EDNA: On-line Data Analysis for imaging and powder diffraction

Example of DiffractionCT
Layout

• Presentation of EDNA
  • On-line data analysis framework
  • Create pipeline by chaining plugins
  • Provides many useful building blocks
• Presentation of the diffractionCT experiment
• What has been implemented
  • Geometrical transformation of spline files
  • Test the equivalence of execution plugins (SPD / FIT2D)
  • 2D powder diffraction caking plugins
  • Storage of hyperspectral data-sets in HDF5/Nexus
• Ongoing developments
  • NiNa, HDF5, NeXus, ...
Example of an existing ODA implementation: The MX Data Collection Pipeline

Future challenges for MX ODA in the Upgrade Programme: Large scale screening (MASSIF), multi-crystal data collection strategy etc.
Features of EDNA

- EDNA is a robust pipe-lining tool for on-line data analysis
  - It has been tested with thousands of tasks at once
- EDNA allows hi-performances
  - Multi-threaded implementation
- EDNA relies on data-models
  - Visual communication with scientists
  - Automatic bindings with the code
- EDNA has a strong testing framework
  - Unit & Execution tests
  - Non regression test before nightly builds
- EDNA is efficient to program
  - Plugin generator for execution plug-ins based on the data-model
  - Re-use of plug-ins already written (by others)
- EDNA is an international collaboration
EDNA is a versatile tool

• Run a plugin on the command line with a single input
  $ edna-plugin-launcher --execute EDPlugin --inputFileFile parameters.xml

• Run a plugin as a tango device server
  $ edna-tango-server

• Run a plugin in parallel on a set of inputs offline
  $ edna-SPD-Cake.py *.edf

• Run a plugin in parallel online on a set of incoming files
  $ edna-SPD-Cake.py --online
EDNA provides scientific building blocks

- **Execution plugins:**
  - MxExecPlugins: 24 exec plugins
  - Exec plugins: 14 exec plugins
    - SPD (4), FIT2D (2), EDF (2), HDF5 (2), thumbnail, video, ...

- **Control Plugins:**
  - MX v1: 26 control plugins, 7 exec plugins
  - MX v2: 3 control plugins
  - DiffractionCT v1: 6 control plugins

- **Other projects:**
  - CCP4 v0: 3 plugins, 1 control, 2 exec
  - Dimple v0: 20 plugins, 5 control, 15 exec
  - Darc v1: 3 plugins, 1 control, 2 exec

- **Total:** 64 Execution plugins, 42 control plugins
Creating a pipeline is already easy

• Define the datamodel
  • What are the input and the output of your plugins (as XSD file)

• Use the plugin generator to create the structure
  • Tests are also automatically generated

• Re-use execution plugins as building blocks

• Chain them by connecting output from the first → To the input of the second

It will even be easier with the forthcoming workflow tool
Example of workflow:
DiffractioCT (ID22)
Part of UPBL4: Nano-imaging / Nano-analysis

Experimental setup @ ID22:

- KB optics with μ-/nano-focused beam
- FReLoN 2k taper: diffraction data
- Fluorescence (single element) detector
- Ion-chamber ($I_0$ and attenuated I)

See e.g. Bleuet et al., Nature Mater. 2008

160 scanning steps/y-line,
60 angular steps/tomo scan

10-15h beam-time required/tomo scan
100 Go data generated/tomo scan
Towards combined contrast modes

• Absorption tomography:
  • record $I_0$ and attenuated $I$ for each scanned position (point)

• Fluorescence tomography:
  • record a fluorescence spectrum (1D) for each position (point)

• Diffraction tomography:
  • record a (powder-)diffraction pattern (2D) for each position (point)

• Computational challenge:
  • Coherent data storage (HDF5/Nexus)
  • Large amount of data (10 000 datasets: each 0D, 1D and 2D)
  • Problems for archiving, transfer (to the user's location)
Work done up to now ...
First DiffractionCT implementation (11/2009)
Second DiffractionCT implementation (06/2010)

EDParallelExecute
on/off-line

EDPluginControl
DiffractionCTv1_1

EDPluginControlDCT
ReadHeadersv1_0

EDPluginControlDCT
PowderIntegrationv1_0

EDPluginHDF5
MapOfSpectrav10

EDF
Filename

Metadata

EDF
Filename

Metadata

Dictionary

EDF
Filename

EDF
Filename

EDF
Metadata

EDF
Read
Headerv1_0

EDF
Read
Headerv1_0

Replace with Fabio

Read EDF

Write EDF

Nice

Python object
or XML data

Write HDF5
Nexus format

Read EDF
On-line data Analysis for NINA

- Focus on most CPU demanding application: DiffractionCT

FReLoN2k
(EDF, 16bit UInt, LittleEndian)

Integration of many tools
Need plug-in management
Plug-ins for azimuthal integration
FIT2D / SPD / Python

- They must all share the same data-model
  - Dark noise & flat field subtraction
  - Converts the distance to the centre in $2\Theta$
- Distortion on the captor using a spline
- Rotation of the detector
- Keep in memory the program
- Equivalence of the plug-ins...
- Need strong testing for non regression !!!!
Tilt as distortion of image ...

...That can be stored in a spline file

\[ \begin{aligned}
B' (x,y) \\
\Phi
\end{aligned} \]

\[ \begin{bmatrix}
x \cdot \cos \Phi \\
y \\
-x \cdot \sin \Phi
\end{bmatrix} \]

\[ \begin{bmatrix}
(\text{d'x} \cdot \cos \Phi)/(\text{d-x} \cdot \sin \Phi) \\
(\text{d'y})/(\text{d-x} \cdot \sin \Phi) \\
0
\end{bmatrix} \]

\[ d = \text{Distance sample to detector} \]

Fit2D definition not SPD

\( \Theta \)

Beam

Sample

Orthogonal detector

Tilted Detector

0 0 0

0 0 d

x (to the center of the ring)

y

z (to the source)
Testing & Profiling of EDNA

- **Plug-ins in EDNA**
  - Each plug-in should be usable alone (use of data-model)
  - Unit tests for each module with test suites (non regression tests)
  - Exchangeable plug-ins by sharing data-models (FIT2D / SPD)

- **Profiling in EDNA**
  - Global profiling using Python Profile
  - Performances evaluation using timings

- **Performances (2k x 2k images on dual-Xeon 5500)**
  - **Fit2D:**
    - < 5 sec / image with plug-in fit2dv1.1
    - Full sinogram at 1 point every 500ms
  - **SPD:**
    - < 2 sec / image with plug-in spdCakev1.5
    - multithreaded
Towards 3D and 4D data storage

• Use HDF5 for data storage
  • Fewer files (currently hundred of thousands)
  • PyMCA can already read, visualize such data-sets
  • Need to get used to Nexus nomenclature
  • Compression is for free

• Store one experiments per HDF5 group / NXMeasurement
  • Absorption, Diffraction, Fluorescence …
  • Need to develop NeXus application definitions

• Problem of order for the data stored:
  • Stack of Images: [Index, y, x]
  • Map of Spectra: [SlowMotor, FastMotor, Spectrum]

• Live visualization of ROI with PyMca
  • No more ROI to be defined in the data-model!
Visualization of HDF5 files for DCT

- Images stored as a stack: Visualization with PyMca
- Live ROI selection for the sinograms with PyMca
- HDFview
Work currently in progress …
And long term milestones
Work in progress for NiNa

- Clean up plugins
  - HDF5 Map of Spectra is not (yet) NeXus compliant
  - SPD Cake is locked to single threaded
- New plugin for HDF5
  - HDF5 map of Images (4D data set !)
  - Extract data for HDF5
    - images to EDF or single image HDF5
    - spectra to spec or xy files
- Full Virtual Experiment
  - Full testing including the spec part, tango, ...
  - Needs especially a virtual camera...
TODO for Nina

• Other contrast modes:
  • NeXus application definitions do not exist at the moment
  • Datamodel to be defined as well

• Aggregation of data inside a single container
  • HDF5/Nexus file format (?)
  • Synchronisation issues
  • Rewrite the EDF-plug-ins in (HDF5)-plug-ins
    • keeping the same data models for all equivalent plug-ins

• On-line tomographic reconstruction with PyHST

• Absorption correction for diffraction data
Work in progress for EDNA

- Clean up in the source repository
  - Make it compatible with Windows (MacOSX is OK)
  - Refactor to new coding convention which are lighter

- Interaction with scientists
  - EDNA plugins should be easily hackable

- Interaction with the beamline
  - Feed back to Spec (tango server, SpecClient? ...)
  - Use a data manager and stop transferring data through files on the disk

- Graphical feed-back of the processing
  - Matthew will work on a GUI for EDNA
  - Integration with a pipelining tool like Passerelle or Knime
Challenges for On-line data analysis

• Performances
  • Time constrained environment
  • Not slow down of the experiment (OK checked on ID22)
  • Use a data-manager (in the future)

• Development with and for scientists
  • Data-model for communication
  • Application definition for files (shared with other analysis software)

• Maintainability
  • Coding conventions
  • Testing framework
  • Shared SVN repository (www.edna-site.org)

• Sustainability and long term support
  • International and multi-site collaboration
  • Should be easy to install for external users (→ packaging)
Acknowledgement

EDNA team
- Olof Svensson
- Alun Ashton (DLS)

Scientists:
- Jonathan Wright (ID11)
- Manfred Burghammer (ID13)
- Peter Boesecke (SPD)

Data Analysis Unit
- V. Armando Solé (PyMca)
- Rainer Wilcke (SPD)
- Alessandro Mirone (PyHST)

Beamline Control Unit
- Cyril Guillou (ID22)