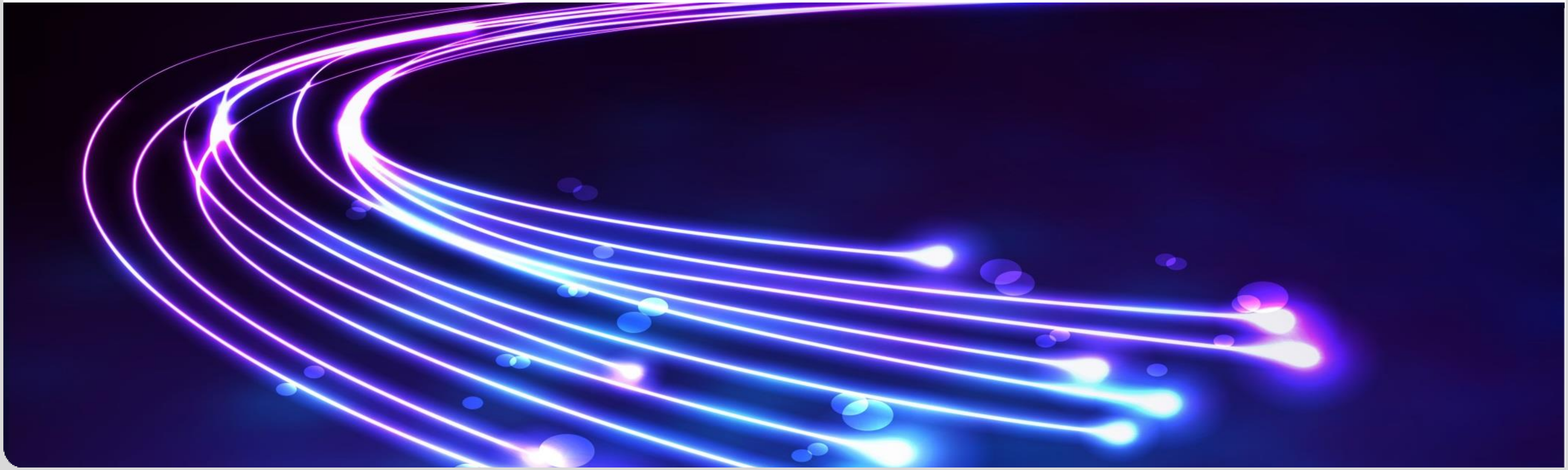
Karlsruhe Institute of Technology

Syris: A Flexible and Efficient Framework for X-ray Imaging Experiments Simulation

Tomáš Faragó

Institute for Photon Science and Synchrotron Radiation (IPS)



Outlook

- Motivation
- Image formation in *syris* **(sy**nchrotron **r**adiation **i**maging **s**imulation**)**
- Implementation
- Code snippets
- Example experiments
  - Algorithm selection for motion estimation
  - Speedup possibilities for CT data acquisition
- Conclusion and Outlook

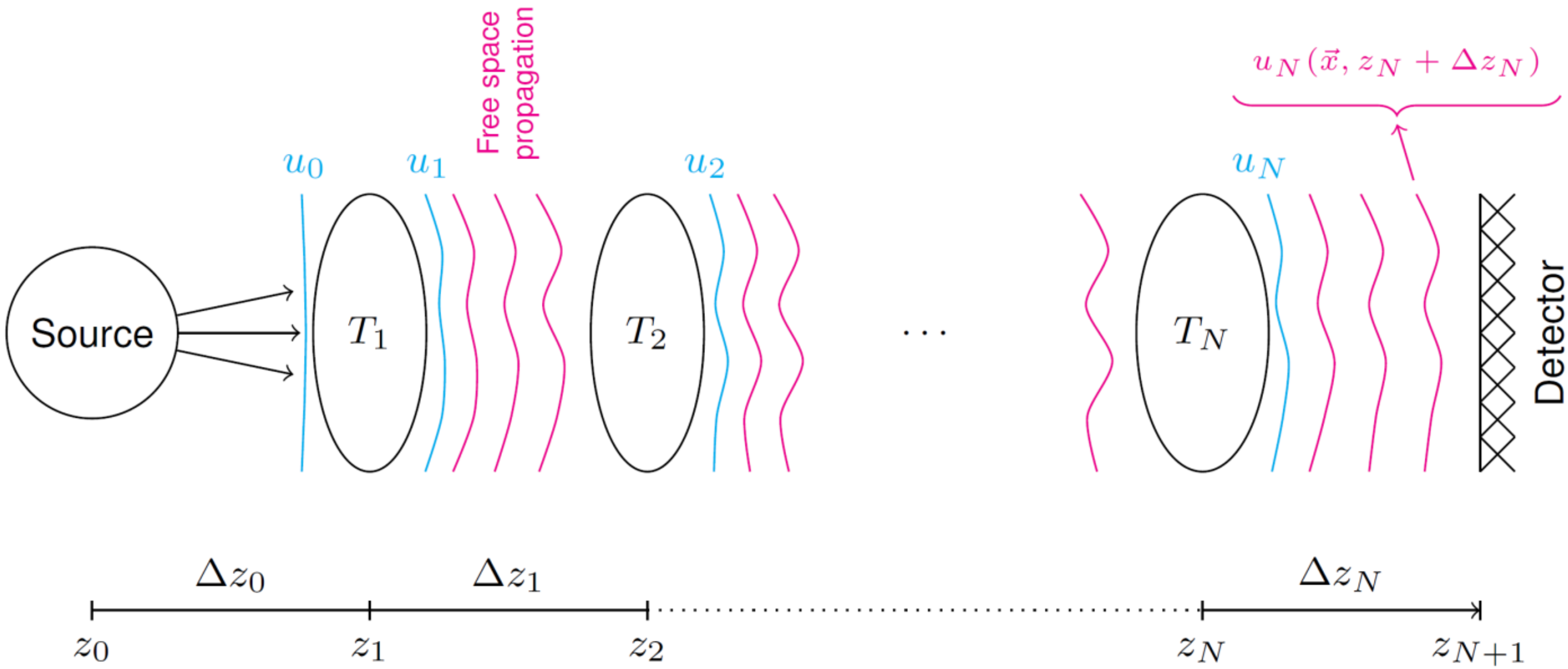
Motivation

- Provide a general X-ray imaging simulation framework
  - 2D up to 4D *high-speed* experiments
  - High physical accuracy
  - High flexibility
  - Fast implementation
- Applications
  - Investigate novel imaging methods
  - Optimize measurement parameters
  - Benchmark data processing pipelines
  - Reveal imaging and data processing parameter dependencies
- Challenge: high computational complexity
  - Suitable physical approximations
  - Most of image formation is multiplication in real or Fourier space
  - GPU implementation

Key Features

- **Shape creation**
  - Metaballs
  - Triangular meshes
- **Motion**: spline-based trajectories with velocity profiles
- **X-ray sources**
  - Bending magnets
  - Wigglers
- **X-ray and matter interaction**: transmission function
- **Free-space propagation**
  - Angular spectrum method
  - Fresnel approximation
- **Spatial coherence**: van Cittert-Zernike theorem
- **Polychromaticity**: superposition of monochromatic intensities
- **Detection**: indirect detectors

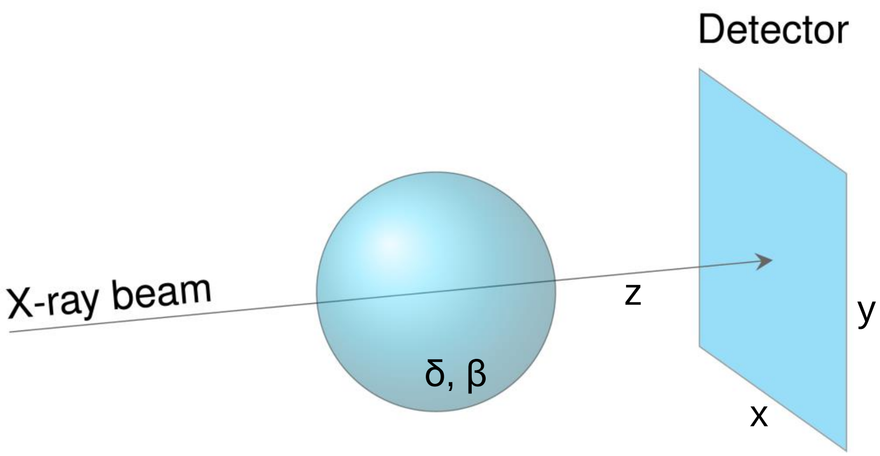
Light Path in *syris*

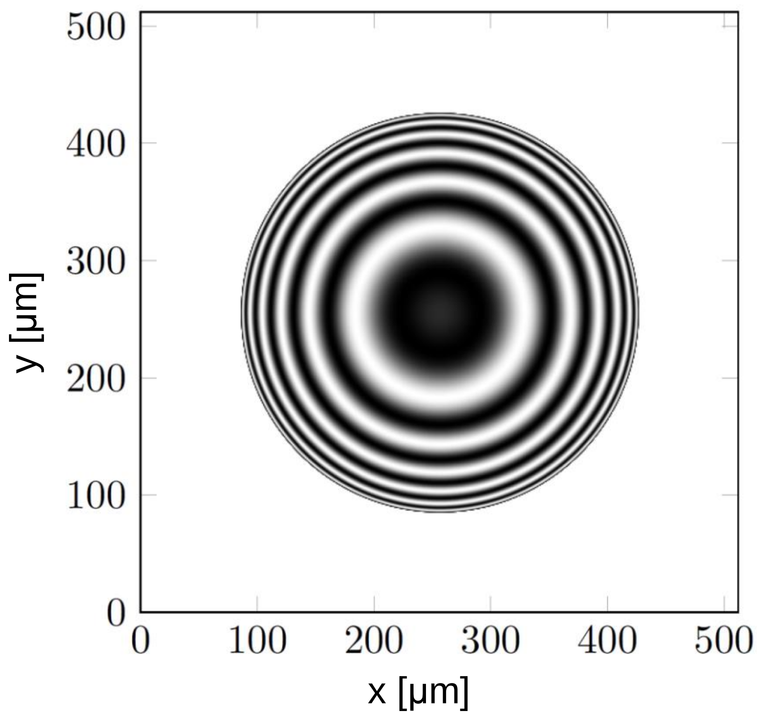


Tomáš Faragó, tomas.farago@kit.edu

X-ray and Matter Interaction

- Transmission function: $T(\vec{x}) = e^{-B(\vec{x})}\left[\mathbf{cos}\left(-\varphi(\vec{x})\right) + j\,\mathbf{sin}\left(-\varphi(\vec{x})\right)\right]$
- Complex refractive index: $n(\vec{x}, z) = 1 - \delta(\vec{x}, z) + j\beta(\vec{x}, z)$
- $B(\vec{x}) \propto \int \beta(\vec{x}, z)\, dz$
- $\varphi(\vec{x}) \propto \int (1 - \delta(\vec{x}, z))\, dz$



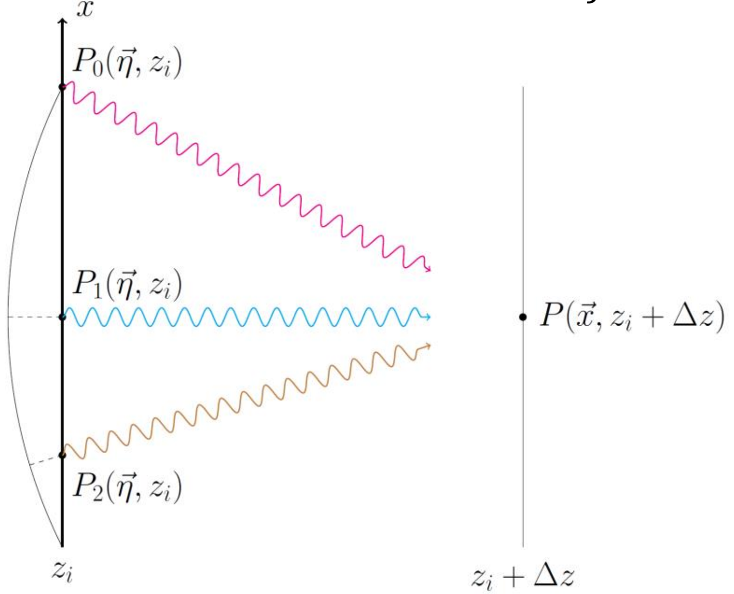


$Re[T(\vec{x})]$ of a sphere

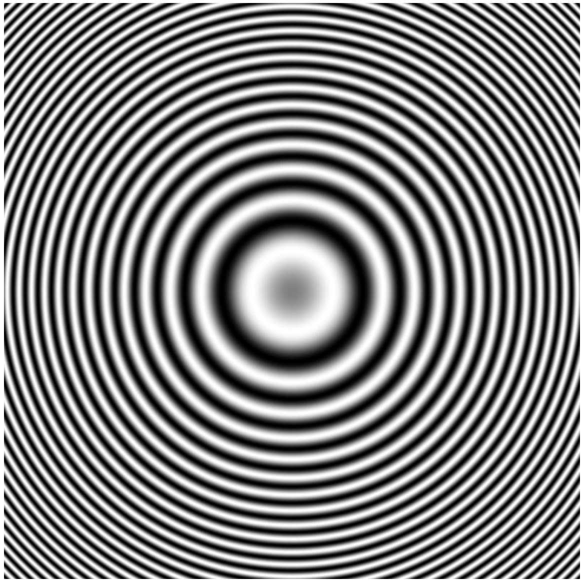
Tomáš Faragó, tomas.farago@kit.edu

Free-space Wave Field Propagation

- Propagated wave field: $u(\vec{x}, z_i + \Delta z) = \mathcal{F}^{-1}\left\{\mathcal{F}[u(\vec{x}, z_i)].\widetilde{K}\left(\vec{\xi}, \Delta z\right)\right\}$
- Propagator in Fourier space: $\widetilde{K}\left(\vec{\xi}, \Delta z\right) = e^{j\frac{2\pi}{\lambda}\Delta z\sqrt{1-\left(\lambda\vec{\xi}\right)^2}}$, $\vec{\xi}$ spatial freq.
- In real space: $K(\vec{x}, \Delta z) = \dfrac{e^{j\frac{2\pi}{\lambda}r}}{j\lambda r}$, $r = \sqrt{(\vec{\eta} - \vec{x})^2 + (\Delta z)^2}$



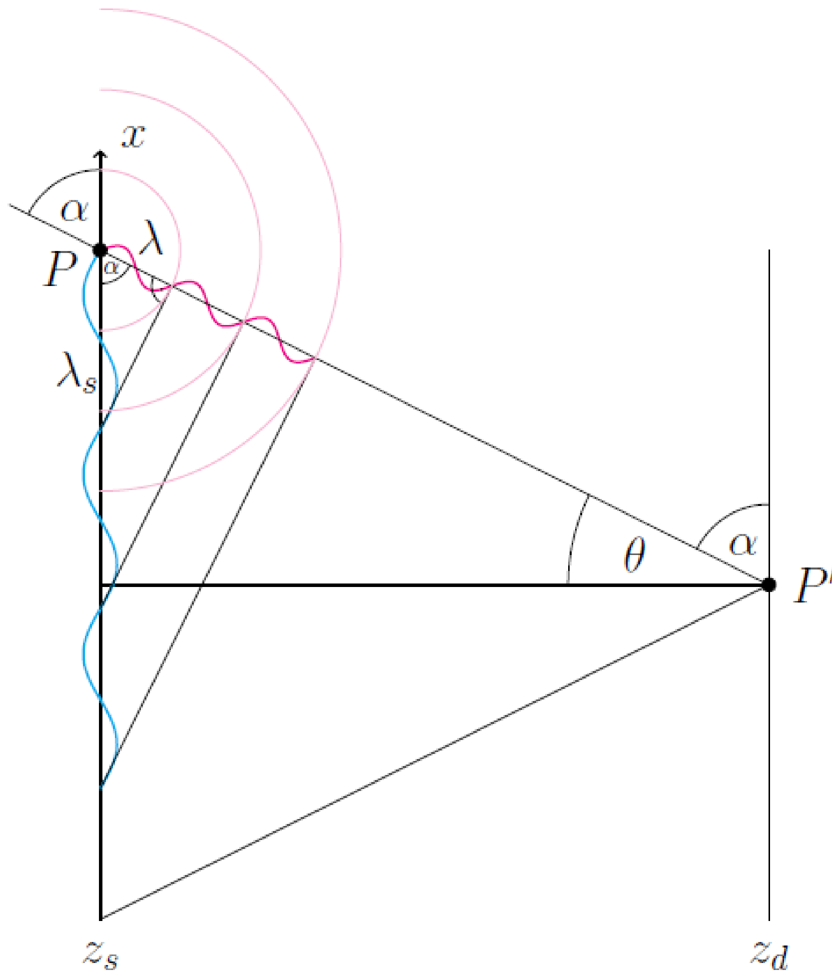
2D propagation scheme



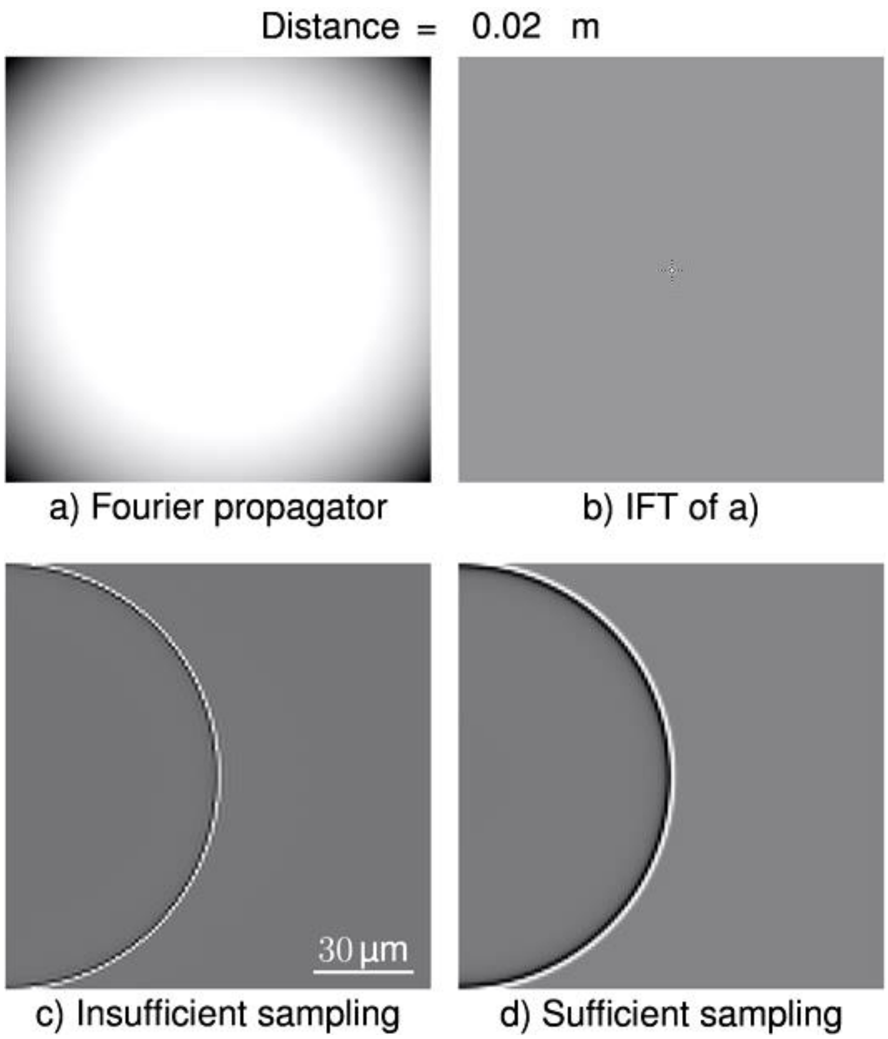
$Re[K(\vec{\eta}, \Delta z)]$

Angular Spectrum Sampling

- $\cos(\alpha) = \sin(\theta) = \frac{\lambda}{\lambda_s}$
- Pixel size
  - $\Delta x = \frac{\lambda_s}{2}$
- $\theta_{max} = \sin\left(\frac{\lambda}{2\Delta x}\right)^{-1} \sim \frac{\lambda}{2\Delta x}$
- Number of pixels
  - $N \sim \frac{2\theta z}{\Delta x}$



Propagator Aliasing

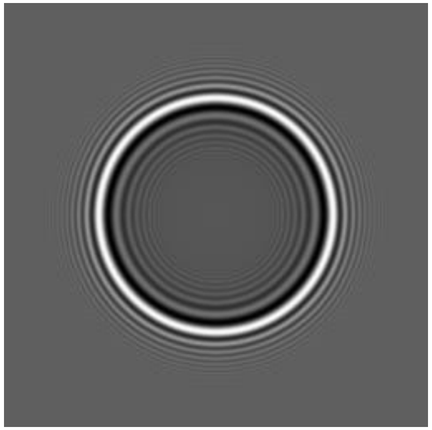


Detection

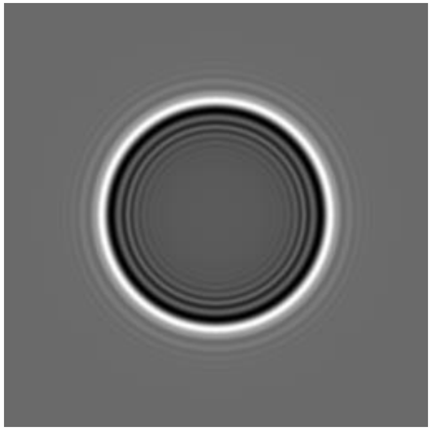
- Indirect detector
  - Convert X-rays to visible light
  - Magnify image by a lens
  - Detect with a conventional camera

- Effects
  - Light attenuation
  - Blurring
  - Noise
    - Shot noise (Poisson distribution)
    - Electronics noise (Normal distribution)
    - Quantization noise (Uniform distribution)

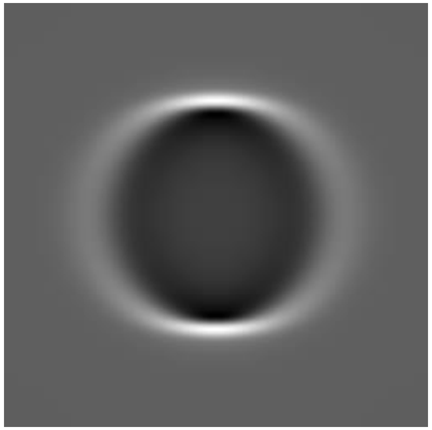
Image Formation Effects



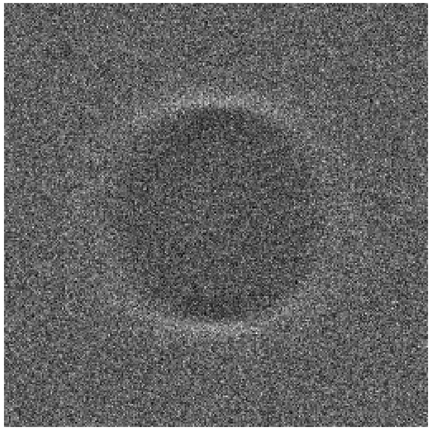
Coherent



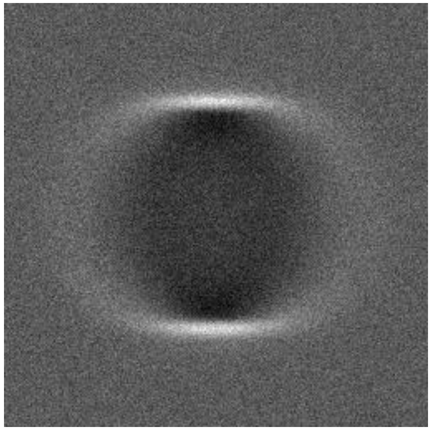
Polychromatic



Reduced spatial Coherence



Noisy



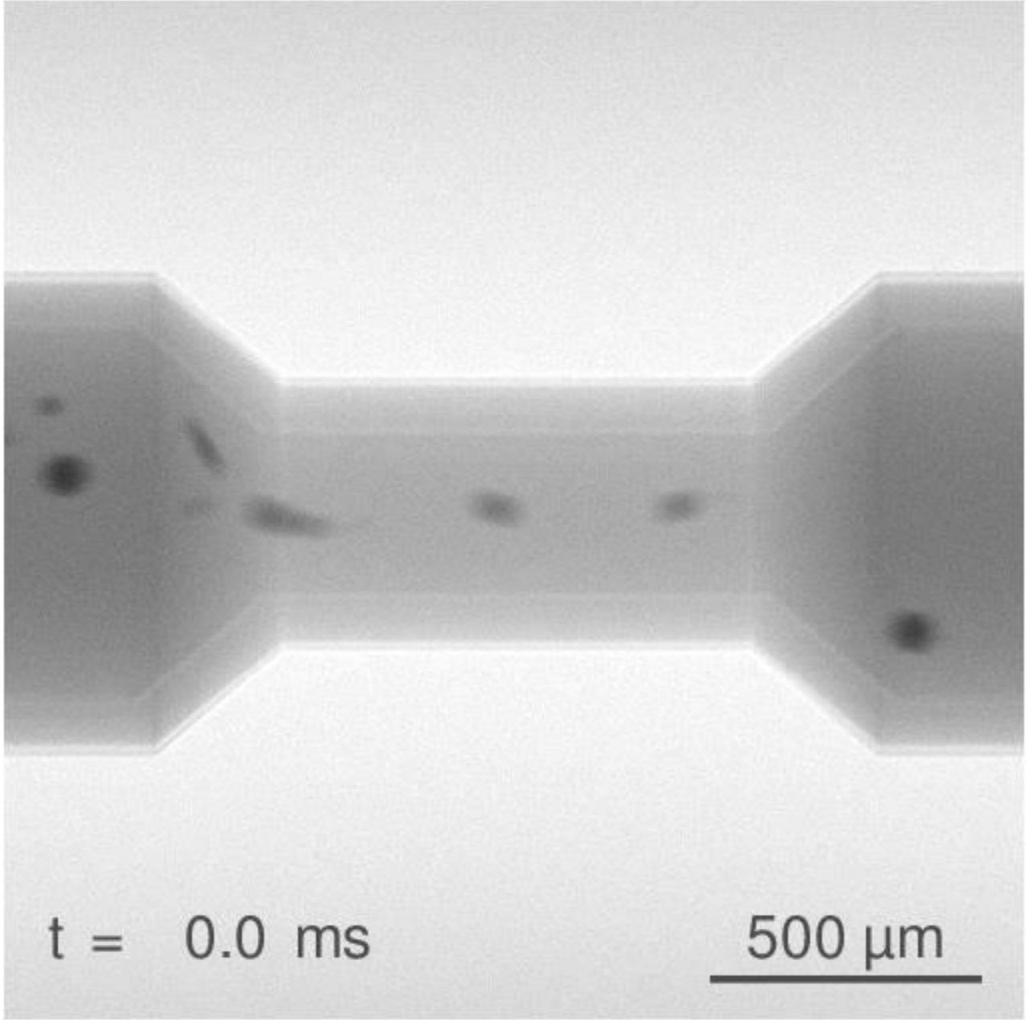
Motion blurred

Tomáš Faragó, tomas.farago@kit.edu

Simulation of a Process

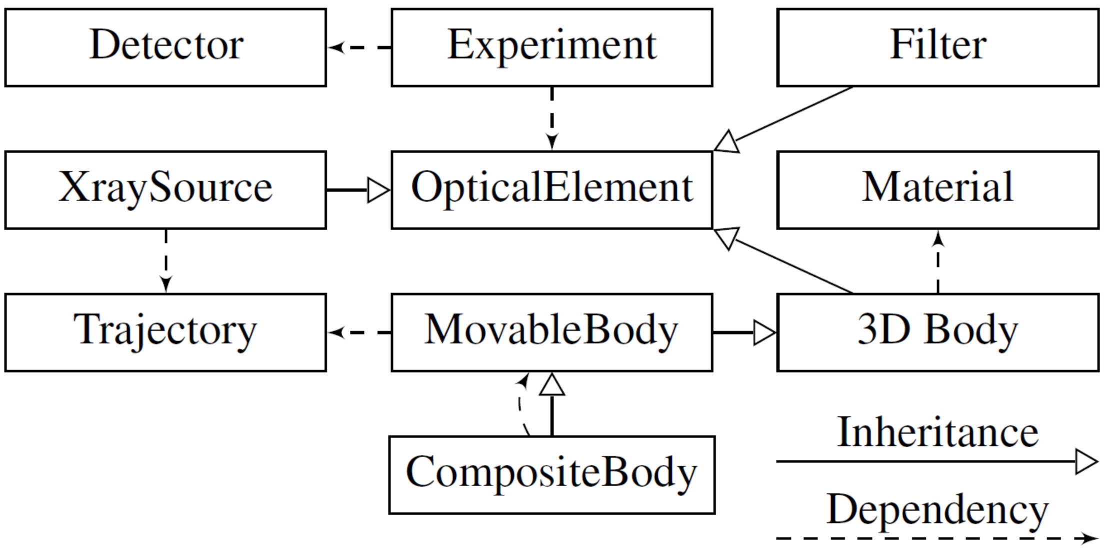
- 5 000 frames/s
- Motion speed

  20 pixels/frame
- *syris* accounts for:
  - Motion blur
  - Noise
  - Beam flicker



Implementation

- Lightweight Python interface
- Physical quantities
- Refractive index
  - CXRO database (web)
  - X0h database (web)
  - pmasf program
- Compute-intensive operations in OpenCL



Simplified class diagram of *syris*

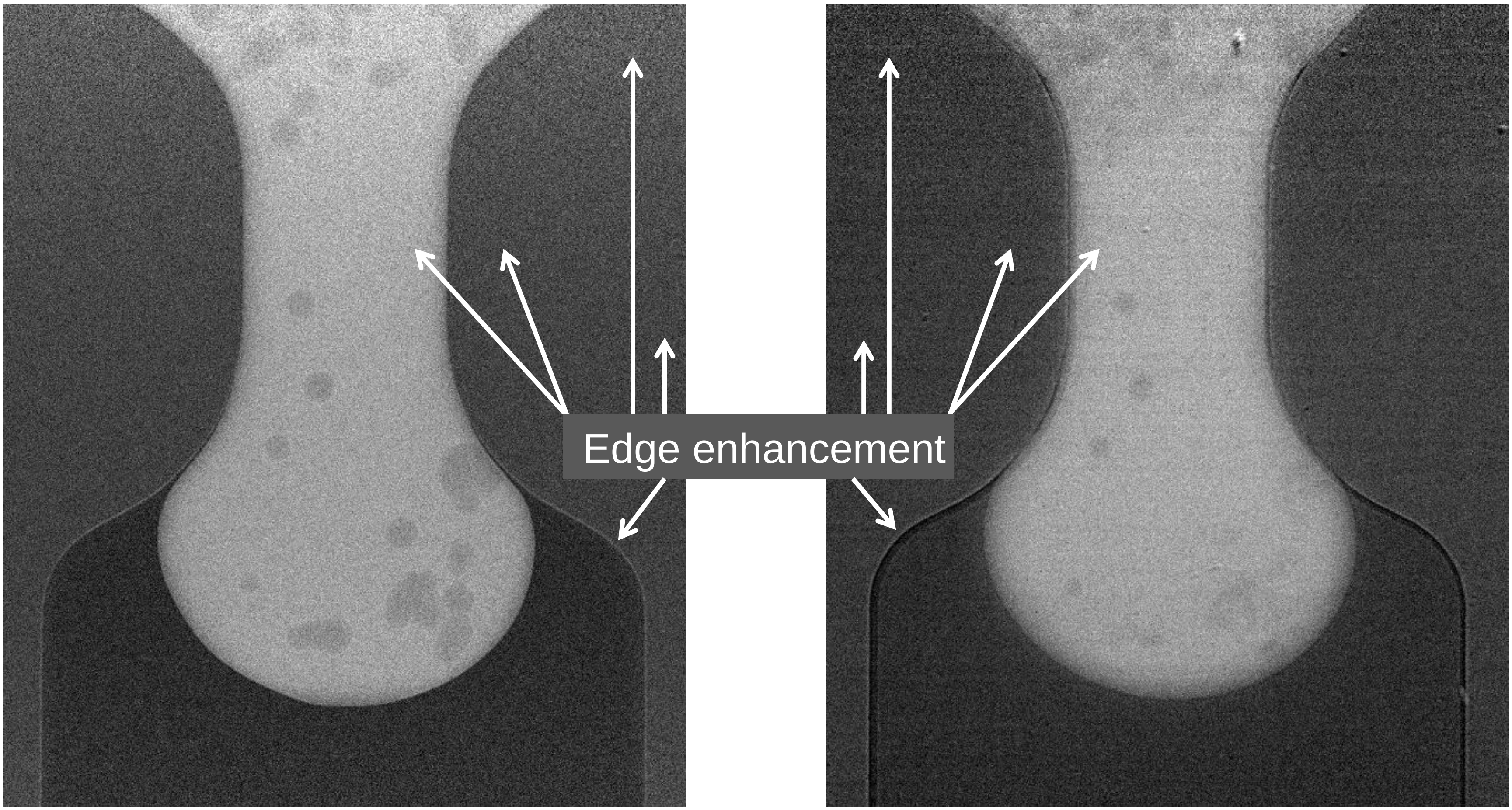
Code Snippet: Motion

```
# Create a linear trajectory
x = np.linspace(0, 1, num=128)
y = z = np.zeros_like(x)

# Create spline control points
control_points = zip(x, y, z) * q.mm

# Trajectory with a constant velocity
trajectory = Trajectory(control_points,
                        velocity=5 * q.um / q.s)

# MetaBall with radius 20 micrometer
body = MetaBall(trajectory, 20 * q.um)
```

Code Snippet: Wave Field Propagation

```
# A dimensionless 2D pixel grid
grid = (1024, 1024)
energies = range(15, 30) * q.keV
pixel_size = 1 * q.um
material = make_pmasf('PMMA', energies)
sphere = make_sphere(grid, 256 * q.um,
                     pixel_size=pixel_size,
                     material=material)

# Propagate to 1 m
result = propagate([sphere], grid, energies,
                   1 * q.m, pixel_size)
```

Realistic Simulation Example



Simulation

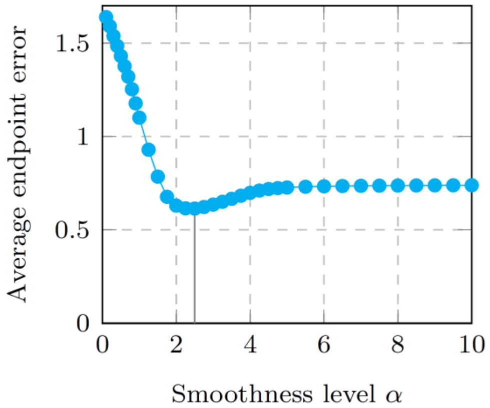
Real data

Motion Estimation Parameter Finding

- Optical flow sensitive to
  - Low contrast
  - Amount of edge enhancement (free-space propagation consequence)
  - Noise
- Benchmark algorithms in terms of the *average endpoint error*
  - $EE = \sqrt{(u_{GT} - u_{res})^2 + (v_{GT} - v_{res})^2}$, u, v motion vector components
- Apply the optimized algorithm on the real data

Algorithm	Average EE
1. Horn and Schunck	0.664
2. 1 + robust flow-driven	0.655
3. 2 + combined local-global	0.624
4. 3 + flow filtering	**0.560**

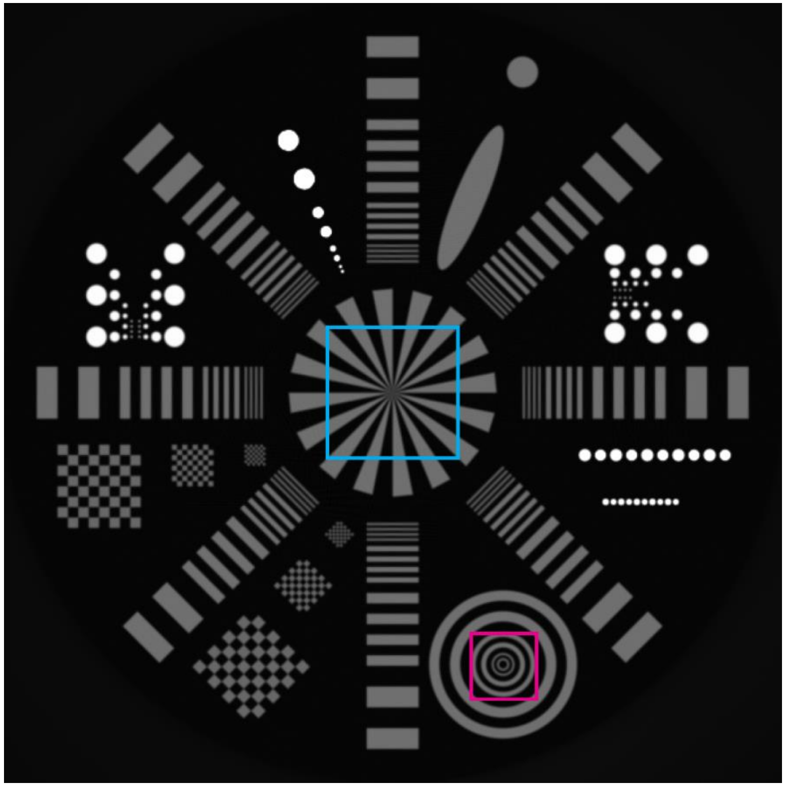
Algorithm comparison



Smoothness level optimization

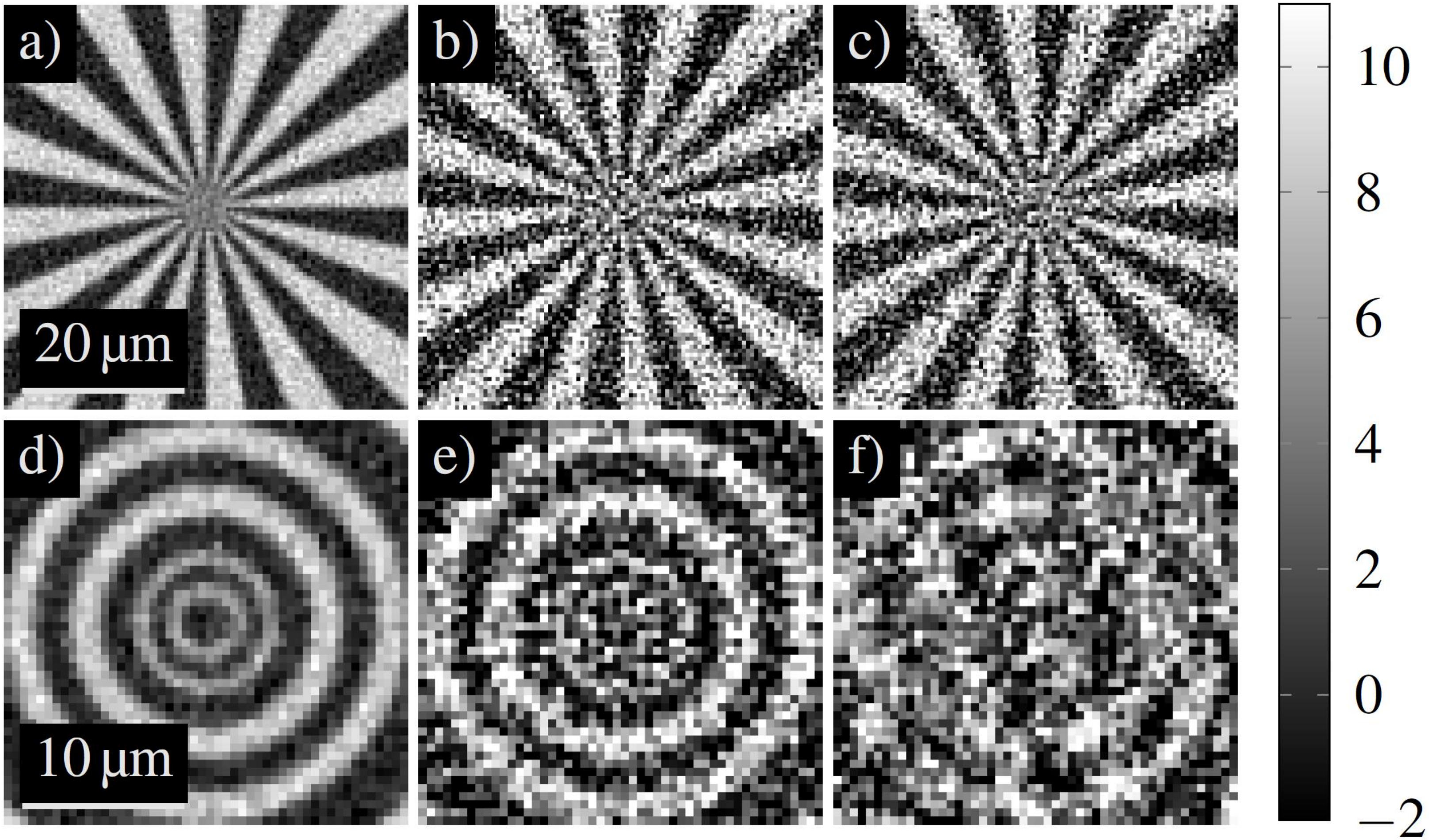
Speedup of CT Data Acquisition

- Simulate CT acquisition strategies and study reconstruction accuracy
- Ideal conditions: *N* projections, *t* exposure time
- **Reduce *t***
  - SNR decreases
- **Reduce *N***
  - SNR decreases
  - Angular sampling violation
- *What to reduce under which circumstances?*



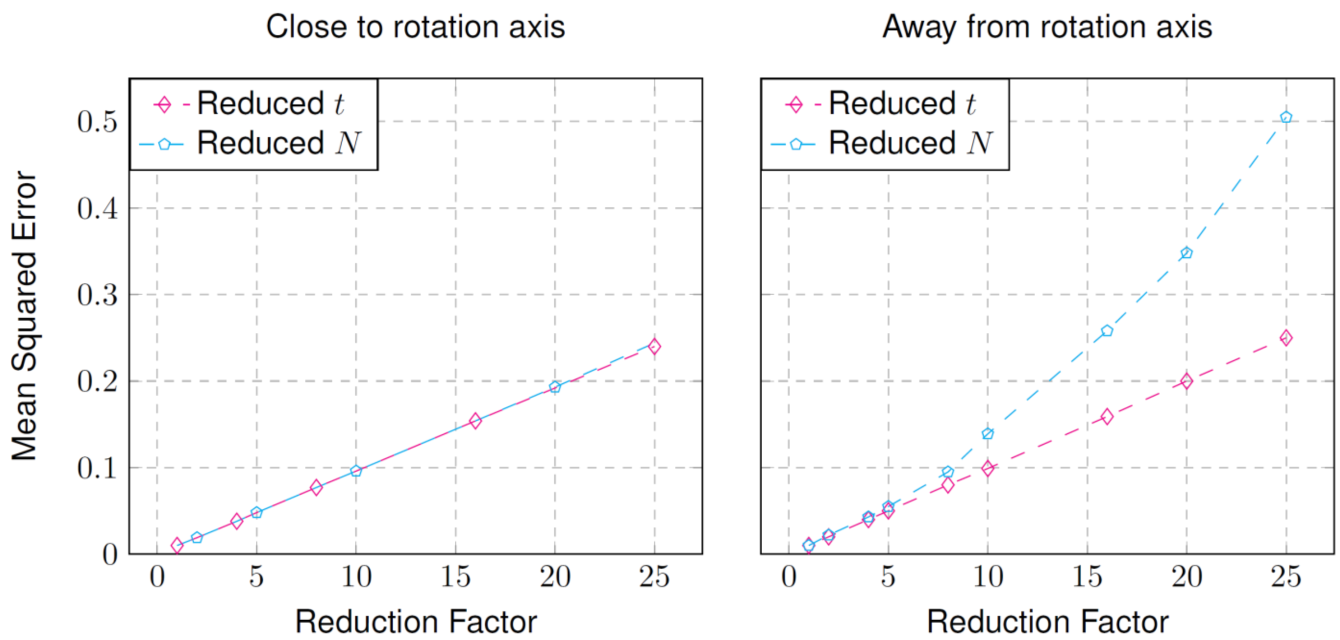
Tomographic phantom

Reconstruction Accuracy Differences



Two reconstructed slice ROIs from different scanning modes

Quantitative Comparison



MSE of the two ROIs as a function of either reducing *t* or *N*

- Result: reduction of *N* up to 5 in both cases provides usable reconstruction
- Far less data to process → faster reconstruction

Conclusion and Outlook

- *syris* provides
  - Lightweight Python high-level interface
  - **Soon open–source**: https://github.com/ufo-kit/syris
  - Model of the complete image formation process
  - Sampling violation detection
  - Motion description
  - Fast computation
- Applications
  - Investigation of novel imaging methods
  - Optimization of experimental and data processing parameters
  - Benchmarking and categorization of data processing pipelines
- Outlook
  - Create a database of data sets for different image processing algorithms
  - Add realistic sample behavior, e.g. mechanical and fluid dynamics
  - Implement more beam line elements, e.g. undulator source