SoftwareX 12 (2020) 100621

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

# XISMuS — X-ray fluorescence imaging software for multiple samples

Sergio A. Barcellos Lins *, Boris Bremmers, Giovanni E. Gigante

*Dipartimento di Scienze di Base e Applicate per l'Ingegneria, University of Rome "La Sapienza", Via Antonio Scarpa 14/16, 00161 Rome, Italy*

ARTICLE INFO

*Article history:*
Received 7 August 2020
Received in revised form 3 November 2020
Accepted 4 November 2020

*Keywords:*
X-ray fluorescence
Imaging
MA-XRF
Data visualization

ABSTRACT

X-rays have long been used as a non-destructive analytical technique to investigate artefacts and objects that can be considered cultural heritage. With the unceasing development of technologies and miniaturization of electronics, X-ray Fluorescence (XRF) analysis has undergone a natural evolution, being now extensively used in a bi-dimensional manner, scanning whole surfaces and generating astonishing amounts of data. Evaluating all this data demands a software distribution or at least a stand-alone algorithm. The former being an obvious choice, as software are easy to install and the learning curve is fast. Moreover, developing a proprietary algorithm is time consuming and not always needed for the average user. In this scope, XIMuS was developed as an *ad hoc* software for macro-XRF analysis (MA-XRF), with an intuitive and simple graphical user interface (GUI). The software provides several built-in tools, staple for the interpretation of X-ray fluorescence data, automated data evaluation and some novelty functionalities as data stitching. XISMuS creates an iterative user database as samples are loaded and compiled through usage, facilitating navigation through different datasets. Simultaneous samples can be loaded in separate instances for comparison purposes. Elemental mapping is made simple, the methods available are described and have each one an application for fast, balanced or precise calculations, supporting parallel computing. Ratios between elements in specific regions of the image can be measured while a region-derived spectrum is shown and updated live. Image correlation can be performed supporting the usage of threshold filters and/or region selection. Batch exporting is also available, cross-normalizing the input datasets.



## Code metadata

| | |
|---|---|
| Current code version | v1.3.1 |
| Permanent link to code/repository used for this code version | https://github.com/ElsevierSoftwareX/SOFTX-D-20-00020 |
| Code Ocean compute capsule | – |
| Legal Code License | MIT License |
| Code versioning system used | git |
| Software code languages, tools, and services used | Python; Tcl/Tk; GNU-GCC |
| Compilation requirements, operating environments & dependencies | Microsoft VS Build Tools 2017; xraylib v3.3.0; numpy v1.18.1; numba v0.45.1; llvmlite v0.31; cython; h5py; opencv-python; scipy; psutils; pywin32 (for 32-bit systems); matplotlib |
| If available Link to developer documentation/manual | https://github.com/linssab/XISMuS/blob/master/HELP.md User Guide pdf is distributed together with the software package. |
| Support email for questions | sergio.lins@roma3.infn.it |

## Software metadata

| | |
|---|---|
| Current software version | v1.3.1 |
| Permanent link to executables of this version | https://sourceforge.net/projects/xismus/files/ |
| Legal Software License | MIT License |
| Computing platforms/Operating Systems | Microsoft Windows 7 & 10 |
| Installation requirements & dependencies | xraylib version 3.3.0 (optional) |
| If available, link to user manual - if formally published include a reference to the publication in the reference list | Provided with the installation package |
| Support email for questions | sergio.lins@roma3.infn.it |

* Corresponding author.
*E-mail address:* sergio.lins@roma3.infn.it (S.A Barcellos Lins).

https://doi.org/10.1016/j.softx.2020.100621

## 1. Introduction

In cultural heritage sciences, priority has always been given to non-destructive analysis, and X-rays fluorescence (XRF), being easy to use, reliable and non-destructive, has been the staple for a long time [1,2]. Not over a decade ago, XRF started being used in a scanning fashion, extrapolating its one-dimensionality and yielding information on the mapping of chemical elements. The first attempts were performed into synchrotron facilities [3], where experts were promptly available to create routines for evaluating the new data format obtained. In fact, the first algorithms started being developed shortly afterwards, but were either method-oriented (focused on the application of new analytical approaches, as clustering, for example) and limited to journal publications or not distributed as "straight-out-of-the-box" code or software. Macro-XRF scanning (as it is widely known nowadays), started to become a trend in heritage science, especially for the analysis of paintings such was the amount of information it provided [4] in respect to the traditional XRF spot analysis [5]. This trend fuelled the creation of commercial MA-XRF systems, with their own data analysis software, and concomitantly, a spike in the development of home-made systems [6–8]. Commercially available systems provide a fast and reliable way of obtaining useable images, but it limits the user in terms of developing new methodologies, since the data formatting is often proprietary or encrypted. Moreover, these systems are quite expensive, and, combined with the limitations imposed by the closed-software provided with them, many research groups rather develop their own MA-XRF systems instead.

This leads to another inevitable problem: processing the data acquired with such systems. The analysis became limited to in-house algorithms, which, due to their inherent complexity, are hardly reproducible or simply not distributed as open-source software. Exceptions were made with PyMca, an open-source standalone software for XRF data evaluation [9] and MAPS [10] (now ported to python as PyMAPS). The latter lacking an installer package [11], making it considerably difficult to get working. A third open-source alternative is Datamuncher [12], focused on the processing of large amounts of data and running on IDL Virtual Machines. PyMca, being open-source and continuously updated, adapted itself to the needs of MA-XRF data evaluation and became the standard software for it [7,13–15]. Partially thanks to its GUI, availability of a distribution package and community collaboration.

One important issue that may arise when analysing large MA-XRF datasets is to stitch data together. When dealing with large samples (often the case of paintings), which exceed the maximum analytical area of the instrument or the working hours available per day, one common strategy is to split the sample into several sub-parts. The stitching is then performed, for each elemental map, by placing the images together in a photo editing software or automatically [12,16,17]. This hinders the complete visualization of the ongoing work on a fast pace and may result in the presence of gaps between sub-parts. Besides, the latter option requires some considerable overlapping between datasets (about 20%), greatly increasing the overall data acquisition time. The possibility to stitch data in-situ, almost immediately after the acquisition of a sub-part, is a desirable feature, allowing a better overall understanding of the completed work, as well as helping plan strategies for the upcoming pieces.

With the continuous increase in applications and development of MA-XRF systems in cultural heritage investigations, and the formation of cross-disciplinary professionals, the need for a simple ad hoc tool, capable of performing in-situ data evaluation and which dismisses a deep understanding of the underlying physics principles, emerged. In this scope, XISMuS was developed as an open-source software, provided with an intuitive Graphical User Interface (GUI) and completely dedicated to MA-XRF imaging. This software does not intend by any means to replace the current existing tools, but instead, to provide one extra, simpler alternative to the complex software already available (sometimes extremely hard to get working) while introducing novelty features as for example; data stitching and a constantly updated user database.

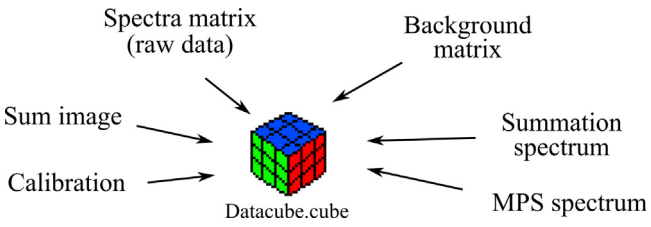


**Fig. 1.** Datacube scheme.

## 2. Software framework and functionalities

The software structure can be very synthetically divided into two groups: the program core and the Mosaic application, used for stitching blocks of data. Each group will be discussed in the following sections separately, highlighting their main functionalities and few of the underlying methods implemented.

### 2.1. Core program

One of the novelty features introduced by XISMuS is the capability of having a local database, making it easy to change between different samples or compare them. This is achieved by creating a datacube (*.cube) file containing all relevant information about the dataset (Fig. 1) whenever new data is compiled.

As it may be assumed from Fig. 1, most of the relevant features used for MA-XRF data interpretation are available and can be readily accessed. The derived spectra, namely maximum pixel spectrum (MPS) [18] and summation spectrum, are calculated automatically. As for the background contribution, it is calculated according to the input method chosen by the user when loading the data. So far, the Statistical Non-Linear Iterative Peak Clipping (SNIPBG) and orthogonal polynomial methods [19] are implemented and available, the former allowing the customization of its parameters and the latter being better suited for the *auto-wizard* method (described in 2.1.3). The background contribution is stored as an independent matrix, in a way that each pixel-spectrum$_{(x,y)}$ has a corresponding background$_{(x,y)}$, where x and y are the pixel spatial coordinates. This choice is not the best memory-wise, but it avoids having to re-calculate thousands of arrays while maintaining the original data. Moreover, as all data is synthetically stored in a datacube object, it is possible to easily implement new data analysis routines and background calculation methods if needed, by either adding them to the datacube object class or loading the datacube object into a custom script.

The energy calibration is either set manually or extracted from the *.mca files header (when available). The calibration is performed by applying a linear regression fit [20] to the channel–energy pairs given. This step is fundamental for a proper extraction of elemental distribution maps, as they rely on the energy axis rather than channels.

As XISMuS is still in its early versions (v1.x.x), only *.mca, *.txt, and, more recently *.h5 files are supported. However, support for

more file types, as *.edf, is expected to be implemented as the software grows.

The software was conceived as part of a scanner development project, and therefore, does not support $z$-axis corrections (the scanning system does not have autofocus on the z-axis) nor considers the motors x–y positions. In the case of mobile or portable instruments, when samples are larger than the analytical area of the instrument, it is necessary to drag the whole system (or sample), to a new starting position. In this way, the motors relative position to the sample lose its reference point, and the recorded x–y positions of the instrument are of no use.

Elemental distribution maps can be obtained by different methods, of which three are currently available:

*2.1.1. Simple roi*

In this method, the chemical element polled has its fluorescence lines extracted from either xraylib [21] or from the internal library, depending whether xraylib is available or not in the host machine. Xraylib has the advantage of being constantly updated and, therefore, contains more precise information. The macro lines (K$\alpha$ and K$\beta$ or L$\alpha$ and L$\beta$) are picked according to the element's atomic number (Z), *e.g.* for Z between 1 and 54 (included), K-lines are picked; while for Z above 54, L-lines are picked. This threshold was chosen based on the typical energy range of MA-XRF analysis, where the X-ray tube works at about 35–40 kV. In any case, the user can always input a custom energy range to map specific regions of interest through the GUI if the previous criterion is insufficient. This limit can also be easily changed in the code itself if needed.

The region of interest (ROI) used for mapping the polled chemical element is calculated based on the element's peaks full-width at half maximum (FWHM). The peak's width (assuming it follows a Gaussian distribution) is related to the function spread in number of channels as: $FWHM = 2.3548 \cdot \sigma$. The standard deviation $\sigma$ can be defined as a function of the energy in eV as [9,19]:

$$\sigma(E_{line}) = \sqrt{\left(\frac{NOISE}{2.3548}\right)^2 + (w_{Si} * FANO * E_{line})} \quad (1)$$

where NOISE is the electronic noise contribution to the peak's width, FANO is the Fano factor for silicon, $w_{Si}$ is the energy in eV required to produce an electron–hole pair in silicon ($\approx$3.85 eV) and $E_{line}$ is the element's fluorescence line energy in eV.

The observed peak centre will not always precisely match the theoretical values, for this reason, a search for a local maximum in a given window must be performed. The search method implemented reads the spectrum and checks if a maximum is found within $\pm[(3 + W) \cdot gain]$ from the theoretical value; where gain is the calibration gain in eV and W is a tolerance factor that can be changed by the user in the *Settings* menu (defaults are: 3 for energies below 4800 eV, 3 for energies above 12 000 eV and 4 for the others). If a maximum is found, the peak centre is updated. For *simple-roi* method, this search is performed only once over the summation spectrum. The continuum is not taken in consideration in this step.

Finally, the ROI is set as an interval between $\pm(2 \cdot FWHM)$, centred at the peak's updated centre, for both alpha and beta lines. To generate the images, the spectra and background matrices are sliced between ROI's lower and upper indexes and subtracted from one another. The calculated areas for each pixel, together with their ratio, are then recorded in a separate text file. The slicing has a clear disadvantage when dealing with overlapping peaks, as no deconvolution is performed and thus, neighbouring peaks cannot be separated. Therefore, this method is only advised for clearly separated peaks. The generated ROI can be checked by accessing the Toolbox dropdown menu, so the user can assess the presence of any signal contamination from a neighbouring peak and assure the results quality.

*2.1.2. Auto roi*

In *auto-roi* method, the fluorescence lines energies and ROI are picked in the same way as in *simple-roi* method. The difference relies in the automatic detection of an element's peak in each individual pixel-spectrum (*i.e.*, the ROI selection is performed for each pixel-spectrum), combined with data filtering to enhance peak detection and secondary criteria. First, the spectrum is smoothed with a 3rd order Savitzky–Golay filter to improve peak detection. Then, a local maximum within the pre-established ROI is searched (to assure the peak is separated and centred). Lastly, signal-to-noise ratio criteria [18] and second differential curve are checked. If a peak is identified, its area is calculated over the raw, non-smoothed spectrum. This method requires a considerably longer processing time in comparison to *simple-roi*, but, in most cases, yields higher contrast and more reliable images.

To accelerate the processing time when polling several elements with *auto-roi* method, multiprocessing option is available, creating one process for each element polled according to available random-access memory (RAM) and physical cores. If the number of elements exceeds the number of cores available, the remaining elements are queued. The most recent version of Python supported by Miniconda3 is 3.7.x, where multiprocessing module must copy the resources needed for each process to memory, thus increasing the amount of RAM required to perform the whole operation. Python 3.8 already supports shared memory between processes with multiprocessing. Nonetheless, the code had to be compiled in a Miniconda3 environment due to incompatibility issues with packages.

*2.1.3. Auto wizard*

This method, differently from the previous ones, fits a series of identified peaks in each spectrum with a Gaussian curve, calculating the peaks areas. The values obtained are, therefore, an approximation but with the great advantage of resolving overlapping peaks (a common issue in cultural heritage materials, especially paintings). The peaks areas are calculated by fitting the following equation:

$$f(peaks) = \left[\sum_{p}^{peaks}\sum_{i}^{nchan} \frac{A_p * gain}{\sqrt{2\pi}\sigma_p} \cdot exp\left(-\frac{\left(E(i) - E_p\right)^2}{2{\sigma_p}^2}\right)\right] + continuum \quad (2)$$

where $E(i)$ is a function that correlates each channel to an energy value, *peaks* is an array containing the energies of the matched peaks in electronvolts, and $A_p$, $E_p$, and $\sigma_p$ are the amplitude, energy, and standard deviation of peak $p$, respectively.

The standard deviation $\sigma$, described in Eq. (1), takes FANO and NOISE into consideration. They are globally fitted for the entire energy range through the summation spectrum and then stored as attributes in the datacube when it is packed for the first time. The fitting engine is capable of fitting both FANO and NOISE for each pixel-spectrum. However, this takes a considerable amount of processing time. The best approach is to use the global values, reducing the fitting variables and speeding up the calculation.

To further shorten the processing time, *auto-wizard* can run in multiprocessing mode, splitting the datacube matrices into chunks according to available physical cores and dispatching the jobs sequentially. The processes, in this case, run asynchronously. Each process saves an *.npy frame file in the end of its execution, in a manner that the data can be re-arranged, building the images after the batch is done processing.

The *auto-wizard* method can be summarized in three main steps: prepare data, automatically find and match peaks, and fit data. In the first step, the datacube spectra matrix is smoothed with a 3rd order Savitzky–Golay filter and the continuum recalculated for the whole dataset. A second summation spectrum

and corresponding continuum are calculated, to be used in the peak-finding algorithm, where each added spectrum has its values below 1 replaced by 1. By doing so, noise from individual spectra is cut-off, avoiding the mis-identification of peaks.

Following, the data is fed to the peak-finding algorithm, which selects and matches the peaks. The method applies a convolution algorithm [22] to the summation derived spectrum, then, its continuum and convolution variance are used as criteria to select which peaks to fit. The identified peaks (having each one being already attributed to a chemical element) are then re-checked by verifying their distance from the theoretical peak centres. Finally, through the GUI, the user is prompted to add more elements to the fit poll if needed, and start the fit process. The datacube remains unaltered by the whole procedure.

Gaussian fit and peak-finding parameters can be modified by the user through the GUI. For the Gaussian fit parameters, the user can change the maximum number of fit cycles per spectrum. Whereas for peak-finding, there are three modifiable parameters, labelled in the *Settings* panel as sensitivity, suppression factor and tolerance. The former sets the peak convolution window (defaults to 9 channels) – smaller windows result in more peaks detected. Suppression factor changes the continuum magnification, used to suppress mis-identified peaks through the convolution (due to data noise or an excessively small window), and the latter is the channel tolerance of a peak from its theoretical energy. If an identified peak, attributed to a chemical element, is far off its corresponding element theoretical peak energy, it is removed from the poll.

Further options include the possibility to save a graphical plot of the last fitted spectrum every n spectra, and, to the author's knowledge, the present software is the first one to easily and promptly provide it for MA-XRF data evaluation, giving the user yet another control to assess results consistency and quality.

Comparisons between the available methods are shown in Fig. 2. *Simple roi* and *auto-roi* methods often create very similar images. However, *auto-roi* images present a higher contrast due to the constant peak centring (Fig. 2a). The advantages of *auto-wizard* can be clearly seen in Fig. 2b, as a significant peak overlapping between gold and mercury exists. For the mercury images, the one obtained with *simple-roi* contains a significant portion of gold signal, therefore representing the pattern of Au rather than that of Hg. In this case, the only way to obtain a proper image is by using *auto-wizard* method.

Selecting one method over the other is more intuitive than it seems. The user can verify the derived spectra, and, from there, decide the best approach by clicking on the elements shown in the Periodic Table window, checking their theoretical positions in the spectra.

Regardless of the method chosen, the datacube only holds one image for each element and Siegbahn notation line (K$\alpha$, K$\beta$ or L$\alpha$, L$\beta$), *e.g.*, the element iron can only have two distinct images, one for its K$\alpha$ line and one for K$\beta$. If any element is remapped with a different method, that element output in the datacube will be replaced by the last acquired image. All images stored in the datacube can be viewed, manipulated, and analysed by accessing the *Image Analyzer* module, which offers few filtering options, area analysis and correlation scatter plots.

### 2.2. Mosaic application

Stitching pieces of data in a MA-XRF perspective has become an issue only when larger samples started being analysed, exceeding the maximum analytical area of the instrumentation, or the working hours available per day. XISMuS offers a dedicated module to stitch together two or more datasets via a simple GUI. The Mosaic module only recognizes XISMuS datacube files, and, therefore, any data to be stitched must be first compiled within the core program. Canvases states can be saved and loaded, to promote inter-lab exchange and the possibility to "re-stitch" the same datasets changing few parameters if needed. Manually modified histograms are saved as well, and loaded only if the corresponding file is available; enforcing backwards compatibility with previous versions of the software, where manual histogram manipulation was not yet available.

Where the module's GUI implementation is concerned, displaying the canvas properly was a great challenge. Dragging the datasets (layers) on canvas, rotating, and moving their priority up or down, demand that the layers position and screen are updated. The majority of functions used where written in Cython, which conveniently converts native Python code into a C++ counterpart stored in a local library. This allows to achieve relatively fast refresh rates when performing such operations. Moreover, the refresh area is limited, requiring only the modified region to be recalculated. Fig. 3 shows how the canvas refreshing and how the render region (final datacube delimiter) are selected.

One of the main problems when stitching different datasets, is to ensure that the data is comparable, *i.e.*, that the different datacubes present compatible counts per pixel and calibration. While re-calibration matching have not been yet solved in the current version, each datacube can be modified to achieve a reasonable visual match. In this regard, three algorithms have been implemented: linear contrast stretching (i), average counts-per-pixel (ii), and histogram matching (iii) scaling. The former method is based on the linear contrast stretching algorithm, used in image processing. The second, instead, simply scales up each datacube by a constant factor, matching the highest average counts-per-pixel found within all datacubes loaded in the Mosaic canvas. Finally, the latter method matches the histograms of the individual datacubes sum-maps to that of the larger datacube found on canvas. Methods (ii) and (iii) improve the output when datacubes with high contrast are loaded, enhancing darker and brighter regions.

Previous versions of XISMuS would scale the raw data. This, however, requires more resources from the system and can be relatively slow. The most recent version of the software, described in this manuscript, creates a scaling matrix (mask) that is applied to the output images individually rather than to the raw data itself.

The scaling mask can be toggled on and off by the user through the *Image Analyzer* window, and the masks created by the different scaling algorithms can be viewed under the corresponding output folder.

A comparison between the outputs obtained with a twelve-piece dataset, from a canvas painting, with the different scaling algorithms is shown in Fig. 4. In the images, it is possible to observe that most of the datacubes achieved a good visual match within the final result. However, when using automated algorithms, as one datacube is always taken as a reference point, one sub-part appears off. Datacubes 3–3, 2–1, and 3–2 (in Fig. 4b, c, and d, respectively) seem darker or brighter, be it because they were taken as a reference or because of an inherently low dynamic range. If the automated algorithm chosen fails, it is possible to manually adjust each datacube histogram, achieving a best visual match (Fig. 4f).

## 3. Results and examples

The results presented herein were obtained from the training data provided alongside the software, so that the reader can easily replicate the outputs presented. The datasets provided were obtained with a prototype MA-XRF scanner, using a tungsten-anode mini X-ray tube from AMPTEK® operating at 40 kV and 100 μA

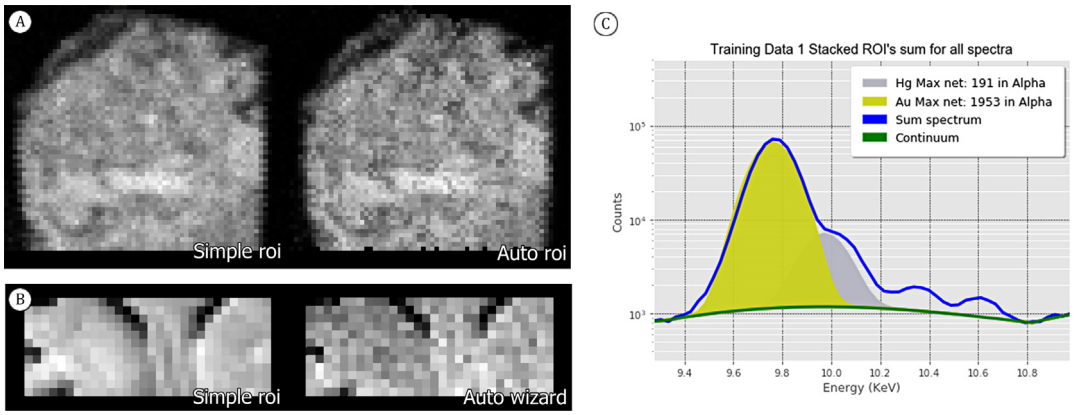


**Fig. 2.** *Simple-* and *auto-roi* images obtained for Fe-K$\alpha$ (a), *simple-roi* and *auto-wizard* images for Hg-L$\alpha$ (b), Au-Hg deconvolution obtained with *auto-wizard* method for the Hg-L$\alpha$ image shown in b (c).

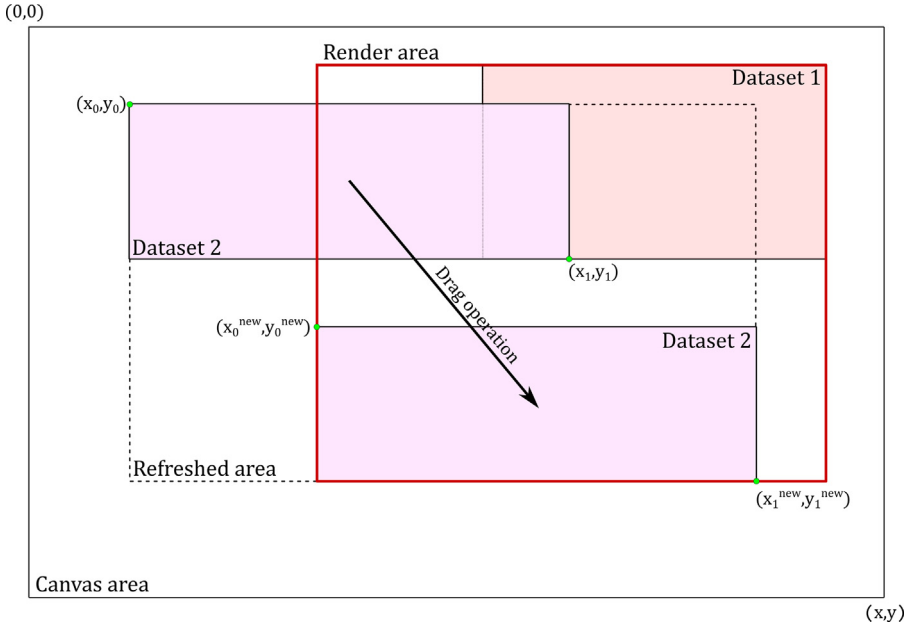


**Fig. 3.** Schematic representation of Mosaic module functioning.

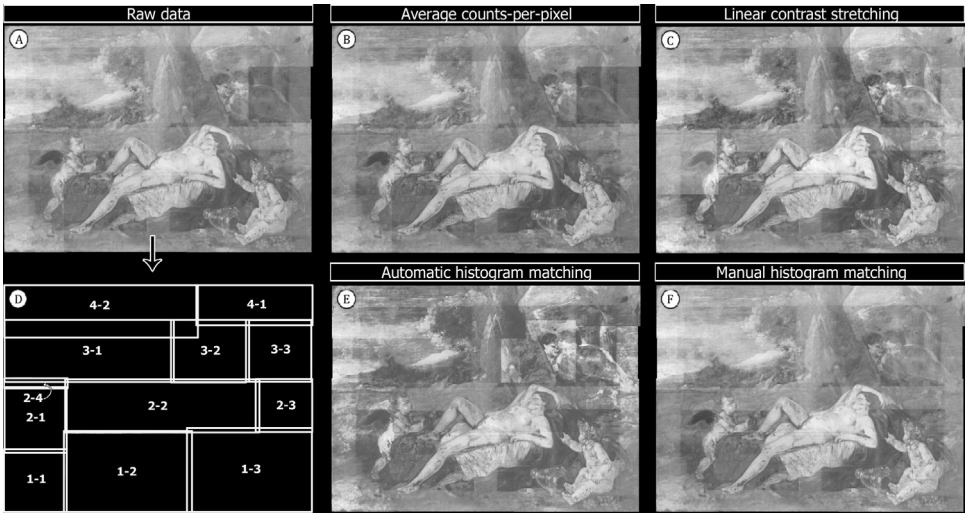


**Fig. 4.** Comparison between the different scaling methods: datacubes as loaded on canvas (a), average-counts-per pixel (b) and linear contrast stretching (c) outputs, datacubes overlapping scheme (d), automatic histogram matching (e) and manual histogram matching (f) outputs.

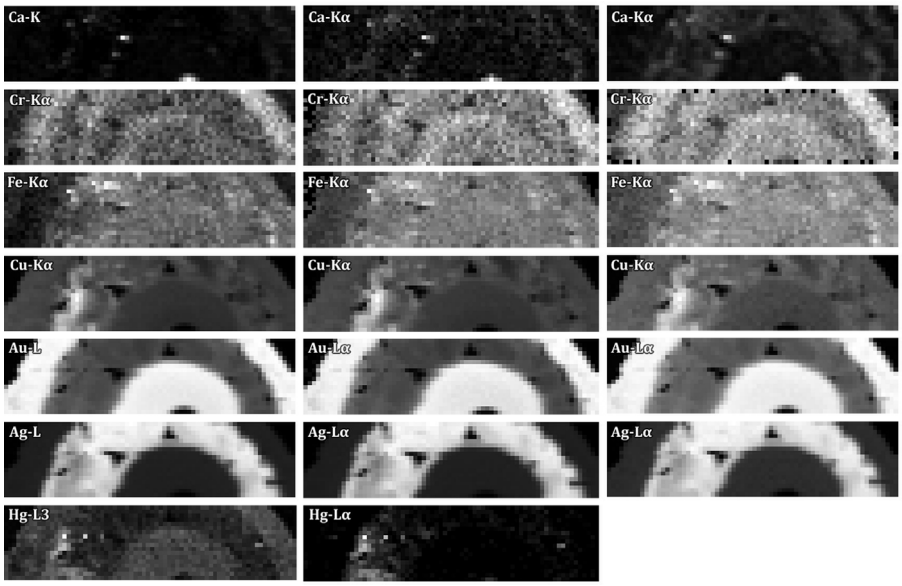


**Fig. 5.** PyMca XRF batch fitting images (left), XISMuS *auto-wizard* (middle) and *auto-roi* images (right).

and an x–y translation stage. The scanning head was fitted with the afore-mentioned tube, collimated to 1 mm and unfiltered, and one AMPTEK® 123-SDD detector. Dwell-time was different for each sample, with the overall acquisition time ranging from 2 to 3 h per dataset. The translation stage step-resolution was set to 1 mm. The data was extracted from ornamental nose-rings from pre-Columbian Peru, made of a gold-silver-copper (*tumbaga*) alloy.

As XISMuS does not intend to replace any existing software, but instead to provide one simpler and *ad hoc* alternative to MA-XRF system developers and end-users, the comparisons presented are solely with the purpose of assessing the results reliability and quality.

### 3.1. Elemental maps

Elemental distribution maps generated with XISMuS were compared with those generated with a standard open-source software for MA-XRF imaging: PyMca. All results were obtained using the same computer system. The system specs are as follows: CPU: Intel Xeon E-2176M @2.70 GHz (with active overclock up to 4.10 GHz), 16 Gb RAM SoDIMM ECC memory and Windows 10 Pro OS.

By using PyMca's ROI imaging tool and XISMuS *simple-roi* method, the outputs were nearly identical. Major differences can be summarized in usability; being XISMuS quicker to load the data and obtain readily useable images of several chemical elements at once. Probably due to the adaptive multi-threading file reading process implemented.

Fig. 5 shows a comparison between images generated with PyMca's XRF batch fitting tool and XISMuS *auto-roi* and *auto-wizard* methods. XISMuS *auto-wizard* and PyMca could resolve peaks and produce cleaner images where peak overlapping was encountered almost identically. The mean squared error (MSE) and structural similarity index (SSIM) scores, between the obtained images, are shown in Table 1.

As the fitting engine built into each software is different, few images, as those of calcium and mercury, may appear slightly dissimilar but still maintaining a reasonable degree of similitude (Table 1). XISMuS disregards the time-consuming and sometimes overly complex fit configuration phase present in the standard software, being almost entirely automatized.

**Table 1**
Image comparison scores - PyMca XRF batch fitting vs. XISMuS.

| Map | MSE | SSIM | MSE | SSIM | MSE | SSIM |
|---|---|---|---|---|---|---|
| | *vs. auto_wizard (polynomial BG)* | | *vs. auto_wizard (SNIPBG)* | | *vs. auto_roi (SNIPBG)* | |
| Ca | 278.42 | 0.52 | 142.70 | 0.60 | 242.97 | 0.59 |
| Cr | 945.25 | 0.80 | 588.12 | 0.80 | 871.29 | 0.83 |
| Fe | 607.60 | 0.83 | 290.26 | 0.84 | 92.83 | 0.91 |
| Cu | 19.51 | 0.99 | 12.23 | 0.99 | 2.94 | 1.00 |
| Au | 269.58 | 0.99 | 49.28 | 0.99 | 7.64 | 0.99 |
| Ag | 1.45 | 1.00 | 1.58 | 1.00 | 1.25 | 1.00 |
| Hg | 2012.68 | 0.45 | 2503.09 | 0.20 | – | |

Mercury image is the most different one, due to the fact that the image produced with PyMca represents the Hg-L3 line while that made with XISMuS was created using Hg-L$\alpha$. When summing Hg-L$\alpha$ and Hg-L$\beta$ images generated with XISMuS, the MSE and SSIM values, in comparison to PyMca Hg-L3 image, change to 850.03 and 0.72 respectively, pointing that the images are much more similar in this case.

In what regards processing time, *auto-wizard*, *auto-roi* (shown in Fig. 5), and *simple-roi* images, for Training Data 2 (18 x 70 *pixels*$^2$) were generated in 101.39, 6.65 and 1.28 s, respectively. Considerably larger datasets, of 989 x 724 (Fig. 4) and 548 x 796 *pixels*$^2$, were processed with *auto-wizard* method (after being merged through the Mosaic module) in 9 h and 6 h, respectively. With *simple-roi*, this time is shortened to 147 and 119 s.

### 3.2. Data stitching

Mosaic tool is a new and innovative feature provided by XISMuS, allowing the user to stitch blocks of data, subsequentially used to create elemental distribution images and compare and study them, quickly and in-situ. The blocks (datacubes) are moved around a 2D canvas, just like in a standard photo editing software, and compiled into a new datacube.

To illustrate this functionality, the example data provided with the software was stitched together and processed. The results are shown in Fig. 6. The tool was tested in the same computer system described before, used for obtaining the elemental distribution images. Larger datasets were also tested with the Mosaic tool, as

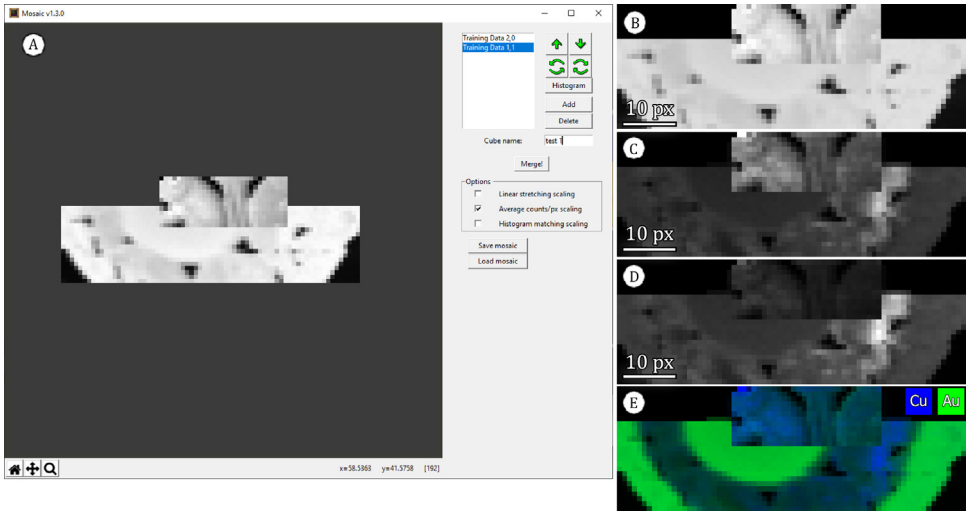


**Fig. 6.** Mosaic module canvas with two datacubes loaded (a); stitched datacube images: summation map (b); copper map with scaling mask *on* (c), copper map with scaling mask *off* (d), and combined map of gold and copper with scaling mask *on* (e).

the one shown in Fig. 4, having a dimension of 989 x 724 *pixels*$^2$ and being merged in about 6 min.

Training Data 1 and 2 were configured and then compiled into datacubes, using the default SNIPBG parameters, *from source* calibration option and *auto-roi* method. Successively, a 100 x 100 canvas was created in the Mosaic module, where both datacubes were loaded and positioned in the canvas accordingly. Rendering priority is given to the last datacube in the list (Fig. 6a). As for the scaling method used, average counts-per-pixel scaling option was ticked. After stitched, copper and gold maps were generated for the newly created datacube.

The scaling methods allow a better and proper visualization of different combined data, by intensifying the signal from datacubes with less counts/pixel than the global data average. In the example given, datacube 1 (Fig. 6a - upper rectangle), representing Training Data 1, has an average of 10 600 counts/pixel, while datacube 2 (bottom) has about 36 900 counts/pixel. For a same-element distribution image, of an element present in both datasets and in similar quantities, datacube 1 signal would be suppressed and barely visible if no scaling factors were introduced (Fig. 6d). Therefore, the use of a scaling method can prove extremely useful for direct comparisons between different datasets as well.

The Mosaic module was tested in-situ during the acquisition of the dataset shown in Fig. 4, by stitching pieces of data immediately after their completion. This approach has proven incredibly helpful, providing a better overview of the scanning progress, the possibility to verify if any pieces were missing (gaps) and to plan the strategy for scanning the next pieces.

## 4. Impact

The software herein presented tackles the described problems by offering an iterative user database (updated through usage), different data extraction algorithms, a dedicated data stitching module and automated data analysis. All within a very simple GUI. These features combined give the users the possibility to easily navigate through different samples, cross-normalize and compare them and to quickly extract data.

For large samples, as it is often the case in paintings case-studies, where the sample is sub-divided into several parts, XISMuS proves to be extremely useful. It provides means to quickly merge several sub-part datacubes into a larger file for a complete visualization of the ongoing data acquisition process.

XISMuS's first build was released on 12 March 2020, and couple results obtained are thoroughly discussed and available in literature [23–25].

## 5. Conclusions

The software herein discussed and presented, demonstrated to serve as an easy-to-use and viable alternative, comparable to the standard software currently available. XIMuS UI, with an efficient error handling, and easiness to use makes it almost unique, requiring a minimal user input but still offering the customization of few data analysis parameters. Furthermore, to the authors knowledge, it is the only open source software available, with a distribution package, that can stitch MA-XRF datacubes together.

The possibility to stitch several datasets (instead of images), in a similar way as one would do in a photo editing software, demonstrated to be the major strength of XISMuS. Stitching data in-situ has proven incredibly helpful to have a better overview of the scanning progress, verify any gaps between sub-parts and plan the scanning strategies for the upcoming parts. Future implementations will aim in converting *.cube files to more standardized file formats, promoting cross-compatibility between different software, as well as supporting more file formats as the need comes.

## Funding

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 766311.

## CRediT authorship contribution statement

**Sergio A. Barcellos Lins:** Conceptualization, Methodology, Software, Validation, Formal analysis, Writing - original draft, Read and agreed to the published version of the manuscript. **Boris Bremmers:** Methodology, Software, Formal analysis, Read and agreed to the published version of the manuscript. **Giovanni E. Gigante:** Writing - review & editing, Data curation, Conceptualization, Methodology, Read and agreed to the published version of the manuscript.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgements

The authors would like to kindly thank Prof. Dr. Roberto Cesareo for providing the datasets used for comparison studies and distributed as training data together with the software. Furthermore, the authors acknowledge and thank Dr. Stefano Ridolfi, for providing a foundation on which the software could be developed, and Prof. Dr. Antonio Brunetti for the helpful comments and support given during the software's development.

## References

[1] Guerra MF. The study of the characterisation and provenance of coins and other metalwork using XRF, PIXE and Activation Analysis. Radiation in art and archeometry. Elsevier; 2000, p. 378–416. http://dx.doi.org/10.1016/B978-044450487-6/50063-8.

[2] Karydas A, Brecoulaki X, Pantazis T, Aloupi E, Argyropoulos V, Kotzamani D, et al. Importance of in-situ EDXRF measurements in the preservation and conservation of material culture. X-rays for archaeology. Springer; 2005, p. 27–53. http://dx.doi.org/10.1007/1-4020-3581-0_2.

[3] Dik J, Janssens K, Van Der Snickt G, Van Der Loeff L, Rickers K, Cotte M. Visualization of a lost painting by Vincent van Gogh using synchrotron radiation based X-ray fluorescence elemental mapping. Anal Chem 2008;80(16):6436–42. http://dx.doi.org/10.1021/ac800965g.

[4] Ravaud E, Pichon L, Laval E, Gonzalez V, Eveno M, Calligaro T. Development of a versatile XRF scanner for the elemental imaging of paintworks. Appl Phys A 2016;122(1):1–7. http://dx.doi.org/10.1007/s00339-015-9522-4.

[5] Cesareo R. Non-destructive EDXRF-analysis of the golden haloes of giotto's frescos in the Chapel of the Scrovegni in Padua. Nucl Instrum Methods Phys Res B 2003;211(1):133–7. http://dx.doi.org/10.1016/S0168-583X(03)01165-0.

[6] Trojek T, Prokeš R, Šefců R, Bilavčíková H, Čechák T. Confocal X-ray fluorescence spectrometer for in-situ analyses of paintings. Radiat Phys Chem 2017;137:238–42. http://dx.doi.org/10.1016/j.radphyschem.2016.02.031.

[7] Campos PH, Appoloni CR, Rizzutto MA, Leite AR, Assis RF, Santos HC, et al. A low-cost portable system for elemental mapping by XRF aiming in situ analyses. Appl Radiat Isot 2019;152:78–85. http://dx.doi.org/10.1016/j.apradiso.2019.06.018.

[8] Ruberto C, Mazzinghi A, Massi M, Castelli L, Czelusniak C, Palla L, et al. Imaging study of Raffaello's "La Muta" by a portable XRF spectrometer. Microchem J 2016;126:63–9. http://dx.doi.org/10.1016/j.microc.2015.11.037.

[9] Solé VA, Papillon E, Cotte M, Walter P, Susini J. A multiplatform code for the analysis of energy-dispersive X-ray fluorescence spectra. Spectrochimica Acta B 2007;62(1):63–8. http://dx.doi.org/10.1016/j.sab.2006.12.002.

[10] Vogt S. MAPS : A set of software tools for analysis and visualization of 3D X-ray fluorescence data sets. J Phys IV (Proc) 2003;104:635–8. http://dx.doi.org/10.1051/jp4:20030160.

[11] Maps github repository. 2020, https://github.com/MapsPy/MapsPy.

[12] Alfeld M, Janssens K. Strategies for processing mega-pixel X-ray fluorescence hyperspectral data: A case study on a version of Caravaggio's painting Supper at Emmaus. J Anal At Spectrom 2015;30(3):777–89. http://dx.doi.org/10.1039/c4ja00387j.

[13] Hocquet F-p, Calvo H, Xicotencatl AC, Micha E, Strivay D. Elemental 2D imaging of paintings with a mobile EDXRF system. Anal Bioanal Chem 2011;399(9):3109–16. http://dx.doi.org/10.1007/s00216-010-4281-8.

[14] Eveno M, Ravaud E, Calligaro T, Pichon L, Laval E. The Louvre Crucifix by Giotto - Unveiling the original decoration by 2d-XRF, X-ray radiography, Emissiography and SEM-EDX analysis. Herit Sci 2014;2(1):1–9. http://dx.doi.org/10.1186/s40494-014-0017-y.

[15] Zeming D, Jun L, Qili J, Qiuli P, Lin C. A type of portable micro-EDXRF spectrometer with laser displacement sensor. Nucl Instrum Methods Phys Res B 2019;442(January):13–8. http://dx.doi.org/10.1016/j.nimb.2019.01.010.

[16] Sciutto G, Frizzi T, Catelli E, Aresi N, Prati S, Alberti R, et al. From macro to micro: An advanced macro X-ray fluorescence (MA-XRF) imaging approach for the study of painted surfaces. Microchem J 2018;137:277–84. http://dx.doi.org/10.1016/j.microc.2017.11.003.

[17] Saverwyns S, Currie C, Lamas-Delgado E. Macro X-ray fluorescence scanning (MA-XRF) as tool in the authentication of paintings. Microchem J 2018;137:139–47. http://dx.doi.org/10.1016/j.microc.2017.10.008.

[18] Bright DS, Newbury DE. Maximum pixel spectrum: A new tool for detecting and recovering rare, unanticipated features from spectrum image data cubes. J Microsc 2004;216(2):186–93. http://dx.doi.org/10.1111/j.0022-2720.2004.01412.x.

[19] Van Grieken RE, Markowicz AA. Handbook of X-ray spectrometry. 2nd ed.. Marcel Dekker, Inc.; 2002, p. 983.

[20] Bevington PR, Robinson DK, Blair JM, Mallinckrodt AJ, McKay S. Data reduction and error analysis for the physical sciences. Computers in physics. 3rd ed.. New York, NY: McGraw-Hill Higher Education; 2003.

[21] Brunetti A, Sanchez Del Rio M, Golosio B, Simionovici A, Somogyi A. A library for X-ray-matter interaction cross sections for X-ray fluorescence applications. Spectrochimica Acta B 2004;59(10–11):1725–31. http://dx.doi.org/10.1016/j.sab.2004.03.014.

[22] Van Espen P, Lemberge P. Ed-Xrf spectrum evaluation and quantitative analysis using multivariate and nonlinear techniques. Adv X-Ray Anal 2000;43(c):560–9.

[23] Iorio M, Graziani V, Lins S, Ridolfi S, Branchini P, Fabbri A, et al. Exploring manufacturing process and degradation products of gilt and painted leather. Appl Sci 2019;9(15). http://dx.doi.org/10.3390/app9153016.

[24] Barcellos Lins SA, Gigante GE, Cesareo R, Ridolfi S, Brunetti A. Testing the accuracy of the calculation of gold leaf thickness by MC simulations and MA-XRF scanning. Appl Sci 2020;10(10):3582. http://dx.doi.org/10.3390/app10103582.

[25] Barcellos Lins SA, Ridolfi S, Gigante GE, Cesareo R, Albini M, Riccucci C, et al. Differential X-Ray attenuation in MA-XRF analysis for a non-invasive determination of gilding thickness. Front Chem 2020;8(March):1–9. http://dx.doi.org/10.3389/fchem.2020.00175.