ISBN: 978-3-95450-236-3 ISSN: 2673-5350 doi:10.18429/JACoW-IBIC2023-WEP046

PROGRESS ON DISTRIBUTED IMAGE ANALYSIS FROM DIGITAL CAMERAS AT ELSA USING THE RABBITMQ MESSAGE BROKER

M. Switka*, K. Desch, T. Gereons, S. Kronenberg, D. Proft, A. Spreitzer
Physikalisches Institut, University of Bonn, Germany

Abstract

In the course of modernization of camera based imaging and image analysis for accelerator hardware and beam control at the ELSA facility, a distributed image processing approach was implemented, called `FGrabbit`. We utilize the RabbitMQ message broker to share the high data throughput from image acquisition, processing, analysis (e.g. profile fit), display and storage between different work stations to achieve an optimum efficacy of the involved hardware. Recalibration of already deployed beam profile monitors using machine vision algorithms allow us to perform qualitative beam photometry measurements to obtain beam sizes and dynamics with good precision. We describe the robustness of the calibration, image acquisition and processing and present the architecture and applications, such as the programming- and web-interface for machine operators and developers.

INTRODUCTION

The ELSA facility [1] deploys around 50 optical beam monitors based on luminescence screens (Chromox or OTR) and synchrotron radiation. The previous installation of cameras with analogue video signal output provided mediocre beam analysis capability due to limited signal quality and the lack of comprehensive camera control. To obtain qualitative beam data from photometry we modernize hardware and software of the optical monitors, primarily utilizing Ethernet-capable digital cameras controlled with non-commercial camera drivers and open source machine vision libraries for image calibration. The`FGrabbit` framework was created [2] to provide resource-optimized camera control, data handling and computing power based on the `RabbitMQ` message broker for distributed analysis. A Gigabit Ethernet (GigE) network between PoE-capable switches links processing computers and image generating devices, such as GenIcam [3] compatible digital cameras, other digitizers or image generating programs. A graphical user interface allows access to the `FGrabbit` framework from the accelerator control system or the facility website and provides comprehensive device and image control.

ARCHITECTURE

The architecture of the `FGrabbit` framework is depicted in Fig. 1. All peripheral devices (cameras, computers) participating in the `FGrabbit` framework communicate through a GigE network, separated from the ELSA control system, but accessible through a gateway computer. As camera control and readout can be achieved through various drivers. We favor the `Aravis` [4] driver in combination with GenICam [3] compatible digital cameras. Streams from other drivers (e.g. Video4linux [1]), VNC[2] servers or image generating programs may also be used as input for the `FGrabbit` framework. Comprehensive image calibration can be applied via the OpenCV computer vision library [5] to account for beam profile distortions resulting from tilted screens (extrinsic properties) or lens imperfections of the imaging system (intrinsic properties). Calibrated images are typically cropped to a useful area of interest (AOI) and ideally visualize an undistorted transverse cross section of an observed beam with correct pixel to millimeter scaling. The calibrated images are typically reduced in size, allowing to economically stream images from multiple cameras through the GigE network. The data load from imaging and image analysis is handled by the `RabbitMQ` message broker. Computing power (e.g. for computational expensive 2D profile fits) is distributed to multiple standard computers within the network. A QT-based [6] graphical user interface

* switka@physik.uni-bonn.de

[1] https://www.kernel.org/doc/html/v4.8/media/v4l-drivers/
[2] virtual network computing

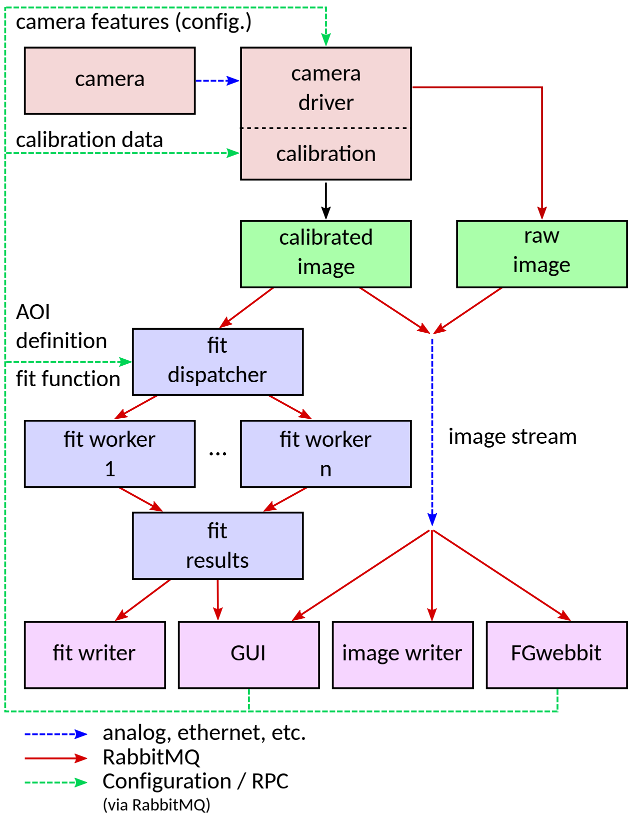


Figure 1: Architecture of the `FGrabbit` framework.

(GUI) allows full-scale user interaction. A web interface allows down-scaled interaction from anywhere via the ELSA website[3].

Performance

A RabbitMQ message is limited to 128 MB, defining the limit for the streamable image size. While the cumulative streaming data rate is limited by the network bandwidth, the computationally expensive fit calculations are outsourced to an arbitrary amount of computers participating in the net- and framework. Computation and network load is conveniently monitored via the RabbitMQ analysis interface. Using standard desktop computers we achieve repetition rates up to 10 Hz of display and beam analysis on a sufficiently large AOI (2D data fit on a hundreds by hundreds large pixel grid).

MEASUREMENT PRECISION

Image Calibration

Beam images taken from e.g. Chromox screens suffer from projection and optical imaging errors which can be classified as intrinsic and extrinsic properties[4]. To obtain a projection-free image without optical errors, a 3×3 projection matrix P can be applied to the uncalibrated image

$$P = K \cdot M, \quad M = H \cdot \vec{t}\,, \tag{1}$$

where K denotes to the intrinsic and M to the extrinsic properties of the individual setup. M is calculated from a rotation matrix H and a translation vector $\vec{t}$. The main influence to the extrinsic uncertainties are image distortions due to the observation perspective, e.g. a screen under a tilt angle of typically 45°. This can be resolved by using a homography matrix as rotation matrix. The matrix entries can be computed from an image taken with a regular pattern of known dimensions (checkerboard or circular patterns consisting of $m \times n$ corners or circles, respectively). Such a calibration image can be temporally installed on the surface of a screen, as demonstrated in Fig. 2. A calibration pattern printed on cardboard with a regular laser printer usually suffices the required image quality. For the calibration we use an OpenCV [5] approach as depicted in Fig. 3. Therewith, a raw image is captured on which an OpenCV detector function finds the position of the corners of the checkerboard (or centers of circles, resp., compare with circular calibration patterns in Appendix). The obtained coordinates are then optimized by performing an additional search for the position using sub-pixel accuracy. Then a coordinate grid that matches the shape of the used pattern is created with a predefined step range. By mapping the found image coordinates into the real world grid the homography matrix can be calculated.

Measurement Error

The radial displacement of the transformed positions to the expected grid is classified as the calibration error. To

[3] https://www-elsa.physik.uni-bonn.de
[4] compare e.g. with https://docs.opencv.org/4.x/

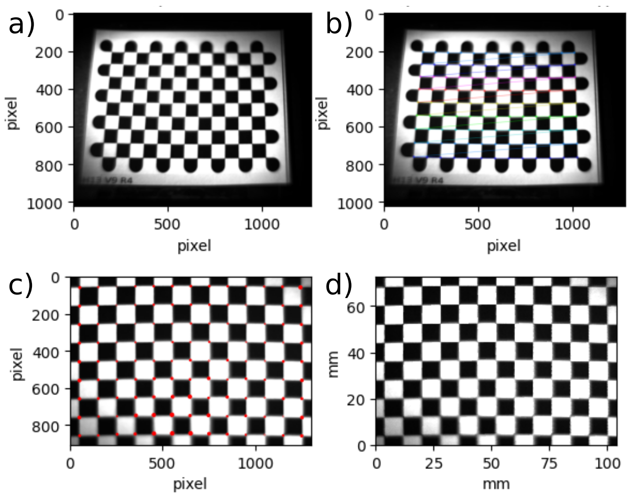


Figure 2: Image calibration using a printed checkerboard pattern on a tilted screen and OpenCV [5] transformation functions: a) original image, b) recognized pattern, c) image after homography transformation including precision errors (red circles), d) scaled and calibrated image.

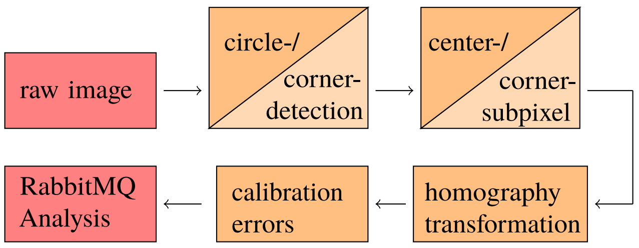


Figure 3: Schematic sketch of the calibration chain. The chain is divided into RabbitMQ framework (red) and OpenCV tasks (orange). The chain can be applied to circular patterns (dark-orange) and checkerboard patterns (light-orange).

minimize statistical fluctuations a sequence of n images is recorded to obtain an averaged radial error, as visualized in Fig. 2 c). Intrinsic properties, such as radial or tangential distortions introduced by the camera lens may be represented by

$$\begin{aligned}
x_{\text{dist,rad}} &= x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)\\
y_{\text{dist,rad}} &= y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)\\
x_{\text{dist,tan}} &= x + \left(2p_1 xz + p_2(r^2 + 2x^2)\right)\\
y_{\text{dist,tan}} &= y + \left(p_1(r^2 + 2y^2) + 2p_2 xy\right)\ ,
\end{aligned}$$

with distortion coefficients $(k_1, k_2, k_3, p_1, p_2)$. OpenCV functions allow to determine these coefficients and to undistort the respective image. The distortion coefficients are obtained from taking multiple pictures of a regular pattern from multiple angles[5]. We find that intrinsic distortions play a neglectable role with usage of industry standard photographic objectives (e.g. $f = 16$ mm) for our monitor setups.

[5] For details see https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html

ISBN: 978-3-95450-236-3 ISSN: 2673-5350 doi:10.18429/JACoW-IBIC2023-WEP046

The calibration error determined from the deviation from the keypoint grid can be interpolated to every pixel, for example using radial basis functions[6], as shown in Fig. 4. Thereon, non-linear weighted fit algorithms can be applied to take local uncertainties into account. For this purpose the GNU scientific library (GSL) [7] back-end of the in-house developed fit library is used. For the unweighted case the common function to be minimized is

$$\Phi(x) = \frac{1}{2}\|f(x)\|^2 = \frac{1}{2}\sum_{i=1}^{n} f_i(x_1,\ldots,x_p)^2,$$

where f is the residual function, n the number of data points (e.g. the number of pixels of the captured image) and p is the number of parameters or the dimension of the fit function. In the weighted case this formula is expanded to

$$\Phi(x) = \frac{1}{2}\sum_{i=1}^{n} w_i \cdot f_i(x_1,\ldots,x_p)^2, \qquad (2)$$

where an individual weight w_i is added for each individual data point. These weights can be utilized for various purposes or represent different effects. Using the obtained inverse calibration variances as weights allows us to suppress potential errors introduced by the calibration process. Furthermore, this allows to mask areas on the screens that exhibit altered behavior to achieve a better fit result. In Fig. 4 a profile captured on a calibrated screen as well as the weighted fit to this data is shown. The inverse variance of the calibration was used as weight, resulting in a profile parameter deviation of up to 2 % comparing weighted and unweighted case.

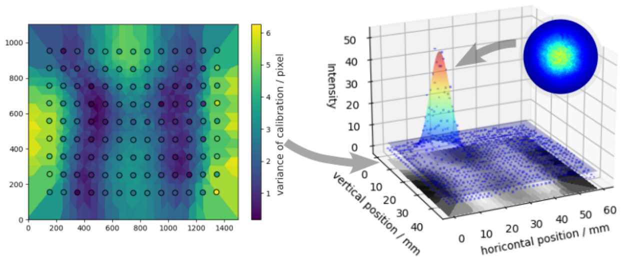


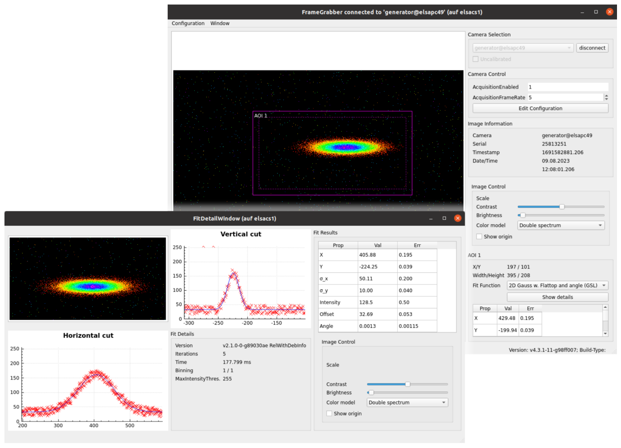
Figure 4: The variance of an exemplary screen calibration with circled nodes and in-between interpolation (left) and a beam profile image on a calibrated screen analyzed with a weighted fit algorithm (right).

FGRABBIT INTERFACES

While RPC[7] commands allow for automated settings and application of the FGrabbit framework, the graphical user interfaces provide practical user interaction and monitoring capability.

Operator GUI

A QT-based [6] graphical user interface allows for convenient control of camera and imaging settings (Fig. 5). The numerous available camera settings are listed and manipulatable, as shown in Fig. 6. Favorite settings can be marked and made available for the main menu. A detail view of the image analysis gives the operator fast and comprehensive feedback of fit parameters and fit quality.



AcquisitionEnabled 1
AcquisitionFrameRate 5

Camera	generator@elsapc49
Serial	25813251
Timestamp	1691582881.206
Date/Time	09.08.2023 12:08:01.206

Scale
Contrast
Brightness
Color model: Double spectrum

AOI 1
X/Y 197 / 101
Width/Height 395 / 208
Fit Function: 2D Gauss w. Flattop and angle (GSL)

Prop	Val	Err
X	429.48	0.195
Y	-199.94	0.039

Version: v4.3.1-11-g98ff007; Build-Type:

Fit Results

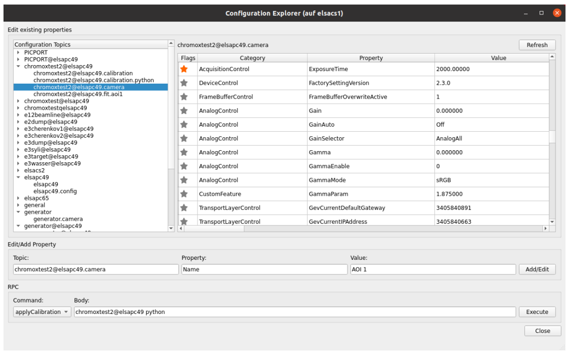
Prop	Val	Err
X	405.88	0.195
Y	-224.25	0.039
σ_x	50.11	0.200
σ_y	10.00	0.040
Intensity	128.5	0.50
Offset	32.69	0.053
Angle	0.0013	0.00115

Fit Details

Version	v2.1.0-0-g89030ae RelWithDebInfo
Iterations	5
Time	177.799 ms
Binning	1 / 1
MaxIntensityThres.	255

Image Control
Scale
Contrast
Brightness
Color model: Double spectrum

Figure 5: FGrabbit graphical user interface with detail view on profile analysis.



Edit existing properties

Configuration Topics
- PICPORT
- PICPORT@elsapc49
- chromoxtest2@elsapc49
  - chromoxtest2@elsapc49.calibration
  - chromoxtest2@elsapc49.calibration.python
  - chromoxtest2@elsapc49.camera
  - chromoxtest2@elsapc49.fit.aoi1
- chromoxtest@elsapc49
- chromoxtestqelsapc49
- e12beamline@elsapc49
- e2dump@elsapc49
- e3cherenkov1@elsapc49
- e3cherenkov2@elsapc49
- e3dump@elsapc49
- e3syli@elsapc49
- e3target@elsapc49
- e3wasser@elsapc49
- elsacs2
- elsapc49
  - elsapc49
  - elsapc49.config
- elsapc65
- general
- generator
  - generator.camera
- generator@elsapc49

chromoxtest2@elsapc49.camera

Flags	Category	Property	Value
★	AcquisitionControl	ExposureTime	2000.00000
★	DeviceControl	FactorySettingVersion	2.3.0
★	FrameBufferControl	FrameBufferOverwriteActive	1
★	AnalogControl	Gain	0.000000
★	AnalogControl	GainAuto	Off
★	AnalogControl	GainSelector	AnalogAll
★	AnalogControl	Gamma	0.000000
★	AnalogControl	GammaEnable	0
★	AnalogControl	GammaMode	sRGB
★	CustomFeature	GammaParam	1.875000
★	TransportLayerControl	GevCurrentDefaultGateway	3405840891
★	TransportLayerControl	GevCurrentIPAddress	3405840663

Edit/Add Property
Topic: chromoxtest2@elsapc49.camera
Property: Name
Value: AOI 1

RPC
Command: applyCalibration
Body: chromoxtest2@elsapc49 python

Figure 6: FGrabbit configuration explorer listing all properties of the digital cameras obtained by the Aravis [4] driver.

FGwebbit

In addition to the Operator GUI a slimmed down Interface is accessible via an in-house developed web application, called FGwebbit. It provides platform independent access to FGrabbit combined with flexible availability via internet and intranet. The responsive architecture (Fig. 7) enables control and views optimized for handheld devices when working on experiments or maintenance tasks in confined spaces. The simplified user interface provides access to live views of multiple camera streams simultaneously. The user interface is complemented by a modular card for each camera. Within a camera card, favorite settings of the attached device are directly available. An expert menu contains all available device parameters for detailed adjustments, accompanied by tooltips for the available settings. Each client can adjust the compression, frame rate, color model and preference for calibrated or uncalibrated data. These settings are saved

[6] e.g. using Python SciPy RBFInterpolator
[7] remote procedure call

and applied only for that particular client, allowing for an optimized application e.g. for mobile use with limited data availability. The web application is prepared to host a global overview of the state of the `FGrabbit` systems, summarizing the status of all associated nodes and devices. Furthermore, the application provides space for the documentation of the `FGrabbit` framework and interfaces.

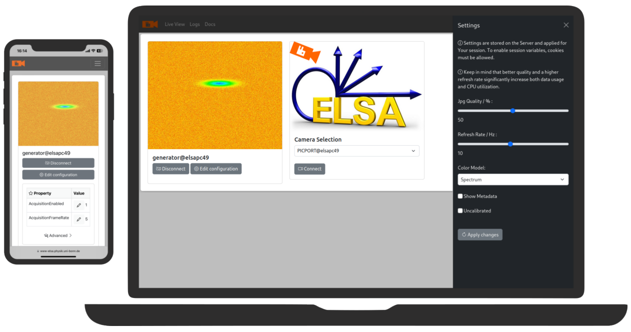


Figure 7: Responsive behavior of the `FGwebbit` interface and active session settings menu.

The `FGwebbit` backend extends the `FGrabbit` `Python` interface [2] and uses the lightweight `Python` web framework `Flask`. To reduce network traffic and data usage in mobile or low bandwidth environments, the live camera feeds are served as compressed JPEG streams. The `OpenCV` library [5] is used for compression and color model application. Further streamlining is achieved by smart handling of TCP connections to the underlying `FGrabbit` framework. Multiple clients and simultaneous streams from or to the same camera device are detected and clustered into a single open TCP connection. This alone significantly improves connection times for additional clients and reduces server load. Unused TCP connections are closed after a predefined timeout, freeing up system resources. For modern design and easy responsive implementation the popular `Bootstrap` frontend toolkit is used, resulting in familiar design elements and usability concepts for `FGwebbit`.

CONCLUSION

Screen calibration for extrinsic properties is the key advantage for beam photometry with monitors installed in environments typical for accelerator beamlines. Calibration errors may be used to improve measurement accuracy using weighted data fit algorithms. The performance of the `FGrabbit` framework is continuously tested and has shown to be robust and scalable. Minor software development is yet carried out to improve usability for long-term accelerator operations.

APPENDIX

Circular Calibration Pattern

Circular calibration patterns are generally more robust (successful) than checkerboard patterns, e.g. when illumination is non-ideal, as shown in Fig. 8.

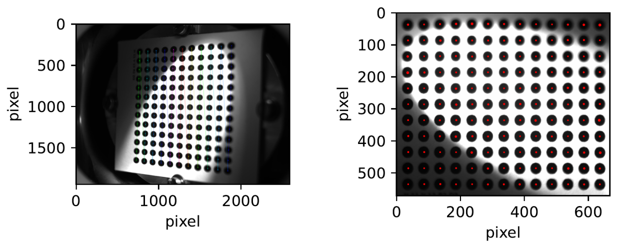


Figure 8: Screen with confined illumination and recognized pattern (left). Calibrated image using circular patterns including precision errors (right).

Virtual Calibration Pattern

Screens with sufficient markings for reference can be calibrated with a virtual pattern, e.g. created with a drawing tool and fed to the calibration algorithm, as shown in Fig. 9.

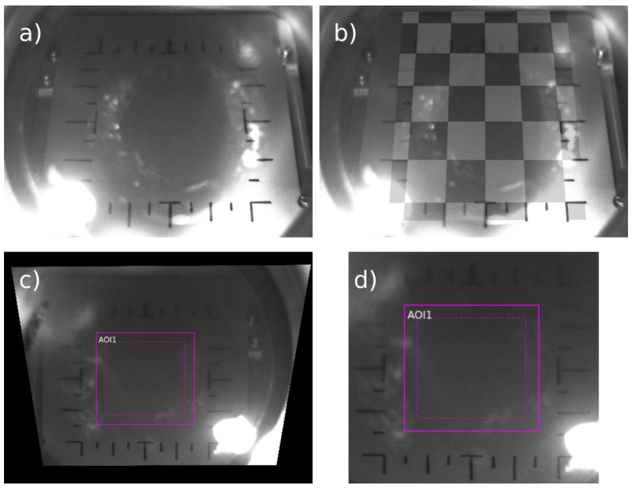


Figure 9: a) Sufficient visible markings may be used to create a b) virtual pattern for c) & d) image calibration.

REFERENCES

[1] W. Hillert *et al.*, "Beam and spin dynamics in the fast ramping storage ring ELSA", *EPJ Web Conf.*, vol. 134, p. 05002, Jan. 2017. doi:10.1051/epjconf/201713405002

[2] D. Proft *et al.*, "Beam profile monitoring and distributed analyis using the RabbitMQ message broker", in *Proc. IBIC'22*, Kraków, Poland, Sept. 2022, pp. 140–143. doi:10.18429/JACoW-IBIC2022-MOP38

[3] European Machine Vision Association, "GenICam Standard", https://www.emva.org/standards-technology/genicam/

[4] A vision library for genicam based cameras, https://github.com/AravisProject/aravis

[5] G. Bradski,"The OpenCV Library", *Dr. Dobb's Journal of Software Tools*, 2000.

[6] The Qt Company, https://www.qt.io/

[7] M. Galassi *et al.*, *GNU Scientific Library Reference Manual*, 3rd ed., ISBN 0954612078, https://www.gnu.org/software/gsl/