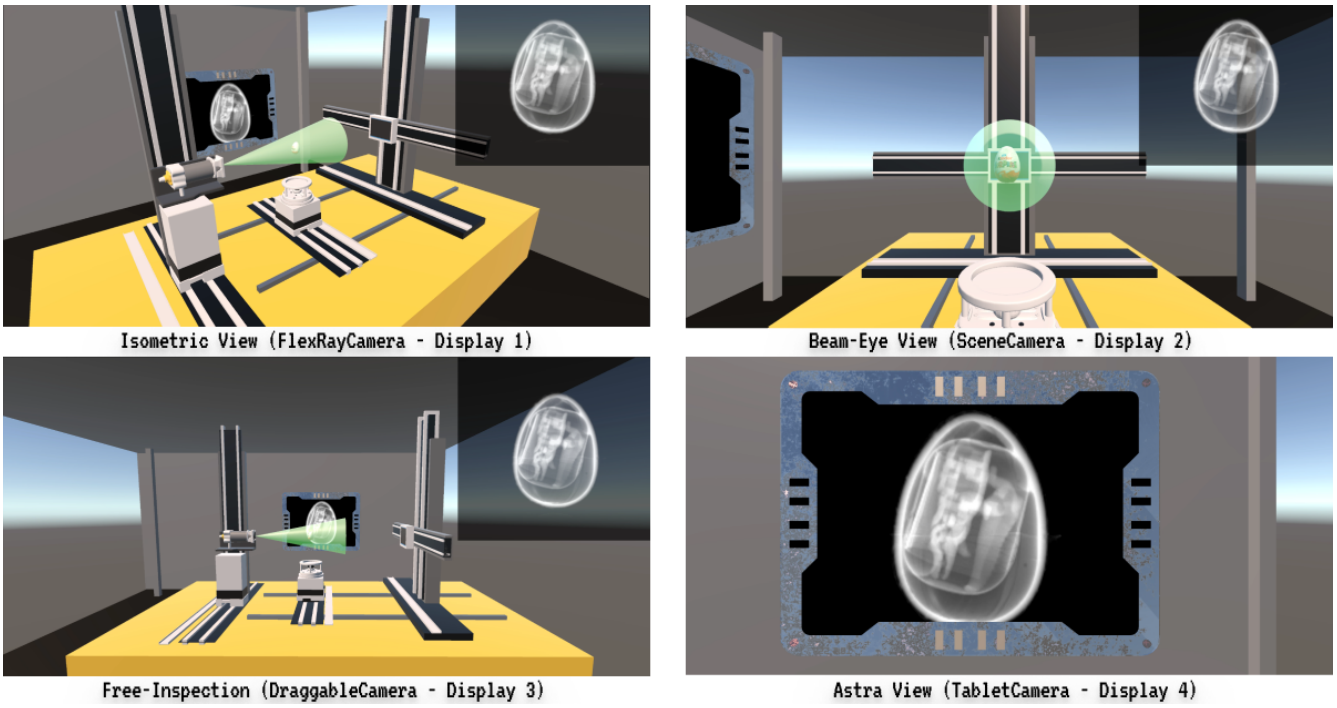


Figure 4.3: A single Unity Scene in different camera angles: Isometric View (Top Left), Beam-Eye View (Top Right), Free-Inspection View (Bottom Left), and Astra View (Bottom Right), along with the Astra projection at that position in the overlay.

4.6.1 Local Interaction and Multi-Viewport System

To provide the user with high spatial awareness during trajectory design, the digital twin utilizes a multi-camera viewport system. This allows for the simultaneous monitoring of the virtual lab from four distinct perspectives:

- **Isometric View** (`FlexRayCamera`)**:** A wide-angle view from the top-right of the source, showing the global state of the laboratory.
- **Beam-Eye View** (`SceneCamera`)**:** Placed at the exact coordinates of the X-ray focal spot, looking toward the detector to identify potential object truncation.
- **Orthogonal Profile** (`TopSideCamera`)**:** A side-view used to monitor longitudinal (Z-axis) clearances between stages.
- **Free-Inspection** (`DraggableCamera`)**:** An interactive camera allowing for orbit, pan, and zoom to inspect specific mechanical interfaces or phantom bead distributions.

Different camera angles along with the ASTRA projection is visible in figure 4.3. Manual motor control is mapped to specific **keyboard bindings**, enabling researchers to interactively explore magnification effects and mechanical limits by translating the Source, Sample, and Detector stages in real-time. Key bindings can be seen in figure 4.4.

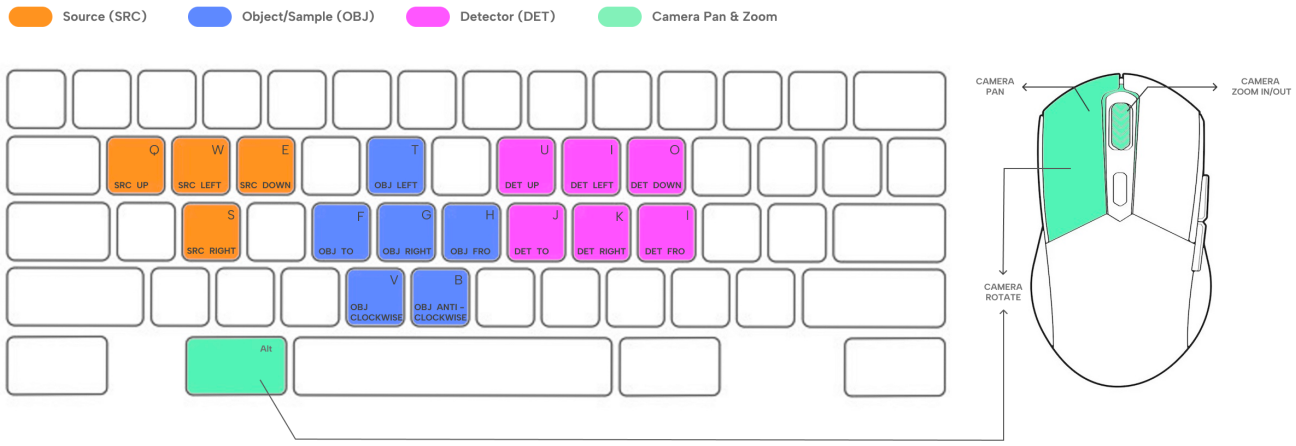


Figure 4.4: Key Bindings for Unity Motor Controls for Source, Sample and Detector Stages along with Camera keys

4.6.2 Remote API and Automation Layer

The system is designed with an API-first architecture, featuring a dedicated TCP Socket Server managed by the `Backend` GameObject. This connectivity allows the digital twin to be treated as an extensible research module, enabling other researchers to integrate the FleX-ray environment into their own external projects.

This external connectivity supports three core automation workflows:

1. **Sample Configuration:** External Python scripts can programmatically swap the `SampleObject` or adjust sample position via JSON-formatted socket messages, facilitating large-scale batch simulations.
2. **Automated Trajectory Execution:** The API allows for the remote "playback" of complex scan protocols. By streaming motor coordinates from an external project, the twin provides a visual and geometric reference for non-standard trajectories before physical deployment.

3. **Calibration Synchronization:** Upon the completion of the Gauss-Newton optimization (Chapter 5), the corrected geometric parameters are pushed back to Unity. This bidirectional data flow ensures that the digital twin remains a living, calibrated replica of the physical machine's current state, allowing users to automate the synchronization of their virtual and physical laboratories.

4.7 Script Parsing and Automated Command Execution

To achieve functional synchronization with the physical laboratory, the digital twin must interpret and execute standard *Aquila* scan scripts—the proprietary command language used by the FleX-ray scanner. This capability allows researchers to validate complex scanning trajectories in a virtual environment before deploying them to the physical hardware. The TCP socket system is used to send the scan script, and a regex-based command parser is built to translate it into motor arm movements while validating the script.

An example of a standard Acquila scan script used to acquire a full 360° cone-beam dataset is shown in Listing 1. The script initialises the tube and camera settings, positions all motor stages, acquires flat-field and dark-field images, and then executes a continuous rotational scan with synchronized image capture.

Listing 1: Example Acquila scan script for a standard circular cone-beam acquisition.

```
scanner     pause                          0.000000            ACK 0
tube        set           kV              90.000000            ACK
tube        set           power           50.000000            ACK
camera      set           mode           HW1SW1High            ACK
camera      set           exposure        99.998199            ACK
camera      set           ROI           32 8 1943 1527         ACK
tra_tube    move          absolute         0.000000            ACK
ver_tube    move          absolute        40.000000            ACK
tra_det     move          absolute        25.319824            ACK
ver_det     move          absolute        33.696938            ACK
mag_det     move          absolute      1060.000000            ACK
mag_obj     move          absolute       660.000000            ACK
tra_obj     move          absolute      -150.000000            ACK
tube        xrayoff                                            ACK
camera      take          image                                ACK
camera      save          image         ...\di000000.tif       ACK
tube        xrayon                                             ACK
camera      take          image                                ACK
camera      save          image         ...\io000000.tif       ACK
rot_obj     set           speed         0.125668970            ACK
rot_obj     move          absolute        -0.503017            ACK
rot_obj     move          absolute      360.503021  0.000000   POS
camera      take and save image sequence  ...\scan_   0        ACK
tube        xrayoff                                            ACK
```

4.7.1 The Command Abstraction Layer

Each line of an Aquila script is translated into a structured `ScriptCommand` object. This object acts as a standardized data container, isolating the operational intent from the low-level Unity implementation. The parser extracts four key attributes for every instruction:

- **Target:** The hardware component for actuation (e.g., `rot_obj`, `tra_tube`, `ver_det`).
- **CommandName:** The specific action, such as `move`, `rotate`, or `pause`.
- **Arguments:** Numeric parameters representing absolute coordinates in millimetres or rotation angles in degrees.
- **Flags:** Synchronization markers, primarily the **ACK (Acknowledge)** flag, which dictates whether the executor should wait for the current movement to reach the `arrivalThreshold` before proceeding.

4.7.2 Running Scan Scripts

The **TCP Socket Server** (operating on port 55000) accepts the request, and to maintain a fluid frame rate in the 3D environment, the server utilizes a dedicated **Background Listener Thread**. This ensures that high-frequency network traffic or large batch-script transmissions do not block Unity's main rendering thread.

The `ScriptParser` utilizes **Regular Expressions (Regex)** to sanitize incoming data, stripping comments and handling irregular whitespace. This robust parsing logic allows the digital twin to ingest raw scripts directly from external Python projects or the ASTRA Toolbox, enabling a decoupled architecture where the simulation engine is entirely driven by external control logic.

The significant challenge in script execution is the transition from near-instantaneous code execution to time-dependent physical motion. The system resolves this using the `CommandProcessor`, which utilizes **Unity Coroutines** to manage the execution flow.

Synchronization and Batching. The processor handles commands in batches. If multiple instructions are sent without an `ACK` flag, the system initiates them simultaneously—for example, moving the source and detector platforms concurrently. If an `ACK` flag is present, the coroutine yields execution, monitoring the `IsBusy()` state of the active `MovableControl` components. The script only resumes once all active stages have arrived at their targets, ensuring the digital twin perfectly mirrors the "step-and-shoot" logic of the physical FleX-ray scanner.

Temporal Scaling (`speedUpFactor`). One of the primary advantages of a digital twin is the ability to bypass the physical time constraints of a real-world scan. While a physical X-ray acquisition requires significant "idle time" for motor stabilization and detector integration, the digital twin can operate in an Accelerated Validation Mode. The `CommandProcessor` implements a `speedUpFactor` (e.g., 2× or higher) that scales the duration of `scanner pause` commands and motor travel times. This allows a researcher to simulate a full 360° scan—which might take an hour in the physical lab—in a matter of minutes. This temporal scaling is essential for rapid iteration, allowing for the quick visual and geometric validation of new trajectories while still maintaining the relative timing relationships between components.

Target Resolution and Routing. The processor features a dynamic routing layer (`ResolveTarget`) that maps abstract script identifiers to the Unity component hierarchy. For

instance, commands targeting `tra_tube` are routed to the `Source` stage, while `rot_obj` targets the *SampleHolder*. This abstraction ensures that the digital twin can be controlled using the exact same API as the physical machine, making the software interface transparent to the user.

4.8 Integration with ASTRA: The Geometry Export Pipeline

The ASTRA Toolbox serves as the mathematical core of the digital twin, responsible for executing high-performance GPU-accelerated forward projections. However, a fundamental technical barrier exists between the Unity engine—a C#-based environment optimized for real-time rendering—and the ASTRA library, which is built on a C++/CUDA core with Python and MATLAB interfaces. To bridge this gap, a dedicated **Geometry Export Pipeline** was developed to facilitate the real-time extraction, transformation, and validation of the virtual scanner's state.

4.8.1 Extraction of Global World Coordinates

To maintain geometric consistency, the pipeline must extract the *global world-space* state of the scanner, independent of the nested local transform hierarchy (Section 4.2). For every frame or motor update, the backend script pulls the following parameters from the Unity scene:

- **X-ray Focal Spot ($\boldsymbol{S}$):** The 3D origin of the cone-beam source.
- **Object Position ($\boldsymbol{O}$):** The 3D centre of the object stage.
- **Object Rotation Angle (α):** The rotation angle of the object about the vertical axis, relative to the magnification axis.
- **Detector Center ($\boldsymbol{D}$):** The 3D centroid of the detector plane.
- **Detector Basis Vectors ($\boldsymbol{u}$, $\boldsymbol{v}$):** The horizontal and vertical pixel spacing vectors of the detector plane, scaled by the detector pixel pitch.

These values are normalized using the 1:1 millimetre mapping established in Section 4.3, ensuring that virtual offsets correspond exactly to physical displacements in the FleX-ray lab.

4.8.2 Coordinate System Mapping and Axis Transformation

The most critical step in the pipeline is the resolution of the coordinate system mismatch. Unity utilizes a **Left-Handed (LHS) Y-up** Cartesian system, whereas ASTRA and standard tomographic frameworks assume a **Right-Handed (RHS) Z-up** frame. To maintain mathematical integrity, the pipeline applies a simultaneous axis-swap and sign-inversion.

The mapping from Unity world coordinates (x_u, y_u, z_u) to ASTRA coordinates (x_a, y_a, z_a) is formally defined by the following transformation matrix:

$$\begin{bmatrix} x_a \\ y_a \\ z_a \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_u \\ y_u \\ z_u \end{bmatrix} \tag{4.1}$$

The following four steps implement this transformation in the pipeline. First, all positions are expressed relative to the object centre, so that the object stage defines the reconstruction origin:

$$\boldsymbol{S}_{rel} = \boldsymbol{S}_{world} - \boldsymbol{O}_{world} \tag{4.2}$$

$$\boldsymbol{D}_{rel} = \boldsymbol{D}_{world} - \boldsymbol{O}_{world} \tag{4.3}$$

Second, all four vectors are rotated about the Unity Y-axis by the current object rotation angle α, so that the ASTRA geometry is expressed in the frame of the rotating object stage rather than the fixed world frame. Third, the Unity Y component is negated to account for the handedness difference between Unity's left-handed and ASTRA's right-handed frame. Fourth, the 12-parameter `cone_vec` row is assembled by permuting the axes from Unity order (x, y, z) to ASTRA order (x, z, y) for each vector:

$$\texttt{cone\_vec} = [S_x,\ S_z,\ S_y,\ D_x,\ D_z,\ D_y,\ u_x,\ u_z,\ u_y,\ v_x,\ v_z,\ v_y] \tag{4.4}$$

The negation and permutation together implement the transformation matrix defined above.

4.8.3 Geometric Consistency and Sanity Checks

Before transmission to the backend, a validation layer performs sanity checks to prevent mathematical irregularities:

1. **Orthogonality Verification:** The detector axes $\boldsymbol{u}$ and $\boldsymbol{v}$ are checked for perfect orthogonality ($\boldsymbol{u} \cdot \boldsymbol{v} \approx 0$) to ensure the virtual detector remains a valid Euclidean plane.

2. **Clearance and SDD Safety:** The pipeline calculates the Source-to-Detector Distance (SDD). If the geometry results in a negative distance (source behind detector), a safety flag is triggered to halt the simulation.

3. **Vector Magnitude Check:** The lengths of $\boldsymbol{u}$ and $\boldsymbol{v}$ are verified against the physical $74.8\mu m$ pixel pitch, ensuring that the simulated magnification factors correctly reflect the hardware reality.

4.9 Projection Simulation and Communication Workflow

Because ASTRA cannot be executed natively within the Unity environment, a **decoupled middleware architecture** was implemented. A separate Python-based tomographic server is instantiated on a local TCP port, acting as the high-performance computational backend.

Whenever a motor movement is detected in Unity, the `AstraClient` encapsulates the 12-parameter `cone_vec` vector and phantom identifier into a binary packet, transmits it to the Python backend via TCP, and awaits the returned projection byte stream, which is loaded directly into a Unity `Texture2D` for display.

4.9.1 Dual-Representation of Objects

A key technical detail is how the simulator handles the scanned sample. While Unity renders the object using a **triangular mesh** for visual interaction and collision detection, ASTRA requires a

voxelized volumetric grid to compute line integrals. The digital twin maintains perfect spatial synchronization by locking the origin $(0, 0, 0)$ of both representations to the center of the virtual rotation stage. This allows for the use of complex objects, such as the LEGO-based bead phantom, where the spiral bead distribution (designed to prevent 2D trajectory crossover) is mirrored in both the visual and mathematical domains.

4.9.2 Multi-Modal Visualization and Data Persistence

To provide the researcher with immediate feedback, the simulated projection is displayed in two distinct modes: it is rendered in **World Space** directly on the virtual detector panel and on a **Game Overlay (HUD)**, allowing for an intuitive 3D understanding of the X-ray alignment and providing a high-resolution 2D view for inspecting fine details, such as sub-pixel bead positions.

For full scanning protocols, the system operates in an automated acquisition mode. As the virtual motor stages iterate through a sequence of angles, the Python backend automatically saves each projection to a designated **project folder** provided as a parameter. This folder structure mirrors the data organization of the physical FleX-ray scanner, ensuring that synthetic datasets can be processed directly by the same reconstruction and calibration pipelines used for real physical data.

4.10 Validation of the Digital Twin Geometry

The geometric consistency of the pipeline was verified by comparing the ASTRA-generated projections against known geometric configurations. For a standard FleX-ray setup with nominal SOD and SDD values, the simulated projections were visually inspected and found to reproduce the expected magnification, detector coverage, and object silhouette. Quantitative validation against real FleX-ray projections is performed in Chapter 5 as part of the calibration evaluation.

5 Mathematical Formulation of the Calibration Problem

5.1 Calibration Objective

The digital twin made for the FleX-ray machine still needs to be calibrated to make sure that it replicates the FleX-ray machine projections accurately. A FleX-ray system has a source section that can translate vertically and traversally, i.e., x and y axes. Note that we are using Unity axis conventions for calibration. Similarly, the sample can move in the x- and z-axes, and the detector in all three axes. Along with this, the sample can also rotate about the y-axis. A cone-beam CT system requires the position of the source, sample, and detector.

The geometry obtained by measuring the physical machine might still be prone to minor inaccuracies. Misalignment is common in modular systems and requires calibration [17, 2]. Accurate geometry is essential for reconstruction quality [17, 30]. At the same time, each sample set for a scan uses an **unknown sample mount**. It is essential, then, to calculate the geometric offsets introduced by the mount to accurately place the virtual sample and plan the scan. For this purpose, an automated calibration technique using a LEGO-based phantom is implemented. The calibration is posed as minimizing the sum of squared reprojection errors across all N projections and K beads:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{n=1}^{N} \sum_{k=1}^{K} \left[\left(x_{n,k}(\boldsymbol{\theta}) - x_{n,k}^{real} \right)^2 + \left(y_{n,k}(\boldsymbol{\theta}) - y_{n,k}^{real} \right)^2 \right] \tag{5.1}$$

where $\boldsymbol{\theta}^*$ is the optimal geometry parameter vector and $(x_{n,k}(\boldsymbol{\theta}), y_{n,k}(\boldsymbol{\theta}))$ are the predicted detector coordinates obtained by projecting the transformed 3D bead position under geometry $\boldsymbol{\theta}$.

The parameter vector θ represents the geometry of the projection defined as

$$\boldsymbol{\theta} = [S_x, S_y,\ O_x, O_y, O_z,\ D_x, D_y, D_z,\ \alpha,\ \tau_x, \tau_z]^T \in \mathbb{R}^{11} \tag{5.2}$$

and update geometry as

$$\delta\boldsymbol{\theta} = [\delta S_x, \delta S_y,\ \delta O_x, \delta O_y, \delta O_z,\ \delta D_x, \delta D_y, \delta D_z,\ \delta\alpha,\ \delta\tau_x, \delta\tau_z]^T \in \mathbb{R}^{11} \tag{5.3}$$

The goal of the research is to design a phantom with a known 3D structure to be used as a reference to implement a calibration technique that can auto-correct the geometric inaccuracies of the digital twin [17, 30]. The reference for the notation used can be seen in the table 5.1.

5.2 Phantom Design

The calibration phantom, also known as a 3D reference structure, is chosen as the calibration mode for calibration of the digital twin. It can provide enough observable features across different projection geometries to be able to reliably do the calibration technique. A phantom created using LEGOs with metallic beads embedded is low-cost, easily reproducible, modular, and provides a flexible configuration. The metal beads provide for high contrast features in the projection, which facilitates accurate bead detection.

The discrete bead-like structure allows for controlled spatial structure to be embedded inside the phantom structure. This known spatial structure can then be reproduced into a virtual 3D, which can be used inside the digital twin. Since the materials used in the phantoms are rigid, there is negligible deformation during scanning.

Table 5.1: Notation used in this chapter.

Symbol	Description
N	Number of projections
K_n	Number of beads visible in projection n
K	Total number of beads
$\boldsymbol{\theta}$	Geometry parameter vector
$\boldsymbol{S}_n$	Source position at projection n, from scan settings $[S_x, S_y, 0]^T$
$\boldsymbol{D}_n$	Detector position at projection n, from scan settings $[D_x, D_y, D_z]^T$
$\boldsymbol{O}_n$	Object stage position at projection n, calibrating $[O_x, O_y, O_z]^T$
α	Initial rotation angle offset of the phantom
ϕ_n	Nominal rotation angle at projection n
$\boldsymbol{R}_y(\alpha + \phi_n)$	Rotation matrix about the y-axis by angle $(\alpha + \phi_n)$
$\boldsymbol{\tau}$	Rotation axis offset vector, $[\tau_x, 0, \tau_z]^T$ from the nominal object centre at $\phi = 0$
M	Total number of residuals
P	Number of active estimated parameters ($P = 6$)
$\boldsymbol{b}_k$	Known 3D position of bead k in the phantom frame
$\boldsymbol{q}_{n,k}$	3D position of bead k at projection n
$\boldsymbol{u}_{n,k}(\boldsymbol{\theta})$	Predicted 2D detector position of bead k at projection n
$\boldsymbol{u}_{n,k}^{real}$	Observed 2D detector position of bead k, $(x_{n,k}^{real}, y_{n,k}^{real})^T$
$\boldsymbol{r}(\boldsymbol{\theta})$	Full residual vector $\in \mathbb{R}^M$
$\boldsymbol{J}(\boldsymbol{\theta})$	Jacobian matrix $\in \mathbb{R}^{M \times P}$
κ	Condition number of the matrix $\boldsymbol{J}^T\boldsymbol{J}$
ϵ_j	Finite difference step size for parameter j
λ	Levenberg–Marquardt damping parameter
η_{conv}	Convergence threshold for parameter update norm
η_{stall}	Stall detection threshold
$\boldsymbol{W}(\boldsymbol{\theta})$	Cone-beam forward projection operator
$\delta\boldsymbol{\theta}$	Active parameter update step at each iteration
$\boldsymbol{\theta}^*$	Optimal estimated geometry parameter vector

During the phantom design, the beads need to be spread across the phantom across all 3 axes x, y, and z. The beads are distributed inside the phantom in a spiral pattern along the y-axis, ensuring that the beads are not aligned along a single axis and avoiding coplanar configurations. This ensures that the bead projections do not overlap with each other, and the spread allows for sufficient separation between the features.

This design also makes sure that all beads are visible across multiple projection angles and distinct trajectories. This lets the phantom act as a ground-truth reference and enables reprojection-based optimization [30].

5.3 Geometry Parameter Vector

The offset geometry is defined by the vector in the equation 5.3. The source is considered to be fixed, as its position is set from the origin (0, 0, 0) in FleX-ray. The remaining parameters in the vector are considered active and used to calibrate the object stage. The detector stage is

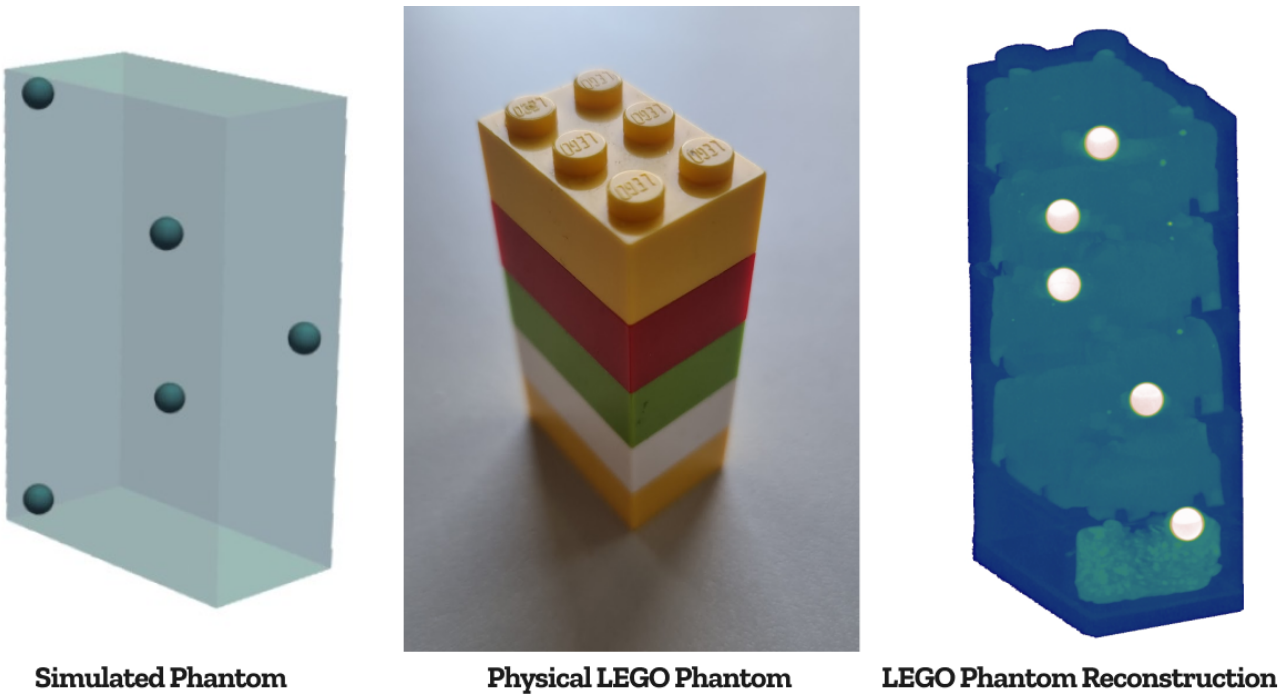


Figure 5.1: images of the simulated phantom, 3D physical phantom and Digital Reconstruction in 3D of the physical phantom

calibrated using the scan settings provided by the FleX-ray machine. The scan settings include fixed parameters such as Source-Object Distance (SOD) and Source-Detector Distance (SDD). The FleX-ray machine uses its proprietary algorithm with Acquila software to find the *piercing point* at the center of the detector.

In the current calibration setup, thus source and detector offsets are fixed ($\delta\boldsymbol{S} = 0$, $\delta\boldsymbol{D} = 0$), and only object-related parameters ($\boldsymbol{O}, \alpha, \tau$) are estimated. The parameters τ_x and τ_z represent the offset of the physical rotation axis from the nominal object center at the initial position ($\phi = 0$). At each projection angle ϕ_n, the 3D position of bead k in the world frame is computed by rotating around this shifted axis and then applying the object translation:

$$\boldsymbol{q}_{n,k}(\boldsymbol{\theta}) = \boldsymbol{\tau} + \boldsymbol{R}_y(\alpha + \phi_n)\,(\boldsymbol{b}_k - \boldsymbol{\tau}) + \boldsymbol{O}_n \tag{5.4}$$

where $\boldsymbol{b}_k \in \mathbb{R}^3$ is the known position of bead k in the phantom frame, $\boldsymbol{\tau} = [\tau_x, 0, \tau_z]^T$ is the axis offset vector, α is the initial rotation angle offset, ϕ_n is the nominal rotation angle at projection n, $\boldsymbol{O}_n = [O_x, O_y, O_z]^T$ is the object stage position at projection n, comprising the nominal position from the scan settings and the estimated correction $\delta\boldsymbol{O}$. The rotation matrix $\boldsymbol{R}_y(\alpha + \phi_n)$ denotes the standard 3×3 rotation about the y-axis by angle $(\alpha + \phi_n)$:

$$\boldsymbol{R}_y(\alpha + \phi_n) = \begin{bmatrix} \cos(\alpha + \phi_n) & 0 & \sin(\alpha + \phi_n) \\ 0 & 1 & 0 \\ -\sin(\alpha + \phi_n) & 0 & \cos(\alpha + \phi_n) \end{bmatrix} \tag{5.5}$$

The calibration vector then becomes:

$$\delta\boldsymbol{\theta}_{active} = [\delta O_x, \delta O_y, \delta O_z,\ \delta\alpha,\ \delta\tau_x, \delta\tau_z]^T \tag{5.6}$$

This can then be calibrated using the *Gauss-Newton nonlinear least-squares method*.

5.4 Forward Projection Model and Residual Formulation

The phantom is projected under different geometric configurations for both the FleX-ray physical system and the digital twin. 3D bead positions are denoted by bold symbols (e.g., $\boldsymbol{q}$), while detector coordinates are represented as 2D vectors $\boldsymbol{u}_{n,k} \in \mathbb{R}^2$. The forward projection model maps a 3D point in object space to detector coordinates:

$$(x_{n,k}(\boldsymbol{\theta}), y_{n,k}(\boldsymbol{\theta})) = \boldsymbol{W}(\boldsymbol{\theta}, \boldsymbol{q}_{n,k}) \tag{5.7}$$

where $\boldsymbol{W}(\boldsymbol{\theta})$ represents the cone-beam projection operator and $\boldsymbol{q}_{n,k}(\theta)$ denotes the transformed 3D position of bead k after applying rotation, axis offset, and translation defined by θ. The bead positions in each projection are identified using a computer vision algorithm and then compared with one another. The differences between these positions are used to compute the residuals, and the pairwise distances between beads are calculated and incorporated as additional residual constraints. The reprojection residuals are defined component-wise in the x and y directions, while inter-bead constraints are formulated as scalar Euclidean distance differences between bead pairs. The inter-bead constraints ensure the rigidity of the phantom and reduce ambiguity between the geometric parameters of the object and the detector.

The total number of residuals is therefore given by

$$M = 2\sum_{n=1}^{N} K_n + \sum_{n=1}^{N} \frac{K_n(K_n-1)}{2} = N\left(2K + \frac{K(K-1)}{2}\right) \tag{5.8}$$

The observed bead positions $\boldsymbol{u}_{n,k}^{real} = \begin{bmatrix} x_{n,k}^{real} \\ y_{n,k}^{real} \end{bmatrix}$ are compared with predicted bead positions $\boldsymbol{u}_{n,k}(\boldsymbol{\theta}) \in \mathbb{R}^2, \quad \boldsymbol{u}_{n,k}(\theta) = \begin{bmatrix} x_{n,k}(\boldsymbol{\theta}) \\ y_{n,k}(\boldsymbol{\theta}) \end{bmatrix}$, where n is the projection index and k is the bead index.

The reprojection residuals are defined component-wise as

$$r_{n,k}^{(x)}(\boldsymbol{\theta}) = x_{n,k}(\boldsymbol{\theta}) - x_{n,k}^{real} \tag{5.9}$$

$$r_{n,k}^{(y)}(\boldsymbol{\theta}) = y_{n,k}(\boldsymbol{\theta}) - y_{n,k}^{real} \tag{5.10}$$

Inter-bead constraints are defined using pairwise Euclidean distances as:

$$r_{n,ab}(\boldsymbol{\theta}) = \|u_{n,a}(\boldsymbol{\theta}) - u_{n,b}(\boldsymbol{\theta})\| - \|u_{n,a}^{real} - u_{n,b}^{real}\| \tag{5.11}$$

The residual vector is defined as

$$\boldsymbol{r}(\boldsymbol{\theta}) \in \mathbb{R}^M \tag{5.12}$$

$$\min_{\boldsymbol{\theta}} \frac{1}{2}\|\boldsymbol{r}(\boldsymbol{\theta})\|_2^2 \tag{5.13}$$

This formulation combines absolute reprojection constraints with relative geometric constraints, improving the conditioning and stability of the optimization problem.

5.5 Jacobian Construction

The Jacobian matrix $\boldsymbol{J}(\boldsymbol{\theta}) \in \mathbb{R}^{M \times P}$ encodes the sensitivity of every residual to each of the $P = 6$ active parameters in $\delta\boldsymbol{\theta}_{active}$. Each row corresponds to a single residual, either a reprojection error component or an inter-bead distance constraint, and each column corresponds to one parameter in $[\delta O_x, \delta O_y, \delta O_z, \delta\alpha, \delta\tau_x, \delta\tau_z]^T$.

Formally, the Jacobian is defined as:

$$\boldsymbol{J}(\boldsymbol{\theta}) = \frac{\partial r(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \in \mathbb{R}^{M \times P} \tag{5.14}$$

Because the cone-beam forward projection operator $\boldsymbol{W}(\boldsymbol{\theta})$ involves trigonometric functions and perspective division, an analytical derivation of $\boldsymbol{J}$ is possible but complex. Instead, the Jacobian is computed numerically using central finite differences. For each parameter j, the j-th column of $\boldsymbol{J}$ is approximated as:

$$\boldsymbol{J}_{:,j} \approx \frac{\boldsymbol{r}(\boldsymbol{\theta} + \epsilon\, e_j) - \boldsymbol{r}(\boldsymbol{\theta} - \epsilon\, e_j)}{2\epsilon} \tag{5.15}$$

where e_j is the j-th standard basis vector and ϵ is a small perturbation. Since the parameters operate on different physical scales, translations in millimeters, $\delta\alpha$ in radians, and τ_x, τ_z as positional offsets, separate step sizes are used per parameter to ensure numerical consistency of column magnitudes in $\boldsymbol{J}$.

5.6 Optimization Method

The calibration problem defined in equation 5.13 is solved iteratively using the Gauss-Newton method with Levenberg–Marquardt regularization, as described generically in Chapter 3. Here, the method is applied specifically to the active parameter vector $\delta\boldsymbol{\theta}_{active}$. The optimization is initialized with correction $\delta\boldsymbol{\theta} = \mathbf{0}$, meaning no correction is applied to the known nominal geometry $\boldsymbol{\theta}_0$ at the start. The algorithm then iteratively estimates the correction $\delta\boldsymbol{\theta}^*$ that best explains the observed bead projections.

At each iteration t, the residual $\boldsymbol{r}(\boldsymbol{\theta}_t)$ and Jacobian $\boldsymbol{J}(\boldsymbol{\theta}_t)$ are computed, and the parameter update $\delta\boldsymbol{\theta}$ is obtained by solving:

$$\left(\boldsymbol{J}_t^T \boldsymbol{J}_t + \lambda I\right) \delta\boldsymbol{\theta} = -\boldsymbol{J}_t^T \boldsymbol{r}_t \tag{5.16}$$

where $\lambda \geq 0$ is the Levenberg–Marquardt damping parameter. The geometry is then updated as:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \delta\boldsymbol{\theta} \tag{5.17}$$

When $\lambda = 0$, the equation 5.16 becomes the standard Gauss-Newton step. When λ is large, the update becomes a scaled gradient descent step, improving stability when the Jacobian is ill-conditioned. Thus, the damping parameter is adjusted adaptively: if the update reduces the cost $\frac{1}{2}\|\boldsymbol{r}\|_2^2$, the step is accepted, and λ is decreased; otherwise, the step is rejected, and λ is increased. This guarantees monotonic cost reduction across iterations [16, 18].

Iterations are run until either the norm of the update $\|\delta\boldsymbol{\theta}\|$ falls below a convergence threshold η_{conv}, or a maximum number of iterations is reached. The complete calibration procedure, including the permutation-based bead matching and stall detection, is summarized in Algorithm 2.

Algorithm 2 Gauss-Newton/LM Calibration for the Digital Twin

Require: Geometry $\boldsymbol{\theta}_0$, real bead detections $\boldsymbol{u}$, projection angles $\{\phi_n\}_{n=1}^{N}$, damping $\lambda > 0$, convergence threshold $\eta_{conv} > 0$, stall window $S_w = 5$, stall threshold $\eta_{stall} = 10^{-8}$
Ensure: Calibrated correction $\boldsymbol{\theta}^*$

$\delta\boldsymbol{\theta} \leftarrow \mathbf{0} \in \mathbb{R}^{11}$, $\lambda \leftarrow 10^{-2}$, history $\leftarrow [\,]$
repeat
  Apply $\delta\boldsymbol{\theta}$ to obtain corrected geometry $\boldsymbol{\theta}_t$
  Generate predicted projections $\{\boldsymbol{W}(\boldsymbol{\theta}_t, \phi_n)\}$ via ASTRA
  Detect bead positions in predicted projections
  Compute residual vector $\boldsymbol{r}_t$ from assigned bead pairs using equations 5.9, 5.10, 5.11, and 5.13
  Compute Jacobian $\boldsymbol{J}_t$ via central finite differences over $\delta\boldsymbol{\theta}_{active}$: Eq: 5.15
  Solve normal equations: Eq: 5.16
  Evaluate trial cost: $c' \leftarrow \frac{1}{2}\|\boldsymbol{r}(\boldsymbol{\theta}_t + \delta\boldsymbol{\theta})\|_2^2$
  if $c' < \frac{1}{2}\|\boldsymbol{r}_t\|_2^2$ **then**
    Accept: $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \delta\boldsymbol{\theta}$, $\lambda \leftarrow \max(\lambda/3,\ 10^{-6})$
  else
    Reject: $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t$, $\lambda \leftarrow \min(5\lambda,\ 10^{6})$
  end if
  Append $\|\delta\boldsymbol{\theta}\|$ to history
  if $|\text{history}| \geq S_w$ **and** $\max(\text{history}_{-S_w:}) - \min(\text{history}_{-S_w:}) < \eta_{stall}$ **then**
    break ▷ Stall detected
  end if
until $\|\delta\boldsymbol{\theta}\| < \eta_{conv}$ **or** max iterations reached
$\boldsymbol{\theta}^* \leftarrow \boldsymbol{\theta}_t$
return $\boldsymbol{\theta}^*$

5.7 Observability and Identifiability

For the calibration problem to be solvable, the system of residuals must provide sufficient constraints to uniquely determine all $P = 6$ active parameters. The necessary condition is that the total number of residuals satisfies $M \geq P$. The sufficient condition for local uniqueness is that the Jacobian has full column rank at the minimizer $\boldsymbol{\theta}^*$ [18]:

$$\text{rank}(\boldsymbol{J}(\boldsymbol{\theta}^*)) = P \tag{5.18}$$

When this condition fails, the normal equations become singular and the parameter update $\delta\boldsymbol{\theta}$ is undefined or unbounded. In practice, rank deficiency manifests as a large condition number $\kappa(\boldsymbol{J}^T\boldsymbol{J})$, which is the ratio of the largest to the smallest singular value of $\boldsymbol{J}^T\boldsymbol{J}$, causing numerical instability and poor convergence even when $M \gg P$ [18].

Several sources of rank deficiency are relevant to this calibration setup. First, **parameter coupling** arises when two parameters produce similar effects on the residuals. In particular, object translations $(\delta O_x, \delta O_z)$ and rotation axis offsets $(\delta\tau_x, \delta\tau_z)$ are partially coupled: a lateral shift of the object is geometrically similar to a shift of the rotation axis, especially with limited angular coverage. Similarly, the rotation angle offset $\delta\alpha$ interacts with the axis offsets τ_x, τ_z, as both affect the apparent trajectory of bead projections.

Second, **phantom design** critically affects observability. If all beads are coplanar, i.e., lying on a single horizontal plane, their projected trajectories under rotation are geometrically equivalent, providing no vertical separation and reducing the effective rank of $\boldsymbol{J}$. The spiral bead distribution described in Section 5.2 is designed to prevent this by ensuring that beads are distributed across all three axes, thereby providing geometrically distinct projection trajectories.

Third, **angular coverage** determines how well the rotational parameters $\delta\alpha$, τ_x, and τ_z are constrained. With few projection angles, the sinusoidal trajectories of bead projections are poorly sampled, and the Jacobian columns corresponding to rotational parameters become nearly linearly dependent. A minimum number of projections is determined in Chapter 6, which is required for the Jacobian to be well-conditioned. In limited-angle scenarios, the calibration problem becomes ill-posed and reliable parameter estimation is not possible [11].

The Levenberg–Marquardt damping term $\lambda\boldsymbol{I}$ in equation 5.16 partially mitigates ill-conditioning by regularizing the normal equations, but cannot substitute for genuinely missing observability. The practical implications of these identifiability conditions, including minimum bead count, minimum angular sampling, and spatial spread requirements, are investigated experimentally in Chapter 6.

5.8 Extension: Detector Lateral Calibration

The baseline calibration fixes the detector position entirely ($\delta\boldsymbol{D} = \mathbf{0}$), relying on the Acquila-reported SDD and piercing point. However, the lateral detector coordinates (D_x, D_y) may carry residual offsets not captured by the nominal scan settings, introducing a systematic bias in all bead projections that, if unmodelled, is absorbed into the estimated object parameters.

Why δD_z is unidentifiable. The axial offset δD_z is structurally coupled to δO_z through magnification: a shift in D_z rescales SDD and therefore changes the projected size of all beads identically to a corresponding δO_z shift. No bead configuration can break this symmetry, so δD_z is fixed at zero and taken from the scan settings.

Extended parameter vector. With δD_z fixed, the lateral offsets δD_x and δD_y are added to the active set, extending it from $P = 6$ to $P = 8$:

$$\delta\boldsymbol{\theta}^{(8)}_{active} = [\delta O_x,\ \delta O_y,\ \delta O_z,\ \delta\alpha,\ \delta\tau_x,\ \delta\tau_z,\ \delta D_x,\ \delta D_y]^T \in \mathbb{R}^8 \tag{5.19}$$

The bead transform (Equation 5.4) is unchanged; δD_x and δD_y enter only through the projection operator $\boldsymbol{W}$, adding two columns to the Jacobian computed by finite difference.

Identifiability and the need for magnification diversity. Including δD_x introduces a new coupling: at a fixed object position, a lateral detector shift and an object translation produce proportional projection shifts, making the corresponding Jacobian columns nearly linearly dependent. This coupling is broken by combining calibration data from two scans at *different magnifications* — that is, different SOD values — into a single joint residual vector. At different SODs, the projection shift due to δO_x scales with magnification while δD_x does not, providing the additional constraint needed to decouple them.

6 Experimentation

This chapter presents the experimental evaluation of the calibration framework described in Chapter 5. Four experiments are conducted to address the research questions **RQ2 - RQ5** mentioned in Chapter 1. Experiment 1 studies the effects of bead count and projection count; Experiment 2 evaluates the impact of bead spatial distribution; Experiment 3 investigates the effect of damping on convergence; and Experiment 4 validates the framework on real FleX-ray data.

6.1 Simulated Experimental Setup

6.1.1 Geometry Scenarios

Five cone-beam geometry configurations, denoted G0–G4, are used across all simulated experiments. Each scenario defines a distinct source, detector, and object position, along with a known ground-truth correction vector $\boldsymbol{\theta}^{gt}$ that represents the true geometric offset to be recovered. The calibration initializes with $\delta\boldsymbol{\theta} = \mathbf{0}$, meaning the correction starts at zero relative to the known nominal geometry $\boldsymbol{\theta}_0$ for each scenario, thereby testing sensitivity to offset magnitude. The five scenarios are summarised in Table 6.1.

Table 6.1: Simulated geometry scenarios. All positions are in Unity world coordinates (mm). δO_x, δO_y, δO_z are the ground-truth object translation offsets and $\delta\alpha$ is the ground-truth initial rotation angle offset to be recovered by calibration.

Scenario	**Source** (x, y, z)	**Detector** (x, y, z)	**Nominal Obj** (x, y, z)	**Unity Obj** (x, y, z)	δO_x	δO_y	δO_z	$\delta\alpha^\circ$
G0	$(0, 45, 0)$	$(0, 0, 674)$	$(0, 20, 570)$	$(10, 15, 589)$	-10	5	-19	10.0
G1	$(8, 50, 0)$	$(-5, 10, 720)$	$(0, 22, 565)$	$(6, 28, 577)$	-6	-6	-12	5.0
G2	$(-10, 55, 0)$	$(8, 20, 760)$	$(0, 30, 560)$	$(15, 42, 582)$	-15	-12	-22	15.0
G3	$(5, 60, 0)$	$(-8, -5, 800)$	$(0, 15, 555)$	$(18, 23, 580)$	-18	-8	-25	25.0
G4	$(-5, 38, 0)$	$(10, 3, 690)$	$(0, 18, 568)$	$(7, 20, 576)$	-7	-2	-8	3.0

6.1.2 Simulated Phantom

The calibration phantom used in the simulated experiments is a cuboid-shaped volume containing K metallic beads distributed in a spiral pattern along the vertical axis. The spiral arrangement ensures that no two beads share the same height, preventing coplanar configurations that would reduce the effective rank of the Jacobian $\boldsymbol{J}(\boldsymbol{\theta})$ as described in Section 5.

The spiral distribution was chosen specifically to maximize parameter identifiability. Beads of radius 2 mm are embedded inside the cuboid volume. The phantom dimensions and bead count K vary per experiment as detailed in Table 6.2. By distributing beads at distinct heights with lateral offsets, each bead traces a unique sinusoidal trajectory on the detector, providing independent constraints on all six active parameters in $\delta\boldsymbol{\theta}_{active}$. This property follows directly from the cone-beam projection geometry, where a point rotating about the axis at height z traces a sinusoid on the detector with amplitude proportional to its lateral offset from the rotation axis [19]. Projections are generated using the ASTRA toolbox forward projector, using the default FleX-ray detector specifications: a pixel spacing of 0.1496 mm and an image resolution of 956×760 pixels (width $\times$ height).

Table 6.2: Cuboid phantom configurations used in Experiments. All dimensions are in mm with beads of radius 2 mm in a spiral arrangement.

Name	Width	Breadth	Height	Characteristics
normal	20	40	60	Good spread
small	20	22	40	Reduced overall volume; moderate spread
square	30	30	30	Equal lateral dimensions; limited height
tall	10	15	80	High vertical spread; poor lateral coverage
wide	80	80	20	High lateral spread; poor vertical coverage
compact	10	15	20	Minimal spread in all axes
coplanar	40	60	10	Near-zero height; tests planar degeneracy

6.1.3 Evaluation Metrics

Three metrics are used to evaluate calibration quality across all simulated experiments:

Mean Absolute Error (MAE): the mean absolute difference between the ground-truth parameter vector $\boldsymbol{\theta}^{gt}$ and the estimated parameter vector $\boldsymbol{\theta}^{*}$, defined as

$$\text{MAE} = \frac{1}{P}\sum_{p=1}^{P}|\theta_p^{gt} - \theta_p^{*}| \tag{6.1}$$

where $P = 6$ is the number of active parameters. A value below 0.2 is considered a successful calibration in the simulated experiments.

Cost reduction (%): the percentage reduction in cost from initialization to convergence, defined as $100 \times (c_0 - c^*)/c_0$. A high percentage of cost change denotes good convergence and not getting stuck in local minima. This metric is particularly useful for real data where ground truth is unavailable.

Iteration count: the number of optimizer iterations until convergence, reported in parentheses alongside MAE in all heatmaps. This reflects the computational cost of each configuration and is used to compare efficiency across optimizer variants in Experiment 3.

6.2 Experiment 1: Effect of Bead Count and Projection Angles on Calibration Accuracy

The first simulated experiment evaluates how the number of beads K and the number of projection angles N jointly affect the accuracy of parameter recovery, directly addressing **RQ4** and **RQ2**. The *normal* phantom is used throughout, with LM damping parameter $\lambda = 10^{-2}$ with $K \in \{1, \ldots, 7\}$ and $N \in \{3, 5, 6, 9, 10, 12, 15, 18, 24, 36, 60, 90, 180, 360\}$. Each combination is evaluated across all five geometry scenarios, G0–G4, giving 490 calibration runs in total.

Table 6.3 summarises the mean parameter recovery error, cost reduction, and iteration count as a function of K, averaged across all N and all scenarios.

The results show a clear threshold at $K = 3$. With $K = 1$, the calibration fails in all 70 runs regardless of N. A single bead provides insufficient constraints to disentangle the six active parameters, as the residual vector has no pairwise inter-bead terms and only two components per projection. With $K = 2$, calibration converges in 70% of cases but remains unreliable at low N,

Table 6.3: Calibration performance by number of beads K, averaged over all N and geometry scenarios G0–G4. MAE is computed over all six active parameters of $\boldsymbol{\theta}$.

K	**MAE**	**Cost reduction (%)**	**Mean iterations**
1	7.89	99.98	43.6
2	0.33	99.97	69.4
3	0.01	100.00	42.8
4	0.01	100.00	47.8
5	0.05	100.00	51.6
6	0.02	100.00	56.8
7	0.02	100.00	57.8

reflected in a high mean iteration count of 69.4. From $K = 3$ onwards, MAE drops below 0.05 across all tested N, with cost reduction reaching 100% in all scenarios.

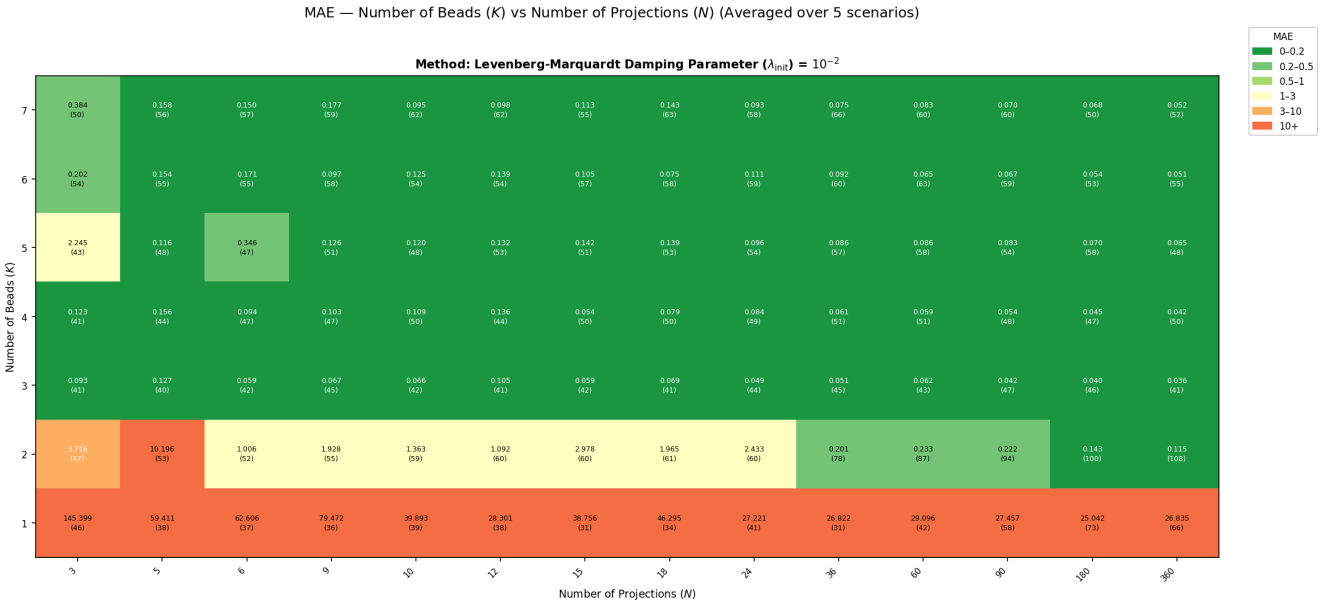


Figure 6.1: MAE across the $K \times N$ grid, averaged over geometry scenarios G0–G4. Green indicates MAE < 0.2. $K = 1$ fails throughout; $K \geq 3$ achieves reliable convergence across all tested N.

The heatmap in Figure 6.1 shows the MAE across the full $K \times N$ grid. This result is consistent with the theoretical analysis of Section 5: with $K = 3$ beads and $N = 3$ projections, the total number of residuals is $M = N\left(2K + \binom{K}{2}\right) = 3(6+3) = 27$, comprising $2K$ reprojection components and $\binom{K}{2}$ pairwise inter-bead distance constraints per projection, and the spiral bead distribution ensures the Jacobian has full column rank confirming that $K \geq 3$ is both necessary and sufficient for reliable parameter identification, directly answering **RQ4**, and demonstrating that the six active parameters are jointly identifiable from bead projections, as posed in **RQ2**.

6.3 Experiment 2: Effect of Phantom Geometry on Identifiability

The second simulated experiment investigates how the spatial distribution of beads within the phantom affects identifiability, directly addressing **RQ3**. Seven phantom configurations are evaluated with $K = 3$ beads embedded, varying the cuboid dimensions and thereby the position of the beads. The experiment uses Gauss–Newton without Levenberg–Marquardt damping ($\lambda = 0$), which is used throughout this experiment to isolate the effect of phantom geometry from optimizer behavior. Each configuration is tested across $N \in \{3, 5, 6, 9, 10, 12, 15, 18, 24, 36, 60, 90, 180, 360\}$ and all five geometry scenarios, giving 70 runs per configuration.

Table 6.4 summarises the mean absolute error, cost reduction, and iteration count for each configuration.

Table 6.4: Calibration performance by cuboid phantom configuration, $K = 3$ beads, Gauss–Newton ($\lambda = 0$), averaged over all N and geometry scenarios G0–G4. MAE computed over all six active parameters.

Configuration	Dimensions (mm)	MAE	Cost reduction (%)	Mean iterations
normal	$20 \times 40 \times 60$	0.01	100.00	23.7
small	$20 \times 22 \times 40$	0.09	100.00	32.5
square	$30 \times 30 \times 30$	0.14	100.00	33.2
compact	$10 \times 15 \times 20$	1.54	99.96	46.1
tall	$10 \times 15 \times 80$	9.31	40.70	12.4
wide	$80 \times 80 \times 20$	13.17	52.33	11.8
coplanar	$40 \times 60 \times 10$	33.27	26.92	9.9

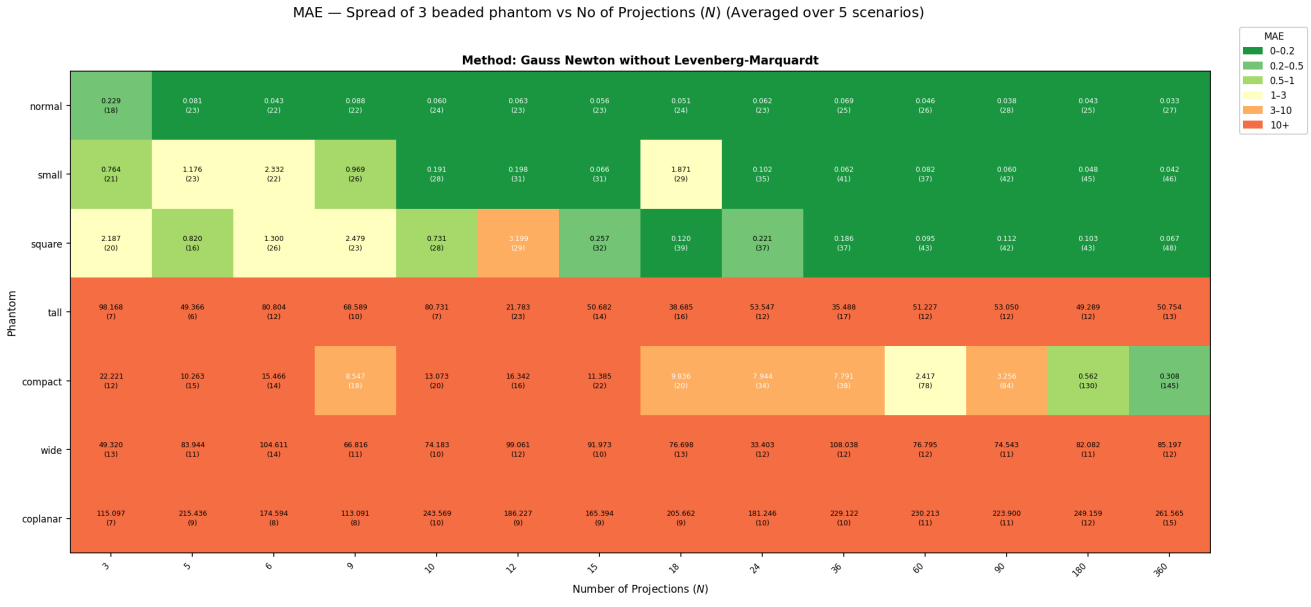


Figure 6.2: MAE by cuboid configuration and N, averaged over geometry scenarios G0–G4. Green indicates MAE < 0.2. The normal, small, and square configurations converge reliably; tall, wide, and coplanar fail due to poor or degenerate bead spread.

The results reveal that both **normal** and **small** configurations achieve reliable calibration, with MAE below 0.2 and cost reductions of up to 100%, because the spiral bead distribution provides adequate vertical separation relative to the phantom size. The **square** configuration also converges reliably despite its equal lateral dimensions, achieving an MAE of 0.14 and full cost reduction, though it requires more iterations and more projections than the normal configuration due to partially symmetric bead trajectories. The **compact** configuration is marginal: despite its spiral bead arrangement, the minimal overall volume limits bead separation, resulting in inconsistent convergence.

The **tall** and **wide** configurations both fail consistently. The tall phantom has minimal lateral spread, meaning bead projections overlap heavily at low rotation angles and the Jacobian columns corresponding to δO_x and $\delta \tau_x$ or δO_z and $\delta \tau_z$ become nearly linearly dependent. The wide phantom has minimal vertical spread, which prevents the rotational parameters from being constrained independently of the translational ones. The **coplanar** configuration represents the most severe failure case: with all beads at the same height, their projected trajectories are geometrically equivalent under rotation, providing no vertical constraint. The low cost reduction for the tall, wide, and coplanar configurations indicates that the optimizer stalls rather than converging to a false minimum, a consequence of an ill-conditioned Jacobian. The Jacobian columns corresponding to δO_y and $\delta \tau_z$ become rank-deficient regardless of N, confirming the theoretical predictions.

These results directly answer **RQ3**: the spatial distribution of beads critically affects identifiability. A phantom that provides adequate spread across all three axes, as in the normal and small configurations, produces well-conditioned Jacobians. Extreme aspect ratios in either the vertical or lateral direction degrade identifiability regardless of N. This justifies the choice of the normal cuboid phantom with spiral bead placement used throughout the remaining experiments.

6.4 Experiment 3: Effect of LM Damping on Convergence

The third simulated experiment investigates how the choice of Levenberg–Marquardt damping parameter λ affects convergence stability and accuracy, directly addressing **RQ5**. The *small* cuboid phantom is used with $K = 3$ beads, tested across the same N values as Experiments 1 and 2 and all five geometry scenarios, giving 70 runs per λ configuration. Four optimizer variants are evaluated: pure Gauss–Newton ($\lambda = 0$), and three LM variants with $\lambda \in \{10^{-4}, 10^{-2}, 1\}$, referred to as *LM_low*, *LM_normal*, and *LM_high* respectively.

Table 6.5: Calibration performance by optimizer variant, $K = 3$ beads, *small* cuboid, averaged over all N and geometry scenarios G0–G4. MAE computed over all six active parameters.

Variant	λ	MAE	Cost reduction (%)	Mean iterations
GN	0	0.09	100.00	32.5
LM_low	10^{-4}	0.04	100.00	55.4
LM_normal	10^{-2}	0.05	100.00	53.9
LM_high	1	0.04	100.00	54.3

Table 6.5 summarises the MAE, cost reduction, and iteration count for each variant, averaged over all N and geometry scenarios.

All four variants achieve full cost reduction across all runs, indicating that none stalls catastrophically. Pure Gauss–Newton converges fastest, requiring a mean of 32.5 iterations, but produces a

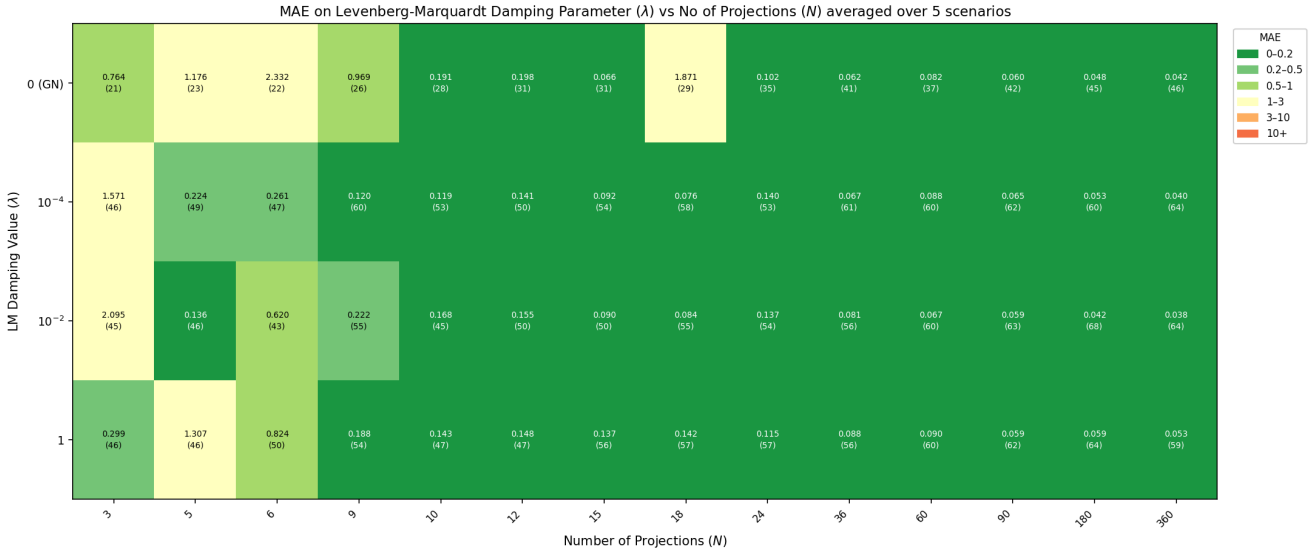


Figure 6.3: MAE by λ variant and N, averaged over geometry scenarios G0–G4. Green indicates MAE < 0.2. GN is faster but less stable at low N; all LM variants improve robustness at the cost of additional iterations.

higher MAE of 0.09 compared to 0.04–0.05 for LM variants, reflecting occasional instability at low N where the Jacobian is poorly conditioned. All three LM variants achieve comparable accuracy, with *LM_normal* ($\lambda = 10^{-2}$) offering a practical balance: slightly fewer iterations than *LM_low* and *LM_high* while maintaining equivalent MAE. These results directly answer **RQ5**: LM regularization improves robustness over Gauss–Newton when projection count is low, and $\lambda = 10^{-2}$ is selected as the recommended setting for the physical phantom experiment in Experiment 4.

6.5 Experiment 4: Calibration using Real FleX-ray Data

6.5.1 Physical Phantom

The physical calibration phantom consists of five LEGO bricks, each measuring 3×2 studs (24×16 mm footprint), stacked vertically, with a metallic bead of diameter 2 mm embedded in each brick. The beads are positioned in a spiral pattern, with each brick rotated relative to the one below. This produces a phantom of dimensions $24 \times 16 \times 49.8$ mm ($160 \times 240 \times 498$ voxels at 0.1 mm/voxel), including the height of the LEGO studs, with beads distributed across all three axes in a non-coplanar arrangement consistent with the design principles established in Section 6.3. Scan 1 used 1434 projections over a full 360° rotation to reconstruct a high-quality 3D volume of the phantom, which served as the reference ASTRA object for predicting bead projections in the calibration pipeline.

6.5.2 Scan Configurations

Eleven scans of the physical phantom were acquired on the FleX-ray system. Scan 1 was used for the phantom reconstruction, Scans 2–6 and 7–11 each used 360 projections with two distinct object placement conditions. Scan 1 and Scans 2–6 represent configurations with small object

translation and rotation offsets, while Scans 7–11 represent configurations with larger rotation offsets reflecting the variability introduced by different sample mounts in practice. Table 6.7 summarises the estimated parameters at $N = 360$ for each scan group.

Table 6.6: Real scan configurations used in Experiment 4. All positions are calibrated in Unity world coordinates (mm). SOD and SDD are derived from the object and detector z-positions, respectively.

Scan	**Source** (x, y, z)	**Detector** (x, y, z)	**Object** (x, y, z)	**SOD (mm)**	**SDD (mm)**	α_0 (°)
Scan1	(0.000, 25.0, 0)	(−25.318, 18.687, 1059)	(0.541, 20, 600)	600	1059	−1.04
Scan2	(5.000, 29.9, 0)	(−25.318, 18.677, 1059)	(0.541, 20, 600)	600	1059	−4.57
Scan3	(5.000, 29.9, 0)	(−25.318, 18.667, 1059)	(0.541, 20, 800)	800	1059	−4.57
Scan4	(−10.000, 30.0, 0)	(−20.002, 33.656, 1059)	(0.541, 20, 800)	800	1059	−4.57
Scan5	(−10.000, 30.0, 0)	(−20.002, 33.656, 959)	(0.541, 20, 600)	600	959	−4.57
Scan6	(0.000, 30.0, 0)	(−6.002, 33.656, 959)	(0.541, 20, 600)	600	959	−4.57
Scan7	(5.000, 29.9, 0)	(−25.318, 18.677, 1059)	(0.541, 20, 600)	600	1059	−4.57
Scan8	(5.000, 29.9, 0)	(−25.318, 18.667, 1059)	(0.541, 20, 800)	800	1059	−4.57
Scan9	(−10.000, 30.0, 0)	(−20.002, 33.656, 1059)	(0.541, 20, 800)	800	1059	−4.57
Scan10	(−10.000, 30.0, 0)	(−20.002, 33.656, 959)	(0.541, 20, 600)	600	959	−4.57
Scan11	(0.000, 30.0, 0)	(−6.002, 33.656, 959)	(0.541, 20, 600)	600	959	−4.57

6.5.3 Results

The calibration reduces the reprojection cost by over 98.5% for all Scans for N=360, confirming that the estimated geometry closely matches the observed bead positions. All six active parameters are recovered consistently across scans: $\delta O_x \approx -0.42\,\text{mm}$, $\delta O_y \approx 4.25\,\text{mm}$, $\delta O_z \approx 15\,\text{mm}$, a rotation angle correction $\delta\alpha$ of 0.71° for Scan 1, $4.2° \pm 0.5°$ for Scans 2–6, and $5.5° \pm 0.8°$ for Scans 7–11, reflecting the initial angular position of the phantom on the stage. The rotation axis offsets distinguish the two scan groups: Scans 2–6 recover $\delta\tau_x \approx -0.06\,\text{mm}$ and $\delta\tau_z \approx 3.66\,\text{mm}$, consistent with Scan 1, while Scans 7–11 recover $\delta\tau_x \approx 14.5\,\text{mm}$ and $\delta\tau_z \approx 13.2\,\text{mm}$, reflecting a substantially different sample mount position. Both groups are recovered with sub-millimeter standard deviation across repeated placements, confirming that the calibration reliably identifies the physical configuration of the object stage.

For Scans 7–11, the final cost after calibration remains higher than for Scans 2–6, despite a comparable percentage cost reduction. This is attributable to the larger initial offsets in Scans 7–11: the higher initial cost means that even after $> 98\%$ reduction, the absolute residual is larger. The estimated $\delta\boldsymbol{O}$ values are nonetheless consistent across all eleven scans, confirming that the calibration recovers physically meaningful geometry corrections.

Since ground-truth geometry is unknown for real scans, calibration quality is measured as the deviation of the estimated parameters from those obtained with the full $N = 360$ projection scan, treated as a reference baseline. Table 6.7 summarises the estimated parameters at $N = 360$ for each scan group.

Figure 6.4 shows the parameter deviation relative to the $N = 360$ baseline as a function of N for each scan. While reprojection cost plateaus around $N \approx 45$, this does not reflect parameter convergence. The heatmap shows that reliable parameter recovery, measured as low deviation from the $N = 360$ reference, requires $N \gtrsim 90$ projections across all scan groups. Below this threshold,

Table 6.7: Estimated calibration parameters at $N = 360$ summarised by scan group. Values are mean ± std across scans in each group. Scan 1 is a single reference scan.

Group	δO_x (mm)	δO_y (mm)	δO_z (mm)	$\delta \tau_x$ (mm)	$\delta \tau_z$ (mm)	$\delta \alpha$ (°)	Cost red. (%)
Scan 1	-0.418	4.197	15.227	0.177	3.671	0.71	99.7
Scan 2–6	-0.420 ± 0.017	4.255 ± 0.043	15.455 ± 0.572	-0.039 ± 0.086	3.657 ± 0.022	4.19 ± 0.48	99.6 ± 0.1
Scan 7–11	-0.403 ± 0.028	4.295 ± 0.049	14.769 ± 0.101	14.506 ± 0.289	13.182 ± 0.483	5.51 ± 0.85	98.9 ± 0.3

individual parameters remain poorly constrained despite high cost reduction, illustrating that cost alone is an insufficient indicator of calibration quality on real data.

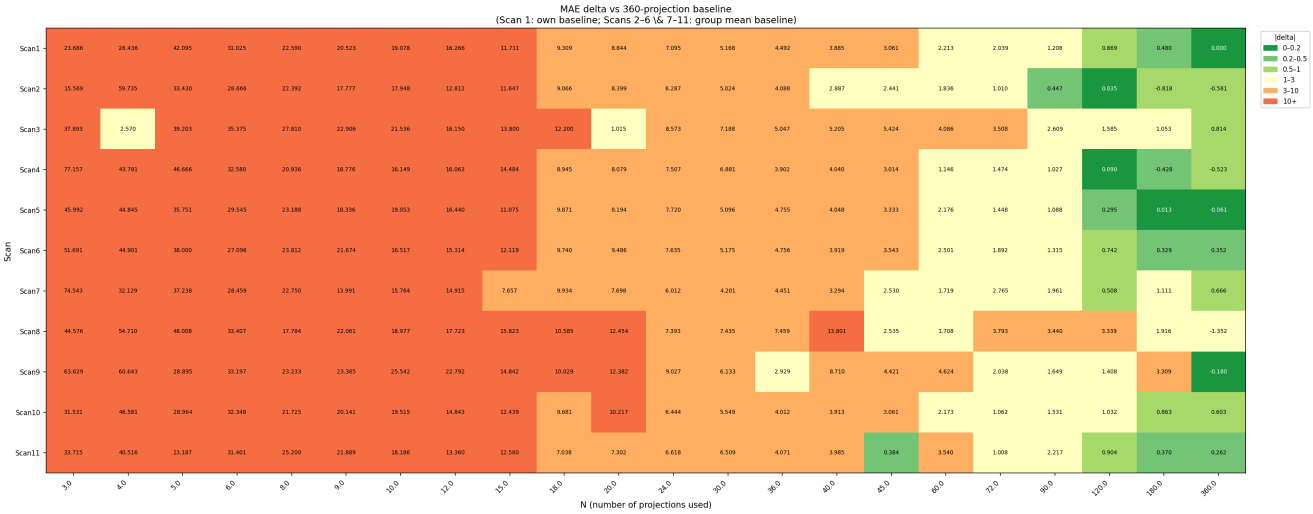


Figure 6.4: MAE delta relative to the 360-projection baseline for each scan, as a function of N. Scan 1 uses its own 360-projection baseline; Scans 2–6 and 7–11 use their respective group mean baselines.

These results collectively validate **RQ2**: the six active parameters $(\delta O_x, \delta O_y, \delta O_z, \delta \alpha, \delta \tau_x, \delta \tau_z)$ are uniquely identifiable from bead projection measurements on real FleX-ray data, with $K = 5$ beads and $N \geq 6$ projections sufficient across all eleven scan configurations tested.

6.6 Additional Experimentation: Detector Lateral Calibration using Multi-Geometry Acquisition

This experiment investigates whether the extended calibration described in Section 5.8 can recover δD_x and δD_y alongside the six object parameters, and how many geometrically distinct object placements are required for reliable recovery. The ground-truth offsets injected are $\delta O_x = -10\,\text{mm}$, $\delta O_y = 5\,\text{mm}$, $\delta O_z = -19\,\text{mm}$, $\delta D_x = 8\,\text{mm}$, $\delta D_y = -7\,\text{mm}$, and $\delta \alpha = 10°$, with $\delta \tau_x = \delta \tau_z = 0$. The *normal* cuboid phantom with $K = 3$ beads and LM damping $\lambda = 10^{-2}$ is used throughout. An MAE < 1.0 is considered a successful calibration, computed across all eight active parameters.

Three sub-experiments are run with varying numbers of object placements, G. In each case, the phantom is scanned at G distinct positions within the scanner. The residual vectors from all placements are stacked jointly into a single vector before solving the normal equations for one shared correction $\delta \boldsymbol{\theta}$. Tiled multi-position acquisition strategy is motivated by the identifiability

argument in Section 5.8: geometric diversity across placements is required to decouple δD_x from δO_x.

Single geometry ($G = 1$): residuals from a single object placement are used. MAE ranges from 24.6 to 88.7 across all tested N with no consistent trend, while the cost reduction is above 99%, indicating that the optimizer finds low-cost solutions with incorrect parameter combinations. At a single object position, the Jacobian columns for δD_x and δO_x are nearly linearly dependent, and the optimizer trades one against the other without penalty.

Two geometries ($G = 2$): the phantom is scanned at two distinct lateral and magnification positions. The residuals from both placements are stacked into a single joint residual vector. MAE drops sharply, falling below 1.0 for $N \geq 6$ in most runs. The geometric diversity between the two placements breaks the coupling between δD_x and δO_x, providing the additional constraint needed for reliable recovery.

Eight geometries ($G = 8$): residuals from eight distinct object positions are stacked jointly, comprising two tiled scans of four positions each at two different magnifications. MAE continues to decrease across all tested N, with diminishing returns beyond $N \approx 36$.

Table 6.8: Detector lateral calibration performance by number of object placements G, averaged over all N and computed across all eight active parameters.

G	Scenarios	MAE	Cost reduction (%)	Mean iterations
1	G1	48.01	99.94	48.1
2	G1, G8	3.53	99.998	116.4
8	G1–G8	0.67	100.00	136.7

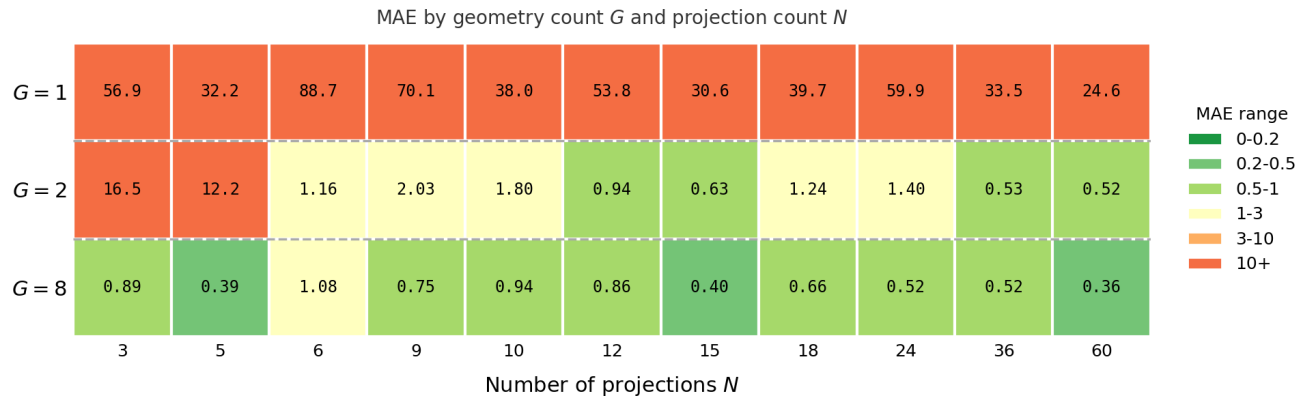


Figure 6.5: MAE as a function of N for $G = 1$, 2, and 8 object placements. A single geometry fails throughout. $G \geq 2$ recovers detector lateral parameters reliably for $N \geq 6$.

These results confirm that δD_x and δD_y are unidentifiable from a single object placement, regardless of N or K, and that stacking residuals from multiple distinct positions resolves this degeneracy.

7 Discussion

This chapter interprets the experimental results from Chapter 6 in the context of the research questions posed in Chapter 1, discusses the practical implications for the FleX-ray digital twin, and identifies the limitations and directions for future work.

7.1 Unity Digital Twin

RQ1 asks what geometric assumptions and design decisions are required to develop a consistent and interactive digital twin of the FleX-ray machine. This is addressed in Chapter 4, where the Unity-based digital twin is constructed at a 1:1 millimeter scale and includes a left-handed-to-right-handed coordinate transformation for FleX-ray/ASTRA compatibility. Joint-based objects that match physical motor limits, along with an event-driven synchronization loop that updates synthetic projections in real time, bring the Digital Twin to life. The calibration results in Chapter 6 validate the geometric consistency of the design, confirming that the nominal geometry extracted from the Unity coordinate system accurately describes the FleX-ray acquisition geometry, including the sample mount offset. The corrections $\delta\boldsymbol{\theta}$ are not a failure of the digital twin geometry but an expected and repeatable physical offset introduced by the sample mount and physical lab motors, which the calibration pipeline is specifically designed to estimate and correct. Chapter 4 together with the calibration experiments answer **RQ1**: a geometrically consistent digital twin requires an accurate coordinate mapping between the simulator and the forward projector, kinematic constraints that match the physical degrees of freedom, and an integrated calibration step to account for sample-specific geometric offsets that cannot be known a priori.

7.2 Interpretation of Calibration Performance

The four experiments collectively demonstrate that the proposed Gauss–Newton calibration framework, augmented with Levenberg–Marquardt regularization, reliably recovers the six active geometric parameters of the FleX-ray object stage from bead projection measurements. In simulated conditions with known ground-truth geometry, mean absolute parameter errors of 0.01–0.05 were achieved consistently for $K \geq 3$ beads and $N \geq 10$ projections, with cost reductions reaching 100% in all converging configurations.

The transition from simulated to real FleX-ray data introduces a measurable performance gap. In simulation, parameter errors are sub-millimeter and sub-degree across all tested geometries. In real data, all six active parameters are consistently recovered, with the rotation-axis offsets $delta\tau_x$ and $\delta\tau_z$ showing greater variance across scan groups. This gap might arise from factors not present in the simulation, such as bead-detection uncertainty in real radiographs, beam hardening effects from the metallic beads, physical tolerances in the LEGO phantom construction, residual mechanical drift between scans, and any mechanical offset already present in the FleX-ray machine relative to an ASTRA projection. Despite this gap, cost reductions of 98–100% are achieved across all eleven real scans, confirming that the optimizer successfully minimizes the reprojection residual even in the presence of real-world noise.

7.3 Observability Limits

Minimum bead count. Experiment 1 (Sec: 6.2) establishes a clear minimum of $K = 3$ beads, directly answering **RQ4**. With $K = 1$, the residual vector has only two components per projection and no inter-bead constraints, insufficient to disentangle the six active parameters. With $K = 2$, convergence reaches approximately 70% but remains unreliable at low N. From $K = 3$ onwards, the mean MAE drops below 0.05 with 100% cost reduction across all configurations. Returns diminish beyond $K = 4$, with mean iteration count rising from 42.8 to 57.8 at $K = 7$ without meaningful MAE improvement. The physical phantom uses $K = 5$ beads as a practical margin above the minimum to obtain more points for residual identification.

Minimum projection count. In simulation, $N \geq 10$ projections suffice to ensure reliable convergence for $K \geq 3$. On real FleX-ray data, calibration plateaus at $N \approx 45$ for Scans 1 and 2–6, while Scans 7–11 show increasing residuals beyond $N \approx 18$, indicating sensitivity to over-constraining when initial offsets are large. These results complete the answer to **RQ4**: both thresholds are geometry-dependent and should be treated as lower bounds rather than fixed rules.

Spatial Coverage Analysis. Among the three experimental variables: bead count K, projection count N, and phantom spatial distribution, the spatial distribution has the largest impact on calibration reliability. As shown in Experiment 2 (Sec: 6.3), configurations with degenerate bead geometry (tall, wide, coplanar) fail regardless of N, while well-distributed configurations (normal, small, square) converge reliably even at low N. This directly answers **RQ3**: spatial bead distribution is the primary determinant of parameter identifiability, more critical than either K or N alone.

Levenberg–Marquardt Damping Analysis Among the different LM Damping values, all achieved good convergence. The Gauss-Newton variant is the fastest but can still fall into a local minima and be unable to get out. The optimizer variant *LM_normal* ($\lambda = 10^{-2}$) is recommended, as it achieves the best accuracy–efficiency balance across all tested configurations in Experiment 3 (Sec: 6.4).

Parameter coupling. The six active parameters are not independent. Object translation δO_x and rotation axis offset $\delta \tau_x$ produce geometrically similar effects on bead projections at low angular coverage, as both shift bead trajectories laterally on the detector. Similarly, δO_z and $\delta \alpha$ interact because a magnification offset changes the apparent bead spacing while an angular offset changes the phase of bead sinusoids. These couplings are resolved by the pairwise inter-bead distance constraints included in the residual formulation, which enforce phantom rigidity and constrain the relative configuration of bead trajectories independently of the absolute geometric parameters. This addresses **RQ2**: the six parameters are jointly identifiable, but only when sufficient angular coverage and bead count are provided to decorrelate their effects.

Extension to detector lateral calibration. The additional experimentation in Section 6.6 demonstrates that extending the active set to eight parameters by including δD_x and δD_y introduces a new coupling: at a single object position, the Jacobian columns for δD_x and δO_x, and δD_y and δO_y are nearly linearly dependent, making the detector lateral offsets unidentifiable regardless of K or N. This manifests as a degenerate optimum: with $G = 1$, the optimizer reduces reprojection cost by over 99% while recovering parameter combinations with MAE exceeding 24, confirming that low cost is not a sufficient indicator of correct parameter recovery when the Jacobian is rank-deficient.

This degeneracy is resolved by combining residuals from geometrically distinct object placements

into a joint residual vector. With $G = 2$ placements, the coupling between δD_x and δO_x breaks because the two parameters produce different residual signatures across the two positions, and reliable recovery is achieved for $N \geq 6$. A tiled acquisition of eight positions, comprising two groups of four placements each at a different magnification, fully decouples all eight parameters and reduces MAE below 1.0 across all tested N, with diminishing returns beyond $N \approx 36$. The primary requirement is geometric diversity across placements rather than a large number of positions, as $G = 8$ diverse positions provides significant improvement over $G = 2$.

7.4 Conditioning of the Jacobian

The conditioning of the Jacobian $\boldsymbol{J}(\boldsymbol{\theta})$ is an important underlying mechanism for observed failure modes. Phantom configurations that fail, specifically the tall, wide, and coplanar configurations, are characterized not only by high MAE but also by dramatically reduced cost reduction (27–52%) and low iteration counts (10 - 12). This pattern indicates that the optimizer stalls early due to an ill-conditioned Gram matrix $\boldsymbol{J}^T\boldsymbol{J}$, rather than converging to a false minimum. In a well-conditioned system, cost reduction approaches 100% because the optimizer can compute meaningful descent directions; in an ill-conditioned system, even large parameter updates produce negligible cost reduction, and the algorithm halts at the stall detection threshold.

The Levenberg–Marquardt damping term $\lambda\boldsymbol{I}$ partially mitigates ill-conditioning by regularizing $\boldsymbol{J}^T\boldsymbol{J} + \lambda\boldsymbol{I}$, making the system invertible even when $\boldsymbol{J}$ is near rank-deficient. However, this regularization cannot substitute for genuine observability: even strong damping cannot recover parameters that are structurally unobservable, as demonstrated by the coplanar failure in Experiment 2 (Sec: 6.3) where cost reduction remains below 27% regardless of N or λ, because the relevant Jacobian columns are structurally zero rather than merely small. This distinguishes the LM role as a stabilizer of near-deficient Jacobians rather than a remedy for fully degenerate ones, directly answering **RQ5**.

7.5 Practical Implications for FleX-ray

Recommended phantom design. Based on the experimental results, the following phantom design is recommended for routine FleX-ray calibration: a cuboid of approximately $20 \times 40 \times 60$ mm (width × breadth × height), with $K = 5$ metallic beads distributed in a spiral pattern ensuring distinct heights and lateral offsets. The normal cuboid achieves the lowest mean MAE (0.01) and fastest convergence (23.7 iterations) of all tested configurations. The LEGO-based construction used in Experiment 4 (Sec: 6.5) is reproducible, low-cost, and rigid enough to maintain bead positions across multiple scanning sessions. $N = 120$–360 projections over a full 360° rotation is sufficient for reliable calibration, based on the plateau observed in Figure 6.4 with LM damping value is $\lambda = 10^{-2}$.

Integration into digital twin workflow. The estimated correction vector $\delta\boldsymbol{\theta}^*$ is pushed back to the Unity digital twin after each calibration run, updating the virtual geometry to match the current physical configuration of the FleX-ray scanner. This completes the calibration loops and generates synthetic projections that are geometrically consistent with the real machine, enabling virtual experimentation, trajectory planning, and scan simulation without requiring physical access to the scanner.

7.6 Limitations of the Proposed Approach

Fixed source and detector assumption. The calibration estimates only object-stage parameters, treating source and detector positions as known from the scan settings. This excludes detector tilt, rotation, and lateral offsets, which may contribute to residuals in extreme configurations. Extending to the full 11-parameter formulation would address this at the cost of increased parameter coupling and a larger minimum K requirement.

Noise sensitivity. Simulated experiments assume noiseless projections. While projection noise on the FleX-ray system is minimal due to the high contrast of metallic bead projections, the gap between simulated and real MAE values suggests that residual calibration errors and mechanical vibration contribute to discrepancies not captured in simulation.

Computational cost. Each iteration requires at minimum $2P+2$ ASTRA forward projection calls: $2P$ for the central finite difference Jacobian and 2 for the current residual and trial step evaluation, with rejected trial steps increasing this further. Each call projects all N angles simultaneously, so the cost per call scales with N. With $P = 6$ and mean iteration counts of 50–55, the total compute time grows with both iteration count and projection count, making high-N configurations significantly more expensive. This is acceptable for laboratory use but would be prohibitive for online calibration.

7.7 Future Work

Extension to full geometry parameter estimation. The most immediate extension is to activate the full parameter vector geometry, including source and detector tilt, rotation, and lateral offsets. This would require multiple geometric positions at different lateral positions and magnification to decorrelate the additional parameters, but would eliminate the assumption that source and detector positions are known exactly.

Adaptive projection sampling. The results show that cost plateau alone is not a sufficient criterion for terminating acquisition, as parameter estimates may continue to shift beyond the point of apparent cost convergence. An adaptive sampling strategy would need to monitor parameter stability rather than cost reduction, adding projections until the estimated parameters stabilize within a predefined tolerance. Establishing this secondary convergence criterion, and validating it across multiple FleX-ray configurations, would make the pipeline more efficient for routine laboratory use without sacrificing parameter accuracy.

Reinforcement learning environment. The calibrated digital twin provides a natural foundation for a reinforcement learning environment, serving as a simulation backend for training autonomous scan-planning agents. An agent could learn acquisition trajectories, source and detector positioning strategies, and reconstruction parameter selection, with reconstruction quality or reprojection cost serving as the reward signal. Because the digital twin replicates the FleX-ray geometry faithfully after calibration, policies learned in simulation could transfer directly to the physical scanner, reducing the need for costly trial scans on real hardware.

8 Conclusion

This thesis investigated the design, implementation, and calibration of a geometrically accurate digital twin for the FleX-ray modular cone-beam CT system at CWI. The work addressed two interconnected challenges: constructing a virtual environment that replicates the physical scanner's geometry and degrees of freedom, and developing an automated calibration method to synchronize the digital twin with the real machine in the presence of unknown sample-mount offsets.

The five research questions posed in Chapter 1 are addressed collectively by the digital twin implementation and the four calibration experiments. **RQ1** establishes that geometric consistency requires a precise coordinate mapping, kinematic constraints matching the physical motor stages, and an integrated calibration step for sample-specific offsets. **RQ2** and **RQ3** together show that parameter identifiability is determined primarily by phantom geometry: the six active parameters are jointly recoverable with $K \geq 3$ non-coplanar beads, while degenerate spatial distributions fail regardless of projection count or optimizer choice. **RQ4** establishes concrete minimum thresholds of $K = 3$ beads and $N = 10$ projections in simulation, rising to $N \geq 45$ on real data, with both values treated as geometry-dependent lower bounds. **RQ5** confirms that Levenberg–Marquardt regularization improves robustness over pure Gauss–Newton at low projection counts, but cannot recover parameters that are structurally unobservable due to phantom degeneracy or insufficient geometric diversity.

8.1 Key Findings

The central finding of this thesis is that **geometric observability**, not optimizer sophistication, is the primary determinant of calibration success. The experiments consistently show that the **geometry of the calibration object** and **angular coverage** govern whether the calibration problem is well-posed, and that optimizer choices matter only within the regime where the Jacobian is already well-conditioned. A coplanar phantom fails with any optimizer and any number of projections; a well-distributed phantom succeeds with Gauss–Newton and as few as ten projections.

A secondary finding is that **parameter decoupling** is the central challenge in extending the calibration to more parameters. Coupled parameters produce **degenerate optima** where the optimizer achieves low reprojection cost while recovering incorrect geometry. Decoupling can be achieved through multiple complementary strategies: a phantom with beads at geometrically distinct positions increases the rank of the Jacobian directly; acquiring data at **multiple object placements** introduces magnification diversity that separates otherwise collinear parameter effects; and **fixing parameters** known from external sources, such as SOD and SDD from the scan settings, reduces the active parameter set and eliminates structural ambiguities that no amount of data can resolve.

A third finding is that the **performance gap** between simulated and real data is primarily attributable to **bead-detection uncertainty** and **beam hardening**, rather than to fundamental limitations of the optimization framework. Cost reductions of 98–100% are achieved across all eleven real scans, confirming that the optimizer minimizes the reprojection residual effectively even in the presence of real-world noise. The recovered parameters are physically consistent across scan groups, demonstrating that the calibration captures **repeatable geometric offsets** rather than overfitting to noise.

A fourth finding is that the **minimum viable calibration configuration** requires only $K = 3$

beads, $N = 10$ projections, and $\lambda = 10^{-2}$, which is substantially more efficient than conventional calibration protocols that use dense angular sampling and large phantom arrays. Furthermore, extending to **eight parameters** for detector lateral calibration requires only a **two-position tiled acquisition**, adding minimal overhead to the standard protocol.

8.2 Implications for Future Digital Twin Systems

The results of this thesis suggest three implications for the future development of digital twins for modular CT systems. First, coordinate-system consistency between the simulation engine and the forward projector must be rigorously established from the outset. The axis permutation matrix derived in Chapter 4 is a necessary component of any system that bridges a game engine with a tomographic toolkit, and errors at this level propagate directly into calibration residuals.

Second, automated calibration workflows are not only feasible but also practically necessary for modular systems in which the geometry changes with every sample mount. The pipeline developed in this thesis, comprising phantom scan, bead detection, iterative optimization, and parameter push-back, executes in under twenty minutes on current GPU hardware and requires no manual intervention after the phantom is placed. This makes it suitable for integration into standard pre-scan protocols.

Third, the digital twin architecture developed here, a decoupled Unity visualization layer communicating with a Python/ASTRA computation backend via TCP, provides a reusable template for future modular CT digital twins. The separation of rendering from computations ensures that geometric accuracy is not compromised by visualization requirements, and the API-first design allows external projects to interact with the twin programmatically, enabling automated scan planning and trajectory validation.

8.3 Final Remarks

This thesis presents a fully functional, geometrically accurate modular digital twin of the FleX-ray machine, providing a foundation for future simulated experimentation and algorithmic study. This thesis also demonstrates that calibration of the digital twin of a modular cone-beam CT system is achievable with a relatively small calibration phantom, a modest number of projections, and an iterative optimization framework that runs in minutes on standard GPU hardware. The key insight is that the design of the calibration phantom and the choice of geometric positions for comparison are more consequential than the choice of optimizer or the volume of projection data.

Acknowledgements

The research for this thesis was conducted from September 2025 to May 2026 at CWI in Amsterdam. I would like to thank Prof. Dr. Joost Batenburg and Prof. Dr. Tristan van Leeuwen for supervising this project and providing valuable comments and remarks. I am also grateful to Dr. Alexander Skorikov for his input and assistance throughout this research; without him, this work would not be in its current form.

I would also like to thank my colleagues Hamid, Xin, Marcos, Nelas, Emma, and Jiayang for creating a good and enjoyable environment during my time at CWI.

Finally, I want to thank my partner Aditi for her unwavering support and motivation throughout this project. I am also deeply grateful to my parents, who have been a constant source of encouragement and whose patience during such a long time away did not go unnoticed.

I would also like to thank my friends in Leiden, Thanos, Dimitra, and Priyanjali, for the kind words, the food, and the weekends.

A special shout-out to all my friends who cheered me on from back home — your support meant more than you know.

References

[1] Sara Sadat Aghamiri, Rada Amin, Pouria Isavand, Sanaz Vahdati, Atefeh Zeinoddini, Felipe C. Kitamura, Linda Moy, and Timothy Kline. Digital twin technology in radiology. *Journal of Imaging Informatics in Medicine*, August 2025. Online ahead of print, PMID: 40760263.

[2] Francien G Bossema, Willem Jan Palenstijn, Arlen Heginbotham, Madeline Corona, Tristan van Leeuwen, Robert van Liere, Jan Dorscheid, Daniel O'Flynn, Joanne Dyer, Erma Hermens, and K Joost Batenburg. Enabling 3d ct-scanning of cultural heritage objects using only in-house 2d x-ray equipment in museums. *Heritage Science*, 9(1):1–15, 2021.

[3] Ismail Buyuksalih, S. Bayburt, Gurcan Buyuksalih, and Alias Abdul Rahman. 3d modelling and visualization based on the unity game engine – advantages and challenges. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-4/W4:161–166, November 2017. License: CC BY 4.0.

[4] Dexela Limited. X-ray detector product specifications, June 2010. Available online: http://file.yizimg.com/344621/2010061015232418.pdf (accessed on 01 April 2026).

[5] Niklas Handke, Yiqun Ma, Anton Weiss, Simon Wittl, Rebecca Wagner, and Gabriel Herl. From uncertainty to calibration: Online pose estimation of an industrial twin robotic computed tomography system with unknown spheres. *Journal of Nondestructive Evaluation*, 44(3), July 2025. Open access, CC BY 4.0.

[6] Nikolaus Hansen. The cma evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

[7] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

[8] Per Christian Hansen and Jakob Sauer Jørgensen. *Computed Tomography: Algorithms, Insight, and Just Enough Theory*. SIAM, 2021.

[9] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[10] Gabriel Herl, Simon Wittl, Alexander Jung, Niklas Handke, Anton Weiss, Markus Eberhorn, Steven Oeckl, and Simon Zabler. RoboCT: The state and current challenges of industrial twin robotic CT systems. *Applied Sciences*, 2025.

[11] Avinash C. Kak and Malcolm Slaney. *Principles of Computerized Tomographic Imaging*. SIAM, classics edition edition, 2001. Originally published by IEEE Press in 1988.

[12] Adrian Kampa. Modeling and simulation of a digital twin of a production system for industry 4.0. *Applied Sciences*, 13(22):12261, 2023.

[13] Deogratias Kibara and Guodong Shao. Building a digital twin of a CNC machine tool. In *Proceedings of the 2024 Winter Simulation Conference*, 2024.

[14] Maximilian Kiss. *Advancing Learned Algorithms for 2D X-ray Computed Tomography*. PhD thesis, Leiden University and Centrum Wiskunde & Informatica (CWI), 2026.

[15] Ali Ahmad Malik. Simulation based high fidelity digital twins of manufacturing systems: An application model and industrial use case. In *Proceedings of the 2023 Winter Simulation Conference (WSC)*, 2023.

[16] Donald W Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[17] V Nguyen, J De Beenhouwer, JG Sanctorum, S Van Wassenbergh, S Bazrafkan, JJJ Dirckx, and J Sijbers. A low-cost geometry calibration procedure for a modular cone-beam x-ray ct system. *Sensors*, 21(8):2674, 2021.

[18] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 2006.

[19] Frédéric Noo, Rolf Clackdoyle, Catherine Mennessier, Timothy A White, and Timothy J Roney. Analytic method based on identification of ellipse parameters for scanner calibration in cone-beam tomography. *Physics in Medicine & Biology*, 45(11):3489, 2000.

[20] NVIDIA Corporation. Nvidia physx physics engine, 2024.

[21] Willem Jan Palenstijn, K. Joost Batenburg, and Jan Sijbers. The ASTRA tomography toolbox. In *Proceedings of the 13th International Conference on Computational and Mathematical Methods in Science and Engineering (CMMSE 2013)*, pages 1–7, 2013.

[22] Henri Der Sarkissian, Felix Lucka, Maureen van Eijnatten, Giulia Colacicco, Sophia Coban, and Kees Joost Batenburg. A cone-beam x-ray computed tomography data collection designed for machine learning. *Scientific Data*, 6:215, October 2019.

[23] Kavitha Srinivasan, Mohammad Mohammadi, and Justin Shepherd. Applications of linac-mounted kilovoltage cone-beam computed tomography in modern radiation therapy: A review. *Journal of Medical Radiation Sciences*, 67(2):147–159, 2020.

[24] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment—a modern synthesis. In *Vision algorithms: theory and practice*, pages 298–372. Springer, 1999.

[25] Unity Technologies. Intro to the unity physics engine (2019.3).

[26] Wim van Aarle, Willem Jan Palenstijn, Jeroen Cant, Eline Janssens, Folkert Bleichrodt, Andrei Dabravolski, Jan De Beenhouwer, K Joost Batenburg, and Jan Sijbers. Fast and flexible X-ray tomography using the ASTRA toolbox. *Optics Express*, 24(22):25129–25147, 2016.

[27] Wim van Aarle, Willem Jan Palenstijn, Jan De Beenhouwer, Thomas Altantzis, Sara Bals, K. Joost Batenburg, and Jan Sijbers. The ASTRA toolbox: A platform for advanced algorithm development in electron tomography. *Ultramicroscopy*, 157:35–47, 2015.

[28] Rongxin Wang, Rui Wang, Pengcheng Fu, and Jian-Min Zhang. Portable interactive visualization of large-scale simulations in geotechnical engineering using unity3d. *Advances in Engineering Software*, 148:102838, June 2020.

[29] Muhammad Waseem and Min Hong. Moving towards large-scale particle based fluid simulation in unity 3d. *Applied Sciences*, 15(17), 2025.

[30] Kai Yang, Alexander LC Kwan, DeWitt F Miller, and John M Boone. A geometric calibration method for cone beam ct systems. *Medical Physics*, 33(10):3695–3706, 2006.

[31] Jun-Feng Yao, Yong Yang, Xue-Cheng Wang, and Xiao-Peng Zhang. Systematic review of digital twin technology and applications. *Visual Computing for Industry, Biomedicine, and Art*, 6(1):10, May 2023. Open access.

A Appendix A: Experimentation Results

This appendix reports the mean absolute parameter error for each of the six active parameters ($\delta O_x, \delta O_y, \delta O_z, \delta\alpha, \tau_x, \tau_z$) across all four experiments, broken down by the primary experimental variable in each case. All values are averaged over all other variables (scenarios, N, or K as appropriate).

A.1 Experiment 1 — Effect of Bead Count

Table A.1 reports mean parameter errors by bead count K, averaged over all N values and all five geometry scenarios G0–G4.

Table A.1: Experiment 1: mean parameter error by K, averaged over all N and scenarios. All values in mm except $\delta\alpha$ (degrees).

K	δO_x	δO_y	δO_z	$\delta\alpha$ (°)	τ_x	τ_z
1	0.085	−0.149	22.017	−10.571	−2.973	1.873
2	0.009	−0.010	0.330	−0.949	−0.283	0.097
3	−0.001	−0.002	0.048	0.005	−0.000	0.000
4	0.000	−0.004	0.067	−0.001	−0.001	−0.000
5	−0.003	−0.015	0.240	0.006	−0.001	0.003
6	−0.000	−0.005	0.090	−0.000	−0.001	0.001
7	0.000	−0.006	0.104	−0.000	0.001	0.001

A.2 Experiment 2 — Effect of Phantom Geometry

Table A.2 reports mean parameter errors by cuboid configuration, averaged over all N and all five geometry scenarios.

Table A.2: Experiment 2: mean parameter error by cuboid configuration, $K = 3$, averaged over all N and scenarios. All values in mm except $\delta\alpha$ (degrees).

Configuration	δO_x	δO_y	δO_z	$\delta\alpha$ (°)	τ_x	τ_z
normal	−0.000	−0.002	0.054	0.005	−0.001	0.001
small	−0.000	−0.027	0.455	−0.024	0.001	0.002
square	−0.006	−0.044	0.685	0.008	0.001	0.006
compact	−0.011	−0.448	7.583	−0.213	0.001	0.026
tall	4.726	8.561	16.170	−14.948	0.226	0.049
wide	9.910	−1.373	54.340	−1.650	1.841	2.091
coplanar	1.662	−8.507	164.872	−2.808	4.327	11.199

A.3 Experiment 3 — Effect of LM Damping

Table A.3 reports mean parameter errors by optimizer variant, averaged over all N and all five geometry scenarios, using the *small* cuboid with $K = 3$.

Table A.3: Experiment 3: mean parameter error by optimizer variant, $K = 3$, *small* cuboid, averaged over all N and scenarios. All values in mm except $\delta\alpha$ (degrees).

Variant	δO_x	δO_y	δO_z	$\delta\alpha$ (°)	τ_x	τ_z
GN ($\lambda = 0$)	−0.000	−0.027	0.455	−0.024	0.001	0.002
LM_low ($\lambda = 10^{-4}$)	0.000	−0.007	0.154	0.000	0.000	0.001
LM_normal ($\lambda = 10^{-2}$)	0.000	−0.010	0.196	0.008	0.000	0.001
LM_high ($\lambda = 1$)	0.000	−0.004	0.130	−0.009	0.000	0.002

A.4 Experiment 4 — Real FleX-ray Calibration

Table A.4 reports the estimated calibration parameters at $N = 360$ for each of the eleven real scans individually, including cost reduction.

Table A.4: Experiment 4: estimated calibration parameters at $N = 360$ for each scan. All positional values in mm, $\delta\alpha$ in degrees.

Scan	δO_x	δO_y	δO_z	$\delta\alpha$ (°)	τ_x	τ_z	**Cost red. (%)**
Scan1	−0.418	4.197	15.227	0.713	0.177	3.671	99.65
Scan2	−0.433	4.205	15.249	3.829	−0.075	3.677	99.66
Scan3	−0.393	4.287	16.436	3.979	−0.146	3.622	99.51
Scan4	−0.423	4.299	15.379	3.717	0.059	3.651	99.56
Scan5	−0.436	4.214	14.950	4.679	0.041	3.667	99.66
Scan6	−0.417	4.270	15.242	4.732	−0.073	3.667	99.66
Scan7	−0.420	4.241	14.920	5.906	14.218	13.625	98.66
Scan8	−0.361	4.334	14.795	4.030	14.272	13.521	99.19
Scan9	−0.395	4.345	14.778	5.749	14.734	12.485	99.16
Scan10	−0.433	4.247	14.678	6.152	14.878	12.881	98.65
Scan11	−0.408	4.309	14.673	5.715	14.427	13.396	98.65

A.5 Experiment 4 — Real FleX-ray Calibration Across All N

Tables A.5 through A.15 report the estimated calibration parameters for each of the eleven real scans as a function of the number of projections N used. All positional values are in mm, and $\delta\alpha$ is in degrees. The $N = 360$ row corresponds to the full-data estimate used as the reference baseline in Figure 6.4.

Table A.5: Experiment 4 (Scan1): estimated calibration parameters as a function of N. Positional values in mm, $\delta\alpha$ in degrees.

N	δO_x	δO_y	δO_z	$\delta\alpha$ (°)	τ_x	τ_z	**Cost red. (%)**
3	−0.228	4.185	37.272	5.085	0.683	−0.635	40.81
4	−0.390	4.218	13.606	30.142	−1.457	1.027	61.50
5	−0.356	4.203	21.547	36.784	−1.346	2.262	76.50
6	−0.376	4.197	18.183	28.847	−1.315	2.509	82.24
8	−0.356	4.180	15.975	22.662	−0.828	2.991	88.62
9	−0.396	4.197	16.113	20.204	−0.895	3.121	90.78
10	−0.390	4.201	17.310	17.548	−0.788	3.244	92.33
12	−0.401	4.201	16.856	15.188	−0.667	3.354	94.51
15	−0.395	4.195	15.633	11.890	−0.514	3.486	96.34
18	−0.408	4.197	15.755	9.441	−0.433	3.479	97.35
20	−0.405	4.198	15.832	8.984	−0.320	3.507	97.81
24	−0.411	4.199	15.561	7.535	−0.240	3.551	98.40
30	−0.411	4.198	15.391	5.853	−0.167	3.551	98.88
36	−0.412	4.197	15.593	4.978	−0.097	3.617	99.13
40	−0.412	4.198	15.320	4.680	−0.049	3.628	99.25
45	−0.414	4.199	15.220	3.970	−0.024	3.637	99.34
60	−0.416	4.197	15.268	3.074	0.037	3.623	99.49
72	−0.414	4.197	15.413	2.694	0.071	3.651	99.55
90	−0.416	4.196	15.308	1.961	0.085	3.645	99.59
120	−0.418	4.197	15.253	1.621	0.122	3.661	99.63
180	−0.417	4.197	15.200	1.238	0.158	3.672	99.65
360	−0.418	4.197	15.227	0.713	0.177	3.671	99.65

Table A.6: Experiment 4 (Scan2): estimated calibration parameters as a function of N.

N	δO_x	δO_y	δO_z	$\delta\alpha$ (°)	τ_x	τ_z	**Cost red. (%)**
3	−0.446	4.101	33.502	2.930	0.787	−1.853	39.89
4	−0.468	4.199	26.640	53.973	−0.739	1.766	66.35
5	−0.402	4.163	18.281	35.088	−1.436	2.110	77.62
6	−0.418	4.194	17.651	30.366	−1.495	2.593	83.27
8	−0.406	4.201	15.617	25.998	−1.205	3.014	89.49
9	−0.397	4.178	14.710	22.374	−1.076	3.093	91.22
10	−0.417	4.189	16.180	21.082	−0.927	3.203	92.75
12	−0.433	4.196	15.910	16.059	−0.942	3.322	94.72
15	−0.428	4.196	16.162	14.828	−0.683	3.400	96.55
18	−0.425	4.200	15.868	12.526	−0.638	3.458	97.59
20	−0.436	4.195	16.203	11.565	−0.574	3.476	97.87
24	−0.427	4.204	15.566	10.123	−0.468	3.549	98.50
30	−0.430	4.201	15.564	8.903	−0.401	3.575	98.94
36	−0.433	4.202	15.570	7.988	−0.335	3.610	99.20
40	−0.431	4.204	15.464	6.934	−0.293	3.611	99.29
45	−0.430	4.204	15.305	6.681	−0.264	3.608	99.37
60	−0.433	4.204	15.464	5.944	−0.206	3.635	99.52
72	−0.432	4.206	15.345	5.289	−0.150	3.639	99.58
90	−0.432	4.205	15.223	4.841	−0.140	3.655	99.62
120	−0.433	4.206	15.332	4.318	−0.117	3.678	99.65
180	−0.432	4.206	15.033	3.817	−0.070	3.674	99.66
360	−0.433	4.205	15.249	3.829	−0.075	3.677	99.66

Table A.7: Experiment 4 (Scan3): estimated calibration parameters as a function of N.

N	δO_x	δO_y	δO_z	$\delta\alpha$ (°)	τ_x	τ_z	**Cost red. (%)**
3	−0.418	4.257	35.906	23.420	0.845	−1.098	46.80
4	−0.359	3.958	18.902	5.103	−1.501	0.798	40.78
5	−0.331	4.256	23.181	35.991	−1.609	1.885	78.11
6	−0.409	4.235	23.740	31.116	−1.480	2.445	83.51
8	−0.364	4.285	19.757	27.286	−1.304	2.865	89.38
9	−0.377	4.279	18.231	23.840	−1.078	3.152	91.26
10	−0.383	4.282	18.624	21.979	−1.156	3.163	92.83
12	−0.366	4.302	17.158	18.062	−1.053	3.259	94.64
15	−0.394	4.273	17.742	15.294	−0.731	3.416	96.51
18	−0.389	4.290	17.685	13.741	−0.621	3.524	97.48
20	−0.315	4.267	15.534	5.174	−0.691	3.084	94.59
24	−0.395	4.281	17.618	10.281	−0.523	3.525	98.32
30	−0.380	4.291	17.049	9.452	−0.484	3.581	98.81
36	−0.396	4.281	16.706	7.706	−0.373	3.636	99.02
40	−0.395	4.282	17.294	7.375	−0.343	3.566	99.13
45	−0.393	4.288	17.199	7.690	−0.288	3.615	99.28
60	−0.396	4.289	16.782	6.760	−0.280	3.629	99.41
72	−0.398	4.272	16.977	6.021	−0.229	3.661	99.44
90	−0.390	4.300	16.380	5.762	−0.182	3.644	99.52
120	−0.395	4.297	16.499	4.676	−0.128	3.639	99.52
180	−0.392	4.293	16.317	4.266	−0.164	3.671	99.55
360	−0.393	4.287	16.436	3.979	−0.146	3.622	99.51

Table A.8: Experiment 4 (Scan4): estimated calibration parameters as a function of N.

N	δO_x	δO_y	δO_z	$\delta\alpha$ (°)	τ_x	τ_z	**Cost red. (%)**
3	0.509	4.162	73.100	26.524	0.747	−0.165	48.61
4	−0.296	4.280	19.815	44.547	−1.399	1.494	66.58
5	−0.211	4.255	25.872	40.719	−1.431	2.227	78.28
6	−0.255	4.300	24.205	28.076	−1.375	2.418	82.99
8	−0.382	4.280	16.015	24.253	−1.057	2.998	89.24
9	−0.324	4.303	19.195	18.880	−1.059	3.064	90.41
10	−0.385	4.272	16.806	18.702	−0.802	3.232	92.57
12	−0.360	4.295	17.389	18.006	−0.777	3.287	94.75
15	−0.339	4.268	19.154	14.720	−0.626	3.427	96.52
18	−0.381	4.283	16.982	11.420	−0.498	3.431	97.45
20	−0.386	4.292	16.407	11.202	−0.374	3.469	97.66
24	−0.387	4.283	17.653	9.291	−0.291	3.651	98.27
30	−0.379	4.285	17.533	8.902	−0.266	3.566	98.83
36	−0.395	4.273	16.256	7.225	−0.203	3.600	99.08
40	−0.399	4.298	16.296	7.329	−0.149	3.620	99.21
45	−0.407	4.286	16.280	6.286	−0.130	3.674	99.23
60	−0.414	4.292	15.294	5.486	−0.063	3.646	99.41
72	−0.392	4.274	17.150	4.114	−0.054	3.540	99.34
90	−0.413	4.294	15.979	4.709	−0.011	3.670	99.53
120	−0.401	4.291	16.453	3.321	0.054	3.619	99.44
180	−0.424	4.290	15.515	3.657	0.075	3.661	99.56
360	−0.423	4.299	15.379	3.717	0.059	3.651	99.56

Table A.9: Experiment 4 (Scan5): estimated calibration parameters as a function of N.

N	δO_x	δO_y	δO_z	$\delta\alpha$ (°)	τ_x	τ_z	**Cost red. (%)**
3	0.216	3.968	51.561	16.736	0.390	−1.171	46.04
4	−0.205	4.200	24.496	41.539	−1.026	1.428	66.36
5	−0.356	4.188	16.437	39.155	−1.418	2.247	78.19
6	−0.361	4.208	16.972	32.165	−1.464	2.424	83.54
8	−0.353	4.188	17.745	24.937	−1.073	2.941	89.60
9	−0.400	4.204	15.920	21.977	−0.975	2.910	91.25
10	−0.374	4.201	16.975	21.428	−0.922	3.202	92.83
12	−0.378	4.199	17.107	18.673	−0.849	3.284	94.88
15	−0.407	4.211	15.570	14.948	−0.569	3.420	96.57
18	−0.408	4.210	15.975	13.317	−0.494	3.517	97.62
20	−0.412	4.209	15.545	12.161	−0.422	3.495	97.89
24	−0.410	4.207	15.937	11.295	−0.347	3.573	98.51
30	−0.425	4.213	15.088	9.605	−0.260	3.556	98.95
36	−0.417	4.208	15.632	8.696	−0.226	3.626	99.19
40	−0.425	4.209	15.342	8.344	−0.159	3.619	99.29
45	−0.422	4.212	15.407	7.600	−0.130	3.612	99.37
60	−0.429	4.211	15.195	6.680	−0.083	3.627	99.52
72	−0.431	4.212	15.101	6.060	−0.055	3.639	99.57
90	−0.431	4.214	15.091	5.733	−0.016	3.653	99.61
120	−0.436	4.214	14.841	5.165	0.017	3.672	99.65
180	−0.435	4.214	14.967	4.728	0.045	3.673	99.66
360	−0.436	4.214	14.950	4.679	0.041	3.667	99.66

Table A.10: Experiment 4 (Scan6): estimated calibration parameters as a function of N.

N	δO_x	δO_y	δO_z	$\delta\alpha$ (°)	τ_x	τ_z	**Cost red. (%)**
3	−0.325	4.095	41.776	31.995	0.627	−0.922	50.62
4	−0.304	4.179	19.945	46.509	−0.927	1.087	67.42
5	−0.356	4.259	18.450	39.402	−1.368	2.214	78.34
6	−0.410	4.265	16.684	29.925	−1.400	2.462	83.38
8	−0.403	4.252	17.087	26.004	−1.205	2.911	89.73
9	−0.396	4.252	17.330	23.620	−1.050	3.077	91.45
10	−0.399	4.249	16.826	18.853	−1.116	3.124	92.59
12	−0.408	4.259	16.384	18.139	−0.919	3.255	94.93
15	−0.405	4.260	16.211	15.199	−0.739	3.355	96.61
18	−0.408	4.265	15.797	13.249	−0.598	3.473	97.63
20	−0.412	4.259	16.320	12.496	−0.550	3.500	97.91
24	−0.410	4.266	15.770	11.240	−0.446	3.553	98.52
30	−0.413	4.269	15.385	9.229	−0.365	3.564	98.95
36	−0.414	4.268	15.389	8.811	−0.319	3.606	99.20
40	−0.414	4.268	15.315	8.087	−0.271	3.613	99.30
45	−0.414	4.268	15.406	7.634	−0.259	3.611	99.38
60	−0.416	4.268	15.456	6.582	−0.205	3.625	99.52
72	−0.417	4.269	15.341	6.089	−0.169	3.658	99.58
90	−0.417	4.270	15.321	5.574	−0.133	3.650	99.61
120	−0.416	4.269	15.261	5.059	−0.107	3.680	99.65
180	−0.418	4.270	15.244	4.703	−0.075	3.671	99.66
360	−0.417	4.270	15.242	4.732	−0.073	3.667	99.66

Table A.11: Experiment 4 (Scan7): estimated calibration parameters as a function of N.

N	δO_x	δO_y	δO_z	$\delta\alpha$ (°)	τ_x	τ_z	**Cost red. (%)**
3	5.082	4.157	30.795	58.448	10.118	18.608	88.67
4	3.385	4.172	13.358	41.435	4.801	17.643	78.33
5	2.691	4.153	18.820	40.311	6.590	17.337	86.03
6	2.190	4.188	16.698	32.995	7.675	17.379	90.10
8	1.517	4.221	15.808	27.427	9.366	17.076	94.14
9	1.316	4.232	12.929	21.205	9.978	16.995	95.03
10	1.166	4.238	14.521	21.393	10.671	16.439	95.88
12	0.839	4.233	15.808	19.324	11.146	16.231	96.88
15	0.584	4.241	13.874	14.219	11.845	15.558	97.56
18	0.396	4.220	15.446	14.809	12.325	15.403	98.01
20	0.310	4.225	15.509	12.571	12.661	15.087	98.11
24	0.205	4.235	14.422	12.013	12.949	14.851	98.33
30	0.061	4.238	14.606	10.152	13.197	14.612	98.48
36	−0.039	4.248	16.000	8.846	13.464	14.519	98.55
40	−0.079	4.235	15.070	8.617	13.544	14.415	98.58
45	−0.116	4.243	14.474	8.544	13.605	14.213	98.61
60	−0.205	4.248	14.570	7.642	13.748	13.972	98.64
72	−0.251	4.235	15.205	7.810	14.017	13.913	98.66
90	−0.289	4.239	14.995	7.233	14.067	13.804	98.68
120	−0.337	4.240	14.874	5.912	14.091	13.720	98.67
180	−0.421	4.238	15.111	6.151	14.212	13.643	98.66
360	−0.420	4.241	14.920	5.906	14.218	13.625	98.66

Table A.12: Experiment 4 (Scan8): estimated calibration parameters as a function of N.

N	δO_x	δO_y	δO_z	$\delta\alpha$ (°)	τ_x	τ_z	**Cost red. (%)**
3	5.016	4.134	42.782	16.667	10.358	18.283	86.55
4	3.540	4.107	28.838	48.544	5.224	17.122	79.08
5	2.816	4.289	26.935	42.275	6.774	17.584	86.35
6	2.263	4.251	20.027	34.564	7.791	17.174	90.52
8	1.595	4.394	12.085	26.223	9.181	16.972	94.55
9	1.364	4.271	16.775	25.394	10.302	16.620	95.62
10	1.182	4.282	17.872	21.550	10.373	16.383	96.33
12	0.893	4.360	19.405	18.141	11.468	16.121	97.34
15	0.645	4.282	18.658	17.148	12.133	15.621	98.10
18	0.445	4.291	17.437	13.366	12.306	15.405	98.53
20	0.350	4.299	19.395	13.056	12.870	15.148	98.63
24	0.223	4.310	16.637	11.001	12.964	14.922	98.87
30	0.105	4.323	16.218	11.459	13.383	14.612	99.05
36	0.008	4.333	16.983	10.649	13.633	14.518	99.11
40	−0.071	4.284	24.602	9.101	13.825	14.582	99.08
45	−0.063	4.323	15.001	7.906	13.802	14.105	99.16
60	−0.150	4.338	14.366	7.572	13.963	13.983	99.20
72	−0.199	4.325	16.114	7.887	14.088	13.846	99.23
90	−0.240	4.328	17.071	6.497	14.155	13.813	99.24
120	−0.283	4.313	17.264	6.217	14.262	13.665	99.24
180	−0.360	4.322	15.983	6.090	14.375	13.452	99.23
360	−0.361	4.334	14.795	4.030	14.272	13.521	99.19

Table A.13: Experiment 4 (Scan9): estimated calibration parameters as a function of N.

N	δO_x	δO_y	δO_z	$\delta\alpha$ (°)	τ_x	τ_z	**Cost red. (%)**
3	5.444	3.903	47.004	29.959	11.350	18.633	87.51
4	3.687	4.160	31.900	49.936	5.666	17.960	79.44
5	2.735	4.295	10.059	40.515	7.177	16.780	86.67
6	2.317	4.260	20.707	32.965	8.311	17.301	90.69
8	1.603	4.278	17.622	25.704	10.179	16.512	94.66
9	1.360	4.254	16.864	26.103	10.889	16.559	95.69
10	1.320	4.289	21.725	23.284	11.331	16.256	96.41
12	1.027	4.305	22.672	19.563	11.944	15.946	97.38
15	0.672	4.299	16.625	17.932	12.753	15.225	98.12
18	0.436	4.310	16.251	13.629	13.072	14.994	98.55
20	0.383	4.314	18.056	14.157	13.474	14.663	98.66
24	0.305	4.277	22.180	6.919	13.507	14.505	98.63
30	0.100	4.317	16.277	10.092	13.857	14.154	99.02
36	−0.001	4.326	15.948	7.451	14.314	13.553	99.01
40	0.120	4.270	28.345	0.951	14.082	13.607	98.48
45	−0.106	4.318	16.061	8.550	14.294	13.757	99.17
60	−0.156	4.308	17.417	7.483	14.656	13.269	99.19
72	−0.228	4.319	15.340	7.024	14.559	13.233	99.22
90	−0.249	4.335	15.519	6.399	14.673	13.140	99.24
120	−0.324	4.325	14.527	7.067	14.682	13.147	99.25
180	−0.371	4.317	17.830	5.535	14.869	13.052	99.23
360	−0.395	4.345	14.778	5.749	14.734	12.485	99.16

Table A.14: Experiment 4 (Scan10): estimated calibration parameters as a function of N.

N	δO_x	δO_y	δO_z	$\delta\alpha$ (°)	τ_x	τ_z	**Cost red. (%)**
3	5.435	4.088	27.437	18.867	10.517	17.851	86.59
4	3.698	4.188	21.484	46.762	6.018	17.096	79.23
5	2.731	4.272	10.324	39.594	7.190	17.518	86.24
6	2.360	4.196	20.988	31.746	8.612	17.111	90.25
8	1.567	4.232	17.264	24.877	10.010	16.440	94.18
9	1.367	4.239	16.360	23.683	10.919	16.238	95.21
10	1.223	4.227	17.385	21.957	11.406	15.981	95.94
12	0.913	4.221	15.759	19.102	11.944	15.569	96.90
15	0.633	4.213	15.963	16.464	12.775	15.055	97.62
18	0.434	4.235	15.515	14.225	13.088	14.849	98.00
20	0.348	4.245	15.893	14.190	13.494	14.712	98.11
24	0.181	4.250	14.168	12.525	13.672	14.314	98.32
30	0.067	4.247	15.516	10.419	14.036	13.928	98.46
36	−0.027	4.248	14.920	9.487	14.195	13.800	98.54
40	−0.069	4.245	15.370	8.956	14.295	13.642	98.57
45	−0.110	4.237	15.435	8.032	14.306	13.604	98.59
60	−0.214	4.248	14.741	7.770	14.486	13.379	98.63
72	−0.254	4.250	14.729	6.659	14.565	13.270	98.64
90	−0.305	4.249	14.509	7.373	14.674	13.085	98.66
120	−0.342	4.252	14.702	6.581	14.767	13.052	98.65
180	−0.424	4.246	15.142	5.930	14.901	12.884	98.65
360	−0.433	4.247	14.678	6.152	14.878	12.881	98.65

Table A.15: Experiment 4 (Scan11): estimated calibration parameters as a function of N.

N	δO_x	δO_y	δO_z	$\delta\alpha$ (°)	τ_x	τ_z	**Cost red. (%)**
3	5.322	4.137	38.519	9.132	10.583	18.687	85.49
4	3.582	4.332	21.985	39.722	6.305	17.255	78.41
5	2.746	4.319	15.404	20.175	5.998	17.209	84.23
6	2.257	4.240	18.328	33.837	8.057	17.348	90.17
8	1.528	4.277	17.320	28.041	9.786	16.912	94.16
9	1.355	4.270	18.361	23.467	10.380	16.722	95.14
10	1.167	4.276	17.122	21.080	10.749	16.458	95.89
12	0.832	4.311	15.786	17.737	11.315	16.044	96.85
15	0.635	4.294	15.571	17.118	12.231	15.396	97.60
18	0.433	4.290	14.405	12.842	12.640	15.093	97.97
20	0.337	4.300	14.730	12.827	12.906	14.866	98.11
24	0.189	4.308	14.972	11.913	13.121	14.780	98.32
30	0.073	4.296	16.572	10.275	13.464	14.493	98.46
36	−0.032	4.310	14.733	9.723	13.723	14.215	98.55
40	−0.072	4.297	15.152	9.201	13.871	14.058	98.57
45	−0.104	4.311	13.608	7.217	13.882	13.927	98.57
60	−0.194	4.306	15.510	8.249	14.120	13.826	98.64
72	−0.242	4.306	13.689	7.658	14.173	13.605	98.65
90	−0.276	4.305	15.256	7.208	14.259	13.577	98.67
120	−0.326	4.308	14.590	6.516	14.302	13.526	98.66
180	−0.413	4.310	14.555	5.918	14.442	13.398	98.65
360	−0.408	4.309	14.673	5.715	14.427	13.396	98.65