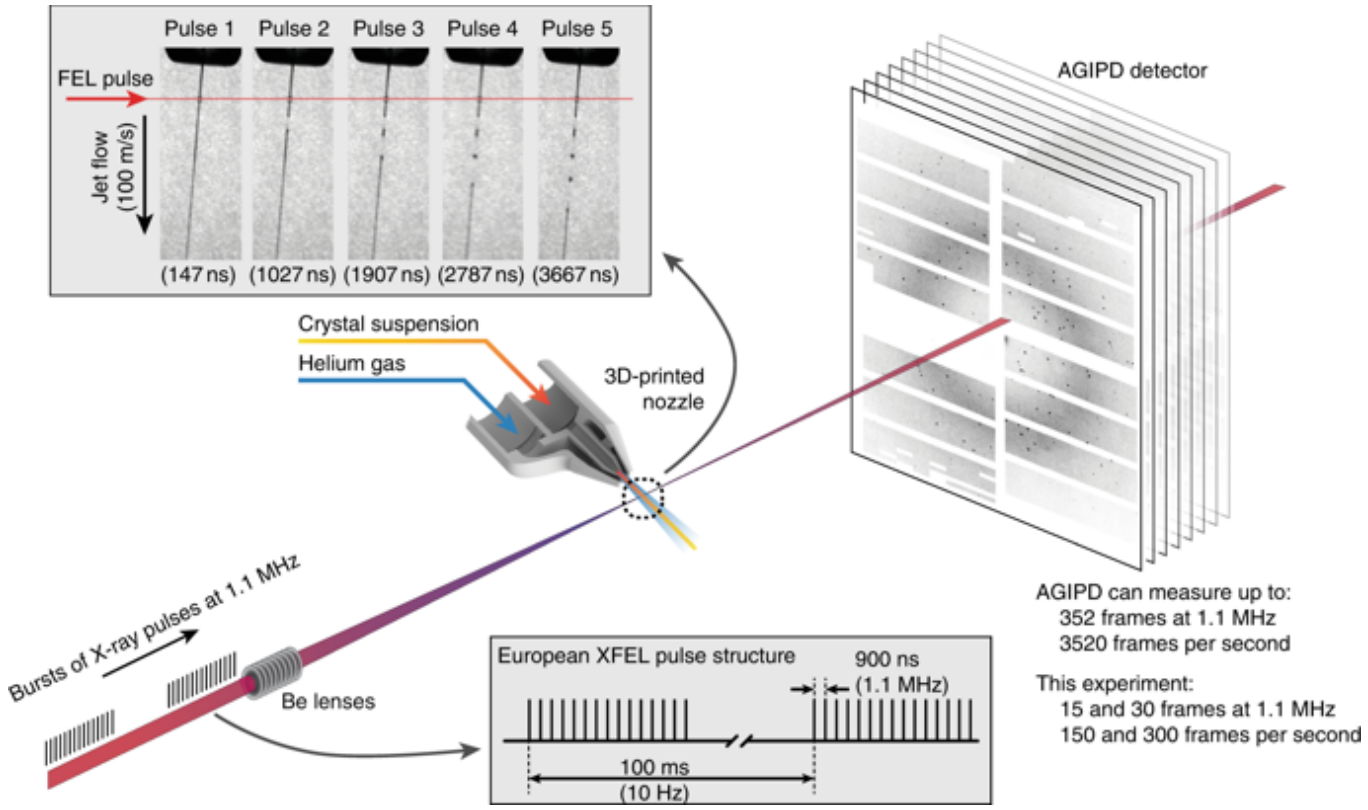
Data reduction at EuXFEL: first steps

Egor Sobolev
Data Analysis Group

XFEL Users Meeting 2022
Schenefeld, 25.01.2022

European XFEL is the fastest X-ray laser in the world

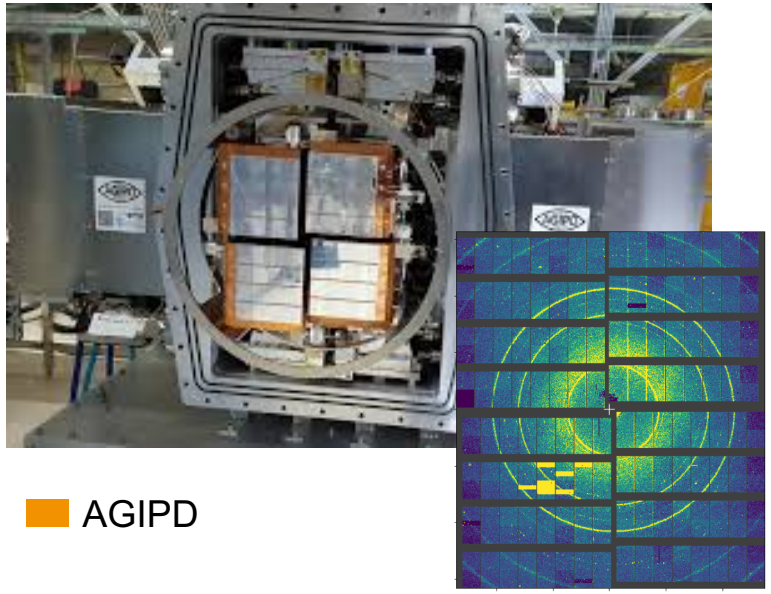
27000 pulses per second



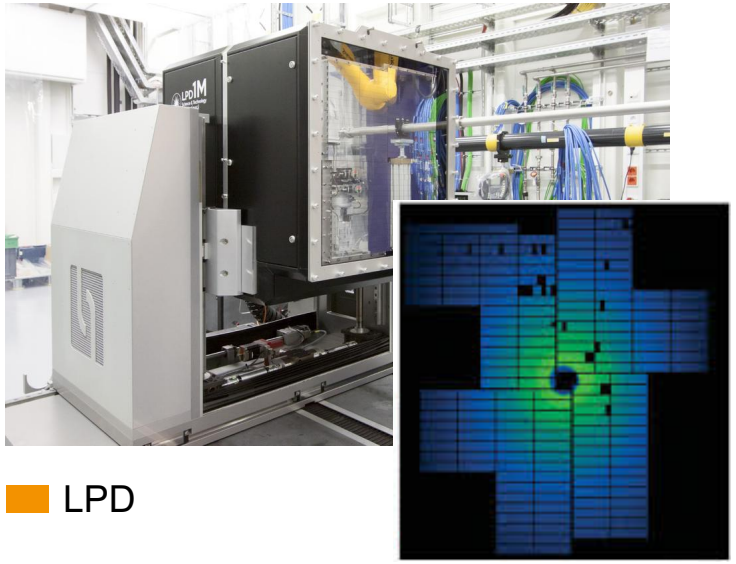
Wiedorn, M.O., Oberthür, D., Bean, R. et al. Megahertz serial crystallography. Nat Commun 9, 4025 (2018)

Where from do the big data come?

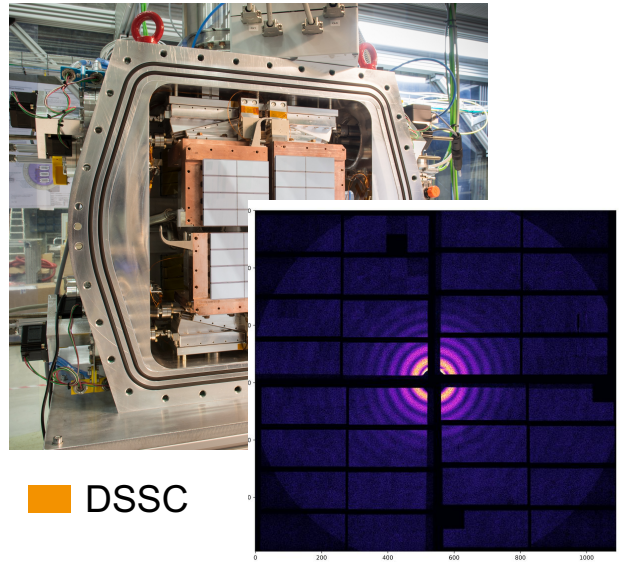
- Fast area detectors:
  - up to 8000 1Mpx frames per second with 14-30 GiB/s (up to 100 TiB/hour)
  - typical amount is about 120TiB per experiment, the biggest > 3.1 PiB raw data



AGIPD



LPD



DSSC

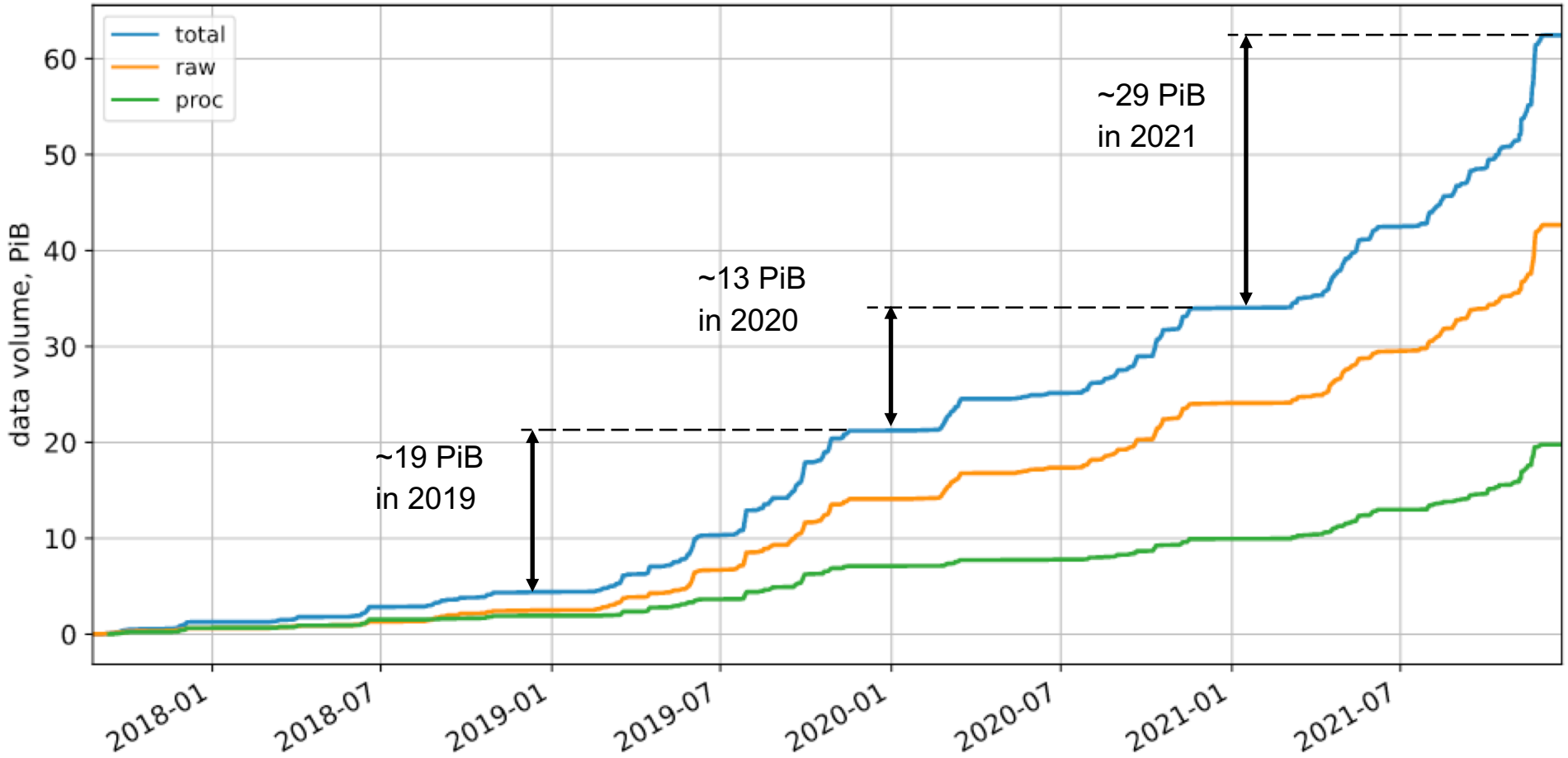
- AGIPD 4M is coming

Amount of data from big area detectors

data_type / aggregator	proc	raw	bytes total
AGIPD	15.9 PiB	33.0 PiB	48.9 PiB
DSSC	332.1 TiB	4.4 PiB	4.7 PiB
LPD	2.4 PiB	2.2 PiB	4.6 PiB
total	19.8 PiB	43.0 PiB	62.8 PiB

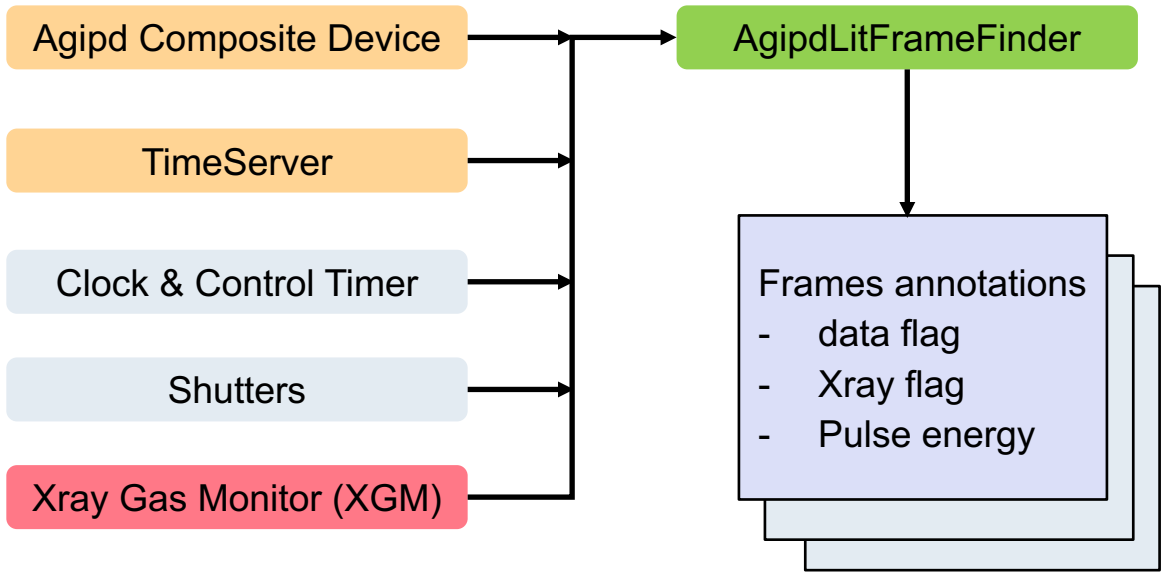
Portion in all data 93%

Total data generated by European XFEL




HOW'S THE BIG DATA PROJECT COMING ALONG, HOSKINS?
© D.Fletcher for CloudTweaks.com

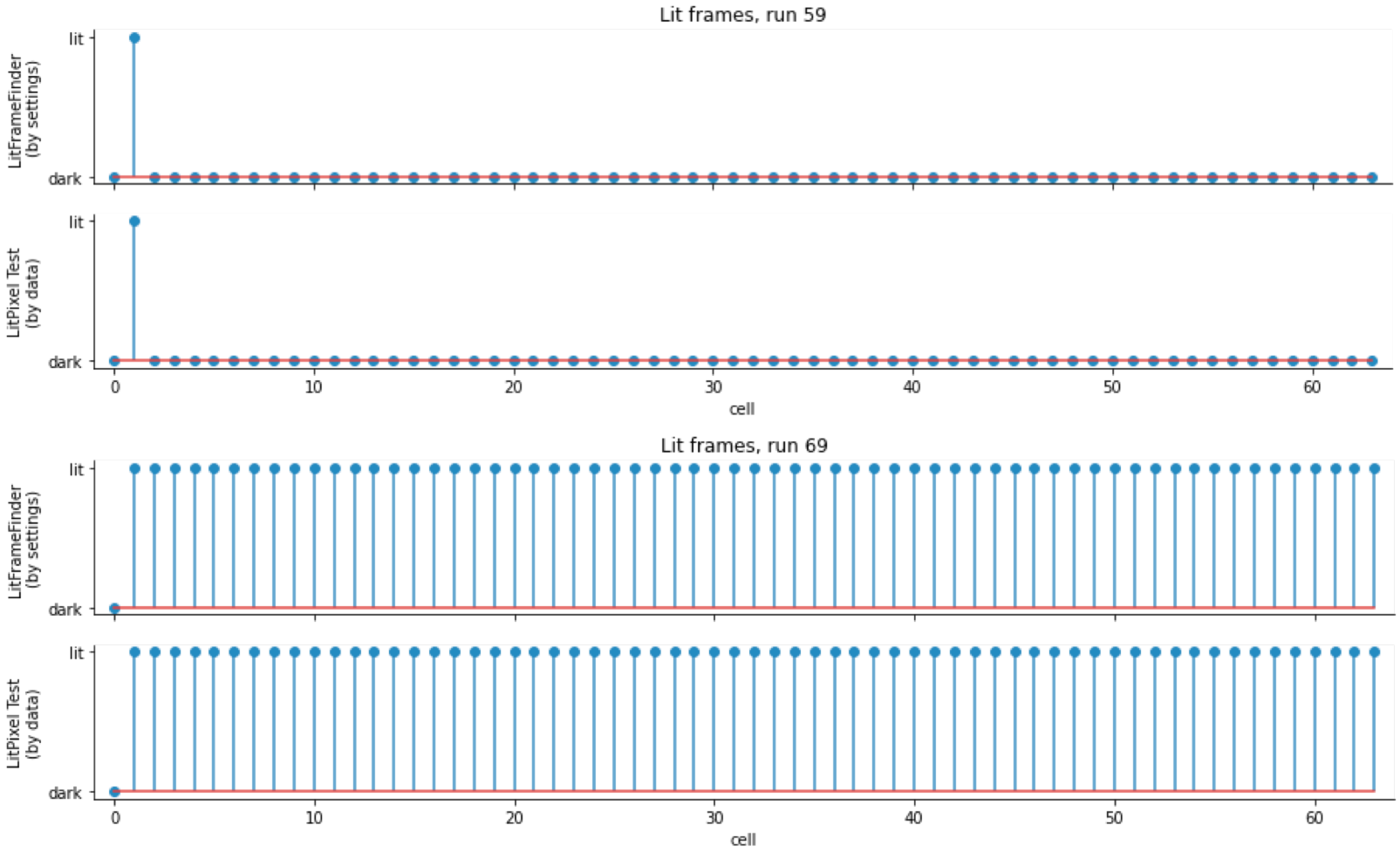
Dark frames annotation and deletion




```
RAW-R0053-AGIPD1MCTRL00-S00000.h5/CONTROL/MID_EXP_AGIPD1M1/REDU
└LITFRM (3 attributes)
  ├dataFrames
  │ ├timestamp  [uint64: 1150]
  │ └value      [uint16: 1150 × 352]
  ├litFrames
  │ ├timestamp  [uint64: 1150]
  │ └value      [uint16: 1150 × 352]
  ├nDataFrame
  │ ├timestamp  [uint64: 1150]
  │ └value      [uint16: 1150]
  ├nFrame
  │ ├timestamp  [uint64: 1150]
  │ └value      [uint16: 1150]
  ├nLitFrame
  │ ├timestamp  [uint64: 1150]
  │ └value      [uint16: 1150]
  ├nPulse
  │ ├timestamp  [uint64: 1150]
  │ └value      [uint16: 1150]
  └referenceFrame
    ├timestamp  [uint64: 1150]
    └value      [int16: 1150]

RAW-R0053-AGIPD1MCTRL00-S00000.h5/INSTRUMENT/MID_EXP_AGIPD1M1/REDU
└LITFRM:output
  └data (3 attributes)
    ├dataFramePattern   [uint8: 1150 × 352]
    ├detectorPulseId    [uint16: 1150 × 352]
    ├energyPerFrame     [float32: 1150 × 352]
    ├energySigma        [float32: 1150 × 352]
    ├masterPulseId      [uint16: 1150 × 2700]
    ├nFrame             [uint16: 1150]
    ├nPulsePerFrame     [uint16: 1150 × 352]
    ├trainId            [uint64: 1150]
    └xgmPulseId         [uint16: 1150 × 2700]
```

- Offline calibration pipeline may use this annotations for selection frames with signal

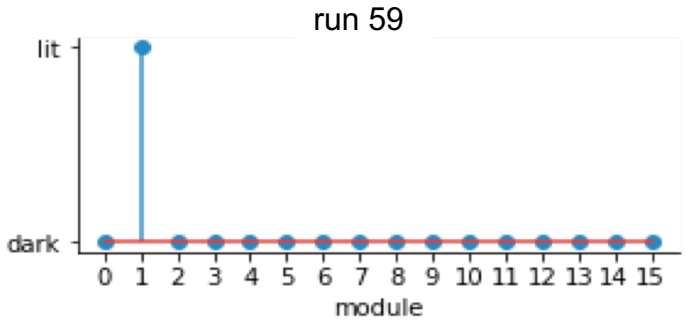
Dark frames annotation and deletion

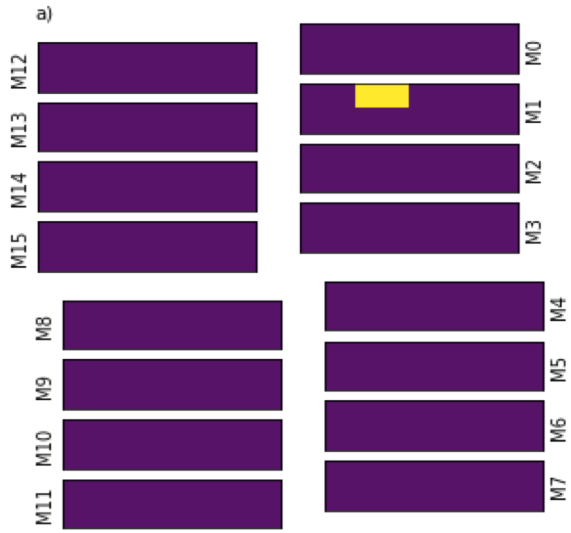


Signal localisation

- Methods:
  - lit-pixel counter binary classifier applied per ASICS base
  - Noise filtering and connected masks building

- Reduction:
  - Module files selection
  - ROI detection

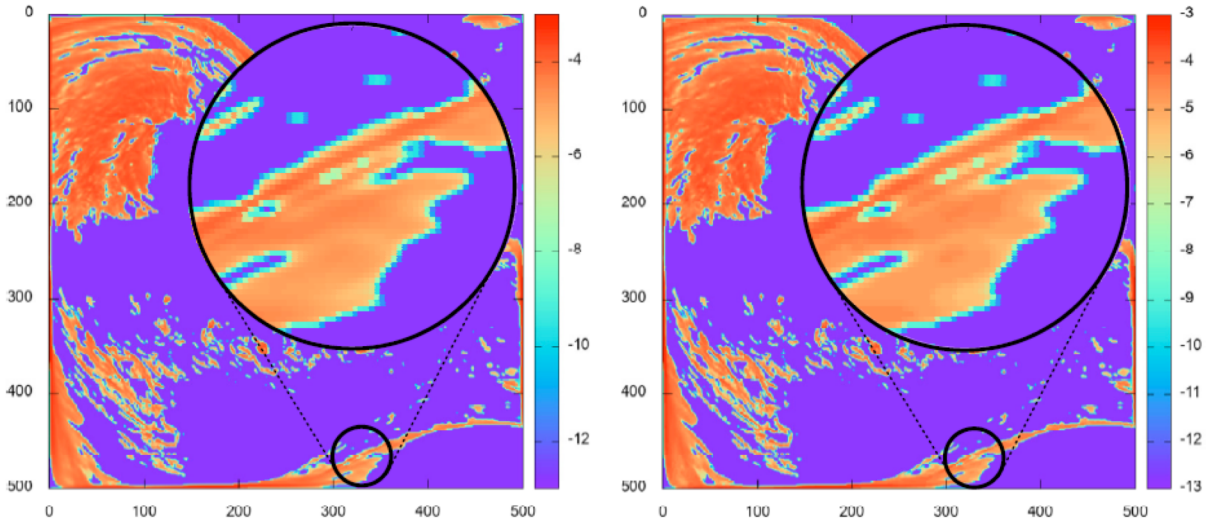




Bragg coherent diffraction imaging

Compression (project)

- Lossless compression
  - Space saving about 30%, too low
- SZ Lossy compression (https://szcompressor.org/)
  - Allows to balance between data quality and compression
  - Compression ratio and compression/decompression time are under investigation



(a) original raw data (b) SZ2.0 (PSNR=51, SSIM=0.997)

- Compression ratio 66:1

Writing in EuXFEL format

at the testing stage

Features:

- writing multi-train arrays at once
- writing data by trains
- check of data consistency
- buffering (direct writing is also an option)
- automatic sequence splitting
- aliases for source name
- parameterization of datasets and sources

```python
from extra_writer import FileWriter, DS

class AzFileWriter(FileWriter):
    gv = DS('@prm', 'geom.fragmentVectors', ('nmod', 'nfrag', 3, 3), float)
    gs = DS('@prm', 'geom.fragmentSize', ('nmod', 'nfrag', 2), int)
    nb = DS('@prm', 'param.numberOfBins', (), np.uint64)
    rlim = DS('@prm', 'param.radiusRange', (2,), float)

    n = DS('@res', 'azimuthal.radial', ('nbin',), float)
    v = DS('@res', 'azimuthal.intensity', ('nbin',), float)
    class Meta:
        aliases = {
            'prm': "{det_name}/x/y"
            'res': "{det_name}/x/y:output"
        }

def calc(det_name, nmod, nfrag, nbin):
    ...
    trains = range(10001, 10101)
    prm = dict(det_name=det_name, nmod=nmod, nfrag=nfrag, nbin=nbin)
    with AzFileWriter('filename-{seq:02d}.h5', **prm) as wr:
         wr.add_trains(trains[:99], [0] * 99)
         wr.add_data([3] * 99, v=np.random.rand(99 * 3, 1000))
         ...
```

Offline processing framework (project)

Frameworks hides under the hat typical batch processing operations:

- Multinode / multicore processing
- Serial async reading into two shared memory buffers
- Data/Event alignment
- Balancing of reading and computations

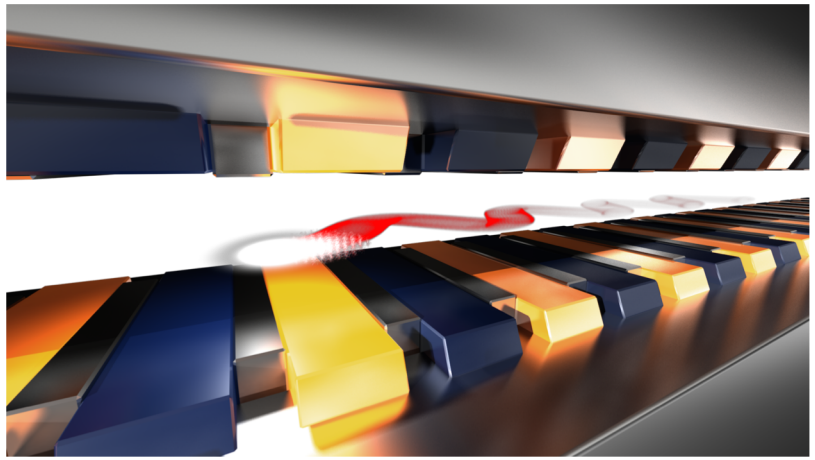
```
class BatchAzimuthalInt(AlgorithmBase):
    ALG_ID = "azint"
    HELP = "Azimuthal Integration"

    @classmethod
    def add_arguments(cls, parser):
      pass

    def configure(self, args):
      pass

    def initialize(self):
      return data_iterator

    def process_train_data(self, idx, train_no, train_id, first, img):
      return img.npulse

    def finalize_chunk_processing(self, chunk_no, idx):
      pass

    def finalize(self):
      pass
```

Experiment specific data reduction (future plans)

Experimental techniques	Reduction method	Ratio	Aggregation method	Ratio
Spectroscopy XES, XAS, etc	ROI, Integration	~10^3	frames averaging	10^2–10^3
Powder diffraction, SAXS/WAXS	Azimuthal integration	10^2–10^3	frames averaging	10^2–10^3
Correlation analysis XPCS, XCCA	Correlation function integration	10^2–10^3	frames averaging	10^2–10^3
SFX	Hit finding	10–100		
SPI/CDI	Hit finding	10^2–10^3		

Summary

- Dark data deletion
  - Dark frame annotation/deletion
  - Signal location
- Lossy compression (in nearest plans)
- Software
  - Library for writing data
  - Offline procession framework (in nearest plans)
- Experiment specific data processing (long term plans)



LCLS Linac Coherent Light Source

LEAPS INNOVATION

CFEL SCIENCE

Thank you for you attention. Questions?

Egor Sobolev, egor.sobolev@xfel.eu
Data Analysis Group, da@xfel.eu, www.xfel.eu/data_analysis
Support: da-support@xfel.eu