Data Processing for Time Correlation

Unknown macro: 'html'

Unknown macro: 'html'

Motivation

Development of this application was stimulated by the discussion with Marcin Sikorski (meeting on 2012-08-30), doing xcs experiments.
Users need in real-time algorithm for calculation of image vs time auto-correlation function

```
g2(tau) = <I(t)*I(t+tau)> / (<I(t)> * <I(t+tau)>),
```

where `I(t)` is an image intensity at time `t`, and `tau` is a delay between two measurements.
Typical experimental condition can be described as follows:

- Run duration is about one hour at frequency up to 120 Hz that gives up to 10^5-10^6 images.
- Currently typical imaging devise is a Princeton camera with 1300x1340 pixels.
- Need to calculate `g2(tau)` for each pixel, averaged over all possible image times `t` with time difference `tau` between images.
- A set of `tau` should have about 30-100 points in log scale uniformly covering the run duration.
- Use for example xcsi0112-r0015: 500 images with 8 sec delay between images.
  Desired time for evaluation of the auto-correlation function should be comparable with run duration <1 hour. Currently this algorithm takes a few hours that can not be used for fast feedback in real time experiment.

In first approximation this problem was solved, as it is described in the Command Line Interface For Time Correlation Analysis. However, in order to be useful, the command line algorithm needs to be integrated in the global analysis system, which *de bene esse* called as **Integrated Analysis Environment for Time Correlation Experiments** and is discussed in IDPE for TCE in XCS - Problems and tentative solutions.

GUI Implementation

Updated on 2013-04-01

As an example of GUI style Marcin suggested to use XPCSGUI - earlier implementation of similar application.

Three versions of GUI were implemented for this application. It was decided to use the `GUIMainTB`
layout, where:

- All input windows are integrated in one with tab-bar for switching and custom tool bar on the top
- Logger and File Browser windows are isolated
- All plots are isolated, although similar plots appear in the same windows.

h3 Plots for intensity in q-static bins

Intensity monitors

FEEGasDetEnergy, XCS-IPM-02, XCS-IPM-mono, XcsBeamline.1:Ipimb.4, and XcsBeamline.1:Ipimb.5

Reflective scattering geometry

Source code from Marcin: q_functions.txt

Code location

Everything resides in `CorAna` package, which is included in LCLS offline releases beginning from `ana-0.7.16`.
References to the specific algorithms can be found in Algorithms for Time Correlation Experiments.
Original data processing code from Marcin is described in Note on XCS code from Marcin.

Modules

In tables below we list modules from packages `CorAna` and `ImgAlgos`, which were developed for this project.
Notations:
✅ - is done (at least it is assumed for now...)
➕ - needs more work
ℹ️ - test or deprecated modules

Package CorAna

Updated on 2013-10-14

All module names beginning with letters `GUI` implements different Graphical User Interfaces.

List of modules in alphabetic order:

Module	Description
✅ AppDataPath.py	Local version with added path to data from src directory.
✅ BatchJob.py	Suprclass for other `BatchJob*.py` modules
✅ BatchJobCorAna.py	
✅ BatchJobData.py	

BatchJobPedestals.py	Class contains methods for batch job submission and monitoring for pedestals.
BatchLogParser.py	
ConfigFileGenerator.py	Class has methods to generate psana configuration and other scripts from stubs located in `CorAna/data/scripts/`.
ConfigParameters.py	Base class for configuration parameters.
ConfigParametersCorAna.py	Sub-class for `CorAna` specific configuration parameters.
CorAnaUtils.py	Junk for eamples.
Drag.py	
DragCenter.py	
DragCircle.py	
DragLine.py	
DragObjectSet.py	
DragPolygon.py	
DragRectangle.py	
DragWedge.py	
EventTimeRecords.py	
FileNameManager.py	Class dynamically generates all file names for current configuration parameters.
GUIAnaPartitions.py	
GUIAnaSettings.py	
GUIAnaSettingsLeft.py	
GUIAnaSettingsOptions.py	
GUIAnaSettingsRight.py	
GUIBlamish.py	Sub-GUI of GUIFiles.py
GUICCDSettings.py	
GUIConfigParameters.py	GUI for the configuration parameters file management.
GUIDark.py	Sub-GUI of GUIFiles.py - manipulations with dark runs
GUIData.py	
GUIELogPostingDialog.py	Dialog window for submission of messages with attachments to ELog using response ID and Igor's new interface: Python module for posting message into ELog
GUIELogPostingFields	Widget with necessary fields for GUIELogPostingDialog
GUIFileBrowser.py	Text file browser for this project
GUIFiles.py	Central GUI for file settings contains tab-bar for other widget selection
GUIFlatField.py	Sub-GUI of GUIFiles.py
GUIHelp.py	WIdget for messages, i.e. help

GUIImgSizePosition.py	
GUIInstrExpRun.py	depricated GUI
GUIIntensityMonitors.py	Control GUI for intensity monitors
GUIKineticMode.py	
GUIListOfTau.py	GUI manipulation with list of tau indexes
GUILogger.py	GUI for logger
GUIMain.py	The first GUI at start of this application
GUIMainSplit.py	The same as GUIMainTB.py with integrated GUILogger.py
GUIMainTB.py	The first GUI at start of this application with tab bar
GUINonKineticMode.py	
GUIRun.py	
GUIRunAuto.py	
GUIRunInput.py	
GUIRunMerge.py	
GUIRunProc.py	
GUIRunSplit.py	
GUISetupBeamZero.py	
GUISetupData.py	
GUISetupEnergyAngle.py	
GUISetupInfo.py	
GUISetupInfoLeft.py	
GUISetupInfoRight.py	
GUISetupPars.py	
GUISetupSpecular.py	
GUISystemSettings.py	
GUISystemSettingsLeft.py	
GUISystemSettingsRight.py	
GUIViewControl.py	
GUIViewResults.py	
GUIWorkResDirs.py	
GlobalExternal.py	
GlobalUtils.py	Module contains all possible global methods.
ImgSpeNavToolBar.py	Re-implemented standard tool-bar (depricated)

Logger.py	Core class for logger
MaskEditor.py	
MaskEditorButtons.py	
Overlay.py	Demonstration of how to draw something on the top of GUI
PlotArray.py	Main class for array presentation plot
PlotArrayButtons.py	Widget for custom button-bar
PlotArrayWidget.py	Widget for graphical window
PlotG2.py	
PlotG2Buttons.py	
PlotG2Widget.py	
PlotGraph.py	
PlotGraphWidget.py	
PlotImgSpe.py	Main class for interactive plot with custom button bar
PlotImgSpeButtons.py	Widget for custom button-bar
PlotImgSpeWidget.py	Graphical image for image and spectral histogram
PlotTime.py	Main class for time record presentation plot
PlotTimeWidget.py	Widget for graphical window
RecordsFromFiles.py	
ThreadWorker.py	
ViewResults.py	
data/scripts/psana-*.cfg	Stub-scripts for ConfigFileGenerator.py

Modules used from package ImgAlgos

Updated on 2013-10-14

List of modules in alphabetic order:

Module	Description
CorAna	Superclass for all CorAna* modules
CorAnaData	Processing of split images, evaluation of correlators
CorAnaInputParameters	Input parameters parser
CorAnaMergeFiles	Merges split files with correlatros in a single file
CorAnaPars.py	Holds common parameters
CorAnaProcResults	Example of stand-alone processing of the file with correlators
CorAnaSubmit.py	Command-line submission procedure for processing
GlobalMethods	Common global methods for `ImgAlgos` package
ImgAverage	Generic `psana` module for image average
ImgCalib	Generic `psana` module for image calibration
ImgIntForBins	Generic `psana` module for intensity(averaged over bin pixels) in bins per event, bin numbers are defined by the map

ImgIntMonCorr	Generic psana module for image normalization on intensity monitors' data
ImgMaskEvaluation	Generic psana module for saturation and noisy pixel mask evaluation from data
ImgTimeStampList	Generic psana module produces the file with inage time stamps and indexes
ImgVsTimeSplitInFiles	Generic psana module accumulates split image for all events in files
IntensityMonitorsData	Generic psana module produces the file with intensity monitor data for all events
PrincetonImageProducer	Generic psana module gets the Princeton camera image and save it as a ndarray<uint16_t, 2> image in the event
Tahometer	Generic psana module for performance report

Functional description

Updated on 2013-10-14
In this section modules are listed in functional order.

PSANA modules

Dark run pre-processing

Scaner
Runs in batch for the dark file to get preliminary information.
✅ `ImgAlgos.ImgTimeStampList`

- counts number of events in the file
- makes file with time stamps
- evaluate time intervals between frames (for dark run)

Pedestals
Runs in batch for the dark file to get averaged pedestals.
✅ `ImgAlgos.PrincetonImageProducer` gets image from event as an `ndarray` object
✅ `ImgAlgos.Tahometer` evaluates performance of the batch job
✅ `ImgAlgos.ImgAverage` produces files with averaged, rms-spread, and hot-pixel mask for images in a given range of events.

Data pre-processing

Scaner
Runs in batch for the data file to get preliminary information.
✅ `ImgAlgos.Tahometer` evaluates performance of the batch job, counts number of events in the data file
✅ `ImgAlgos.ImgTimeStampList` makes file with time stamps and time record counters for tau
✅ `ImgAlgos.IntensityMonitorsData` makes file with intensity monitor records

Average
Runs in batch for the data file to get averaged image
✅ `ImgAlgos.Tahometer` evaluates performance of the batch job
✅ `ImgAlgos.PrincetonImageProducer` gets image from event as an `ndarray` object
✅ `ImgAlgos.ImgAverage` produces file with averaged and rms-spread for raw images in a given range of events
✅ `ImgAlgos::ImgMaskEvaluation` - module is configured to evaluate masks:

- saturated mask - pixel is considered as saturated if its amplitude exceeds the threshold at least once per run
- noise mask - pixel is considered as noisy if its amplitude exceeds the MEAN+5*RMS in 5% of events. The "noise" MEAN and RMS are evaluated for 8 (or less on the boarder) surrounding pixels. This mask is not used in current analysis.

Data processing

Split
✅ `ImgAlgos.Tahometer` for performance evaluation.
✅ `ImgAlgos.PrincetonImageProducer`
✅ `ImgAlgos::ImgCalib` module is configured to:

- subtract pedestals obtained for dark run
- account for threshold (LLD) constant or in number of RMS

✅ `ImgAlgos.ImgIntMonCorr` image normalization on intensity monitors data
✅ `ImgAlgos.ImgVsTimeSplitInFiles` split image and save blocks for all events in separate files.
✅ `ImgAlgos.ImgIntForBins` evaluate intensity (averaged over bin pixels) in bins per event, bin numbers are defined by the map, results are saved in the file.
✅ `ImgAlgos.ImgAverage` produces file with averaged and rms-spread for images with subtracted pedestals and allied LLD in a given range of events

Process
Data processing is implemented in stand alone (non-psana) modules
✅ `ImgAlgos.CorAna.cpp`
✅ `ImgAlgos.CorAnaInputParameters.cpp`
✅ `ImgAlgos.CorAnaData.cpp`

Merge
✅ `ImgAlgos.CorAna.cpp`
✅ `ImgAlgos.CorAnaInputParameters.cpp`
✅ `CorAnaMergeFiles.cpp` - saves binary file for float(32) with shape (Ntau,3,rows,cols), where 3 stands for <Ip>, <If>, and <Ip*If>
Can be acessed in python as

```
    sp.cor_arr = np.fromfile(sp.fname, dtype=np.float32)
    <image-size> = rows * cols
    nptau = <file-size>/<image-size>/3
    sp.cor_arr.shape = (nptau, 3, rows, cols)
```

GUI

The system of GUIs, consisting of dozens of `CorAna.GUI...` modules is implemented in the draft approximation. Roughly it reproduces all features of the old program.

Input parameters GUIs

✅➕ `CorAna.GUI...` most of them are available. Will be added or extended if necessary.
✅ Files - define input files and do pre-processing
✅ Setup Info
✅ Analysis Info
✅ System
✅ Intensity Monitor

Run GUI

✅ Input - short summary of input info for data processing
✅ Split - control and monitoring for the 1st stage of processing
✅ Process - ... 2nd stage ...
✅ Merge - ... 3d stage ...
✅ Auto - ... for all 3 stages ...

View Results GUI

Contains a set of control fields for presentation of results
✅ Direct and reflected beam geometry is implemented in `ViewResults.py`. Currently the switch between the direct and reflected beam geometry is used from tab status variable: `cp.exp_setup_geom.value()`.
➕ If the PV variable will be used, then the switch should be changed in `ViewResults.get_q_map(sp)` module.

Graphics

✅ `PlotArray*.py` - for intensity monitors
✅ `PlotImgSpe*.py` - for images, partition maps, masks etc.
✅ `PlotTime*.py` - for time stamp monitoring
✅ `PlotG2*.py` - G2 plot
✅ `PlotGraph*.py` - I(q) and I(q,t) plot
✅ `MaskEditor*.py`, `Drag*.py` - Mask Editor for

- region of interest (ROI-mask)
- blemish mask
  ✅ `ViewResults.pyQ` - all evaluations for resulting array of correlators
- maps for pixel x,y coordinates, r, phi
- q maps for transmission and reflective geometry
- etc.

Infrastructural modules

Infrastructural modules provide basic infrastructure of the project.
✅ `ConfigParameters.py`, `ConfigParametersCorAna.py`, and `GUIConfigParameters.py` provides convenient approach for maintenance of all configuration parameters.
✅ Infrastructural modules `Logger.py` and `GUILogger.py` provides a generic approach to logging system.
✅ Module `ConfigFileGenerator.py` use current settings of configuration parameters and stub-file scripts from `CorAna/data/scripts/` and generates the psana configuration files.
✅➕ Module `FileNameManager.py` is a single place which provides a dynamic file names for current version of the configuration parameters.
✅ `GlobalUtils.py` - global utilities for common operations.
✅ `BatchJob.py` - superclass for batch job submission.
✅ `BatchJobPedestals.py` - pre-processing for dark run files.
✅ `BatchJobData.py` - re-processing for data files.
✅ `BatchJobCorAna.py` - main data processing - calculation algorithm.
✅ `RecordsFromFiles.py` - class helps to access data in files.

Status of the project

- This project is in the stage of comparison of results between old Marcin's scripts and this application.

Comparison of results

Conditions

1. dark run: `/reg/d/ana12/xcs/xcsi0112/xtc/e167-r0020-s00-c00.xtc` use all 75 events
2. data run: `/reg/d/ana12/xcs/xcsi0112/xtc/e167-r0015-s00-c00.xtc` use all 500 events
3. do not use any intensity monitor selection or correction
4. do not use any mask including hot pixel, saturation, blemish, ROI, and restriction on the image size.
5. use LLD as a constant ADU threshold = 20
6. use a single q-phi static bin
7. use a single q-phi dynamic bin
8. q value is not an issue for current comparison, so geometry does not matter.

Compare g2 at 14 tau values with indexes:

```
1 2 3 4 5 6 7 8 9 10 12 14 16 20
```

See Progress section for more details on comparison.

Progress

Weekly progress of the IDPE project for TCE

To-do list

Processing (at psana Split level)
⛔ Get `cp.photon_energy`, `cp.nominal_angle` and other PV variables from data scan and setup confiruration

Comparison of results

- Search for the reason of difference in results for single-q-bin g2.

View Results
more plots for results
✅➕ g2(tau) for q-dynamic
✅➕ Average intensity and Intensity(q-static)
✅➕ Intensity(q-static, t)
✅➕ Histogram for intensity monitor
⛔ Fit results for function g2(tau|pars) = C*exp[-(2t/tau0)**beta] + B

- plot for all fits with beta=1 and float,
- tau0 vs q-dynamic
- C vs q-dynamic
- B vs q-dynamic
- beta vs q-dynamic

References

Photon Correlation Spectroscopy, article in wikipedia
Algorithms for Time Correlation Experiments
Note on XCS code from Marcin