ToraBaC – TORATOM batch correction tool

doc. Ing. Michal Vopálenský, Ph.D.

Institute of Theoretical and Applied Mechanics of the Czech Academy of Sciences, v.v.i.

Centre Telč

Telč, Nov 2020

This work has been financially supported by the project of the Ministry of Culture of the Czech Republic: Mobile device devoted to imaging and analysis of layered paintings and the polychromy of the works of old art (DG18P02OVV006).

1. Motivation

During computed tomography (CT), a set of images is taken from different angles, covering usually one full rotation (360°). These images will be called *projections* in the further text. The set typically consists of hundreds to thousands of projections. The projections usually contain high image noise, caused by the non-uniform response of individual pixels. The intensity profile of the beam causes gradients of the background and there is very often considerable offset signal from the detector dark currents. All these problems can be overcome by making the basic correction of the projections – the so-called flat field correction. Dark field (DF) and open beam (OB) images are needed for this projection. DF image is acquired from the detector without illumination, OB image is acquired when the same settings of the beam, the geometry and of the acquisition are applied, but no object is in the imaging path. It is recommended to acquire a set of DF and OB images and make a „master" DF and OB image as the median or average of the DF and OB set.

If the intensity in of the monochromatic radiation at the region of a given pixel without any attenuation is I_0, then the attenuated intensity behind a thickness d of a material with the attenuation coefficient μ is

$$I = I_0 \cdot e^{-\mu d} \tag{1}$$

The corresponding pixel value PV is then

$$PV = k \cdot I + PV_{DF} \tag{2}$$

In (2), k is the conversion constant (can be considered as sensitivity) and PV_{DF} is the pixel value in the absence of any radiation (offset or DF value). Consequently,

$$\frac{PV - PV_{DF}}{PV_{OB} - PV_{DF}} = \frac{I}{I_0} = e^{-\mu d} \tag{3}$$

In (3), PV_{OB} is the pixel value in open beam (non-attenuated radiation). It is obvious that under the (not fully justified) assumption of linear sensitivity k in each particular pixel, operation (3) recovers the information on the attenuation of the radiation, eliminating the influence of the dark field pixel value and open beam pixel value. In this way, the non-uniformity of the pixel sensitivity as well as the influence of the beam intensity profile is also eliminated. This technique is commonly used even in the case of polychromatic radiation and it is called the flat-field correction.

In the tomographical practice, there are often requirements on even other image manipulation and calculation than the flat field. The software tool ToraBaC enables to make a flat-field correction as well as a general calculation with up to three images involved. The calculations can be made on a batch of files. The only limitation is that the result of the calculation is an image.

2. ToraBaC user manual

2.1 General purpose

ToraBaC is a software that is used for processing of tomographical images in a batch. It features a flat-field correction, or general calculation on the images with up to three images involved in the calculation. It has been developed in the Centre Telč, Institute of Theoretical and Applied Mechanics, Czech Academy of Sciences. The software has its own graphical user interface (GUI) made in PyQt5. The core algorithms are implemented in Python 3.7 and are compatible with Python 2.7.

2.2 System requirements and installation

ToraBaC was developed for computers using Windows operating system. Its usage on computers with different operating systems should be possible, but was not tested. The software requires Python with PyQt5, numpy and PIL packages installed on the computer. The whole software package consists of five files: `ToraBaC.ui`, `loadingBinDialog.ui`, `progressWindow.ui`, `EZRT.py` and `ToraBaC.py`. The .ui files are the components of the GUI, `EZRT.py` is the file with implementation of header for EZRT raw files and the `ToraBaC.py` is the core code. All these files have to be located in one directory.

The software does not require installation, it can be launched by executing `ToraBaC.py`. Dependent on the platform, this may require a simple doubleclick, or it has to be launched from the commander line going to the program directory and typing

```
python ToraBaC.py
```

If everything goes well, the console commander line will start up together with the basic GUI face of ToraBaC.

2.3 Description of the ToraBaC main window

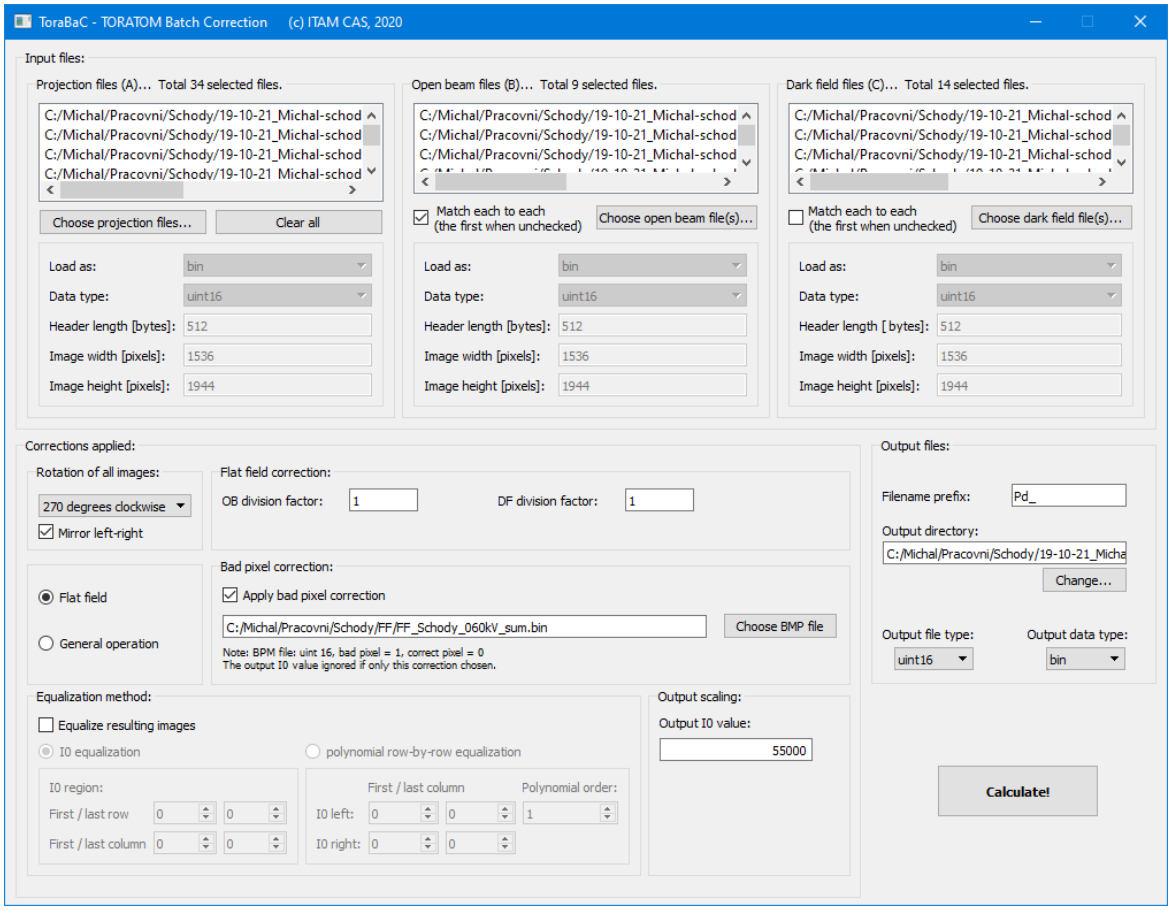

Fig. 1: Main ToraBaC window

The ToraBaC window is densely populated, but the placement of individual dialogs tries to follow a clear logic. The upper part consists in three file containers, designated also A, B and C. The A container is used for the projection files, the B and C containers serve to the Open Beam and Dark Field files, respectively, if Flat Field correction is performed. If a General operation on the images is required, the B and C containers can be used to load the necessary images, and the designation “Open beam files” and “Dark field files” will remain, although these files are not necessarily open beam / dark field.

Below the file-selecting tool, the whole operation chain parameters can be set, including the format and prefix of the resulting file. Clicking Calculate! begins the calculation process, which can take up to hours in case of 1000+ projection files.

2.4 File selection

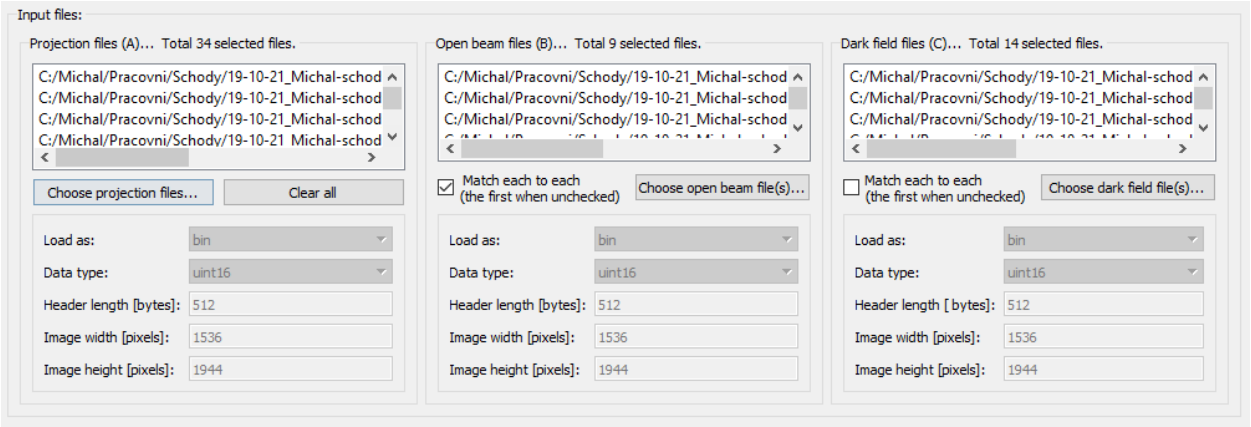

Fig. 2: Selection containers

Clicking Choose projection files…, the standard windows opening dialog appears and let the user select multiple files containing the projection images. By selecting them and specifying the format, they are *added* to the already existing files in the A container. To clear the container, button Clear all has to be used. The B and C containers do not have the Clear all button and their contents is cleared by selecting new files after Choose file(s) button is pressed, i.e., newly selected files are not added to the existing list, but they replace the existing list. In the container header, there is the number of selected files indicated for the convenience of the user.

After selecting the files, a dialog appears for the format selection. The supported formats are raw, binary and text. The Raw file is a file in the format used by Fraunhofer EZRT, including the header. In fact, this format are binary encoded data with 2048 bytes header. Binary file is the most universal importing option, as it supports integer, unsigned integer and float encoded data from 8 – 64 bits. The encoding has to be selected in the combo box provided. If there is a header, its length must be specified in bytes, as well as the width and height of the image in pixels (see Fig. 3). The last supported input file is a text file. The structure of such a file is values in the specified encoding separated with space.

Beneath the file lists, the format for import of the selected files is displayed.

It is not necessary that the A, B and C container files are of the same format, but after importing, they have to be of the same size at least.

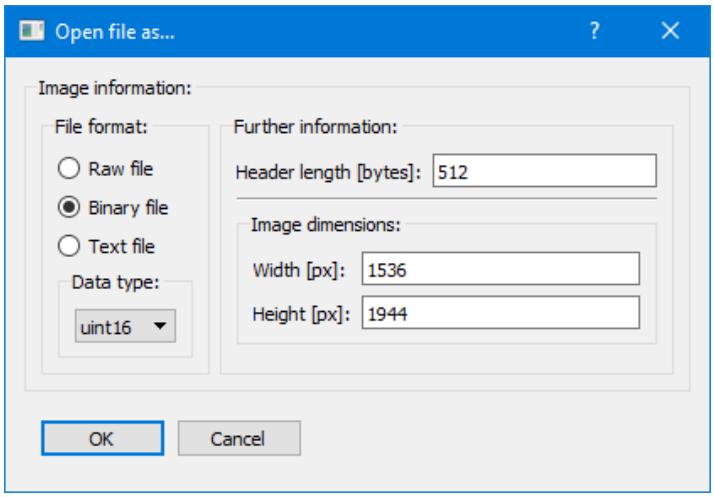

Fig. 3: Input file format dialog

For example, the projection files can be imported as binary uint16 files with a header of 512 bytes, but the open beam and dark field files, already as averages, can be without the header and in float32 format etc. For the calculations, the image data are in any case internally converted into float64 format.

There is an important feature in ToraBaC, consisting in the possibility of making calculation with matching files. In certain applications, it is necessary that the projection file is corrected with "its own" open beam and/or dark field image. This possibility is supported by ticking the Match each to each checkbox beneath the B and C container. If the checkbox is unchecked and there are more files in the container (as in the container C in Fig. 2), all files but the one on the top are ignored and the file on the top of the list is used for all the calculations. For the one-to-one calculation, the number of files in corresponding lists must be identical. If not, ToraBaC will throw an alert and will not continue to the calculation. Hence, the setup in Fig. 2 is not correct, having 34 projection files, which are supposed to be matched with 9 open beam files. The container C contains 14 files, but the Match each to each is unchecked, so that this would not lead to an alert and the file on the top would be used for all the projection files.

Figure 4 shows the processing order of the calculations.

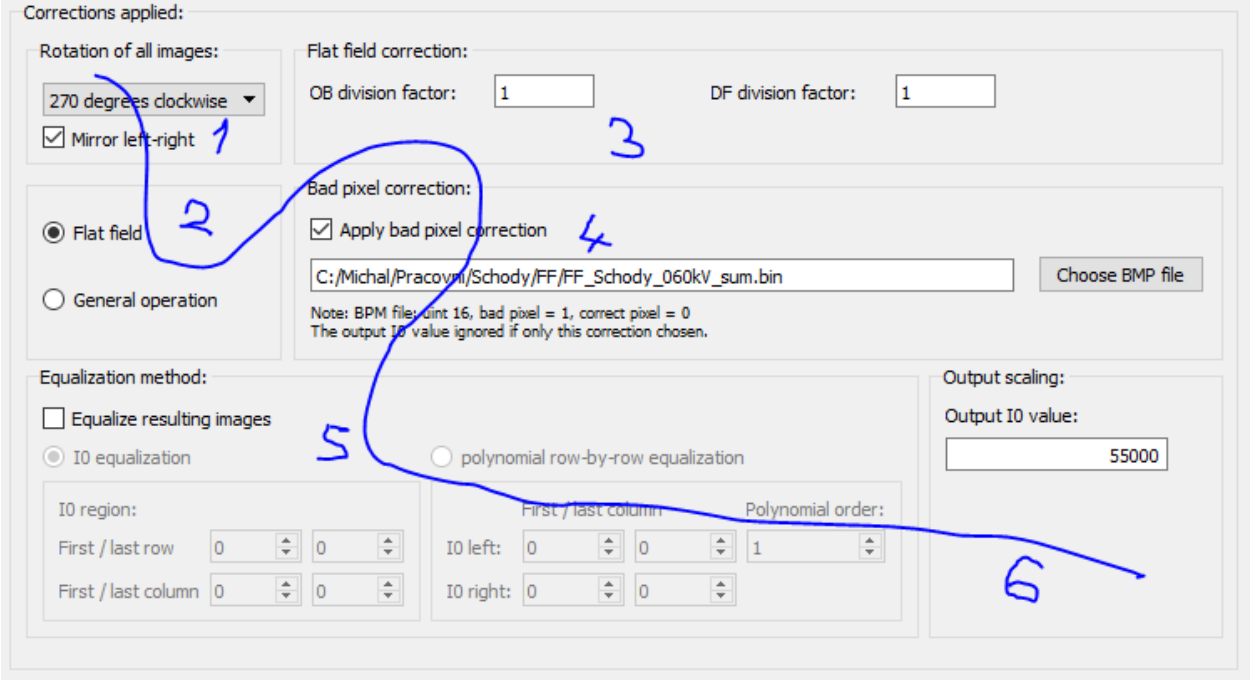

Fig 4: Processing order in ToraBaC

The processing chain begins with importing the corresponding file from the A, B and C container and their conversion to float 64 format. After that, all the three images are rotated and mirrored as specified in the Rotation of all images section.

The rotation set in the Rotation of all images section is applied to the image from A, B and C container as well as the Bad Pixel Map image, before further processing.

In the next step, flat field correction or general operation on the images is performed, depending on the selected option. In case of flat field correction, it is possible to set the division factor for the open beam and dark field image. This feature is used if the open beam or dark field images were acquired as a sum of more images. After the calculation of the flat field image, having the range 0 – 1 and stored internally as a float 64 image, the bad pixel correction is applied.

Bad pixel map (BPM) is an image encoded as **uint 16 with 0 in all good pixels and 1 in all the defect pixels**. There are several ways how to obtain the bad pixel map image, based usually

on the comparison of the open beam and dark field image. The common (and imperfect) way of generation of the BPM image in the Laboratory of X-ray tomography in Centre Telč is comparing the dark field and open beam image and declaring as defectious the pixels in which the difference of intensity is below a chosen value.

The BPM can be switched on or bypassed, depending on the check box in its section. If chosen, the BPM image has to be provided, otherwise there will be an alert from the ToraBaC and the calculation will not be performed. Again, the BPM image is rotated and mirrored in the same manner as all the other input images are. The bad pixel correction is based on linear interpolation of the bad pixel value from the neighboring pixels. This method is very simple, though quite effective and sufficient in most of the cases.

Equalization of the image is the next step after the bad pixel correction. It consists in rescaling of all the corrected images in order that the mean value of the background be 1 (note that the corrected image is scaled from 0 to 1). In this way, the variability of the intensity of the beam and its influence on the intensities in the images is eliminated. The equalization is possible in two ways. The **I0 equalization** requires a region that is always outside the projection of the investigated object to be defined. The mean value of the intensity in this region is calculated and a factor is derived by which all the intensities in the image have to be multiplied to obtain the mean value 1 in the same selected region in all the images.

The I0 intensity region is defined on the rotated and mirrored images. This has to be taken into account when finding a suitable I0 region on the input images that can be oriented in a different way!

Sometimes there are notable gradients on the background on the flat-fielded images. This phenomenon is in detail described in [1]. ToraBaC enables to eliminate this phenomenon in the way described in [1] by choosing the **Polynomial row-by-row equalization**. However, this operation is considerably more time consuming than the I0 region equalization. Equalization can be switched on or bypassed by checking / unchecking the corresponding check box in the Equalization section.

The last step of the flat field correction is rescaling the images to an appropriate range of intensities. This step is necessary if the images are intended to be saved in other than floating point formats. Until now, the corrected image has been internally treated as float 64 image, with the intensity range between 0 and 1. In reality, the range can be a little bit different, spanning from small negative numbers up to numbers higher than 1 in some pixels as a consequence of

noise and equalization (the mean value in I0 region is set to 1, but this means that some intensities will be higher than 1 if working with real, noisy images). It is very common to save the images in uint 16 encoding. As the maximum value of uint 16 is 65535, it is reasonable to set the scaling factor somewhere to 50 000 – 60 000 to avoid overflow in pixels with intensities higher than 1. The scaling factor is set in the **Output scaling** section.

The output image of the flat field correction made by ToraBaC, not talking about the bad pixel correction and equalization, is mathematically expressed as

$$PV_{out} = \frac{PV_{proj} - PV_{DF}}{PV_{OB} - PV_{DF}} \cdot SF \tag{4}$$

where

PV_{out} is the pixel value in the output image;

PV_{proj} is the pixel value in the projection image;

PV_{DF} is the pixel value in the dark field image;

PV_{OB} is the pixel value in the open beam image;

SF is the scaling factor.

2.5.2 General operation

If General operation is selected in the respective section, the Flat field correction section is replaced with the General operation section, see Fig. 5.

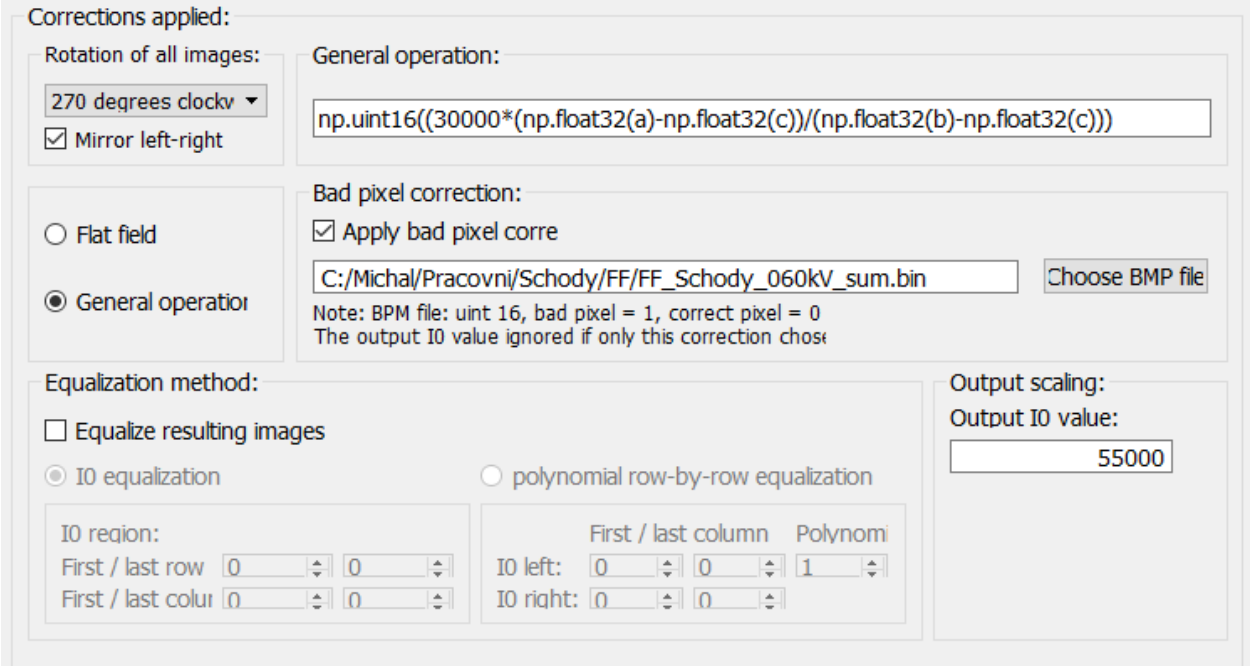

Figure 5: General operation engaged

The required mathematical formula can be written into the edit line in the General operation section. By default, there is the formula np.uint16((30000*(np.float32(a)-np.float32(c))/(np.float32(b)-np.float32(c))), which correspond to the flat-field correction with pre-conversion of the input images into the float 32 format. There are several rules for the formula:

- the result of the calculation must be an image, i.e., a 2-D array
- parameters derived from the images can be used, e.g. np.uint16(a*np.mean(a)) – here the mean value is used as a scaling factor for the image from container A
- image from containers A, B and C are referred to as a or A, b or B and c or C, respectively
- the formula can generate and image of different size than that of the input images, e.g. np.float32(a[100:200, 100:200] will generate a floating point ROI in the area of 100 to 200, 100 to 200 from the A-container image
- the formalism, or syntax, used here is fully compatible with the Python / Python numpy commands. Numpy is imported as “np”, therefore this prefix must be used in front of each numpy function. Pure Python commands are also available.

- this functionality can be used to perform rather complicated tasks, using advanced numpy functions. The only complication is the need to put the formula precisely and use correctly the brackets. In case the formula is not entered correctly, ToraBaC will issue an alert and the operation will not be performed.
- it is not necessary to use all the containers for the general operation. Therefore, this functionality can be used for example for conversion of input images into a different number format (e.g., np.float64(a)) or to make a batch rotation of the images (in that case, only "a" would be entered as operation).

The rest of the processing chain stays the same as in the case of the flat field correction. The General operation functionality is a very strong and versatile tool, but it is very sensitive to the operator's fault. It is therefore recommendable to test the developed formula on a couple of images before the whole process is triggered. As well, one must be very careful about the number formats used, their conversion, overflows etc. For example, if the operation leads to uint16 type intensities, whose values are up to 65 535 and the Scale factor is higher than 1, then there will be massive overflows in the resultant image.

2.6 Output files

In the Output files section, the path, file type, data format and prefix of the output files is set, see Fig. 6.

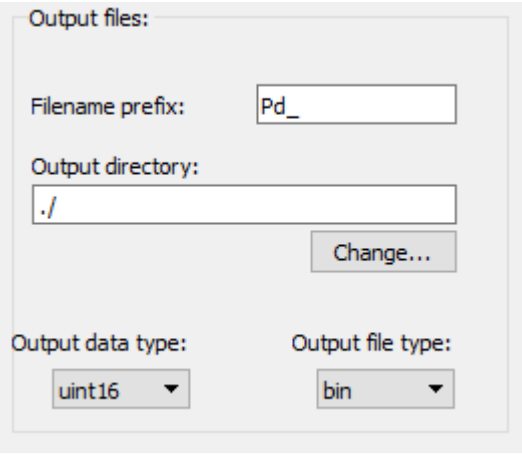

Figure 6: Output file settings

The Filename prefix is a prefix which is given to each A-container file, forming thus the output file name. The Output directory can be typed into the provided edit line, or selected by clicking the Change button. At last, the Output data type and Output file type must be selected. The available

data types as int, uint and float from 8 bit to 64 bit, as the output file type one can choose from a plain binary file (bin), the EZRT raw file (raw), the text file (txt), the jepg file (jpg) and tiff file (tif).

There is no back check of the suitability of the selected data type! For instance, if scaling factor in case of Flat Field correction is 1, the output files are floats with the range from 0 to 1. Saving them to uint16 file will lead to images where the pixel values will be only 0 or 1! This fact is even more important to be considered carefully in case of General operation mode.

3. References

[1] M. Vopálenský, I. Kumpová, D. Vavřík: Suppression of residual gradient in the flat-field corrected images. NDT.net, issue 2019-03, https://www.ndt.net/article/ctc2019/papers/iCT2019_Full_paper_87.pdf