High-performance coherent X-ray imaging data analysis using PyNX

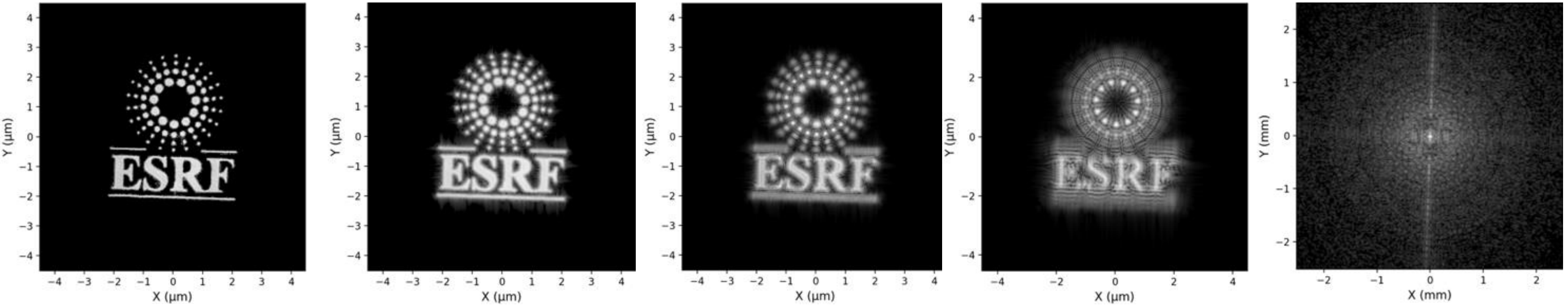
Vincent Favre-Nicolin

Algorithms & scientific Data Analysis group

ESRF | The European Synchrotron

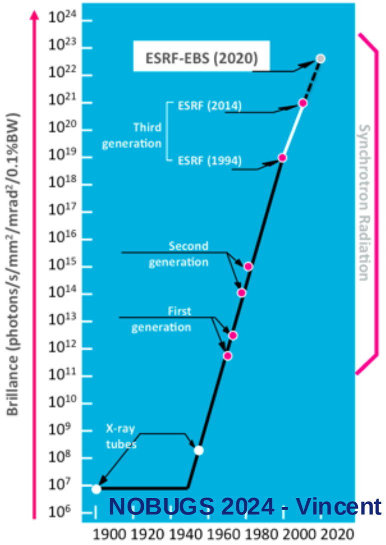
COHERENT X-RAY IMAGING @ESRF

Propagation distance



Near Field

Far Field

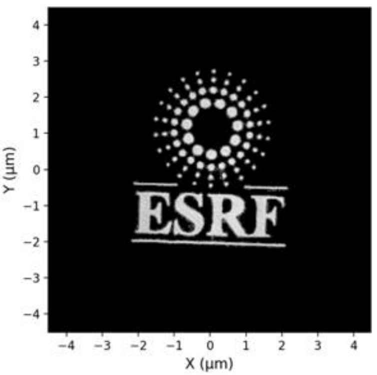


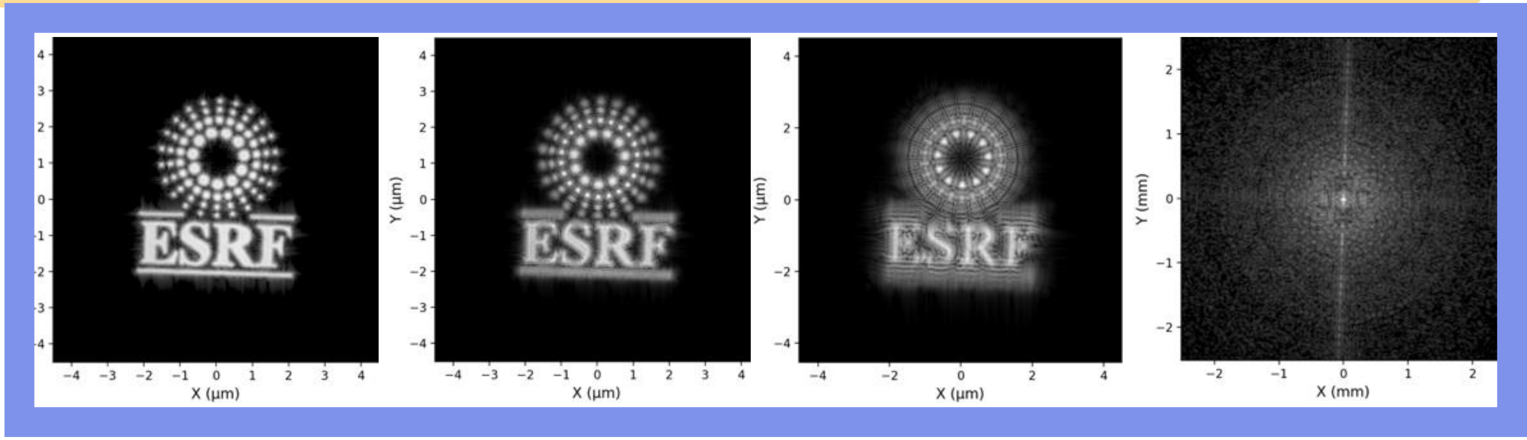
- 10-100x more coherent flux since 2020
- Need for faster, more efficient data analysis
- Many techniques (near and far field, small angle and Bragg, 2/3D, tomo,..)

The European Synchrotron | ESRF

COHERENT X-RAY IMAGING: ALGORITHMS ?

Propagation distance



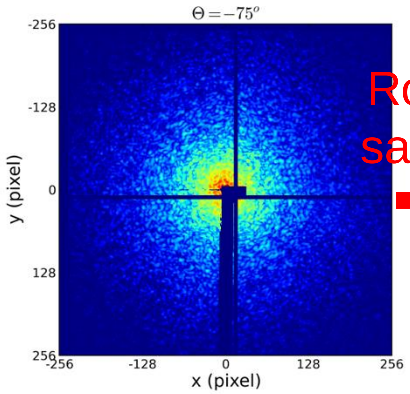


Object

The Phase problem

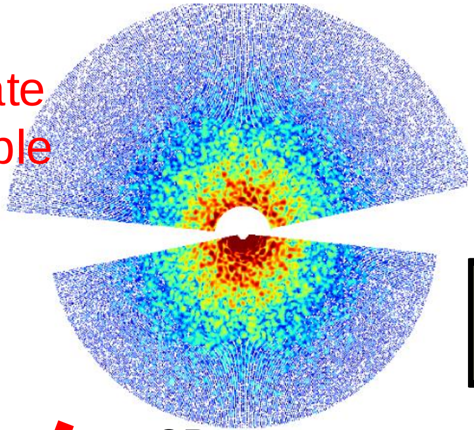
- Only the intensity is measured
- Complex algorithms required to reconstruct the object
- Iterative processes are used to yield the highest resolution

COHERENT DIFFRACTION IMAGING: THE PHASE PROBLEM

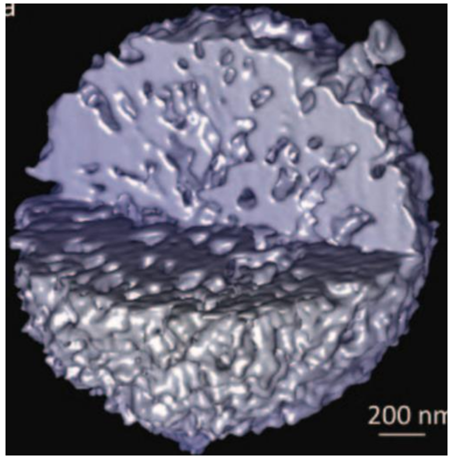


2D pattern

Rotate sample



3D pattern



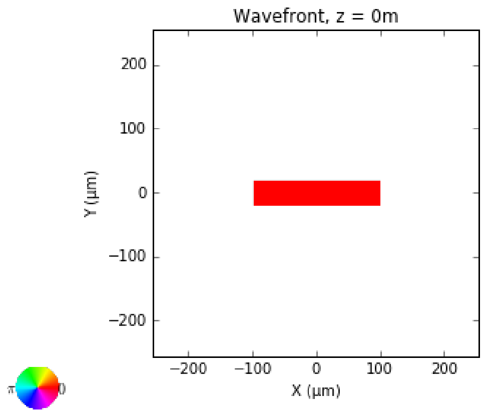
3D object

$$\begin{tikzcd}
\boxed{\text{random phases } \varphi_i} \arrow[r, dash] & \boxed{\sqrt{I_{obs}}} \arrow[d] \\
\boxed{A_{calc}(\vec{k})} \arrow[r, "\varphi_{calc}"'] & \boxed{A_{obs}(\vec{k})} \arrow[d, dash, "FT^{-1}"] \\
\boxed{\rho} \arrow[u, "FT"'] & \boxed{\hat{\rho}} \arrow[l]
\end{tikzcd}$$

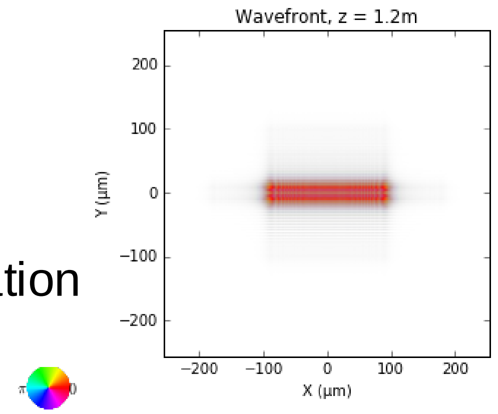
Density modification:

- Positivity (*)
- Finite support

COHERENT IMAGING: OPERATORS & GPU-FRIENDLY DATA



Near field propagation



All operations on coherent wavefronts can be described as mathematical operators:

$$W_{z=1.2m} = P(dz=1.2) * W_{z=0}$$

- Coherent imaging algorithms all work on large arrays (10^5-10^9 pixels/voxels)
- => **massively parallel computing** using **GPUs** can be used
- GPUs can read/write arrays at a speed of >500 GBytes/s
- Need optimised software: PyNX@ESRF

COHERENT IMAGING: OPERATORS

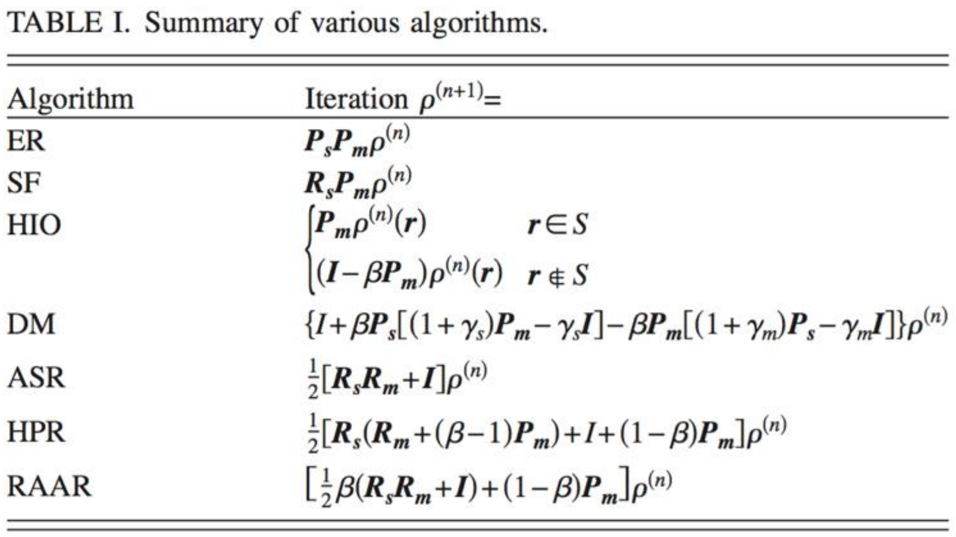


TABLE I. Summary of various algorithms.

Algorithm	Iteration $\rho^{(n+1)}=$
ER	$P_s P_m \rho^{(n)}$
SF	$R_s P_m \rho^{(n)}$
HIO	$\begin{cases} P_m\rho^{(n)}(r) & r \in S \\ (I-\beta P_m)\rho^{(n)}(r) & r \notin S \end{cases}$
DM	$\{I+\beta P_s[(1+\gamma_s)P_m-\gamma_s I]-\beta P_m[(1+\gamma_m)P_s-\gamma_m I]\}\rho^{(n)}$
ASR	$\frac{1}{2}[R_s R_m + I]\rho^{(n)}$
HPR	$\frac{1}{2}[R_s(R_m+(\beta-1)P_m)+I+(1-\beta)P_m]\rho^{(n)}$
RAAR	$[\frac{1}{2}\beta(R_s R_m+I)+(1-\beta)P_m]\rho^{(n)}$

Marchesini, S. ‘*A unified evaluation of iterative projection algorithms for phase retrieval*’. Review of Scientific Instruments **78** (2007), 011301

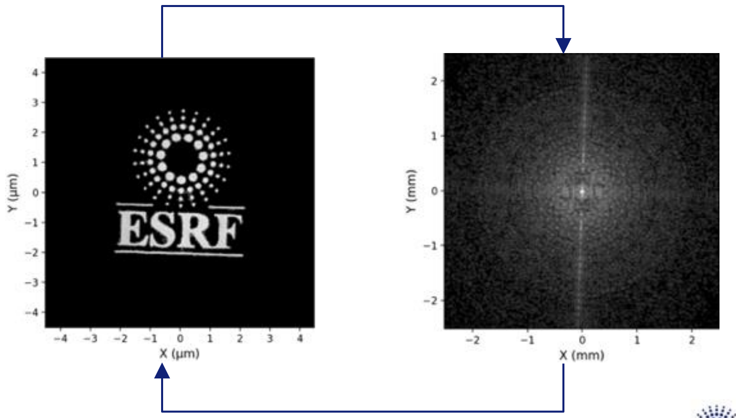
Similar for ptycho:
Marchesini *et al*, Inverse Problems 29 (2013), 115009

P_m:

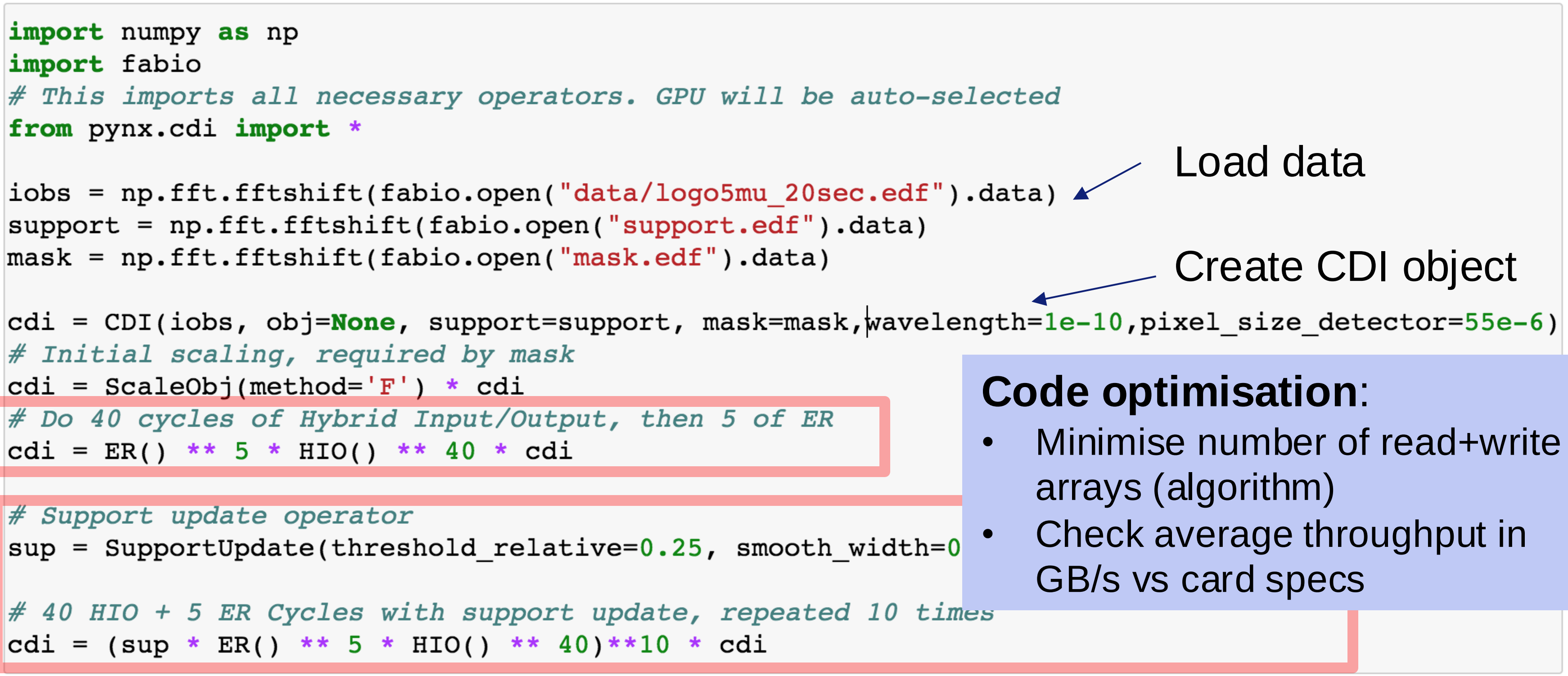
- Fourier transform the object
- Impose magnitude in Fourier space from observed intensity
- Back-Fourier Transform

P_s

- Replace density by zero outside of support



PYNX CDI OPERATORS (PYTHON API)



```
import numpy as np
import fabio
# This imports all necessary operators. GPU will be auto-selected
from pynx.cdi import *

iobs = np.fft.fftshift(fabio.open("data/logo5mu_20sec.edf").data)
support = np.fft.fftshift(fabio.open("support.edf").data)
mask = np.fft.fftshift(fabio.open("mask.edf").data)

cdi = CDI(iobs, obj=None, support=support, mask=mask,wavelength=1e-10,pixel_size_detector=55e-6)
# Initial scaling, required by mask
cdi = ScaleObj(method='F') * cdi
# Do 40 cycles of Hybrid Input/Output, then 5 of ER
cdi = ER() ** 5 * HIO() ** 40 * cdi

# Support update operator
sup = SupportUpdate(threshold_relative=0.25, smooth_width=0

# 40 HIO + 5 ER Cycles with support update, repeated 10 times
cdi = (sup * ER() ** 5 * HIO() ** 40)**10 * cdi
```

Load data

Create CDI object

Code optimisation:

- Minimise number of read+write of arrays (algorithm)
- Check average throughput in GB/s vs card specs

All operations are executed on the GPU **asynchronously** – i.e. the python code will finish before the queued operations executed using CUDA or OpenCL

GPU VS CPU COST EFFICIENCY (AMAZON)

	GPU (V100), CUDA		Xeon E5-2686 4 cores, FFTW
2D FFT (16x1024x1024)	0.81 ms	$\xrightarrow{\text{x47}}$	38 ms
3D FFT (128**3)	0.16 ms		4 ms
3D FFT (256**3)	1.04 ms		60 ms
3D FFT (512**3)	12.1 ms	$\xrightarrow{\text{x46}}$	550 ms
Amazon price/hour	3 €	$\xleftarrow{\text{x7}}$	0.4 €
Cost per 10**6 2D FFT	0.04 €	$\xrightarrow{\text{x7}}$	0.26 €
Cost per 10**6 3D FFT	10 €	$\xrightarrow{\text{x6}}$	61 €

NB: timing does not include data transfer to GPU (implies long on-GPU computing)

- GPU are ~50x faster compared to CPU (4 cores), for float32
- **Price per FFT is ~1 order of magnitude cheaper per FFT**
- GPU memory max ~~32 48~~ 80 Gb

2020

Notes: Xeon E5-2686 test on the Amazon V100 machine (4 core=8 vCPUs). 256 and 512 3D FFTs are 10-20% faster on ESRF scisoft14 (Xeon Gold 6134). FFTW with FFTW_MEASURE

GPU VS CPU COST EFFICIENCY (AMAZON)

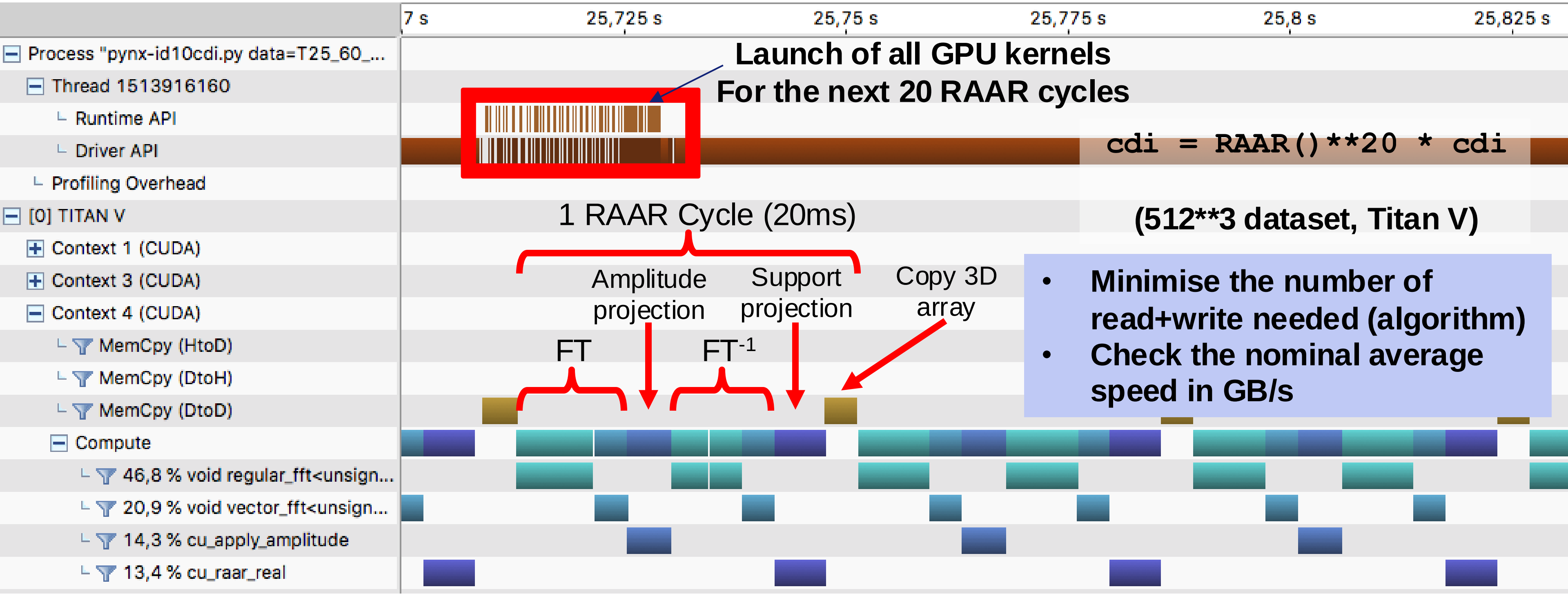
	GPU (V100), CUDA		4 cores, FFTW
2D FFT (16x1024x1024)	0.81 ms	x47 →	38 ms
3D FFT (128**3)	0.16 ms		4 ms
3D FFT (256**3)	1.04 ms		60 ms
3D FFT (512**3)	12.1 ms	x46 →	550 ms
Amazon price/hour	4 €	← x16	0.24 €
Cost per 10**6 2D FFT	0.056 €	x3 →	0.16 €
Cost per 10**6 3D FFT	13 €	x3 →	36 €

NB: timing does not include data transfer (implies long on-GPU computing)

- GPU vs CPU costs are increasing
- Still favourable: parallel processing more efficient, hardware acceleration of transcendental functions etc…
- But borderline for non-iterative transforms…

2024

PYTHON: ASYNCHRONOUS GPU EXECUTION

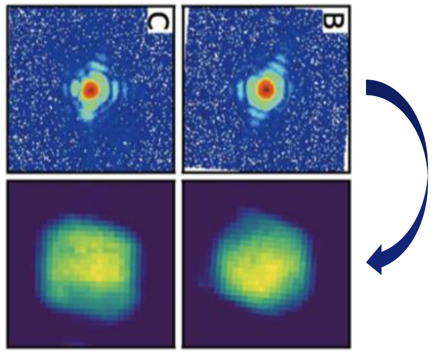


```
cdi = RAAR()**20 * cdi
```

- **Minimise the number of read+write needed (algorithm)**
- **Check the nominal average speed in GB/s**

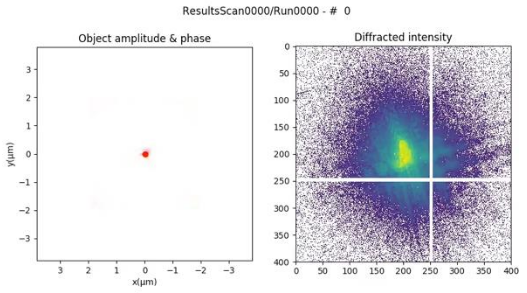
All operations are **queued** from Python, then executed **asynchronously**.
No Python<->GPU latency, both in CUDA and OpenCL
… except when reading back data from GPU (LLK, graphical display,..)
Typical performance for CDI/Ptycho: *average* 600 Gbytes/s on a V100

COHERENT IMAGING APPLICATIONS

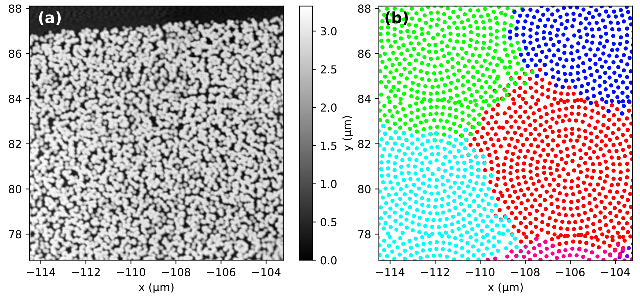


CDI- unsupervised Reconstruction (MaxIV)

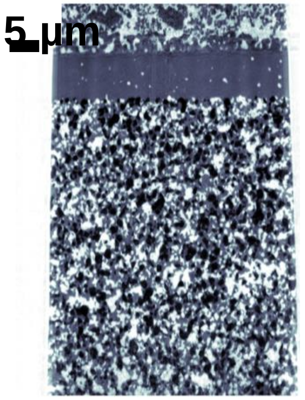
J Synchrotron Rad 26 (2019), 18300



Ptychography



Multi-GPU Ptychography (MPI, asynchronous)



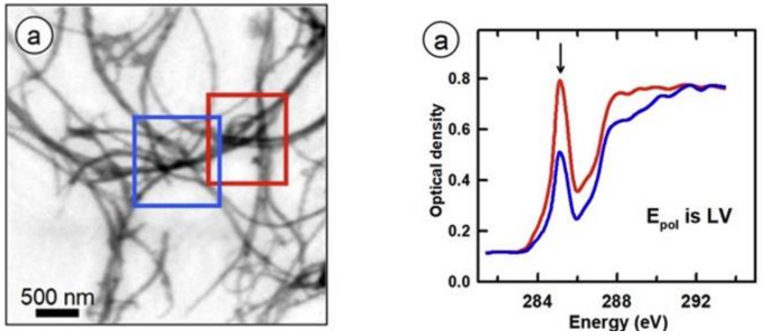
Near-field ptycho-tomography (ID16A)

1200 projections

Processing: 1'/projection

(faster for weak phase objects)

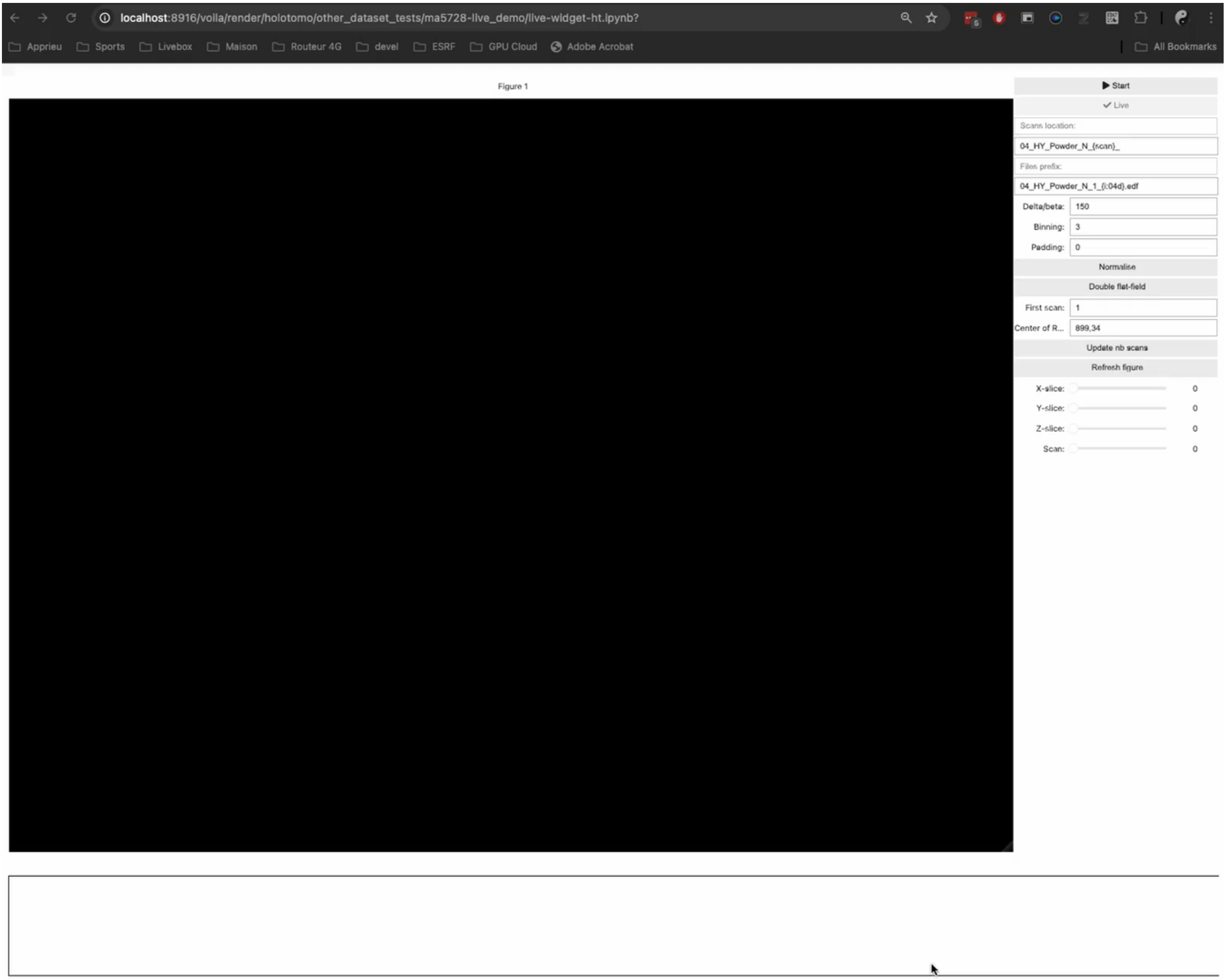
Materials Characterization 187 (2022) 111834



Spectro-Ptychography (Hermès@Soleil)

Commun Mater 3, 8 (2022)

LIVE HOLO-TOMOGRAPHY (ID16B)



localhost:8916/voila/render/holotomo/other_dataset_tests/ma5728-live_demo/live-widget-ht.ipynb?

Scans location: 04_HY_Powder_N_{scan}_

Files prefix: 04_HY_Powder_N_1_{i:04d}.edf

Delta/beta: 150

Binning: 3

Padding: 0

First scan: 1

Center of R... 899,34

X-slice: 0

Y-slice: 0

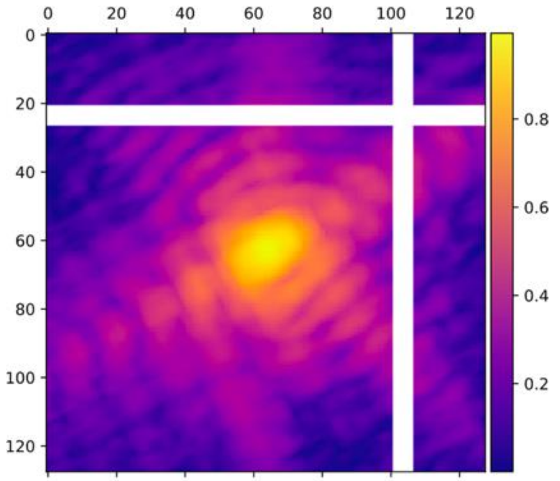
Z-slice: 0

Scan: 0

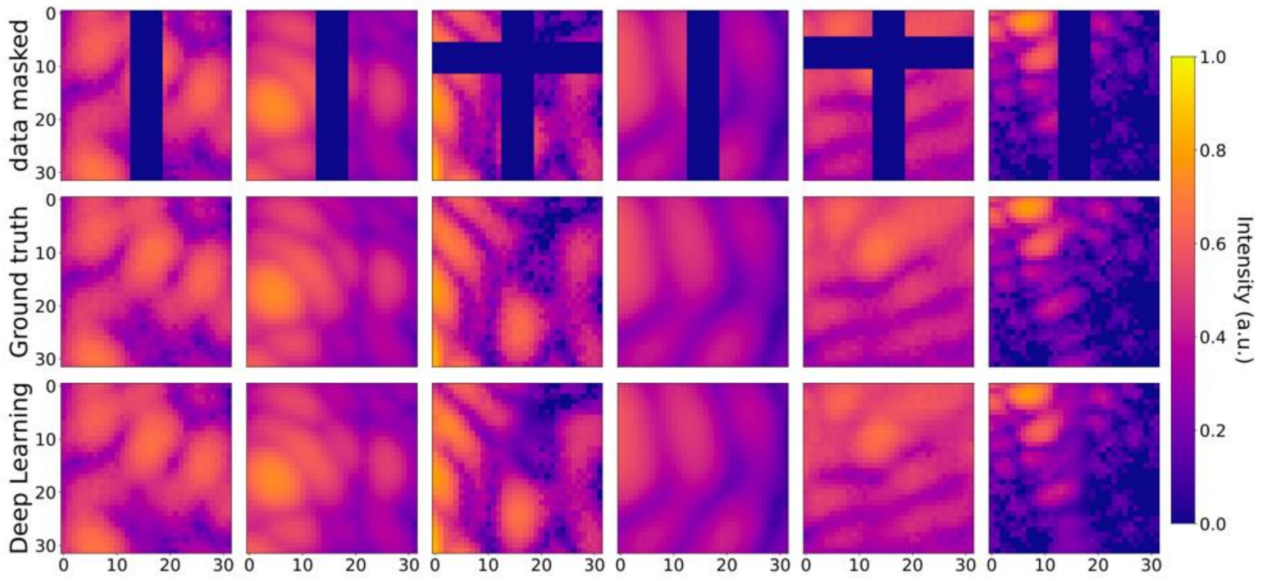
- data 2048x2048, 2500 frames
- ID16B: 1 tomogram every 3-7s for two hours
- Binned x3 for ‘live’ reconstruction
- Iterative phasing (10 cycles) + filtered back-projection (Nabu)
- 1 volume (~700^3) every 1.8 s (32 cores + 1 L40s GPU)
- multiprocessing + shared memory

- Simple web/python interface using ipywidgets + voilà (1 day of development)
- Remote interface to a compute node
- Good for prototyping user interfaces (or more..)

CDI + MACHINE LEARNING: GAP INPAINTING



Pixel detectors are great for coherent diffraction imaging… but gaps create lots of artefacts



Using deep learning it is possible to learn the shape of CDI oscillations and estimate the masked data

Learning using small patches (e.g. 32 pixels wide) instead of the whole data should allow the transfer of the neural network to a larger number of datasets

The European Synchrotron | ESRF

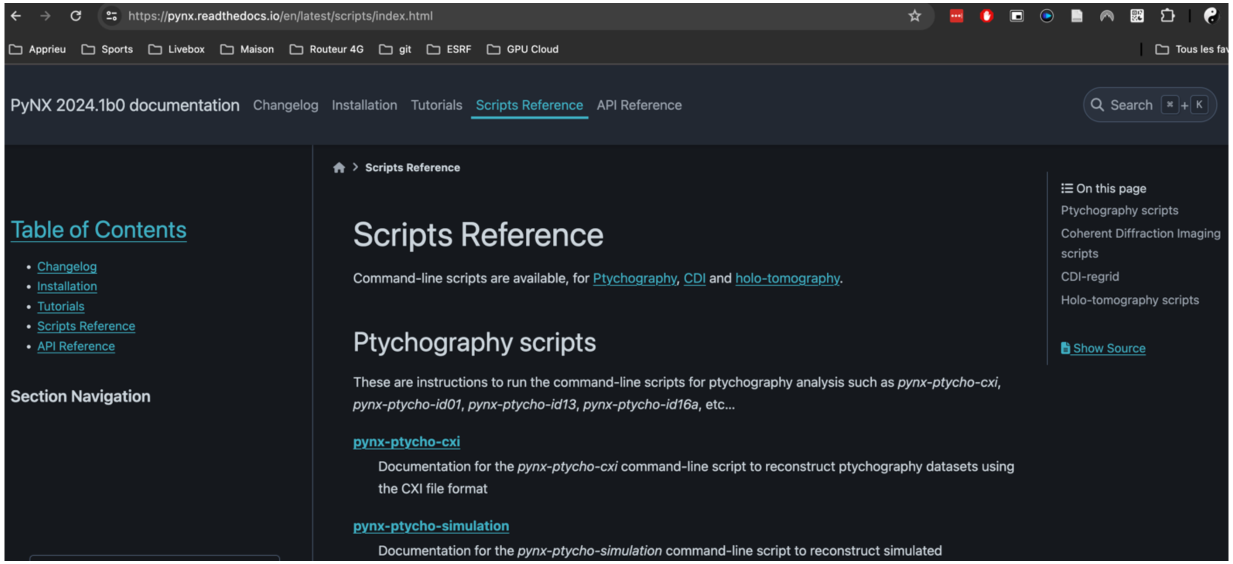
M. Masto's PhD, J. Appl. Cryst 57 (2024), 966 https://arxiv.org/abs/2403.08596

GET PYNX

PyNX 2024.1

Open-source, available from https://gitlab.esrf.fr/favre/PyNX

Doc from https://pynx.esrf.fr


https://pynx.readthedocs.io/en/latest/scripts/index.html

PyNX 2024.1b0 documentation Changelog Installation Tutorials Scripts Reference API Reference Search

Table of Contents
- Changelog
- Installation
- Tutorials
- Scripts Reference
- API Reference

Section Navigation

Scripts Reference

Scripts Reference

Command-line scripts are available, for Ptychography, CDI and holo-tomography.

Ptychography scripts

These are instructions to run the command-line scripts for ptychography analysis such as *pynx-ptycho-cxi*, *pynx-ptycho-id01*, *pynx-ptycho-id13*, *pynx-ptycho-id16a*, etc...

pynx-ptycho-cxi
Documentation for the *pynx-ptycho-cxi* command-line script to reconstruct ptychography datasets using the CXI file format

pynx-ptycho-simulation
Documentation for the *pynx-ptycho-simulation* command-line script to reconstruct simulated

On this page
Ptychography scripts
Coherent Diffraction Imaging scripts
CDI-regrid
Holo-tomography scripts

Show Source

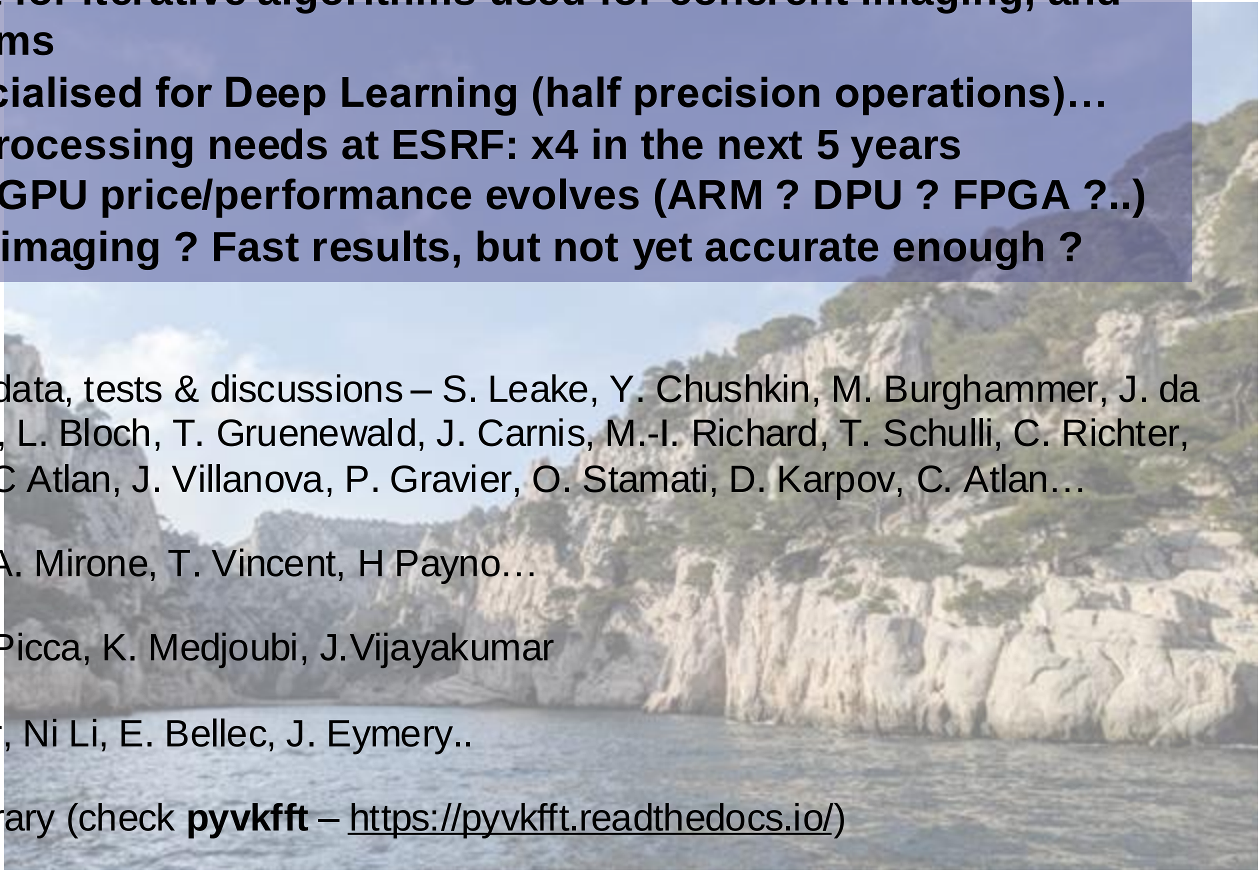
PyNX is developed at ESRF, but is also used at Soleil, Petra-III, Sirius, NSRRC/TPS…

Scripts for other instruments (data format) can be easily added for Ptycho & CDI.

ESRF The European Synchrotron SOLEIL SYNCHROTRON DESY. NSRRC LNLS sirius

CONCLUSION & ACKNOWLEDGEMENTS

- **GPUs remain highly efficient for iterative algorithms used for coherent imaging, and other highly parallel algorithms**
- **But get more expensive/specialised for Deep Learning (half precision operations)…**
- **Expected growth in data & processing needs at ESRF: x4 in the next 5 years**
- **Need to watch how the CPU/GPU price/performance evolves (ARM ? DPU ? FPGA ?..)**
- **What about ML for coherent imaging ? Fast results, but not yet accurate enough ?**



Acknowledgements:

- ID01, ID10, ID13, ID16A&B for data, tests & discussions – S. Leake, Y. Chushkin, M. Burghammer, J. da Silva, P. Cloetens, A. Johannes, L. Bloch, T. Gruenewald, J. Carnis, M.-I. Richard, T. Schulli, C. Richter, T. Zhou, E. Zatterin, G. Girard, C Atlan, J. Villanova, P. Gravier, O. Stamati, D. Karpov, C. Atlan…
- P. Paleo, N. Vigano, J. Kieffer, A. Mirone, T. Vincent, H Payno…
- Soleil: D. Simonne, N. Mille, F. Picca, K. Medjoubi, J.Vijayakumar
- CEA: M.-I. Richard, C. Chatelier, Ni Li, E. Bellec, J. Eymery..
- D. Tolmachev for the **VkFFT** library (check **pyvkfft** – https://pyvkfft.readthedocs.io/)
- **All the PyNX users for feedback and interesting data**