LIMA: A GENERIC LIBRARY FOR HIGH THROUGHPUT IMAGE ACQUISITION

A. Homs#, L. Claustre, A. Kirov, E. Papillon, S. Petitdemange, ESRF, Grenoble, France.

Abstract

A significant number of 2D detectors are used in large scale facilities' control systems for quantitative data analysis. Common control parameters and features can be identified in these devices, but most of manufacturers provide specific software control interfaces. A generic image acquisition library, called LIMA, has been developed at the ESRF for a better compatibility and easier integration of 2D detectors to existing control systems. Its design is driven by three main goals: i) control system independence; ii) a rich common set of functionalities, providing alternative software solutions when not implemented by hardware; and iii) intensive use of multi-threaded algorithms synchronised by events for ensuring performance of high throughput detectors. LIMA currently supports a dozen of image detectors from different manufacturers, half of them being integrated by collaborating institutes and manufacturers. Although still under development, LIMA features so far basic image transformation, processing and reduction, fast data saving on different file formats, among other features. Its modular design allows not only the extension of generic hardware and software functional blocks, but also the integration of user-defined post-processing algorithms.

INTRODUCTION

The standard beamline (BL) control software at the ESRF is basically built on top of SPEC, a versatile hardware control application, and a variety of distributed device servers accessed through the TACO and TANGO middlewares. A considerable number of 2D detectors have been interfaced in the past using common generic CCD TACO and TANGO device server interfaces, notably simplifying the client SPEC code. However, the real code implementing the configuration and control of the image acquisition process was copied and/or rewritten on each new detector. Moreover, new features had to be manually added on existing detectors, which resulted unpractical in many cases.

In order to go a step further in the standardisation of the control of these devices, we have developed a generic Library for IMage Acquisition (LIMA) [1]. It provides the common functionality expected from 2D detectors, exploiting the hardware capabilities to their maximum extent, and complementing by software the rest features not provided by the detector.

BUILDING BLOCKS

A key design goal was to not degrade the performance of high throughput detectors. This implied the use of parallelisable multi-threaded algorithms for an optimum usage of multi-CPU/cores in modern PCs, and their corresponding event-based synchronisation. The result was the development of ProcessLib, an auxiliary multi-threaded image processing framework. It manages the execution of arbitrary sequences of tasks, combined in chained and/or parallel configurations. Once added to the run queue, a task is executed as soon as a working thread is available, normally when a CPU/core is idle. The asynchronous notification of the task end is implemented through callbacks.

Another basic element developed for LIMA is a message logging framework for code tracing and debugging. It can be dynamically configured to enable/disable individual functional blocks, and, in an independent way, to change the log verbosity. The output trace is indented following the function call stack, which is different for each active thread. Special efforts were made to reduce code execution slowdown when no trace is active, while keeping this dynamic, thread-safe behaviour.

The core of the library has been written in C++ and Python wrapping through SIP has been implemented for most of the classes.

LIBRARY STRUCTURE

The general LIMA layout is shown in Fig. 1.

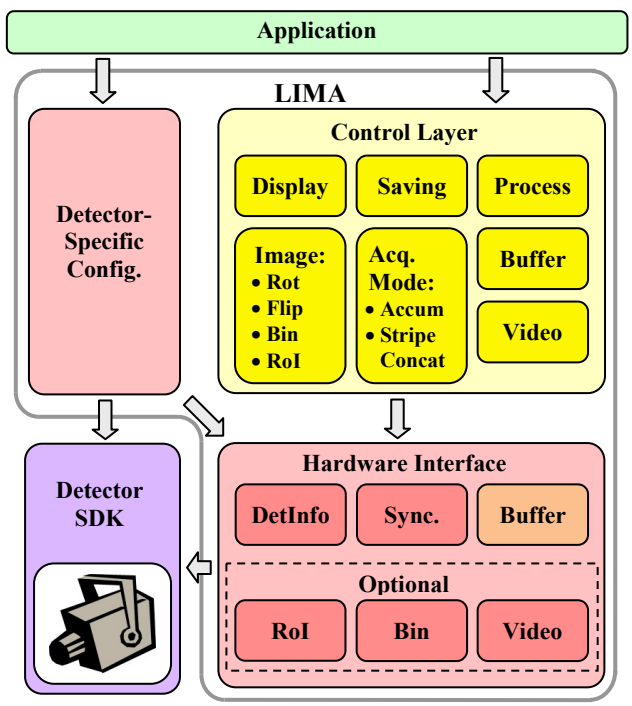

Figure 1: General LIMA layout.

#alejandro.homs@esrf.fr

The library structure divided into two main layers: control, containing the common control and processing code, and hardware, implementing the detector-specific part. The control layer provides the library interface to the high level application. User requests to configure and control the acquisition are gathered by the control layer, so the hardware layer functionality is limited to the generation the image frames in a best-effort basis. The control layer is responsible of: i) adapting the received image geometry if it does not match the user requests, and ii) execute the frame processing chain.

Hardware Layer

The detector-specific code, called "hardware plug-in" must implement a well-defined "hardware interface". It provides basic acquisition start/stop/status control as well as entry points to functional blocks, corresponding to the hardware capabilities. Only three control blocks are mandatory:

- Detector information: name, model, dimensions.
- Synchronisation: internal/external trigger, exposure time and sequencing.
- Image buffer management, for which helper classes are provided with a default behaviour.

All the other capabilities are optional. They include:

- Pixel binning (scaling).
- Sub-image / region-of-interest (RoI).
- Image horizontal / vertical flip.
- Shutter control.
- Video cameras: standard mono/color image formats and gain settings.

This modular design simplifies the integration of new hardware functionality without recoding existing hardware plug-ins.

One important requirement imposed on the hardware plug-in is its responsibility of generating events on every acquired frame. In case the detector API does not provide this feature and relies on blocking/polling interfaces, auxiliary thread management helper classes are provided to simplify the implementation of frame ready callbacks.

As a general rule, the hardware layer is queried to check if it can completely or partially satisfy a particular user request. For instance, when the user asks for a pixel binning combination that is not implemented, the hardware layer replies with the highest binning it can provide. The same best-effort approach should be followed with RoIs, where some hardware alignment constrains might require returning a region bigger than the requested one. The user can always select if a feature is to be implemented entirely by hardware, by software or by a combination of both.

It is worth to remark that to add support for a new detector its only required to implement:

1. The detector information control block
2. The synchronisation control block
3. Acquisition start/stop/status control and frame-ready callbacks

This can be coded either in C++ or Python. Such a minimal implementation will already provide the user with a fully operational system exporting all the common software features.

Control Layer

A modular approach is also found in the control layer, the core of LIMA and its most complex element. Different functions are activated through individual blocks, all of them coordinated by a global control class. The user configures the acquisition using these different modules, many of them implementing parameter caches to minimise hardware access. Once all the settings have been fixed, a "prepare acquisition" command synchronises the hardware configuration, including the (sometimes slow) memory buffer allocation. This minimises the delay in the "start acquisition" command, which can add undesirable latencies in the software synchronisation with other devices like motors or LIMA detectors.

The dynamics of the control layer starts with the "hardware frame ready" callback. The first task to accomplish is the software frame reconstruction, needed if the detector pixel readout sequence does not correspond to its real geometry [2]. Even if this is a detector-specific functionality, it is performed in the control layer to exploit the ProcessLib capability of parallel processing different frames on different CPU/cores.

Individual counters are updated as the frame progresses in the processing chain: last-image-acquired (hardware), last-base-image-ready (geometric reconstruction and transformations), last-image-ready (basic processing), last-image-saved (data storage), etc. The user can register to an acquisition status callback that notifies each time these image counters progress.

Detector-specific Configuration

The control layer provides a user interface to standard acquisition parameters. However, virtually all detectors implement specific settings, which can range from chip timing and readout configurations to generic I/O signal management and ADC gain/threshold levels. To avoid a complex generic infrastructure for these controls, the hardware plug-in is responsible to directly export them to the user through a detector-specific control block. The control layer is not aware of such parameters, so mechanisms have been foreseen to notify important changes that affect the acquisition. For instance, a "max image size changed" callback must be implemented by the hardware if the user can select among detector profiles with different effective image sizes.

AVAILABLE FEATURES

Geometric Transformations

Four basic image transformations are implemented: image rotation (90º, 180º, 270º), horizontal and/or vertical flip, pixel binning (scaling) and RoI (sub-image). From the user coordinates point of view, they are applied in that

order. This means that binning factors include the rotation mode, and RoI coordinates are relative to binned pixels. The simultaneous activation of these transformations requires more complex calculations when some of them are either totally or partially done by hardware. In addition, the implementation of arbitrary pixel binning combinations, including values that are not integer divisors of the detector size, is under development.

Some scientific techniques exploit 2D detectors to measure 1D spectra in image stripes. Pixel binning in one dimension is normally used to improve signal to noise ratio. Such configurations can lead to a very high frame rate (above 1 KHz), interesting in time-resolved experiments [3]. To keep performance under these conditions, LIMA implements the stripe concatenation mode, where a long sequence of many frames can be read/saved at once as a single concatenated image without additional memory copies. In case the original 2D images are required for analysis (i.e., no binning applied), the same stripe calculation can be obtained by the independent RoI-to-spectrum software operation.

Basic Image Processing

Hardware detector constrains can limit the pixel integration either in exposure time (Frelon) or in dose (Maxipix [2]). A simple solution to this problem is provided through the frame software accumulation. It is activated by specifying a maximum frame exposure time; the control layer programs the real number of hardware frames depending of the requested total exposure. In addition to the accumulated image, a saturation pixel mask can also be obtained from the algorithm, important to detect non-linear software artefacts.

Basic image processing algorithms like background subtraction, flat-field correction and pixel masking are already integrated. Standard calculations used for Beam-Position-Monitoring (BPM) including intensity sum, average, std. deviation, min/max and centroid position are also available in a per-RoI basis. That is, the so-called "RoI-counters" are calculated in parallel on multiple sub-images for each frame. The history of the all the RoI-counters is available during and/or after a sequence of images (scan).

Data Saving

Data storage is a key element in high performance image acquisitions. In addition to the detector images, meta-data describing the acquisition environment, including user-supplied meta-data, must be saved. The meta-data concept is called "frame-header" in LIMA and is implemented in the core of ProcessLib as key→value maps. Three levels of meta-data are identified:

- Static: does not change during the life of the process (detector-specific: model, serial number).
- Common: is shared by all the frames in an acquisition sequence (user-defined sample name, sequence start date/time).
- Frame: specific information when frame was taken (high resolution time stamp, instantaneous internal and/or external counters values).

There are three file saving modes currently implemented in LIMA. They differ in the way to trigger the saving of each frame: manual (user request), auto-frame (frame is ready) and auto-header (both the frame and its user-defined header are ready).

The following file formats are currently supported:

- EDF: ESRF Data Format
- Nexus/HDF5: part of the Common Data Model (CDM), developed by SOLEIL and ANSTO
- CBF: Crystallographic Binary Files. It is optimised in LIMA with parallel frame compression.
- Raw

The library also allows the parallel saving of multiple file streams on different medias. This data replication technique has proven to be efficient in high throughput detectors when multiple endpoints (PC, file servers) must receive the same data. Data endpoints examples include: online data analysis workstation, central data storage server (backup, long-term archiving) and local NAS with BL user disks (to be brought to their home institute).

Online Visualisation

Online data visualization is required in must of the cases as a direct feedback of the acquisition evolution. Current LIMA display mechanisms export the image data to a channel, to be read by a separate client application that performs the real visualisation. One method publishes the data on the standard ESRF SPEC Shared memory (SPS). A second method exports the generic LIMA video interface through the TANGO LIMA server. In each case, a dedicated client Qub/Qt4-based application shows the live image.

External Software Plug-ins

Finally, user-defined software plug-ins can be used to execute arbitrary image-based operations. An entry point in the control layer completely exports the ProcessLib functionality, allowing an external code to be called on every frame. Again, the external software operation can be implemented C++ or Python.

PROJECT STATUS

Supported Detectors

The list of currently supported detectors is:

- Camera simulator: a pure software plug-in
- ESRF Espia-based Frelon camera and Maxipix detectors (Single chip, 2x2 and 5x1 chip arrays)
- Dectris Pilatus and Mythen
- GigE Basler and Prosilica cameras
- ADSC and MarCCD detectors
- XPAD pixel detector
- Roper Scientific cameras through PVCAM
- PCO.Dimax (under development)

Current Applications

The first application on top of LIMA was the ESRF Frelon TACO device server; aimed to be 100 % compatible with its predecessor. The existence of an important amount of SPEC code for different Frelon-based experimental setups allowed the fast commissioning and validation of the basic LIMA elements.

In parallel, a generic LIMA TANGO device server was developed, with the ESRF Maxipix detector as its first specific hardware plug-in. As new hardware plug-ins were added, this new TANGO interface gradually exported the full LIMA potential: multi-chip reconstruction, RoI-counters, frame accumulation, background/flat-field, video and so on. New SPEC code has been written to efficiently exploit this new 2D detector functionality.

A different YAT-based TANGO server interface has been developed and maintained by SOLEIL Synchrotron, allowing LIMA integration in their control system.

Finally, an X-BPM TANGO server on top of LIMA has been developed; its installation currently uses Basler cameras.

Operating Systems

LIMA has been mainly developed on Linux operating systems, mainly on SuSE Linux 8.2 (i686) and on RedHat EL 5 (x86_64). It has been later ported to RedHat EL 4 (i686) and OpenSuSE 11.2 (i686 & x86_64).

A Windows portage has been made by coding basic thread management primitives. After the validation of the camera simulator, real hardware plug-ins like the PCO.Dimax are under development.

Performance

The first validation of LIMA performance has been made with the ESRF Frelon HD camera controlled by a Transtec X2100 Workstation (Dual Intel Xeon @ 2.66 GHz, 32-bit). Continuous Frame-Transfer-Mode acquisitions at 30 frames/sec (120 MBytes/sec) can be performed with active SPS online display and parallel data saving. Higher frame rates (about 1000 frames/sec) can be achieved on the same PC in stripe (spectrum) mode with hardware binning and Kinetics hardware RoI. As with its non-Lima predecessor, debugging output must be disable to not slowdown acquisition.

The most recent high performance LIMA validation has been done at the ID03 ESRF BL with the Maxipix detector in single-chip configuration. The control PC is a more powerful Ecrin/Trenton server (Dual Quad Core Xeon E5335 @ 2.0 GHz). Acquisitions at the maximum frame rate (about 1500 frames/sec), limited only by the maximum Espia board data rate (180 MBytes/sec), are possible with parallel saving and 5 active RoI-counters definitions.

Code Repository

The LIMA source code is available under the Gnu Public License at the European Phonton-Neutron (EPN) Science Campus Web site [1]. Its development is managed by GIT; the repository can be accessed at [4]. The official project documentation can be found at [5].

ACKNOWLEDGEMENTS

Important contributions in hardware plug-ins and Nexus file saving format have been made by F. Langlois, A. Noureddine, A. Buteau and X. Elattaoui from SOLEIL Synchrotron inside a collaboration framework on LIMA development. In the same way, M. T. Nuñez-Pardo-de-Vera (PETRA III/DESSY) and C. Nielsen (ADSC) have also implemented specific hardware plug-ins.

CONCLUSIONS

A generic Library for IMage Acquisition (LIMA) has been developed for controlling high throughput 2D detectors. It allows the optimum exploitation of hardware optimizations, like pixel binning and RoIs, but it also provides software alternatives for detectors that do not implement them. A common set of software image operations (geometric transformations, processing, calculations, saving and visualization) is available for all the detectors. Their multi-threaded nature notably increase acquisition performance on multi CPU/cores PC. The modular library design simplifies the integration of new hardware and software functionality (plug-ins). An increasing number of supported detectors are already used in Synchrotron facilities with good performance results.

REFERENCES

[1] http://forge.epn-campus.eu/projects/lima
[2] C. Ponchut, J.M. Rigal, J. Clement, E. Papillon, A. Homs and S. Petitdemange, "MAXIPIX, a Fast Readout Photon-Counting X-Ray Area Detector for Synchrotron Applications", JINST 6 (2011) C01069.
[3] J.C. Labiche, O. Mathon, S. Pascarelli, M.A. Newton, G.G. Ferre, C. Curfs, G. Vaughan, A. Homs and D. Fernandez-Carreiras, "The Fast Readout Low Noise Camera as a Versatile X-Ray Detector for Time Resolved Dispersive Extended X-Ray Absorption Fine Structure and Diffraction Studies of Dynamic Problems in Materials Science, Chemistry and Catalysis", Rev. Sci. Instrum. 78/9 (2007) 091301.
[4] git://git.epn-campus.eu/repositories/Lima
[5] http://lima.blissgarden.org/