THE LIMA PROJECT UPDATE

S. Petitdemange, L. Claustre, A. Homs#, E. Papillon, R. Homs-Regojo, ESRF, Grenoble, France.

Abstract

LIMA, a Library for Image Acquisition, was developed at the ESRF to control high-performance 2D detectors used in scientific applications. It provides generic access to common image acquisition concepts, from detector synchronization to online data reduction, including image transformations and storage management. An abstraction of the low-level 2D control defines the interface for camera plugins, allowing different degrees of hardware optimizations. Scientific 2D data throughput up to 250 MB/s is ensured by multi-threaded algorithms that exploit multi-CPU/core technologies. Eighteen detectors are currently supported by *LIMA*, covering CCD, CMOS and pixel detectors, and video GigE cameras. Control system agnostic by design, *LIMA* has become the de facto 2D standard in the TANGO community. An active collaboration among large facilities, research laboratories and detector manufacturers joins efforts towards the integration of new core features, detectors and data processing algorithms. The *LIMA* 2 generation will provide major improvements in several key core elements, like buffer management, data format support (including HDF5) and user-defined software operations, among others.

INTRODUCTION

The ESRF beamline (BL) control system is composed by a variety of devices installed over the optics and experimental hutches. Most of these equipments are controlled by dedicated PCs, and accessed through the network using the TACO and TANGO middlewares [1]. The user scientist performs the experiment from the main BL control workstation by means of SPEC, a versatile hardware control application that communicates to the distributed device servers. A significative number of 2D detectors have been integrated in the past using generic interfaces. A first generic layer is given by SPEC image interface, which allowed the development of unified macros. A second layer was a common CCD TACO/TANGO server interface, so only one generic SPEC controlled is needed. Finally, efforts were made to reuse the device server code implementing the configuration and control of the image acquisition process. However, back-porting new features to existing detectors was often unpractical.

The next step in the standardisation of 2D detector control was the development of LIMA, a generic Library for IMage Acquisition [2,3]. It provides a common functionality for various detectors, using hardware acceleration capabilities when available.

#alejandro.homs@esrf.fr

THE LIMA LIBRARY

Goals

The LIMA library design was driven by four main goals. First, to be control system-independent, so it can be used by different laboratories. A second goal was to address high speed detectors, so it i) exploits the hardware optimisations to their maximum extent; ii) makes intensive use of multi-threaded algorithms; and iii) minimises unnecessary memory copies. Another requirement for the library was to provide the same high level interface for all detectors, both from the configuration and frame processing point of views. This implies the activation of a software implementation when the corresponding functionality is not available in the hardware, like pixel binning or the selection of a sub-image/region-of-interest (RoI). The last goal was to design the library in a modular way, so extensions can be easily added in the future, both in the low hardware level and the application interface level.

Library Structure

LIMA separates the image generation from its software processing by defining two levels: the hardware layer and the control layer, respectively. The hardware layer is responsible for exporting the detector capabilities and for controlling them accordingly. On top of it and below the application layer, the control layer must configure the acquisition parameters through the hardware layer. Depending on the missing hardware optimisations, the control layer must also activate the fallback software algorithms if their functionality was requested by the user. Finally, any pure-software frame processing, like data reduction algorithms, is managed by the control layer.

The hardware layer implementation, called “camera plugin”, must provide a well-defined hardware interface, an abstraction of the low-level 2D detector control. It normally calls the Software Development Kit (SDK) API. In addition to the basic start/stop/status functions, the hardware interface contains a list control objects associated to the available hardware capabilities. Three capabilities are mandatory: detector information, synchronisation with external devices and buffer management. The remaining capabilities are considered as optimisations and are optional. Some of them affect the image geometry, such as pixel binning, RoI and the horizontal/vertical flip (mirror). Others are completely unrelated, like shutter control and “native saving”, which allows the SDK to automatically save the raw data if no additional image processing is needed. The modular structure of the hardware interface has simplified the integration of new low-level functionality. For instance, the interface for video cameras and the native saving were added to LIMA without affecting the existing detectors.

ISBN 978-3-95450-139-7

The control layer is the generic detector interface towards the application. It also follows a modular design with blocks that control the different configuration areas: image, saving, buffer management, video, among others. Depending on the configured parameters and the detector capabilities, some software processing tasks are activated; these are known as "internal software operations". Additionally, "external software operations" can be added to the frame processing chain, either provided by the LIMA task repository or completely user-defined. Once the parameters are set by the user and the acquisition starts, it will independently progress without the need of intervention by the application layer. The user can either poll on the acquisition status or request to be informed of its evolution, in a complete passive role.

Almost every detector exports specific configuration parameters with limited or null relevance to the other detectors. Typical examples are sensor readout modes, ADC gains, DAC thresholds, among many others. The control layer is not aware of such particularities in order to avoid a complicated interface to generic parameters. It is the camera plugin role to export all the detector-specific configuration parameters and procedures to the application layer.

LIMA is written in C++, and a Python wrapping with SIP is provided for most of the classes. Camera plugins as well as user-defined frame processing tasks can be written either in C++ or in Python. The library is available on several GNU/Linux distributions: RedHat EL 5, openSUSE 11 and Debian 6, both 32-bit and 64-bit architectures. LIMA runs on Windows XP and 7 32-bit as well; a porting to Windows 7 64-bit has been recently developed and is under test.

Available Features

Several common concepts have been identified in the abstraction of the 2D detector data processing. The first one to be applied is the frame reconstruction, necessary when the detector readout sequence does not follow the real sensor geometry. Typical cases are detectors with parallel readout of different areas, like CCDs with multiple ADCs or multi-chips modules (ESRF Maxipix). If the SDK supplies such kind of "geometrically incorrect" raw data, LIMA can systematically call a detector-specific reconstruction function. Some detectors have a limited integration capacity, either in time or in dose. For these cases, the accumulation mode divides the requested exposure into shorter intervals and transparently integrates them into a single frame.

Basic geometric transformations can be performed on each frame: image rotation, horizontal/vertical flip, pixel binning and RoI extraction. Except rotation, these operations might be partially or completely performed by the detector; LIMA complements the remaining transformations by software to fulfil the user request. Figure 1 shows the evolution of an image through different hardware and software transformations.

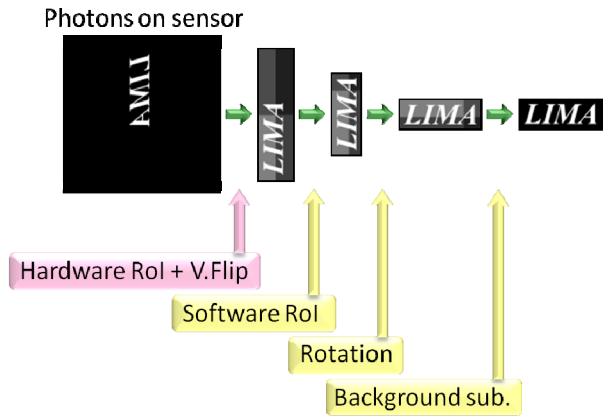

Figure 1: Example of an image transformation sequence.

Once the image has the appropriate shape, pre-processing algorithms are executed: background subtraction, flat-field correction and pixel masking. At this stage the image is considered ready for saving and/or for further data reduction processing. The 2D data can be saved manually (on user request) or automatically (on each "image ready" event). User-defined metadata is composed in LIMA by two different parts: common and per-frame. The common part is a fixed information block used for all the frames in an acquisition, like the sample name or scan conditions. In addition to that, specific per-frame parameters can be supplied, such as the instantaneous beam intensity or sample temperature. If per-frame metadata is used, the image is not saved until the corresponding information is received from the application level. Currently supported formats are EDF (ESRF Data Format), CBF (Crystallographic Binary Files) and Nexus/HDF5 (through the Common Data Model developed by SOLEIL and ANSTO). Data compression is used in CBF files, and is optional in EDF files using gzip.

Additional data reduction tasks are available for the LIMA frame processing chain: statistic calculations on pixel intensity (known as RoI-counters), centroid determination for Beam Position Monitoring (BPM), and image projection (binning) on one dimension (RoI-to-spectrum). Azimuthal regrouping and integration has been recently added to LIMA through pyFAI [4], useful for X-ray scattering and diffraction techniques. It provides either 1D (azimuthal integration) or 2D (polar transformation) data reduction, taking into account not only the experiment setup geometry, but also possible spatial distortions in the detector (by optical fiber tapers). Finally, LIMA offers the possibility to insert a user-defined task in the frame processing chain. This external software operation will be called at every frame and can generate either a new image to be supplied to another operation (link task) or another kind of data that LIMA is not aware of (sink task).

ISBN 978-3-95450-139-7

Frame Dynamics

In order to exploit the modern multi-CPU, multi-core platforms, the LIMA software processing is based on an auxiliary framework developed for parallel task execution: ProcessLib. When a new "hardware frame ready" event is generated by the camera plugin, a chain of tasks is built and scheduled for execution. As their name suggests, link tasks, like image rotation and background subtraction, must be executed sequentially. However, tasks reading the same frame can be parallelised; this is typically the case of data reduction sink tasks such as file saving and RoI counters. In most of the cases the processing of a frame is independent of the following one, so LIMA also parallelises task chains associated to different frames. A particular task requiring a sequential handling of frames, like file saving, can force the serialisation internally. A pool of working threads is used to execute tasks as they are scheduled in order to not overload the operating system.

Callbacks are used in LIMA for asynchronous notifications, including task completion events. Each time an event marks a relevant stage in the frame processing, a dedicated frame counter in the acquisition status is incremented so the user can follow its progress. For instance, the "hardware frame ready" and "frame saved" events increment the "last image acquired" and "last image saved", respectively. An example of a frame processing chain and its associated events is shown in Figure 2.

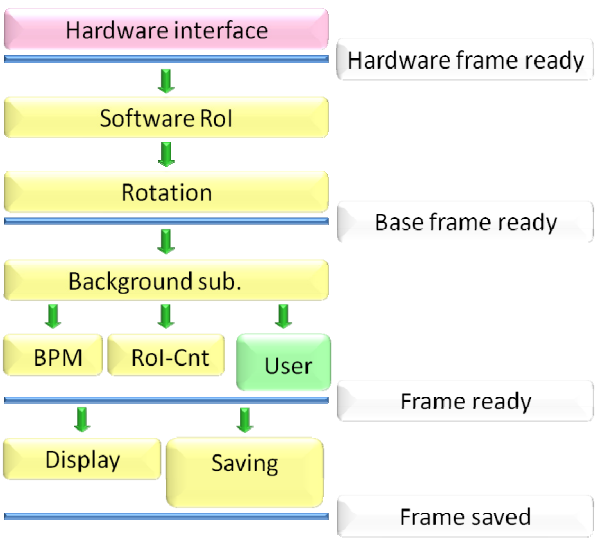

Figure 2: Frame processing chain and events.

CURRENT STATUS

Supported Detectors

The following detectors are interfaced to LIMA:

- ESRF Frelon and Maxipix (single chip, 2x2, 5x1)
- Dectris Pilatus and Mythen
- Basler, Prosilica, Point Grey and IDS uEye GigE cameras
- ADSC, MarCCD and RayonixHS detectors
- Andor I-Kon
- XPAD pixel detector
- Roper Scientific cameras through PVCAM
- PCO Dimax and Edge cameras
- Perkin Elmer flat panel detector
- PhotonicsScience
- DSG/STFC Xh 1D Ge detector

Scientific Applications

LIMA has been in production level for about 3 years in more than 20 BLs at the ESRF. The main application is a TANGO server that exports different devices with interfaces to LIMA core functionality, detector-specific parameters and data-reduction tasks. On top of that, a BeamViewer HTTP server can be started, which is very useful as display for BPM applications.

Many X-ray techniques exploit LIMA capabilities. Fast imaging experiments notably use 2D detectors, like radiography, tomography and ptychography. Area detectors are also key elements in diffraction and scattering techniques, from CDI to XPCS and GISAXS, including macromolecular crystallography. Time resolved XAS and powder diffraction experiments use as well LIMA as primary detector control. Video cameras are used for both sample visualization and BPM applications.

In terms of performance, LIMA allows data acquisition at the maximum rate supported by the detectors in the framework of the above-mentioned applications. For instance, data rates of 250 MB/s are achieved on the PCO.Dimax CMOS camera with local disk saving. The multi-threaded approach of its frame processing chain allows LIMA to follow the detector data rate, again to the extent of the current applications. As an example, azimuthal integration with pyFAI is performed on every frame acquired with a Frelon HD camera in Frame-Transfer-Mode (half-CCD) at full speed (30 fps, 120 MB/s), controlled by a dual six-core PC.

LIMA Collaboration

Shortly after LIMA entered into the production phase, a spontaneous collaboration among different European synchrotron facilities and national laboratories was born. SOLEIL, PETRA III/DESSY, ALBA, MAX-LAB, FRM-II/TUM and ILE/LULI/Ecole Polytechnique have either contributed with camera plugins, installations scripts and bug fixes, or installed LIMA for operation/test and helped in the debugging on different scenarios. In the same direction, camera manufacturers like ADSC, Rayonix and DSG/STFC have developed LIMA hardware plugins for their detectors, notably simplifying their integration in the instrumentation community.

This active collaboration motivated the organization of a LIMA Workshop, which took place in March 2013 at the ESRF. A review the application of LIMA at the different institutes as well as their requests represented a starting point for fruitful discussions on the future developments of the project.

ISBN 978-3-95450-139-7

Code Repository

The LIMA source code is available under the Gnu Public License at Git-Hub global repository manager [5]. The official project documentation can be found at [3].

TOWARDS LIMA 2.0

In addition to the needs for integration of new detectors, several limitations were identified by the ESRF and the collaborators during the LIMA Workshop. A global restructuration of the image buffer management was agreed to better fit the current and future applications. One of the goals of the new interface is to allow the start of a new acquisition while the previous one is still processing acquired frames, with the aim of reducing the gap between scans. Extensions also foresee the support of detector-generated metadata, as well as more flexible, multi-client callback interface. A major redesign of the saving interface would also be desirable for a flexible integration of more generic schemas of the HDF5 data format. The diversity of the experimental techniques that exploit LIMA seems to require such extended flexibility. Finally, new data reduction algorithms would be appreciated by the scientific community to further improve the online data analysis. In particular, a formal framework for sinogram generation is very useful in tomography experiments, as well as the implementation of RoI counters in polar coordinates (simplified azimuthal integration). The integration of the latter data reduction tasks in LIMA is already in progress.

CONCLUSIONS

The LIMA library has consolidated its role as generic framework for 2D detector control not only at the ESRF but also at several large facilities and national laboratories. The supported detectors and implemented features simplify its integration into experimental control system, especially if they are based on (or able to connect to) the TANGO middleware. High speed acquisitions at the maximum rate required by the experiments are performed with LIMA, including diverse data reduction algorithms. The LIMA community actively collaborates in the development of new plugins and software tasks, and important improvements in the core are envisaged in the near future to better fit scientific and instrumentation needs.

ACKNOWLEDGEMENTS

As expressed above, important contributions in hardware plug-ins, saving interfaces, installation scripts and TANGO extensions have been made by collaborators, in particular at SOLEIL, PETRA III/DESY, FRM-II/TUM, MAX-LAB, ALBA, ADSC, Rayonix and DSG/STFC. Nexeya Systems has worked in the improvement of the project documentation. Finally, Matias Guijarro and Jerome Kieffer from the ESRF ISDD software group have developed the BeamViewer web server interface and pyFAI, respectively, and Alessandro Mirone has contributed with useful discussions.

REFERENCES

[1] A. Homs-Purón, D. Beltrán, A. Beteva, M. C. Domínguez, P. Fajardo, A. Götz, J. Klora, E. Papillon, M. Pérez, V. Rey: “LINUX/PCI: The ESRF beamline control system modernisation”, Proceedings of ICALEPCS2003, MP565, Gyeongju, Korea.

[2] A. Homs, L. Claustre, A. Kirov, E. Papillon, S. Petitdemange: “LIMA: A generic library for high throughput image acquisition”, Proceedings of ICALEPCS2011, WEMAU011, Grenoble, France.

[3] http://lima.blissgarden.org

[4] J. Kieffer, D. Karkoulis, “PyFAI, a versatile library for azimuthal regrouping”, Journal of Physics: Conference Series 425 (2013) 202012.

[5] git://github.com/esrf-bliss/Lima.git

ISBN 978-3-95450-139-7