IOP Conf. Series: Journal of Physics: Conf. Series **849** (2017) 012059 doi:10.1088/1742-6596/849/1/012059

dada – a web-based 2D detector analysis tool

Markus Osterhoff

Institute for X-ray physics, University of Göttingen, Friedrich-Hund-Platz 1, D–37077 Göttingen

E-mail: mosterh1@gwdg.de

Abstract. The *data daemon*, *dada*, is a server backend for unified access to 2D pixel detector image data stored with different detectors, file formats and saved with varying naming conventions and folder structures across instruments. Furthermore, *dada* implements basic pre-processing and analysis routines from pixel binning over azimuthal integration to raster scan processing. Common user interactions with *dada* are by a web frontend, but all parameters for an analysis are encoded into a Uniform Resource Identifier (URI) which can also be written by hand or scripts for batch processing.

1. Introduction

Basic analysis of modern (2D) pixel detector images is hindered by seemingly trivial technical problems: varying file names and different formats between beamlines, detectors, and sometimes these change over time. In fact, just reading a detector image as a university's first-time bachelor student can be rather difficult, although it should be a standard task. Then, the correct image orientation (flipping, rotating) has to be found, before basic pre-processing like binning and region of interest (ROI), and empty division can be applied. As a small university group with fluctuating students taking data at different end-stations, the sheer number of file name formats, folder structures, and detector formats can be quite overwhelming when analysis scripts are adapted from older beamtimes.

Comparing *dada* to other common software: FabIO [4] is a Python library for 2D detector images; here we add also file name unification and basic analysis. Dawn [6] has emerged as a very powerful Eclipse based analysis program that features an impressive set of tools for crystallography and fluorescence mapping. Here we focus on holographic imaging and scanning applications. Another famous program is of course ImageJ [5], which has a large user base not only in the optical microscopy community. Oposed to these programmes, *dada* can be run both on a local PC/laptop, on a dedicated analysis machine, and on small clusters. Long-running jobs continue when the interface is closed.

On the other hand, certain pre-processing steps are common to different experiments; more advanced analysis methods that have been investigated during students' thesis works are now understood and need to be frozen in a specific version, with a minimum of interaction (i.e. change of parameters). With our *data daemon* – or *dada* for short – we are solving these two problems in one modularised programme, which is entirely written in C. In the following section 2, we describe more detailed the steps of file name translation, file interpretation, image orientation and basic pre-processing; in section 3 some analysis routines are described and illustrated with

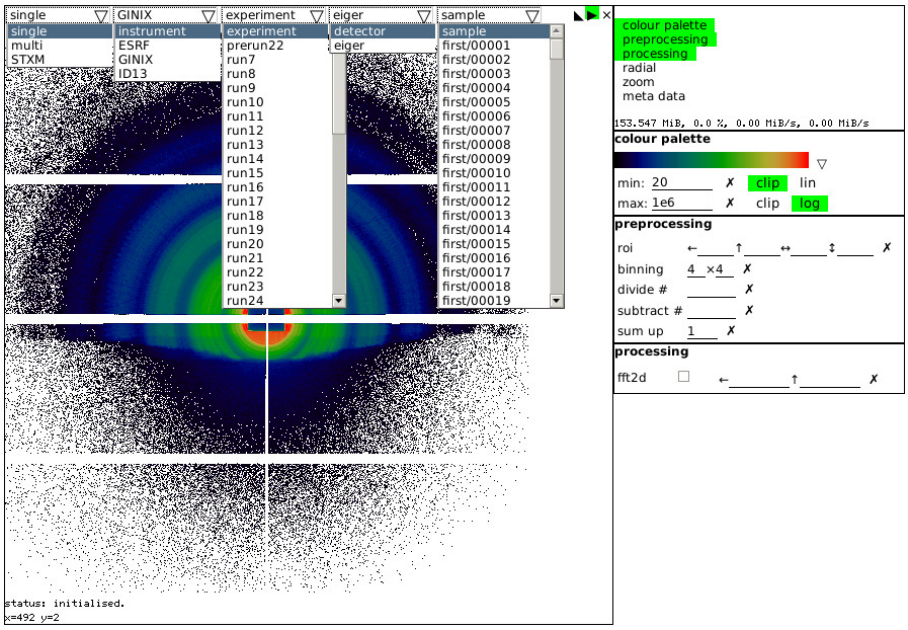

Figure 1. Screenshot of *dada*'s web frontend. In the top row, the selection menus for instrument, experiment, detector and sample are shown; on the right panel, the colour palette can be defined, along with certain pre-processing steps.

exemplary data. Section 4 briefly describes the web-frontend (see Fig. 1) that the user interacts with. We conclude with a summary and outlook in section 5.

2. Image read-in, pre-processing

A single 2D-detector image is defined by the names of instrument (beamline), experiment, sample and detector, and the image number. The "messy" translation into the correct file name is then carried out in the backend and can easily be adapted for new setups. The image data is then read in, dealing with decompression together with interpretation, correcting the image orientation, and applying specific bad pixel masks. These steps are coded in three functions (*filename.c*, *open,c*, *pixelmask.c*); generic versions apply for one detector family, but the subtle differences between experiments are accounted for with specific versions of the routines in instrument/experiment specific subfolders.

Pre-processing starts with 2D region-of-interest (ROI) and binning (pixel decimation) and can include division by and subtraction of (empty) images, and stack operations like adding up sequential images, or finding the maximum value of each image pixel of the stack. These operations are carried out in the *show_single_image* module.

For quick visual inspection, the image data can then be colour-coded into a "common" image file (PNG, or PDF with basic annotations) and delivered to the user, e.g. via web browser. For the colour code, linear and logarithmic scaling and overriding of min/max is possible; also different colour palettes can be chosen. The final image files are cached by the backend for faster access in the future.

A "live version" of *dada* can be used at the beamline, where the latest detector image is shown on the screen, with feedback of basic statistics (total counts, min/max) and the possibility to directly print the image for the analogue lab book.

A 2D array lay-out of raster scanned images is possible using the module *show_composite_image*: it places N_y by N_x of pre-processed single images in a regular grid;

with reasonable ROI and binning, a large 2D raster scan of 2D image data can then be assessed visually. This method can also be used to visualise 1D scans, see an example in the following section.

All detector routines and the *show_** modules are loaded as dynamic libraries. This makes it possible to adjust the programme during execution: detectors can be added at run-time, processing can be adapted and developed without the need to re-start *dada*.

So far, we have implemented data read-in routines for Pilatus detectors (cbf[1] and tiff, also cbf.gzip), Maxipix and other EDF-detectors, certain CMOS detectors (tiff), and recently Eiger and Lambda (HDF5[2]).

Instead of rendering a colour-encoded visual image file, the pre-processed data can be requested as an array of *doubles*[3]; this allows for a unified access of detector images across file name conventions and detector formats from third-party software.

3. Routine Analysis Methods

The most basic analysis for single images is of course min/max and total intensity (sum over all pixel values) after pre-processing. In the "live version" for the beamline, these values can be used as feed-back to the control software SPEC[4] via network, so quantitative scanning on 2D detector ROIs is possible with a typical overhead of for example 15 ms for a Pilatus 300k detector. A quick azimuthal integration for radial beam intensity profiles with user-defined beam centre is possible, and intensity between q-ranges (given as circular discs, in pixel units) is reported. Also the anisotropy of scattering signals is quantified using the principal component method developed in [1].

The aforementioned statistics can then be used in the STXM module (scanning transmission x-ray microscopy) as the observables on a regular 2D grid to analyse raster scans. Thus, a 2D colour-coded image is formed based on e.g. the total intensity of detector images (after ROI, binning, and further pre-processing as described above) for a 2D scan of the sample. Other observables are dark-field (scattering outside a specified region), centre-of-mass in horizontal and vertical direction (to quantify e.g. differential phase contrast); intensities in circular discs; anisotropy and orientation from the principal component analysis to investigate ordered structures in the sample.

4. Web Interface

The *dada* backend uses an *HTTP* server[5] as its primary interface. Thus, all requests (single image as raw data or colour-coded visual image, STXM analysis, meta data) are made via a Uniform Resource Identifier (URI), e.g. of the form

http://dada-demo.roentgen.physik.uni-goettingen.de/demo/stxm/GINIX/run42/p300/cm3/1?horz=201&vert=201&signal=omeg&palette=wb&scale=lin&cmin=0.0&cmax=0.7&radial.pb.x=232&radial.pb.y=314&radial.r1=72&radial.r2=93&radial.r3=200

which requests a STXM analysis of orientations in a 201 × 201 raster scan with user-defined beam centre and colour palette.

A screenshot of the web frontend showing the selection menus for instrument, experiment, detector and sample is shown in Fig. 1.

[1] See, e.g., `https://www.dectris.com/technical_pilatus.html?file=tl_files/root/support/technical_notes/pilatus/Pilatus_CBF_Header_Specification.pdf` or `http://www.esrf.eu/computing/Forum/imgCIF/cbf_definition.html` for definition / examples.

[2] C.f. `https://support.hdfgroup.org/HDF5/`.

[3] So far, this is just a plain memory dump; but more accessible formats are foreseen, e.g. an HDF5 export.

[4] SPEC, by Certified Scientific Software.

[5] The GNU libmicrohttpd library, see `https://www.gnu.org/software/libmicrohttpd/`.

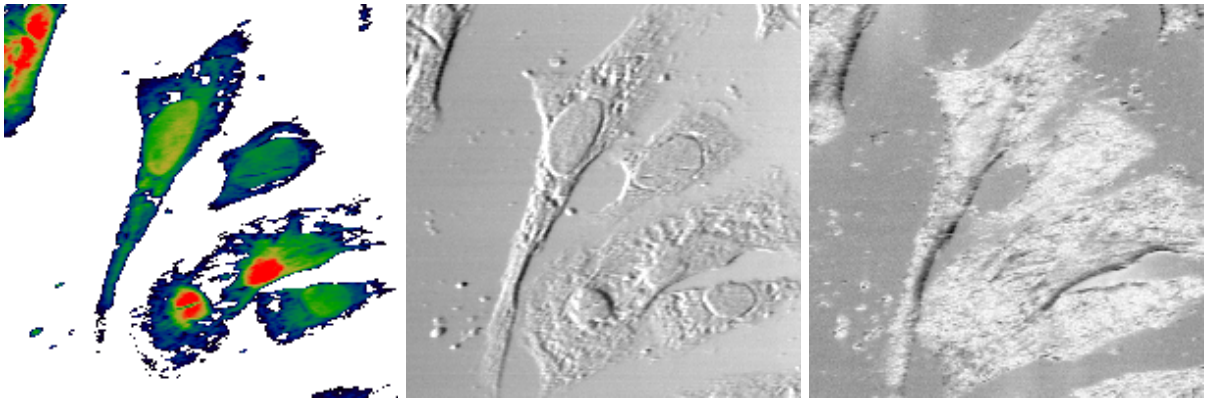

Figure 2. Three different observables calculated for a 2D STXM taken with a Pilatus detector show total scattered intensity (darkfield), centre-of-mass (comparable to differential phase contrast) and the order parameter ω [1, 2] to visualise actin fibre bundles within a cardiomyocyte cell.

In principle, all analyses are requested from URIs in this form; in fact, this allows for a batch analysis by scripts generating such URIs. For most uses, however, a web interface based on jQuery is available that builds up the URI from the user's choices. We stress that ongoing long-running jobs continue after the user has closed the web frontend, and the results can be accessed later on; the web frontend is reachable world-wide (authentication and encryption are possible separately), while the computations are carried out on dedicated machines.

In addition, also an image stack can be viewed in sequence, and individual images of a STXM scan are accessed by clicking the pixel. For radial analysis (azimuthal integration, intensity inside q discs, anisotropy), the beam centre is be defined via mouse clicks, and an overlay of circles in q-space can be enabled. Currently, everything works on a pixel-basis with no sub-pixel interpolation; also the beam centre is defined by the user in full pixels (after binning). Refinements might be added in future.

5. Summary and Outlook

In summary, the *data daemon* (*dada*) allows for a unified access to x-ray pixel detector data, that (i) acts as a central backend hiding technical trivial, but tedious problems from the user, and (ii) implements standardised pre-processing and analysis algorithms. The web frontend helps users in building the request URI that fully determines the job, but for batch analysis this URI may also be generated from users' scripts. Long running jobs continue even if the web frontend is closed, and are shown later on.

Currently, various detectors like Pilatus and Eiger, Maxipix and Lambda, and certain TIFF-writing sCMOS cameras are supported; file name generation is implemented for several ESRF beamlines, the cSAXS at SLS, and the GINIX end-station at PETRA III. The list will certainly grow and be updated.

Also, further analysis routines such as the automized detection of diffracton streaks typical for fibre scattering (streak-finder algorithm, [3]), Fresnel propagation and (holographic) phase retrieval methods, as well as rudimentary tomographic functions (sinogram generation, filtered back projection) will be added as analysis modules.

The *dada* will be published to the user community in 2017 on an open license.

Acknowledgements
We acknowledge funding by Deutsche Forschungsgemeinschaft through SFB 755, INF project, and thank Tim Salditt for his continuous support and feedback.

References

[1] M. Bernhardt, M. Priebe, M. Osterhoff, C. Wollnik, A. Diaz, T. Salditt, and F. Rehfeldt, "X-Ray Micro- and Nanodiffraction Imaging on Human Mesenchymal Stem Cells and Differentiated Cells", Biophys. J. 110, 680–690 (2016).
[2] M. Bernhardt, J.-D. Nicolas, M. Eckermann, B. Eltzner, F. Rehfeldt, T. Salditt: "Anisotropic X-Ray Scattering and Orientation Fields in Cardiac Tissue Cells" (in review)
[3] M. Priebe, M. Bernhardt, C. Blum, M. Tarantola, E. Bodenschatz, and T. Salditt, "Scanning X-Ray Nanodiffraction on Dictyostelium discoideum", Biophys. J. 107, 2662–2673 (2014).
[4] E.B. Knudsen, H.O. Sørensen, J.P. Wright, G. Goret, J. Kieffer: "FabIO: easy access to two-dimensional X-ray detector images in Python", Journal of Applied Crystallography 46 (2013).
[5] Rasband, W.S., ImageJ, U. S. National Institutes of Health, Bethesda, Maryland, USA, `http://imagej.nih.gov/ij/`, 1997-2016.
[6] M. Basham, J. Filik, M.T. Wharmby, P.C.Y. Chang, B. El Kassaby, M. Gerring, J. Aishima, K. Levik, B.C.A. Pulford, I. Sikharulidze, et al., J. Synchrotron Rad. 22 (2015).