GRAPTOR — Graz Application for Tomographic Reconstruction

Richard Huber Martin Holler Kristian Bredies*

March 6, 2019

GRAPTOR (Graz Application for Tomographic Reconstruction) is a software tool for solving tomographic reconstruction problems related to Radon-transform inversion. The reconstruction framework is based on a variational approach and Tikhonov regularization. The software accompanies the publication [7] and was designed to carry out a coupled TGV-based [3, 2, 6] regularization for multi-channel tomographic reconstruction problems, particularly for applications in analytical *Scanning Transmission Electron Tomography* (STEM) via EDX or EELS, see e.g. [8, 10, 11].

The tool is implemented in Python and uses a parallelized OpenCL implementation for multi-core CPUs and GPUs to accelerate reconstruction for big data sets. Moreover, a graphical user interface (GUI) allows for easy and intuitive use also for non experts.

This manual has the purpose to explain the usage of the program, the corresponding Graphical User Interface (GUI) and its functionality and options. In particular, Section 4 is dedicated to explaining how to find suitable parameters for reconstruction.

Contents

*Institute of Mathematics and Scientific Computing, University of Graz, Heinrichstraße 36, 8010 Graz, Austria. Email: `richard.huber@uni-graz.at`, `martin.holler@uni-graz.at`, `kristian.bredies@uni-graz.at`.

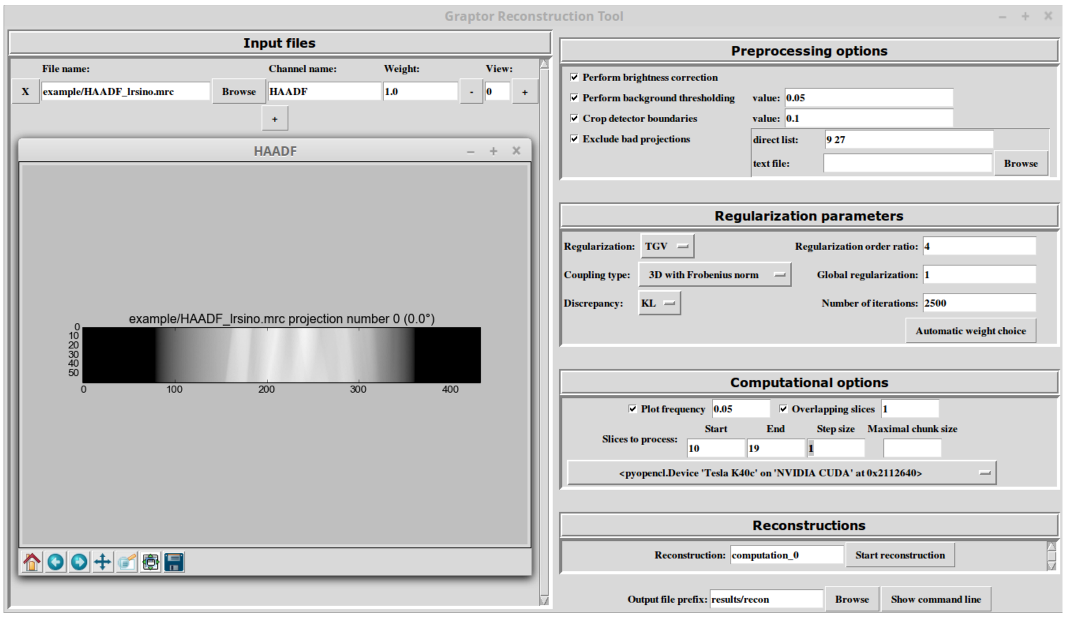
1 Getting started — A first example

Before describing the GUI and its capabilities in detail, we provide a first basic example of the typical work flow with the software. For the following, we assume that Python, all required Python packages as well as at least one OpenCL device is available on the system (see the '`readme.md`' that comes with this software for details). Then, we are ready to start the program by calling '`GUI.py`', e.g., by changing into the program folder and typing '`python GUI.py`' on the terminal.

Assume we want to reconstruct a volumetric image from the file '`HAADF_lrsino.mrc`' in the '`example`' folder. To this aim, we first select the file using the *Browse* button on the top left of the GUI. Since this data set represents HAADF density data we call the channel 'HAADF' by entering this name in the field of the *Channel name* column next to the *Browse* button. At first, we take a look at the data by clicking the + button (once or several times) in the *View* column right of the *Channel name* field. Note that for this visualization, the software has already applied the preprocessing options *Perform brightness correction* and *Perform background thresholding* in the *Preprocessing options* box, which are activated by default (see Section 3.2 for details).

When browsing through the individual projections using the '−' and '+' button, we notice that there is quite some black space with empty measurements on both sides of the projections. While this is not an issue for the reconstruction method in principle, it still unnecessarily increases computation time and we can remove this black space by checking the *Crop detector boundaries* box in the *Preprocessing options* box. After setting the corresponding value to 0.1 and clicking for example on the '+' button again to update the viewer, we see that this removes the black space on both sides.

Continuing to browse through the projections, we also notice that in particular projections 9 and 27 suffer from rather strong artifacts. Since this might degrade reconstruction quality, we can leave out these two projections in the reconstruction by checking the *Exclude bad*


Graptor Reconstruction Tool

Input files

File name: example/HAADF_lrsino.mrc — Channel name: HAADF — Weight: 1.0 — View: 0

Preprocessing options

- ☑ Perform brightness correction
- ☑ Perform background thresholding — value: 0.05
- ☑ Crop detector boundaries — value: 0.1
- ☑ Exclude bad projections — direct list: 9 27 — text file:

Regularization parameters

Regularization: TGV — Regularization order ratio: 4
Coupling type: 3D with Frobenius norm — Global regularization: 1
Discrepancy: KL — Number of iterations: 2500

Computational options

☑ Plot frequency 0.05 — ☑ Overlapping slices 1

	Start	End	Step size	Maximal chunk size
Slices to process:	10	19	1	

<pyopencl.Device 'Tesla K40c' on 'NVIDIA CUDA' at 0x2112640>

Reconstructions

Reconstruction: computation_0

Output file prefix: results/recon

Figure 1: Screenshot of the GUI for the first example.

projections box in the *Preprocessing options* and adding the numbers 9 and 27 (separated by spaces) into the *direct list* field.

As a first try, we leave all parameters in *Regularization parameters* at the default values, since for most standard problems TGV and KL with 3D Frobenius coupling is a suitable choice, and 2500 iterations should suffice.

In order not to compute all slices available in the data we enter 10 and 19 in the *Start* and *End* fields right next to *Slices to process* in order to only use those slices in our computation. As OpenCL device we use the default choice such that, after the steps above, the GUI should look as in Figure 1.

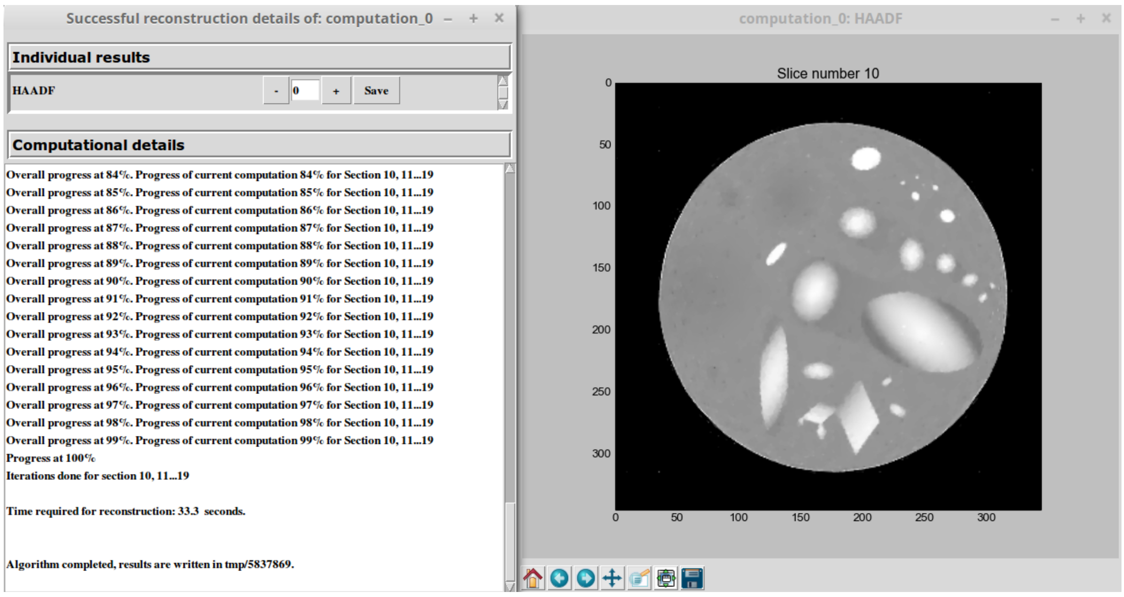
Now we click on the *Start reconstruction* button and, after a quick terminal plot, observe how the reconstruction with the current parameters evolves along the iterations of the reconstruction algorithm. Once the reconstruction is complete, a window appears that shows the terminal output and, via clicking the '$-$' and '$+$' buttons and the *Save* button, allows to view and save the reconstruction as depicted in Figure 2.

This completes the first example of using the software and we refer to the subsequent sections of this manual for further information.

2 The reconstruction approach

For a detailed explanation of the reconstruction method and the numerical implementation we refer to [7], while here we give only a brief overview. When using this software package in your research, we ask you to cite this reference.

We consider N input channels (for example $N = 4$ for STEM data with inputs from HAADF and elemental maps of aluminum, silicon and ytterbium) and corresponding given sinogram data $\{f_i\}_{i=1}^N$. The reconstruction corresponds to solving a Tikhonov approach of



```
Overall progress at 84%. Progress of current computation 84% for Section 10, 11...19
Overall progress at 85%. Progress of current computation 85% for Section 10, 11...19
Overall progress at 86%. Progress of current computation 86% for Section 10, 11...19
Overall progress at 87%. Progress of current computation 87% for Section 10, 11...19
Overall progress at 88%. Progress of current computation 88% for Section 10, 11...19
Overall progress at 89%. Progress of current computation 89% for Section 10, 11...19
Overall progress at 90%. Progress of current computation 90% for Section 10, 11...19
Overall progress at 91%. Progress of current computation 91% for Section 10, 11...19
Overall progress at 92%. Progress of current computation 92% for Section 10, 11...19
Overall progress at 93%. Progress of current computation 93% for Section 10, 11...19
Overall progress at 94%. Progress of current computation 94% for Section 10, 11...19
Overall progress at 95%. Progress of current computation 95% for Section 10, 11...19
Overall progress at 96%. Progress of current computation 96% for Section 10, 11...19
Overall progress at 97%. Progress of current computation 97% for Section 10, 11...19
Overall progress at 98%. Progress of current computation 98% for Section 10, 11...19
Overall progress at 99%. Progress of current computation 99% for Section 10, 11...19
Progress at 100%
Iterations done for section 10, 11...19

Time required for reconstruction: 33.3  seconds.


Algorithm completed, results are written in tmp/5837869.
```

Figure 2: Reconstruction results for the first example.

the form

$$\min_u \sum_{i=1}^{N} \mu_i D(Tu_i, f_i) + R(u), \tag{1}$$

where $u = \{u_i\}_{i=1}^N$ represents the tomographic reconstruction we aim to compute, and Tu_i the Radon transform of the ith channel of u. The function D corresponds to a data fidelity, i.e., is a measure of misfit between Tu_i and f_i, R is a regularization functional intended to promote desired properties of the reconstructions (e.g., piecewise smoothness) and the μ_i are positive weights that are used to balance data fidelity with regularization.

For the Radon transform T we use the custom implementation described in [7], which is given as

$$Tu(s,\theta,z) = \sum_{(x,y):\, |(x,y)\cdot\theta^\perp - s| < \Delta s} \left(1 - \frac{|(x,y)\cdot\theta^\perp - s|}{\Delta s}\right) u(x,y,z), \tag{2}$$

where the sum is taken over all positions (x,y) in a discrete pixel grid.

The default choice for D in our software package (and also the one used in [7]) is the Kullback–Leibler divergence [1], which is, from a Bayesian perspective, the appropriate choice in the case of Poisson distributed noise. It is, for non-negative densities $\tilde{u}, f$ (up to constants) given as

$$D(\tilde{u}, f) := D_{\mathrm{KL}}(\tilde{u}, f) := \int \tilde{u} - f\log(\tilde{u}),$$

where we use the convention $0\log(0) = 0$ and $-a\log(0) = \infty$ for $a > 0$. An alternative choice offered by our software package is a squared L^2-fidelity (the appropriate choice in the case of Gaussian noise), which is given as

$$D(\tilde{u}, f) = D_{L^2}(\tilde{u}, f) = \frac{1}{2}\int (\tilde{u} - f)^2.$$

For regularization, the choice proposed in [7] (again the default of the software) is a multi-channel TGV [3] functional using a Frobenius norm coupling of different channels. That is,

$$R(u) = \mathrm{TGV}_\alpha^2(u) = \min_w \alpha_1 \| |\nabla u - w|_{n_1} \|_1 + \alpha_0 \| |\mathcal{E} w|_{n_2} \|_1, \tag{3}$$

Here, ∇ denotes a channel-wise discrete gradient operator, mapping vector valued images to their matrix-valued discretized derivatives, and $\mathcal{E}$ denotes a channel-wise symmetrized gradient mapping the matrix field w to a tensor field. The norms $|\cdot|_{n_1}, |\cdot|_{n_2}$ are pointwise norms and the norm $\|\cdot\|_1$ is the sum over all voxels. The default choice of the software is the respective Frobenius norm for both pointwise norms $|\cdot|_{n_i}$, i.e., the root sum of squares of all matrix or tensor entries. This is also what is used in [7] and is motivated by the goal of enforcing joint sparsity, in particular of $\nabla u - w$, and with that, to enforce an edge alignment between the different contrasts. An alternative choice offered by the software is to choose both $|\cdot|_{n_i}$ to be decoupled norms, i.e., to take the Frobenius norms of the different channels of $\nabla u - w$ and $\mathcal{E} w$ separately. This is equivalent to a separate reconstruction of each channel and is a good baseline to compare the benefit of coupling different channels. A further choice offered by the software is to pick $|\cdot|_{n_1}$ to be a pointwise nuclear norm (and again $|\cdot|_{n_2}$ to be the Frobenius norm), see for instance [9] for an application of this idea in the context of biomedical imaging. The nuclear norm can be interpreted as a convex relaxation of a rank penalization and is again expected to enforce an alignment of level sets.

Besides the usage of different coupling norms, the software also allows to choose between slice-wise 2D or volumetric 3D regularization. For the former, the derivative operators $\nabla, \mathcal{E}$ only evaluate derivatives in two dimensions while for the latter, derivatives are taken with respect to all three space dimensions. Generally, volumetric regularization will produce better results than slice-wise regularization and is also the default choice of the software as well as the method used in [7]. Slice-wise regularization has the advantage of requiring less memory and is computationally cheaper, thus it is suitable for systems with less memory or computational power.

A second option instead of second order TGV regularization that is offered by the software is total variation regularization [12], i.e.,

$$R(u) = \| |\nabla u|_{n_1} \|_1.$$

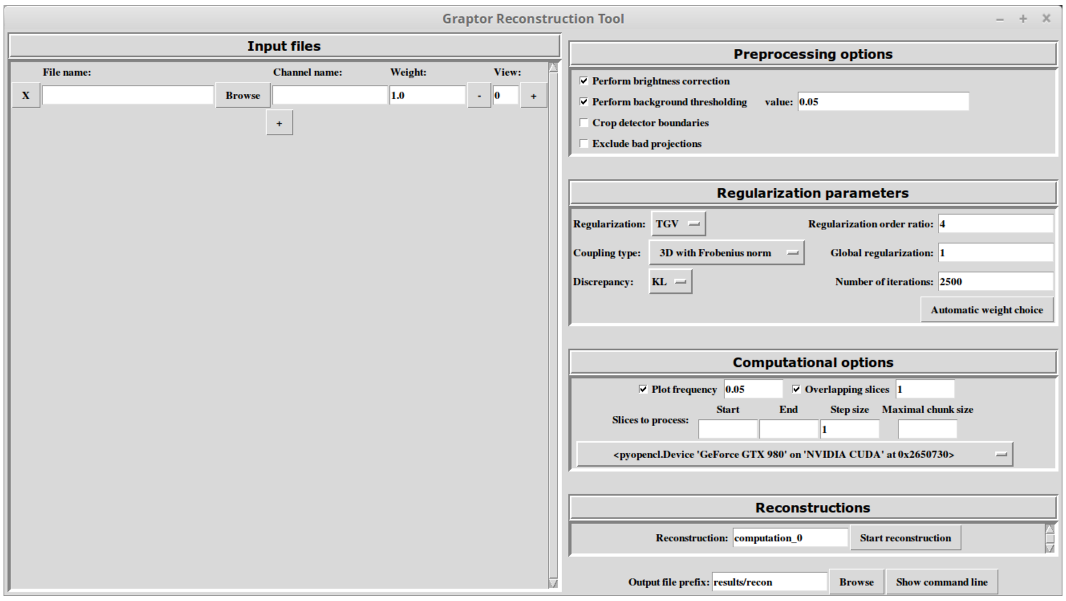
Also in this context, the pointwise norm can be chosen to be a decoupled, Frobenius or nuclear norm and the regularization can be chosen to be in 2D or 3D.

Remark. *We note that a nuclear-norm based coupling of the channels together with 3D regularization, though experimentally implemented in our software, is currently not available as option in the GUI due to its experimental character and potentially unexpected behavior.*

3 The GUI

For non-expert users, the most convenient way to use the software package will be via the graphical user interface (GUI). The GUI allows to load data, enter desired parameters, start and handle reconstructions and save the corresponding results in an easy visual way, see Figure 3.

We will now provide a step by step explanation on the functionality of the GUI, which inputs are required and what the meaning of the available options is. The GUI can be



Weight: 1.0
View: 0

☑ Perform brightness correction
☑ Perform background thresholding value: 0.05
☐ Crop detector boundaries
☐ Exclude bad projections

Regularization: TGV
Regularization order ratio: 4
Coupling type: 3D with Frobenius norm
Global regularization: 1
Discrepancy: KL
Number of iterations: 2500

☑ Plot frequency 0.05
☑ Overlapping slices 1

Slices to process:	Start	End	Step size	Maximal chunk size
			1	

<pyopencl.Device 'GeForce GTX 980' on 'NVIDIA CUDA' at 0x2650730>

Reconstruction: computation_0

Output file prefix: results/recon

Figure 3: The graphical user interface after startup.

started by '`python GUI.py`' or a similar call with '`python3`' or from integrated development environments (IDEs) inside the program folder.

3.1 Input files

3.1.1 Data input and format

The first step towards a reconstruction is to designate the files to load the projection data from. For each data channel, the software requires an MRC file [5] (ending with '`.mrc`') containing the sinogram data, and in addition a file containing information on the projection angles. All sinograms for the different channels need to have the same dimensions (3D array of size #projections × #detectors × #slices). The angle file needs to either have the suffix `.rawtlt` and contain the corresponding angle values in degree (one per line), or it can be a csv file (same name with suffix `.csv`) where the angles are delimited by '`,`'. The order of the angles in the file must correspond to the order of projections in the MRC file. The GUI provides basic functionality to create an angle file in case it does not exist: It will then open a dialog with fields *Start angle*, *Final angle* and *Step size* that allow to specify these parameters for regularly spaced angles, and will create a respective '`.rawtlt`' file when the button *Create file* is pressed.

After pressing *Browse*, the user is asked to choose the MRC file (alternatively the path to the file can be entered directly in the *File name* field). The software assumes that a projection-angle-file with the same name (and suffix either '`.rawtlt`' or '`.csv`') is available in the same folder as the sinogram file. See also the example data set (in the folder '`example`') for a reference. Additional files for additional channels can be entered by clicking the '+' button at the bottom of the file list.

The algorithm is intended for data such that 0 (or the minimal value) represents no mass and the maximal value (or 1) represents the maximal amount of mass. The algorithm normalizes all data to values between 0 and 1 to make data comparable. If the data is saved in a way such that the maximal value corresponds to no mass and the minimum value corresponds to mass, the values need to be negated before loading the data. Also, note that it is assumed that the sinogram data is centered around the rotation axis.

3.1.2 Channel name, weight and view

Right to the *Browse* button, the user has the option to define a name for each data channel. The channel name needs to be unique and is used for reference in the software and for saving the data. If no channel name is entered, the channels will be numbered by the software.

Once the correct data is selected, the software allows to visualize the different projections of the entered data. The fields below the *View* headline allow to define the projection number to visualize. It can be entered directly or adapted with the '−' and '+' buttons on the left and right. To open the visualization, either click on one of the '+'/'−' buttons or press *enter* after entering a number in the field. Note that the viewing option shows the projection *after* applying the data preprocessing options (except the removal of bad projections, see Section 3.2 below). This is in particular useful to confirm that the correct preprocessing options are chosen, e.g., that a suitable amount of thresholding is used, whether brightness correction is necessary, or whether it is possible to reduce the number of detector pixels (again see Section 3.2 for details).

The field *Weight* is used to choose adequate values μ_i in (1) for each channel. More precisely, the weights μ_i will be set for each channel to the value in this field times the value of the field *Global regularization* on the right hand side of the GUI. Thus, the value of *Global regularization* allows to define the overall trade off between data and regularization while the values in the fields *Weight* allow to balance data fidelity between the different channels. The GUI also provides a heuristic for an automatic choice of these weights by clicking on *Automatic weight choice* on the right (see Section 3.3). This choice uses the mean values of the normalized data to provide a first guess for the weight parameters. We also refer to Section 4 for more information on how to find suitable weights.

3.2 Preprocessing options

The preprocessing options of the GUI allow to correct for particular errors or model inconsistencies in the data via basic preprocessing steps. We refer to the supporting information of [7] for more detailed information on the impact of data errors or model inconsistencies to the reconstruction and on the basic preprocessing steps available here.

3.2.1 Perform brightness correction

A common issue with practical measurements are changing overall pixel intensities for different projection angles, which might occur due to technical limitations such as partial detector shadowing, or changing intensities of the beam. This creates model inconsistencies as our method assumes all projections to correspond to the same overall mass. To correct for such inconsistencies, a brightness correction is available in the GUI (and active by default). When using the brightness correction, the projection data for the different angles is normalized such that all projections have the same overall mass.

3.2.2 Perform background thresholding

Measured sinogram data might, in addition to Poisson noise, also be affected by low level thermal (Gaussian) noise due to electronics involved. While this is not severe for voxels with mass, it leads to model inconsistencies for background voxels where no mass should be present. To correct for this, the GUI allows to apply a background thresholding. When the option *Perform background thresholding* is active, all pixels with value below the number provided in the field on the right will be set to zero (in this context, note that the data is normalized *a-priori* to have range from 0 to 1). The effect of the thresholding can be observed by using the data viewing option of the GUI (in *Input files*, see Section 3.1), as it shows the data with this preprocessing step already applied.

3.2.3 Crop detector boundaries

This option allows to remove a certain percentage of boundary pixels along the horizontal space dimension, uniformly for each sinogram measurement. It is useful if the sinogram data contains large stripes of zero measurements at the boundaries. Though this is not a a theoretical issue, removing such zero measurements will reduce memory requirements and computation time, while essentially not affecting the result as long as small stripes of empty measurements are still present. If the option *Crop detector boundaries* is set, the value in the field to its right should be between 0 and 0.5 and defines the fraction of overall image pixels to remove from both the left and right sinogram boundaries. A value 0 will remove no pixels, while the value 0.5 will remove all pixels. The result of this option is visible in the viewing functionality of the GUI.

3.2.4 Exclude bad projections

Some projections present in a data set might not be usable altogether, due to significant errors in measurement, e.g., strong diffraction contrasts. Thus, considering these projections in the reconstruction might in fact be worse than just skipping these measurements. This is what can be done with the *Exclude bad projections* option. There are two options to do so: One is to use the *direct list* option to directly enter the numbers of projections to be left out separated by spaces (you can use the viewing option to determine the corresponding numbers). Alternatively, one can use the *text file* option to load a '`.txt`' file containing these numbers (one per line). If both options are used, the projections that are given in either of the two ways will be left out.

3.3 Regularization parameters

The regularization parameter box allows to define the type of regularization and data fidelity to use as well as some regularization parameters and the number of iterations for the algorithm.

3.3.1 Regularization

This option allows to choose between total variation (TV) and second order total generalized variation (TGV) regularization as described in Section 2. The value TGV regularization is the default choice, as it allows to reconstruct piecewise smooth data with jump continuities and is also the method used in the experiments of [7]. The option *TV* is computationally cheaper and might be better suited in case the desired reconstruction is known to be piecewise constant.

3.3.2 Regularization order ratio

This option only has an effect when TGV is used for regularization. It allows to set α_0 in (3), while $\alpha_1 = 1$ is fixed. Usually the default value of 4 should be a good choice and no adaptation of this parameter should be necessary. In any case, a greater choice for this parameter will drive TGV towards a TV-like behavior (subject to globally affine intensity shifts) and a smaller value will drive it towards a second-order TV-like behavior, in particular leading to rather smooth reconstructions with smoothed jumps. It is also important to note that adapting this value will affect the overall amount of regularization, in particular in the limit $\alpha_0 \to 0$, no regularization will be carried out. To compensate for that (and achieve the second-order TV-like smoothing effect as described before) the *Global regularization* parameter needs to be increased accordingly.

3.3.3 Coupling type

This allows to define the type of coupling used in the joint regularization of multiple data channels as described in Section 2. The following options can be chosen:

- *2D uncoupled*: Regularization incorporating derivatives only in each 2D slice and no coupling of the different channels. Equivalent to separate reconstruction of the slices and channels.
- *2D with Frobenius norm*: Regularization in 2D as above, but coupling different channels with the Frobenius norm. Equivalent to separate reconstruction of the 2D slices.
- *2D with nuclear norm*: Regularization in 2D as above, but coupling the matrix-valued derivatives with a nuclear norm and (only applicable for TGV) coupling the tensor valued derivatives with a Frobenius norm. Equivalent to separate reconstruction of the 2D slices.
- *3D uncoupled*: Regularization incorporating derivatives in all three space dimensions but no coupling of the different channels. Equivalent to separate reconstruction of the different channels.
- *3D with Frobenius norm* (default choice): Regularization incorporating derivatives in all three space dimensions and coupling different channels with the Frobenius norm.

We note that the option 3D with nuclear norm, i.e., regularization incorporating derivatives in all three space dimensions and coupling the matrix- and tensor-valued derivatives with a nuclear and Frobenius norm, respectively, though implemented in the software, is not yet available in the GUI since this option is still experimental and might lead to unexpected behavior.

We also note that the different coupling options have no effect if only one channel is used and that there is no difference between 2D and 3D regularization in case only one slice is used.

3.3.4 Discrepancy

This option allows to set the data fidelities in the minimization problem (1) either to the Kullback–Leibler divergence (option *KL*, default) or a squared L^2-norm (*L2*) as described in Section 2. We note that both fidelity types might yield reasonable results even though the noise model might not exactly fit to the data and that both types might require to use different regularization parameters.

3.3.5 Global regularization

The value in the field *Global regularization* adjusts the regularization for all channels uniformly. More precisely, the weights μ_i in (1) will be set for each channel to the value in this field times the value of the field *Weight* of the corresponding channel on the left hand side of the GUI. We refer to Section 4 for more information on how to suitably choose these parameters.

3.3.6 Number of iterations

The field *Number for iterations* defines the number of iterations of the primal-dual optimization algorithm [4] that is used to solve (1) (for details on the concrete implementation we refer to [7]). Since the algorithm will converge to a global optimum of (1), the only constraint on the number of iterations is to be high enough such that the optimum is approximated sufficiently well. On the other hand, increasing the number of iterations will naturally increase computation time such that one has to find a good balance between computational cost and close approximation of the optimal reconstruction. Typically, oscillations and wavelike artifacts in the reconstruction (see, e.g., Figure 6) are an indicator that the method has not yet converged and a higher number of iterations is needed. In this context, one should also note that the parameter *Global regularization* affects such artifacts and the level of noise in the reconstruction.

3.3.7 Automatic weight choice

The option *Automatic weight choice* proposes a choice of the weights for each contrast (see the fields *Weights* on the left) depending on the mean of each data channel (which, in the case of Poisson noise, correlates with the present noise level). This can give a rough first idea for suitably choosing the weights which, together with the *Global regularization* parameter, define the overall tradeoff between regularization and data fidelity (see also the Section 4).

3.4 Computational options

The *Computational options* box allows to choose parameters regarding the slices of the data that should be processed as well as options for the visualization of intermediate results and the OpenCL device to be used.

3.4.1 Plot frequency

The option *Plot frequency* (when activated) plots intermediate results of the reconstruction process. The value in the right-hand-side field defines the update frequency for the plot with intermediate results, as a fraction of the overall number of iterations. That is, the default value 0.05 means that the plot will be updated whenever a fraction of 0.05 of the total number of iterations (see *Number of iterations*) is completed. Alternatively, an integer value $x > 1$ means that the results after every xth iteration will be plotted.

3.4.2 Overlapping slices

If the OpenCL device used for the computation does not possess a sufficient amount of memory, the program will try to split the problem (splitting the slices being reconstructed). This might lead to issues at the slices where the split occurs. Slightly overlapping the subproblems could lead to more consistency. The parameter in the field right to this option specifies the number of slices the resulting subproblems overlap. E.g., if the problem has slices [1,2,...,10] and we can compute 6 slices in parallel, an overlap of 1 would lead to slices [1...6] being processed but only [1...5] of the results being saved, and computing [5...10] but saving only [6...10]. So this overlap leads to larger subproblems and hence, to redundancy and slightly increased computational effort. The default value is 1, i.e., one overlapping slice is used. A value of 0 or unchecking the box *Overlapping slices* deactivates the feature. Depending on the size of the slices, greater overlaps might also be reasonable.

3.4.3 Slices to process

The fields in this option define the slices of the data that should be processed by the algorithm. The field *Start* and *End* define the first and last slice to be processed, while the field *Step size* allows to define a stride on this slide interval. That is, a step size of x means that every xth slice in the given interval will be processed, starting from the slice given in *Start*. One single slice can be computed by choosing *Start=End*, though reconstructing with multiple slices can be expected to yield superior results when 3D reconstruction is used in the *Coupling type* option.

The optional field *Maximal chunk size* allows in addition to manually limit the number of slices of the data to be used per computation. If an integer number is provided in this field, which is lower than the total number of slices, the program will try to split the computations for the problem into (slightly overlapping) subproblems where in each subproblem only as many slices as given with this parameter will be used. If the field is left empty, no limit on the number of used slices is set. This option allows to manually limit the total memory that will be required by the software. While generally the software will try to split the problem automatically in case of insufficient memory, this option is intended in particular for cases where the automatic splitting delivers unsatisfactory results.

3.4.4 Choice of OpenCL device

The box below the *Slices to process* option allows to choose the OpenCL device to be used for the computations by selecting one item in the list of available devices. An error message is displayed in case no suitable device is found and in this case, the reconstruction code can not be executed. The most likely reasons for this are that PyOpenCL is not installed, is not configured correctly, or a GPU driver is missing.

3.5 Reconstructions

The *Reconstructions* box allows to start, save and view the reconstructions using the parameters defined in the other boxes. The field right to *Reconstruction* allows to enter a name for each reconstruction to compute (if no name is entered, the software will number the reconstruction instances automatically).

Computing reconstructions. Clicking on *Start reconstruction* computes the reconstruction using the current parameters. Also, an additional line will appear that allows to start further reconstructions (potentially with different parameters) while the previous ones are still being computed or available for visualization or saving. Any running reconstruction can be stopped and afterwards restarted using the *Abort* and *Restart* buttons, respectively.

Storing reconstructions. Once a particular reconstruction is complete, it can be saved in '`.mrc`' format using the *Save* button. This will open a dialog where the user is asked to choose a file name. Both the reconstruction as well as its sinogram data will be saved using this file name, where the channel name for each channel as well as the suffix '`sinogram`' will be appended if applicable. Also, a '`.config`' file will be saved with the same file name where all computational parameters are stored. In case more than one channel is used, the software will in addition save two files (with suffix '`all_channels.mrc`' and '`all_channels_normalized.mrc`') where the reconstructions for all channels are concatenated side by side, once in the original grayscale range and once where all channels are rescaled to the same range.

Viewing reconstructions. The *Details* button allows to show the terminal output of the reconstruction procedure, to view the reconstructions, and to save particular channels or joint views. The viewing functionality is the same as in the *Input files* box. Besides viewing and saving the different channels, the *Details* dialog also allows to jointly view and save the reconstructions for all channels, either in original scaling or rescaled to the same range.

Deleting reconstructions. As long as a particular reconstruction instance is visible in the list of the *Reconstructions* box, it can be viewed and saved to disk. The *Delete* button deletes reconstructions from this list. While this will delete the temporary storage files associated with a particular reconstruction, it will not delete reconstructions that have already been saved to disk using the saving functionality of the GUI.

3.6 Command line execution

As alternative to using the GUI, the reconstruction software can also be called directly from the terminal. This might in particular be useful for a scripted execution of the code, e.g., for a parameter sweep. The button *Show command line* on the right bottom of the GUI shows the terminal command for executing the reconstruction with the current parameters. In the field *Output file prefix* to its left, the user can define a prefix for the output files. Note that this only affects the terminal command, not the path that is used with the *Save* button in the GUI. For further information on how to use the software via the terminal we refer to the help function of the reconstruction code that can be called with '`python Reconstruction_coupled.py --help`'.

4 Parameter choice strategies

In this section we consider the question of how to actually find suitable weights and parameters, which is a very relevant aspect of the approach since a good reconstruction can only be obtained with suitable parameters. As it is generally a good strategy for parameter choice, we first deal with an appropriate parameter choice for a single channel reconstruction and then extend our considerations to multi channel data. At the end of this section, we provide a concrete example.

4.1 General considerations

4.1.1 Choice of a single weight

When first dealing with the reconstruction software, it is recommended to start with a single density image (ideally the best data set, e.g. the HAADF data set in electron tomography) and to tune the regularization parameter for this single image first. To this aim, it is only necessary to adapt the *Global regularization parameter*. As a general rule, note that large values of this parameter will lead to noisy results while low values lead to smooth and blurry results. Hence, when aiming to find a good parameter choice, it is recommended to sequentially increase and decrease this parameter until a good balance between remaining noise and smoothing is achieved. The best solutions are usually obtained with the largest weight that yields reconstructions without noise or noise artifacts. Note that, for such a parameter search by bisection, in order to get a first estimate of a good parameter it is usually sufficient to carry out only a few iterations (such as, e.g., 500) since this typically already reveals whether the solution will be noisy or blurry, even though some artifacts due to the low iteration number might still be left. Also, it is recommended to initially use only a few slices of the data to get a first estimate of this parameter and only afterwards include the entire data set for fine tuning.

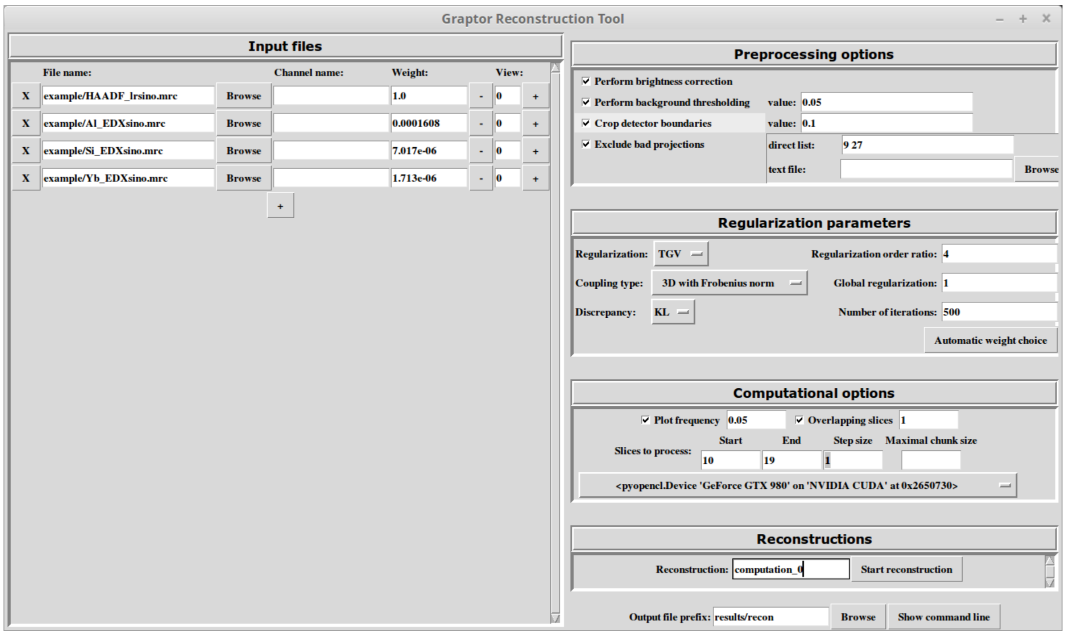
4.1.2 Choice of multiple weights

Once a good parameter for a single channel is found, additional data channels can be loaded with the software. With multiple channels, the reconstruction quality depends again on the *Global regularization* parameter but also on the individual *Weight* parameter for each channel. To deal with multiple channels, it is recommended to use the *Automatic weight choice* button of the GUI in order to get a first estimate on reasonable weights. After clicking this button, the *Global regularization* parameter can again be adapted by bisection to find a good balance between smoothness and noise, starting with what was a good choice for a single data channel. After a good parameter is found, the user can manually fine-tune the *Weight* parameter to further improve the results. Note that, in some cases, it might also be beneficial to iterate between tuning the *Global regularization* parameter and the channel weights.

4.2 Example on parameter choice

Let us reconstruct all 4 channels in the '`example`' folder by loading them into the GUI. We use the parameters that have already been chosen in the first example in Section 1 (i.e., *Crop detector boundaries* set to 0.1 and *Exclude bad projections* set to 9,27, *Global regularization* set to 1.0), assuming that those are a good choice for the single channel data. Pressing the button for *Automatic weight choice* yields values as depicted in Figure 4.

We start the reconstruction with these weights, *Global regularization* set to 1.0, and 500 iterations. The result (all channels with normalization) is shown in Figure 5. We see that the reconstructions are rather smooth and blurry, which is not satisfactory. This affects all


Graptor Reconstruction Tool

Input files

	File name:		Channel name:	Weight:		View:	
X	example/HAADF_lrsino.mrc	Browse		1.0	-	0	+
X	example/Al_EDXsino.mrc	Browse		0.0001608	-	0	+
X	example/Si_EDXsino.mrc	Browse		7.017e-06	-	0	+
X	example/Yb_EDXsino.mrc	Browse		1.713e-06	-	0	+

Preprocessing options

- ✓ Perform brightness correction
- ✓ Perform background thresholding — value: 0.05
- ✓ Crop detector boundaries — value: 0.1
- ✓ Exclude bad projections — direct list: 9 27; text file: (empty)

Regularization parameters

- Regularization: TGV
- Coupling type: 3D with Frobenius norm
- Discrepancy: KL
- Regularization order ratio: 4
- Global regularization: 1
- Number of iterations: 500

Computational options

- ✓ Plot frequency 0.05
- ✓ Overlapping slices 1
- Slices to process: Start 10, End 19, Step size 1, Maximal chunk size (empty)
- <pyopencl.Device 'GeForce GTX 980' on 'NVIDIA CUDA' at 0x2650730>

Reconstructions

Reconstruction: computation_0

Output file prefix: results/recon

Figure 4: The graphical user interface for the whole sample data set in the ‘`example`’ folder after automatic weight choice.

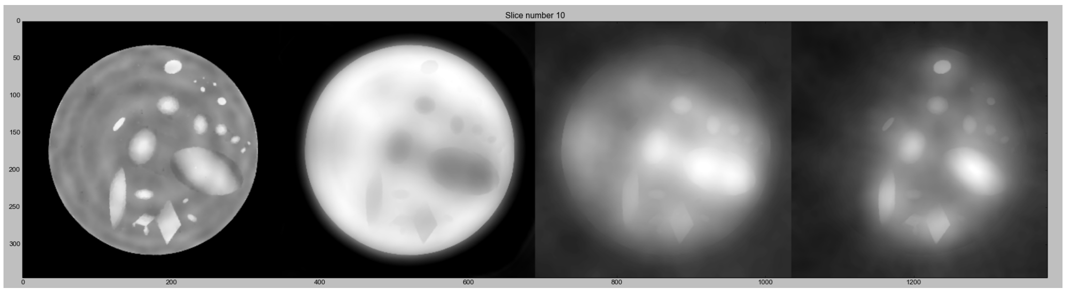


Figure 5: Preliminary reconstruction results with automated weight choice, *Global regularization* set to 1, and after 500 iterations.

channels, meaning that the *Global regularization* parameter could be too small. So we try again, setting the *Global regularization* parameter to 10. This step yields the results shown in Figure 6. We see that now, the result has generally improved, but in particular the silicon and ytterbium channels still appear too smooth. To compensate for that, we increase the corresponding *Weight* parameter by a factor of 10, setting it to 7.017e-05 for silicon and to 1.713e-05 for ytterbium. After 500 iterations, we obtain the results depicted in Figure 7. Being satisfied with this for the moment (though more fine tuning would certainly beneficial) we now compute 2500 iterations with the same parameters, resulting in the reconstruction as shown in Figure 8. This seems satisfactory for our first try and concludes this example.

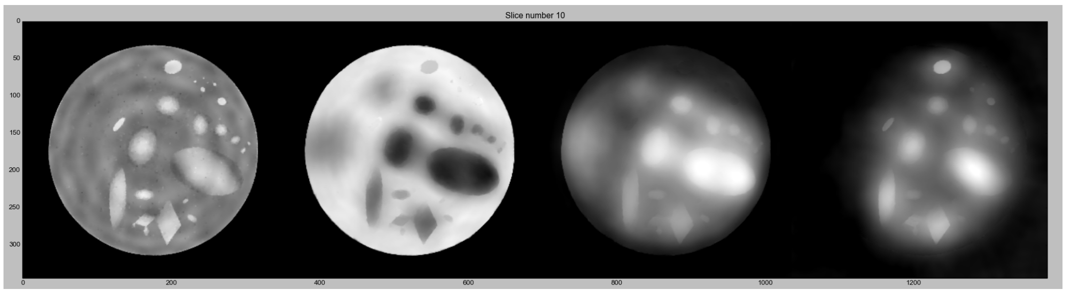


Figure 6: Preliminary reconstruction results with automated weight choice and *Global regularization* set to 10.

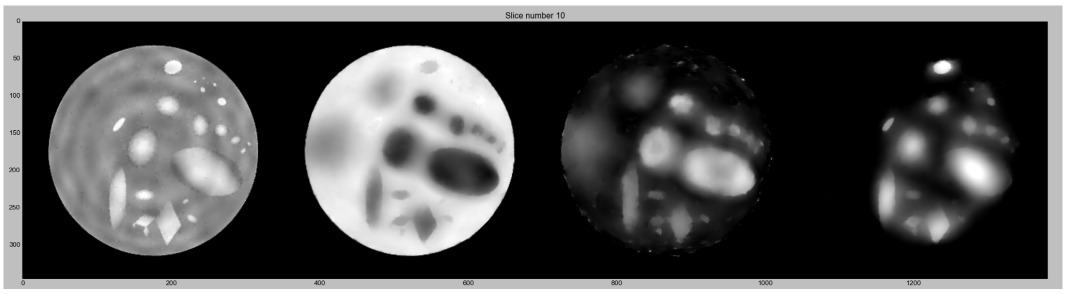


Figure 7: Preliminary reconstruction results with adapted weights and *Global regularization* set to 10.

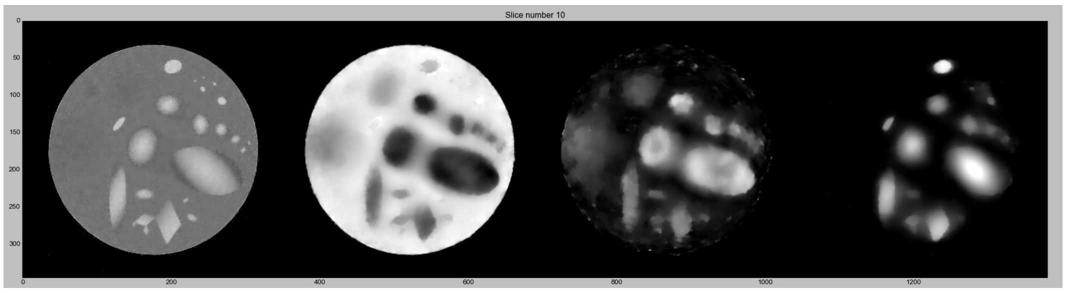


Figure 8: Final reconstruction results with HAADF weight $= 1.0$ aluminum weight $\approx 1.6 \cdot 10^{-4}$, silicon weight $\approx 7.0 \cdot 10^{-5}$, ytterbium weight $\approx 1.7 \cdot 10^{-5}$, *Global regularization* set to 10 and 2500 iterations.

References

[1] Jonathan M. Borwein and Adrian S. Lewis. Convergence of best entropy estimates. *SIAM Journal on Optimization*, 1(2):191–205, 1991.

[2] K. Bredies. Recovering piecewise smooth multichannel images by minimization of convex functionals with total generalized variation penalty. In Andrés Bruhn, Thomas Pock, and Xue-Cheng Tai, editors, *Efficient Algorithms for Global Optimization Methods in Computer Vision*, volume 8293 of *Lecture Notes in Computer Science*, pages 44–77. Springer Berlin Heidelberg, 2014.

[3] Kristian Bredies, Karl Kunisch, and Thomas Pock. Total generalized variation. *SIAM J. Imaging Sci.*, 3(3):492–526, 2010.

[4] Antonin Chambolle and Thomas Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *J. Math. Imaging Vision*, 40(1):120–145, 2011.

[5] Anchi Cheng, Richard Henderson, David Mastronarde, Steven J. Ludtke, Remco H.M. Schoenmakers, Judith Short, Roberto Marabini, Sargis Dallakyan, David Agard, and Martyn Winn. MRC2014: Extensions to the MRC format header for electron cryo-microscopy and tomography. *Journal of Structural Biology*, 192(2):146–150, 2015.

[6] M. Holler, R. Huber, and F. Knoll. Coupled regularization with multiple data discrepancies. *Inverse Problems*, 34(8):084003, 2018.

[7] Richard Huber, Georg Haberfehlner, Martin Holler, Gerald Kothleitner, and Kristian Bredies. Total generalized variation regularization for multi-modal electron tomography. *Nanoscale*, 2019. DOI: 10.1039/C8NR09058K.

[8] Konrad Jarausch, Paul Thomas, Donovan N. Leonard, Ray Twesten, and Christopher R. Booth. Four-dimensional STEM-EELS: Enabling nano-scale chemical tomography. *Ultramicroscopy*, 109(4):326–337, 2009.

[9] F. Knoll, M. Holler, T. Koesters, R. Otazo, K. Bredies, and D. Sodickson. Joint MR-PET reconstruction using a multi-channel image regularizer. *IEEE Trans. Med. Imag.*, 36(1):1–16, 2017.

[10] Rowan K. Leary and Paul A. Midgley. Analytical electron tomography. *MRS Bulletin*, 41(7):531–536, July 2016.

[11] K. Lepinay, F. Lorut, R. Pantel, and T. Epicier. Chemical 3d tomography of 28nm high K metal gate transistor: STEM XEDS experimental method and results. *Micron*, 47:43–49, 2013.

[12] Leonid I. Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1–4):259–268, 1992.