On-Axis-View: a GUI library to enhance the sample environment control

Jordi S Andreu, Fulvio Becheri, Guifré Cuní, Roberto J Homs, Gabriel Jover-Manas and Daniel Roldan

ALBA Synchrotron Light Source, Carrer de la llum 2-26, 08290 Cerdanyola del Vallès, Barcelona, Spain

E-mail: jandreu@cells.es, ctbeamlines@cells.es

Abstract. Many X-Ray experiments performed in a synchrotron facility (like Macromolecular Crystallography, Non-Crystalline Diffraction or Powder Diffraction experiments) require precise control of the sample position and orientation (centering procedures), the proper positioning of the X-Ray beam and its morphological analysis and also, a friendly integration of all the different optical and motorized instruments into the main control interface. The *On-Axis-View* library provides such functionalities, enhancing the development of customized Graphical User Interfaces according to the specific user requirements and also a better user experience. In addition, the usage of the library greatly reduces the required time to develop new solutions and eases its maintenance.

1. Introduction

A mandatory feature of any synchrotron beamline is the remote monitoring and operation of all its devices: motors, cameras, valves, detectors, etc. From the control system side, there are two important issues to handle prior to operate. The first issue is the fine-tuning, alignment, monitoring and characterization of the target sample at the final experimental position. The second is the correct characterization and alignment of the incident beam at the position of the sample. In order to visualize the scene, an area scan camera is usually mounted with its optical axis aligned with the incident beam trajectory and focused on the position of the sample. This video image provides a visual control of the sample *i.e.* its position and orientation during basic operations like mounting/unmounting or during the centering operations. With no sample mounted, the same camera can be also used to visualize the X-ray beam by placing a Yttrium-Aluminium-Garnet (YAG) crystal film at the position of the sample. The images obtained with the YAG crystal are used to characterize the X-ray incident beam by calculating its position, width and height.

The ALBA Control System [1, 2] is based on the Sardana package [3, 4], a software for Supervision, Control and Data Acquisition (SCADA) and the Taurus framework [5, 6] for creating and supporting Graphical User Interfaces (GUI) and command line applications, dedicated to experiment control and data acquisition of both the accelerator and beamlines. Sardana is built on top of the Tango framework [7] extending it with a powerful python-based macro development environment. This allows to plug in custom procedures and complex macros via its *MacroServer*. It also provides a comprehensive access to the hardware through the *Device*

Pool and based on common and dynamic interfaces. The Taurus framework was originally conceived as an in-house solution for connecting client side applications to Tango device servers. It provides a user interface code for the Sardana suite based on the *Model-View-Controller* approach. Based on the Sardana and Taurus frameworks, the aim of the library is to embed in a single product the tools required by the scientists for sample and beam operation, providing a high-level and user-friendly interface integrated with the ALBA control system. To achieve this goal, the library has to act as a reliable link between the different beamline equipment involved in those procedures and the GUI delivered to the final users. The effort to integrate the OAV library to other control systems can be greatly reduced thanks to the capability of taurus to support models other than Tango.

The On-Axis-View (OAV) library could be conceptually divided in three different parts. The first one is the library core containing the abstract classes. These classes provide the engine to update the representation of the different beamline objects on the displayed image (canvas). The second one is a specific Taurus widget which serves to display the video image via the guiqwt library [8] and related image tools encapsulated in a menu bar. Third and last, a configuration module to define how the real equipment is integrated in the control system. According to a given configuration, the library creates specific objects representing all the elements involved in the sample/beam management and characterization. This improves the transferability of the library to different beamlines.

2. Technical Details

The OAV library has been written in python 2.7 [9] and organized as follows. First, a base core composed by two abstract python classes which provide a link between the real object in the experimental setup and its representation in the canvas. One class is the *ZoomAware* class which provides the set of transformations used for repainting any real object represented in the canvas according to the camera zoom. These methods keep track of the current zoom value and implement the coordinates transformation between the real coordinates respect to the laboratory coordinate system and the canvas coordinates. The other abstract class included in the library core is the *ObjectAware* class, which inherits from the ZoomAware class. It provides an additional set of transformations used for recalculating the canvas object position according to any change of the position of a certain real object. The second part is the front-end widget, which is a specialization of a TaurusWidget class. This class contains an ImageDialog class (from guiqwt) embedding the video image and a configurable toolbar menu as main access point to the tools. In our case, the ImageDialog object contains the video image supplied by the *video_last_image* attribute from a LImA [10] Tango device server, responsible for managing the OAV camera at the position of the sample. Since this widget inherits from a TaurusWidget class, it can be easily added to any other Taurus-based application. Finally, a configuration module which is used for setting up the library at each beamline according to the specific integration of the real equipment.

The library comes with a set of tools extending the default guiqwt ones and completing the user interface for the scientist to operate their experiments with control, confidence and precision (see Table 1). The group of tools currently provided with the library could be separated in three categories:

- **Base tools:** Specialization of default guiqwt tools which make no use of the OAV core classes.
- **Simple tools:** Customized default guiqwt tools which inherits from any of the OAV core classes.
- **Complex tools:** Customized simple OAV tools with a higher events flow complexity.

Table 1: The current OAV tools catalog is organized in three different categories: base*, simple† and complex‡ tools. Any of these tools can be added to the OAVWidget by adding its name to the list of tools passed as optional argument.

Tool	Description
`cross`*	Automatically draws a cross-hair at the optical axis position, which is assumed to be provided by an external source.
`save`*	Saves to a file the current video image including the canvas objects.
`centering`†	Calculates the distance between a point set by a mouse click (usually the sample position) and the beam position, and moves the selected point-object to the beam position.
`n-click centering`†	Calculates and centers the sample making the target point invariant over translations and sample rotations.
`circle`†	Draws a circle element in the canvas and displays its diameter on screen. The circle can be drawn in sample aware mode, which implies that it will move as attached to the sample.
`ruler`†	Draws a ruler on the canvas which can be used to measure any element in the image. The measured value is shown on screen. The ruler can be drawn in sample aware mode, which implies that it will move as attached to the sample.
`scale`†	Draws a reference scale on the left-bottom corner showing the length value of each axis. The values are dynamically updated for each change of the zoom value.
`beam`†	Draws a rounded shape according to the beam position and size.
`select position`‡	Selects a set of points in the canvas and stores the corresponding motor sample positions which can be retrieved for future operations.
`raster scan`‡	Define a grid of points whose positions are used to generate a 2D-scan. The grid of points defined by the tool can be used to execute a complex collection of data at each point via an external program (in our case, a Sardana macro executed by the MacroServer device). The tool also provides a callback method which changes the spot color background associated to each grid position (which represents the incident beam) according to a given color scale.

3. Tools implemented at ALBA beamlines

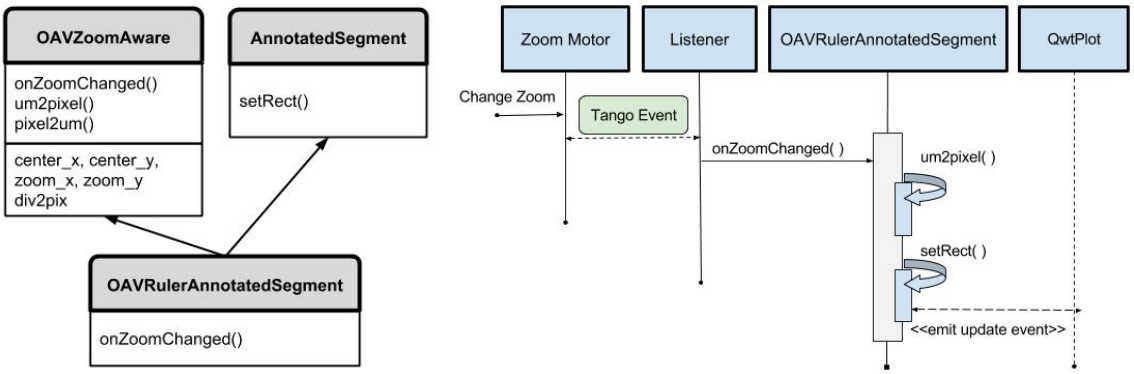
The OAV library is used on three of the seven beamlines at ALBA: the Powder Diffraction (MSPD-BL04), the Non-Crystalline diffraction (NCD-BL11) and the Macromolecular Crystallography (XALOC-BL13) beamlines. As we have previously mentioned, the OAVWidget class inherits from the TaurusWidget class and then, the addition of the OAVWidget to any other GUI is straightforward. Thanks to this, a common set of generic tools is shared across those beamlines, ready to be used by just configuring the tools through the configuration module.

Although the OAV library is shared across all beamlines, some tools need to be re-implemented according to the different equipment integration. For example, this is the case of the *n-click centering* tool, used at MSPD and XALOC, which has a different implementation due to their differences in the motorization of the sample. In other situations, the requirements

are strongly dependent on the beamline workflow and equipment, driving us to design beamline-specific tools. An example is the *raster scan* tool used at XALOC, which provides the control over the whole experiment and shows the results on the GUI. A similar case is the *select position* tool used during a microdiffraction experiment at MSPD. Although the tool does not provide the control of the experiment, the user is able to select a set of points which can be used later on by a Sardana macro to measure the diffraction.

3.1. A simple tool: the ruler

One basic requirement from any beamline scientist is to measure the length of the objects found at the sample position i.e. the sample, the incident beam, etc. The *ruler* tool provides this functionality by enabling the user to draw a straight line between two different points in the canvas. The main object encapsulating this tool is the *OAVRulerAnnotatedSegment* class, which inherits from the ZoomAware class and the AnnotatedSegment class (Figure 1a). When the camera zoom changes, a Tango event is generated carrying the new zoom value and it is received by a listener contained in the OAVRulerAnnotatedSegment object (Figure 1b). Since the shape object owns the necessary methods to calculate its length in the real space (*i.e.* the distance between the two points in the real world corresponding to the two canvas points) the ruler object is redrawn and the new distance is reported as a text label next to the canvas object. Once the ruler object is created, the user can also modify the object by editing its edges and the displayed value gets updated automatically (Figure 3a).



(a) UML class diagram

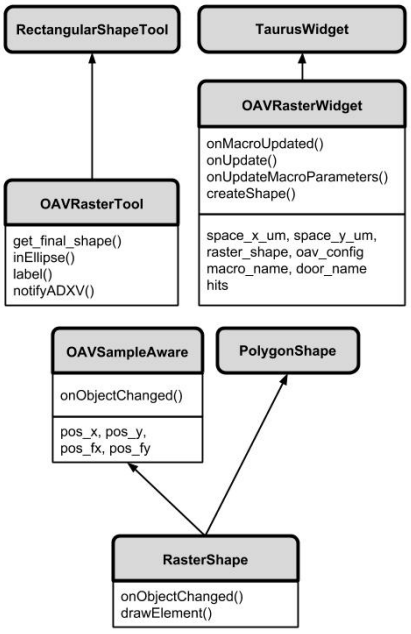
(b) UML sequence diagram

Figure 1: UML class and event sequence diagrams corresponding to the ruler tool.

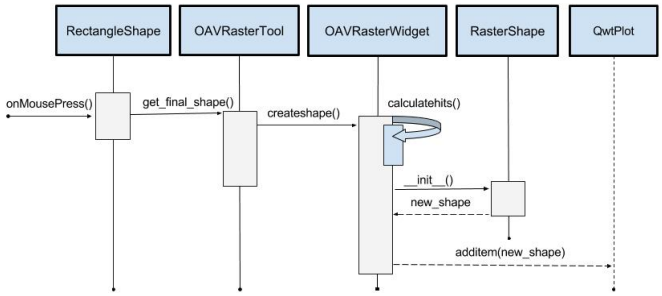
3.2. A more complex tool: the raster scan

Among the set of experiments conducted in a beamline, most of them can be carried out by executing a python code from a Command Line Interface (CLI). However, for some of them, a specific GUI greatly simplifies the experiment setup, monitoring and data acquisition. For instance, the proper characterization of the sample in any experiment of macromolecular crystallography is a key factor to decide the best sample region from where to collect the diffraction images. The raster tool helps the scientist by automatically define a series of collections to be performed at different points of the sample.

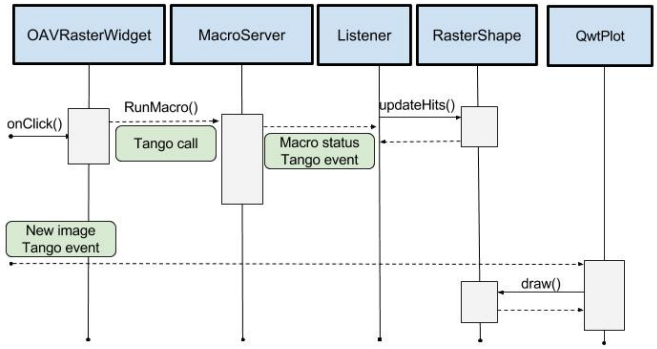
We show in Figure 2a the UML class diagram for the raster tool. The tool is composed of three different objects: the tool itself, containing the objects to link with the ImageDialog class; the *RasterShape* class, a specialization of the shape class to provide the custom behavior for the shape, in this case, a grid of rounded shapes; and finally, the *RasterWidget*, which is used as entry point to modify the current shape (*i.e.* adding or substracting scan points) and to start the scan process from the GUI.



(a) UML class diagram

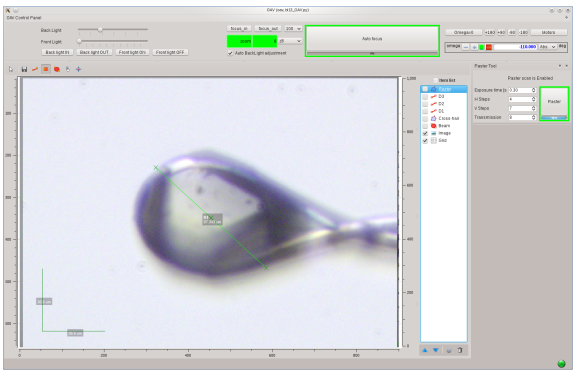


(b) Grid creation events diagram

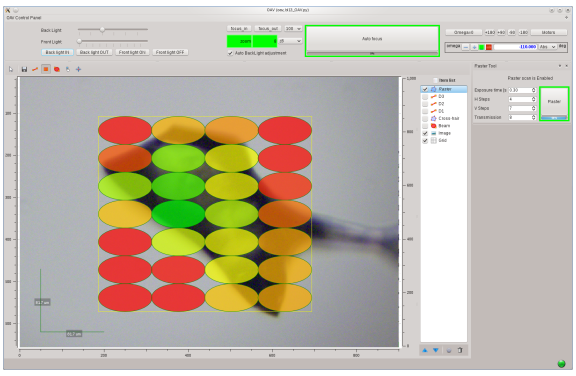


(c) Results display events diagram

Figure 2: UML class and event sequence diagrams of the raster scan tool.



(a)



(b)

Figure 3: Two snapshots of OAV-Raster application from XALOC beamline (BL13). **Left:** The ruler tool helps to measure the size of the sample in a pin. **Right:** During a raster scan the target spots are colored according to a color scale indicating the best spots for diffraction (in green).

Initially, the user selects a rectangular region over the sample image which is filled with rounded objects representing the incident beam. This generates a regular grid of points each of them representing a target position for a data collection (see Figure 2b). Once the grid is drawn, the user can add or substract target points manually by using the OAVRasterWidget controls. The scan process is an external procedure which returns the results of the scan via Tango events. Once the event is received, the results can be managed by the raster object. In our case, the results are represented coloring the spots according to a figure of merit function. The ellipses objects are redrawn with the proper color triggered by the Tango event responsible for updating the video image (see Figure 2c). In this way, the scientist can visualize which is the most convenient spot from where to take the complete collection of images. In Figure 3b we show a snapshot of the XALOC-BL13 beamline GUI used to perform the raster scans thanks to the this tool.

4. Conclusions

The OAV library provides a framework for developing either custom simple tools or more complex solutions which can be easily embedded in any other application. The common API and the configuration files gives the developer an easier and faster way to integrate the library. Apart from the generic tools shared across different beamlines, we have also shown how the library is used to provide beamline-specific solutions. Several tools has been designed and implemented for Powder Diffraction experiments, Non-Crystalline Diffraction or Macromolecular Crystallography taking advantage of the OAV library, obtaining clean and user-friendly GUIs while fulfilling the requirements of the final users.

Acknowledgments

The authors wish to acknowledge the work done by the whole ALBA controls group. Also acknowledge the support and feedback provided by the scientist from Powder Diffraction (MSPD-BL04), Non-Crystalline Diffraction (NCD-BL11) and Macromolecular Crystallography (XALOC-BL13) beamlines, an important part of the refinement design process of the library and tools. The authors also thanks R. Pastor for reviewing this manuscript.

References

[1] Fernández-Carreiras D *et al. The design of the ALBA control system: a cost-effective distributed hardware and software architecture* Proc. ICALEPCS 2011 Grenoble, FRBHMUST01.
[2] ALBA website: http://www.albasynchrotron.es
[3] Reszela Z, Cuní G, Fernández-Carreiras D, Klora J and Pascual-Izarra C *Sardana - A python based software package for building scientific SCADA applications* Proc. PCaPAC 2014 Karlsruhe, WCO206.
[4] Sardana website: http://www.sardana-controls.org
[5] Pascual-Izarra C, Cuní G, Falcón C M, Fernández-Carreiras D, Reszela Z, Rosanes M and Coutinho T M *Efforless creation of controls & data acquisition graphical user interfaces with taurus.* Proc. ICALEPCS 2015 Melbourne, THHC3003
[6] Taurus website: http://www.taurus-scada.org
[7] Tango website: http://www.tango-controls.org
[8] guiqwt website: http://pythonhosted.org/guiqwt
[9] Python website: http://www.python.org
[10] Homs A, Claustre L, Papillon E and Petitdemange S *LImA: a generic library for high throughput image acquisition.* Proc. ICALEPCS 2011 Grenoble, WEMAU011