Supplementary information for the article:

PyNX: high performance computing toolkit for coherent X-ray imaging based on operators

Vincent Favre-Nicolin, Gaétan Girard, Steven Leake, Jérôme Carnis, Yuriy Chushkin, Jérôme Kieffer,
Pierre Paléo and Marie-Ingrid Richard

1 OpenCL vs CUDA FFT performance

Both OpenCL and CUDA languages rely on the same hardware. Generally speaking, the performance is almost identical for floating point operations, as can be seen when evaluating the scattering calculations (Mandula et al, 2011). However the FFT performance depends on low-level tuning of the underlying libraries, namely the cuFFT and clFFT libraries, which are respectfully optimised for Nvidia and AMD devices.

The performance of both libraries has been evaluated for an Nvidia V100 GPU, for 2D and 3D FFT of all sizes for which the largest prime factor decomposition is at most 7 (note that clFFT allows up to 13, and cuFFT allows values larger than 7 but with a degraded performance). The configuration used for the comparison was: Nvidia driver 435.21, CUDA version 10.1, clFFT v2.12.2, pyopencl 2019.1.2, pycuda 2019.1.2, gpyfft git commit 2c07fa8e7674757.

As performance on a GPU is limited by the memory throughput rather than the floating-point operations, we report here the average processing speed in GB/s, taking into account the N read and write operations for the N-dimensional FFT. Each test is done by performing two pairs of backward and forward FT in single precision (32-bit floating point), and the test is repeated four times, the best time being kept for reporting. In the case of clFFT, each possible order for the axes transforms (a N-dimensional FT is a succession of N 1-dimensional FT) for the FT is tested and the best time is used.

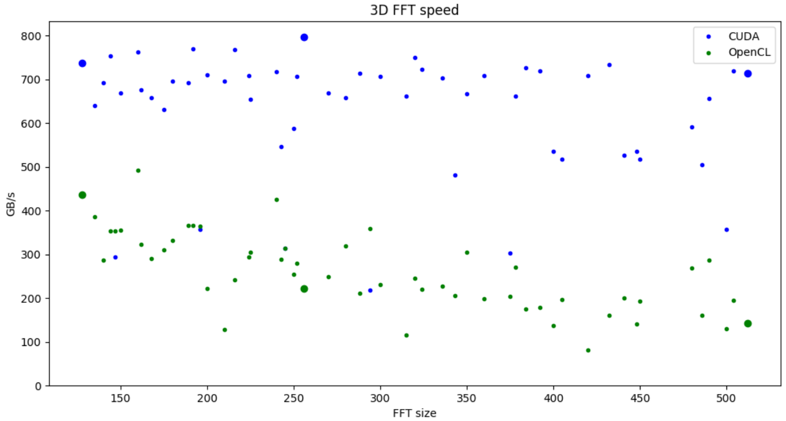


Figure 1: 3D FFT performance, measured on a Nvidia V100 GPU, using CUDA and OpenCL, as a function of the FFT size. The obtained speed can be compared to the theoretical memory bandwidth of 900 GB/s. Larger dots are shown for power-of-twos transforms

Note that **these tests do not imply that cuFFT is superior to clFFT *in general*, but rather that it is at least the case *on Nvidia hardware***. This is expected as clFFT is optimised for AMD GPU. One notable difference is the warp-size which is 32 for Nvidia GPU, whereas for AMD the wavefront is 64 – both numbers correspond to the number of low-level compute threads which are executed in parallel on a compute unit – a difference which can explain that clFFT tuning is not optimal on Nvidia hardware.

Also note that 'in-place' cuFFT transforms require 2x the amount of memory for the transform, in order to optimise memory transfers (it is faster to copy values from adjacent memory, so the transforms along the different axis of the FFT are better optimised by re-arranging the memory). clFFT does not require this (which may be a reason for a lower performance), and can thus handle larger transforms, which can be useful for large 3D CDI FFT.

All the tests can be reproduced using the function:

```
pynx.test.speed.plot_fft_speed()
```

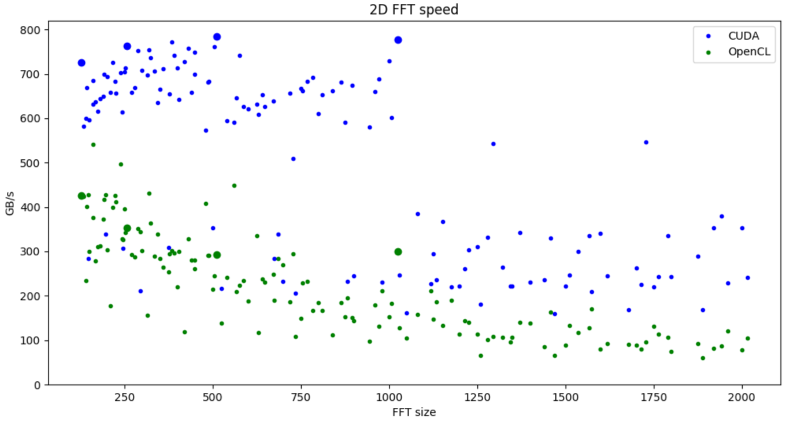
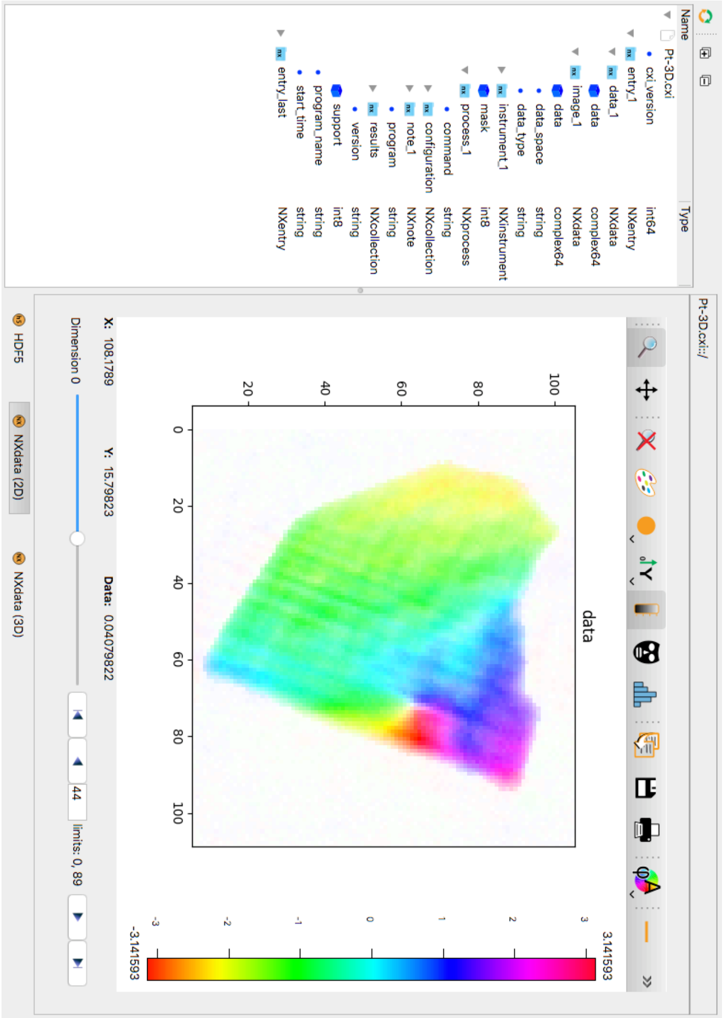


Figure 2: 2D FFT performance, measured on a Nvidia V100 GPU, using CUDA and OpenCL, as a function of the FFT size up to N=2000. The obtained speed can be compared to the theoretical memory bandwidth of 900 GB/s. Larger dots are shown for power-of-twos transforms

2 Larger version of Figure 5a and b



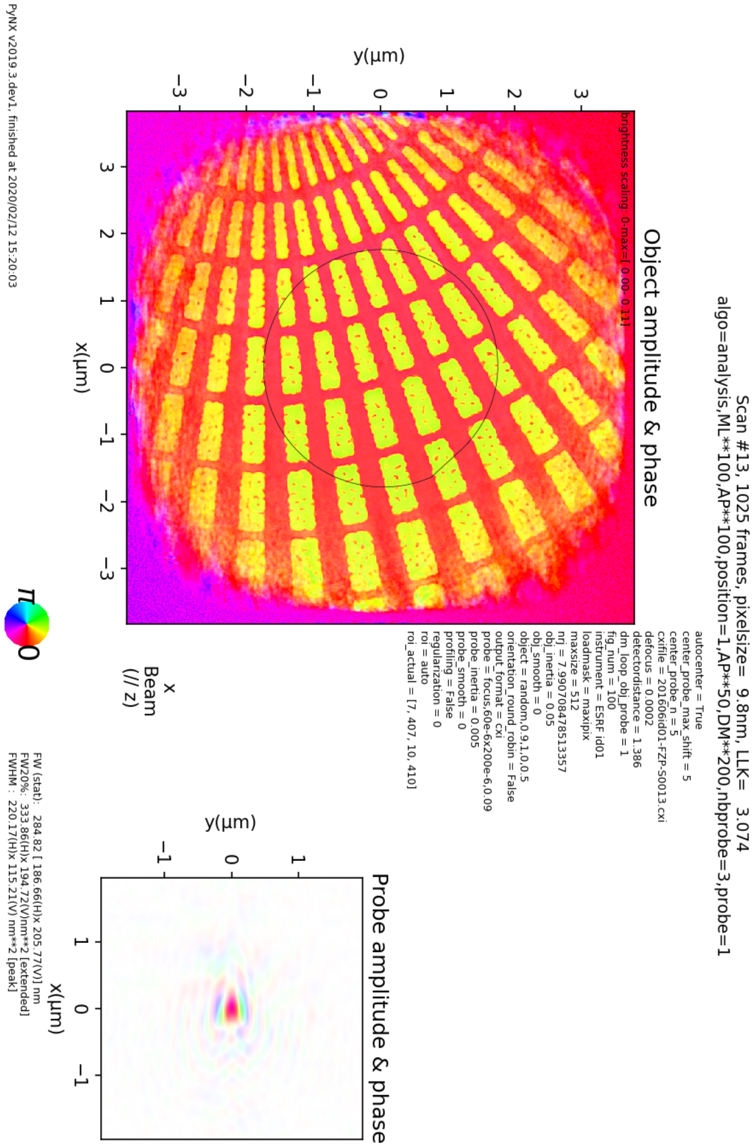
Pt-3D.cxi::/

Name	Type
Pt-3D.cxi	
cxi_version	int64
entry_1	NXentry
data_1	NXdata
data	complex64
image_1	NXdata
data	complex64
data_space	string
data_type	string
instrument_1	NXinstrument
mask	int8
process_1	NXprocess
command	string
configuration	NXcollection
note_1	NXnote
program	string
results	NXcollection
version	string
support	int8
program_name	string
start_time	string
entry_last	NXentry

X: 108.1789 Y: 15.79823 Data: 0.04079822

Dimension 0 44 limits: 0, 89

Figure 3: larger version of article Figure 5b. Display of the CXI file, using the silx toolkit viewer, obtained as a result from a CDI analysis. As the CXI file is NeXus-formatted, the view automatically opens the relevant object. The HSV colour map gives information both about the amplitude and the phase of the object. This example is the result of Bragg CDI on a Pt nano-crystal with a dislocation. As can be seen in the left of the image, different fields in the CXI/hdf5 file include information about the object as well as the process and parameters used for the analysis.



Scan #13, 1025 frames, pixelsize= 9.8nm, LLK= 3.074
algo=analysis,ML**100,AP**50,DM**200,nbprobe=3,probe=1

```
autocenter = True
center_probe_max_shift = 5
center_probe_n = 5
cxifile = 201606id01-FZP-S00013.cxi
defocus = 0.0002
detectordistance = 1.386
dm_loop_obj_probe = 1
fig_num = 100
instrument = ESRF id01
loadmask = maxipix
maxsize = 512
nrj = 7.990708478513357
obj_inertia = 0.05
obj_smooth = 0
object = random,0.9,1,0,0.5
orientation_round_robin = False
output_format = cxi
probe = focus,60e-6x200e-6,0.09
probe_inertia = 0.005
probe_smooth = 0
profiling = False
regularization = 0
roi = auto
roi_actual = [7, 407, 10, 410]
```

```
FW (stat):   284.82 [ 186.66(H)x 205.77(V)] nm
FW20%:  333.86(H)x 194.72(V)nm**2 [extended]
FWHM :  220.17(H)x 115.21(V) nm**2 [peak]
```

PyNX v2019.3.dev1, finished at 2020/02/12 15:20:03

Figure 4: larger version of Fig.5a - example output plot of a ptychography experiment, showing both the object and the probe in RGBA, as well as all parameters used for the analysis

3 Larger view of the MPI-ptychography reconstruction of the modulator

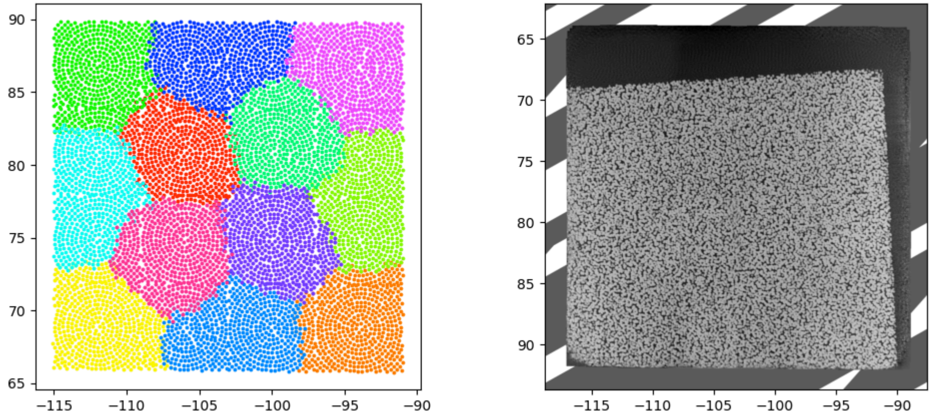


Figure 5: overview of the reconstructed modulator, shown in the article Figure 3. Left: Division of the scanning positions into 12 domains with ~575 points each, including ~30 shared with the neighbours. Right: complete view of the reconstructed object phase

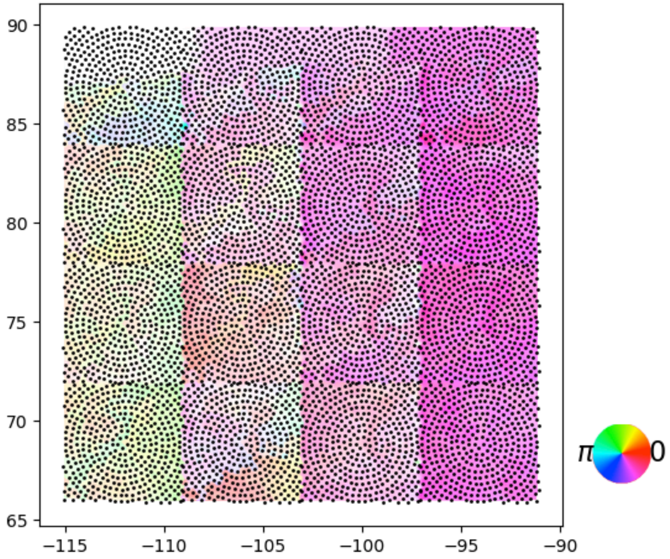


Figure 6: complete heat map of the optimised position shifts (see article fig 3c for explanations), which shows that the overall trend is a drift vs time for the 16 successive scans which compose the entire dataset. In this representation, the maximum displacement is 313nm, and the reference region at the top left.