# Ettention User Guide (1.0)

## Installation

### Ettention

Download etomo.zip. The zip file contains following files: etomo.jar, makecomfile, ettentionsetup, ettention.cmd, and ettentionsetup.adoc. Copy the first four files into %IMOD_DIR%\bin and the ettentionsetup.adoc into %IMOD_DIR%\autodoc. Note that you are going to be asked to replace existing etomo.jar and makecomfile (to be on the safe side, make backup of these two files before you replace them). The only difference between the original and the patch files is the support for Ettention in the latter ones, i.e. the functionality of Etomo remains unchanged.

Download binaries for Ettention (or use your own build) and eTomo plugi. Copy following files to the %IMOD_DIR%\bin:

- reconstructConsoleApp.exe
- libReconstruction.lib
- EtomoPlugin.dll
- EtomoPlugin.lib
- EtomoPlugin.exp

Rename recosntructConsoleApp.exe to ettention.exe. Done. Try to run Etomo again. In Tomogram Generation step is now in addition to WBP and SIRT also Ettention.

### IMOD

The current version of Ettention patch supports **IMOD v 4.7**.
To download IMOD visit:
http://bio3d.colorado.edu/imod/
All necessary information regarding IMOD installation can be found here:
http://bio3d.colorado.edu/imod/doc/guide.html
Once installed, try to run Etomo to see, if it works properly.

## Usage via eTomo

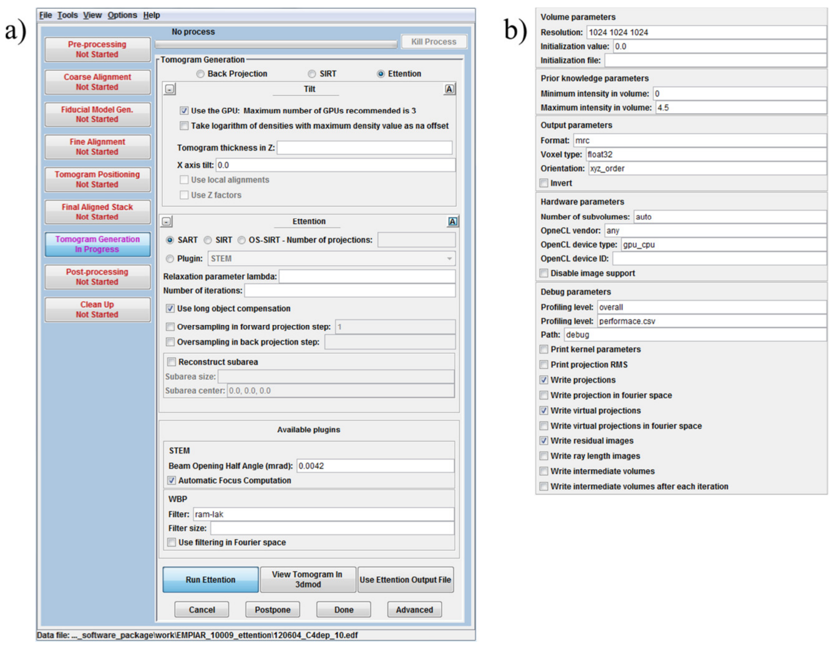
The simplest way to start using Ettention is via the eTomo Graphical User Interface (GUI) in the IMOD package. ETomo is a user interface written in Java, which provides easy access to the individual algorithmic steps in IMOD, implemented as separate executables. The interface covers the entire workflow for electron tomography, including projection preprocessing, CTF correction, and alignment as well as tomogram region-of-interest selection, reconstruction, postprocessing, and filtering. Ettention plugs in to seamlessly replace the tomographic reconstruction step in this pipeline.

The components necessary to incorporate Ettention into IMOD package and eTomo GUI ship with the Ettention distribution and consist of two parts: a modified version of eTomo and a generic IMOD adapter for Ettention. Adding Ettention into an existing IMOD/eTomo installation requires two steps: installing Ettention and copying the modified eTomo .jar file into the IMOD folder.

Once installed, additional GUI elements are displayed in the eTomo User Interface (Figure **Fehler! Es wurde keine Folge festgelegt.**), which allow to access frequently used functions. That includes reconstruction algorithm (Ettention SART, Ettention SIRT or Ordered Subset SIRT), relaxation parameter, number of iterations, and oversampling options for forward and back projection.

In order to access only occasionally used functions, the GUI can be set to advanced mode. This exposes a large number of parameters such as hardware device selection (GPU), functions for debugging and profiling, options to load initial volumes, and more detailed choices for the output format.

As will be explained later, new user-assembled algorithms can be added to Ettention via a plugin mechanism. These new algorithms may also be made available in IMOD/eTomo with little effort. Every plugin consists of a shared library (.dll or .so) and an user interface extension file (.xml). The dll/so file is copied into the Ettention plugin folder, the xml file into the eTomo folder.  The plugin can then be activated using a radio button, as show in (Figure **Fehler! Es wurde keine Folge festgelegt.**). By default, the following plugins ship with the Ettention distribution: STEM tomography using Combined Tilt- and Focal Series, Weighted Back Projection, Weighted SIRT, and tomography with Geometric Prior Knowledge.


a)
File Tools View Options Help
Pre-processing Not Started
Coarse Alignment Not Started
Fiducial Model Gen. Not Started
Fine Alignment Not Started
Tomogram Positioning Not Started
Final Aligned Stack Not Started
Tomogram Generation In Progress
Post-processing Not Started
Clean Up Not Started
Kill Process
Run Ettention
View Tomogram In 3dmod
Use Ettention Output File
Cancel
Postpone
Done
Advanced
b)

No process

Tomogram Generation

Back Projection SIRT [✓ Ettention]

Tilt

[✓] Use the GPU: Maximum number of GPUs recommended is 3
[ ] Take logarithm of densities with maximum density value as na offset

Tomogram thickness in Z:
X axis tilt: 0.0
[ ] Use local alignments
[ ] Use Z factors

Ettention

[✓ SART] SIRT OS-SIRT - Number of projections:
Plugin: STEM
Relaxation parameter lambda:
Number of iterations:
[✓] Use long object compensation
[ ] Oversampling in forward projection step: 1
[ ] Oversampling in back projection step:
[ ] Reconstruct subarea
Subarea size:
Subarea center: 0.0, 0.0, 0.0

Available plugins

STEM
Beam Opening Half Angle (mrad): 0.0042
[✓] Automatic Focus Computation

WBP
Filter: ram-lak
Filter size:
[ ] Use filtering in Fourier space

Data file: ..._software_package\work\EMPIAR_10009_ettention\120604_C4dep_10.edf

Volume parameters
Resolution: 1024 1024 1024
Initialization value: 0.0
Initialization file:

Prior knowledge parameters
Minimum intensity in volume: 0
Maximum intensity in volume: 4.5

Output parameters
Format: mrc
Voxel type: float32
Orientation: xyz_order
[ ] Invert

Hardware parameters
Number of subvolumes: auto
OpneCL vendor: any
OpenCL device type: gpu_cpu
OpenCL device ID:
[ ] Disable image support

Debug parameters
Profiling level: overall
Profiling level: performace.csv
Path: debug
[ ] Print kernel parameters
[ ] Print projection RMS
[✓] Write projections
[ ] Write projection in fourier space
[✓] Write virtual projections
[ ] Write virtual projections in fourier space
[✓] Write residual images
[ ] Write ray length images
[ ] Write intermediate volumes
[ ] Write intermediate volumes after each iteration

*Figure* ***Fehler! Es wurde keine Folge festgelegt.****: The integration of Ettention into eTomo GUI allows accessing the Ettention functionality from this widespread tool. a) The basic view allows convenient access to the most common parameters as well as new, user-assembled algorithms via the plugin mechanism. b) The advanced view allows to access additional parameters, such as functions for debugging and profiling or detailed output format configurations.*

## Tilt box

This box contains general parameters regarding input projections and tomogram (e.g. tomogram thickness, projections to exclude, X axis tilt…). All parameters from this box are stored in tilt.com. Note that in case of "Take logarithm of densities" the value that follows in tilt.com is ignored – the values in projection stack are logarithmized w.r.t. their maximum value. Currently, local alignments and Z factors are not supported.

**Ettention box**

In this box, there are all parameters concerning Ettention. All parameters are stored in ettentionsetup.com.

Ettention supports block-iterative schemes also known as Ordered-Subset SIRT (OS-SIRT). Therefore, you can choose if the volume is updated after each projection has been processed (SART), after all projections have been processed (SIRT) or to after given number of projections have been processed (OS-SIRT). In latter case you also have to specify the number of projections for one volume update. Although any value between 1 and projections stack size is allowed, values between 5 and 10 are reasonable.

## Usage via the command line

As an alternative to using IMOD/eTomo, Ettention can also be operated from the command line on Windows or Linux. The command "reconstructConsoleApp --help" lists the available parameters. Most importantly, a configuration file can be specified using the parameter "--configfile filename.xml", which allows storing setups in xml files for repeated execution.

In the following, the file format for ettention configuration files is specified. All parameters can be specified in an xml configuration file. The config file itself is specified by the commandline parameter –configfile. They must be individual subnodes of a group node with their name, and the value of the parameter must be specified as the "value" attribute of the node. For example, the parameter projections from the group input is specified as follows:

```
reconstructConsoleApp.exe -configfile "sart.xml"

<input>
       <projections value="input.mrc" />
</input>
```

All options can also be overwritten using the command line in the reconstructConsoleApp executable. In this case, options must be given as parameter with leading minus "-" and the group as a prefix, separated by a dash ".". The value must be separated by a white space. For example the parameter above can be overwritten with the command line option:

```
-input.projections "input.mrc"
```

### Options Group Input

All options in this category are grouped in the XML node <input> or specified with the prefix "input." on the command line.

**input.projections (filepath, required)** This parameter specifies the path to the input stack containing the measured projection images. The path should be an absolute path if it is specified via the command line. In an xml configuration, the path can be either absolute, or relative to the location of the xml file. In the case of file formats that consists of more than one file (for example a series of tif-files in a directory), the path of the directory should be specified, without any filename.

**input.tiltAngles (filepath)** This parameter specifies the path to an separate tilt angles file (.tlt). The tilt file must be an ASCII file with one line per projection that contains the tilt angle in degrees. Tilt files can only be used with for single tilt geometry. The specification of a tiltAngles file is required if an mrc stack is used as input projections.

**input.xAxisTilt (float, default: 0.0)** This parameter specifies an angle between the tilt axis and the y-axis, i.e. it allows to tilt the geometry by a fixed angle around the x-axis.

## Options Group Output

**output.filename (filepath, required)** This parameter specifies the path to the output volume that will be written after the completion of the reconstruction. The filename must be in an already existing directory. Relative files are handles as explained for “input.projections”.

**output.format (mrc|rek|…, optional)** This parameter specifies in which file format the output volume is written. Legal ids are “mrc” (MRC stack) and “rek” (Fraunhofer REK format). It the parameter is not specified, the system uses the extension of the filename in the output. Additional file formats can be provided via plugins.

**output.options.voxelType (unsigned8|unsigned16|float32, default: float32)** This parameter specifies the encoding of an individual voxel in the output stack. For the floating point voxel type float32, values are stored “as they are”. For the fixed precision voxel types (unsigned8 and unsigned16), values are scaled to exploit the available data range as best as possible, i.e. the lowest value is mapped to zero and the highest value in the volume to the highest available integer value in the encoding. If the parameter is not specified, the system assumes the voxel type that was found in the input projections, i.e. it will generate a 16 bit volume from 16 bit projections, and so on.

**output.options.orientation (xzy_order|xyz_order, default: xzy_order)** This parameter specifies, in which orientation the output volume is written to disc, i.e. what coordinate layout order is used. Default is the xzy layout.

**output.options.invert (bool, default: false)** If set to true, the output volume will be inverted in the sense, that the absolute range of the values are preserved by the role high- and low contrast are exchanged, i.e. bright voxels become dark and vice versa.

## Options Group Reconstruction Volume

**volume.dimensions (Vec3f, required)** Specifies the size of the reconstruction volume in world space units.

**volume.origin (Vec3f, default: 0/0/0)** Specifies the origin of the reconstruction volume in world space units. Together with *volume.dimensions* this can be used to specify the bounding box of the reconstruction, i.e. perform an area of interest reconstruction.

**volume.resolution (Vec3ui, optional)** Specifies the resolution of the reconstruction volume. This always corresponds to the resolution of the output volume. The voxel size is given implicitly by dividing *volume.dimensions* by *volume.resolution*. If the parameter is left unspecified, the system assumes a voxel size of 1/1/1 and computes the resolution from *volume.dimensions*.

**volume.initValue(float, default=0.0)** This parameter specifies an initial, constant value for the reconstruction volume. The parameter is exclusive with volume.initFile, i.e. one can either specify an initValue or an initFile, not both.

**volume.initFile (filepath, optional)** This parameter  specified an volume file to load as an initialization of the reconstruction process. The parameter is exclusive with volume.initValue, i.e. one can either specify an initValue or an initFile, not both.

## Options Group Algorithm

**algorithm (string, required)** Specifies the identifier of the used algorithm. Legal values are "sart", "sirt" and "blockIterative". Additional legal values can be specified via plugins, i.e. plugins can add additional reconstruction algorithms, in which case the corresponding identifiers are also valid.

**algorithm.numerOfIterations (uint, required)** Specifies how often the algorithm should iterate. One iteration hereby corresponds to processing all input projections once. Typical values are 1-5 for SART and 5-15 for SIRT.

**algorithm.lambda (float, required)** The relaxation parameter for the reconstruction. Legal values are in the interval -2.0 to 2.0, exclusively.

**algorithm.useLongObjectCompensation (bool, default: false)** If set to true, the system will perform long-object compensation in the sense of (Xu et. al. 2010).

## Options Group Forward Projection

**forwardProjection.samples (uint, default: 1)** Specifies the number of samples used in the forward projection. The default value "1" means that for each pixel, a single ray is used. Legal values are natural numbers that have integer square roots, i.e. 2, 4, 9, 16 and so on.

## Options Group Back Projection

**backProjection.samples (uint, default: 1)** Specifies the number of samples used in the back projection. The default value "1" means that for each voxel, the center point of the voxel is used as a representative sample. Legal values are natural numbers that have integer square roots, i.e. 2, 4, 9, 16 and so on.

## Options Group Prior Knowledge

**priorknowledge.volumeMinimumIntensity (optional float, default: none)** Specifies a previously known minimal intensity of the tomogram gray values. A typical value is zero, which specifies that tomogram gray values must be positive.

**priorknowledge.volumeMaximumIntensity (optional float, default: none)** Specifies a previously known maximum intensity of the tomogram gray values.

## Options Group Hardware

**hardware.openclVendor (string, default: any)** This value specifies, which OpenCL devices should be used for the reconstruction. The default value is "any", other legal values are "intel", "nvidia" and "amd". By setting a value, access to opencl devices can be restricted to devices from a certain vendor.

**hardware.openclDeviceType (string, default: gpu_cpu)** This value specifies, which OpenCL devices should be used for the reconstruction. The default value is "gpu_cpu", other legal values are "cpu_gpu", "gpu" and "cpu". The different options have to following effect:

- gpu_cpu If available, the system uses a GPU. If no GPU is available, the system uses a CPU device.
- cpu_gpu If available, the system uses a CPU. If no CPU is available (i.e. no OpenCL driver is installed for the CPU), the system uses a GPU device.
- gpu The system uses a GPU device. If no GPU device is available, an error message is produced.
- cpu The system uses a CPU device. If no CPU device is available, an error message is produced.

**hardware.openclDeviceId (int, optional)** If this parameter is set, the id if an opencl device can be specified explicitly. The parameter should not be used together with openclVendor and openclDeviceType. In other words: you can either give hints using specifiy openclVendor and openclDeviceType and let the system decide which device to use, or explicitly specify the device using openclDeviceId, not both.

**hardware.disableImageSupport (bool, default: false)** By default, the system detects if an OpenCL device has support for 3D textures (images) and uses this type of memory for the forward projections. On platforms without support for 3D textures, the behavior is emulated. By setting the parameter to true, the system will use the emulation on any platform, disregarding native texture support.

**hardware.subVolumeCount (unsigned int, optional)** By default, the system automatically determines how much memory is available on the OpenCL device. If the available memory is insufficient to store the entire volume (plus all additional buffers), a number of subvolumes is used, i.e. the system incrementally transfers parts of the volume for processing. By setting the parameter, the automatic choice for the number of subvolumes can be set manually. However, setting a number lower than the system suggestion is likely to result in an out-of-memory error.

## Options Group Debug

**debug.outputKernelParameters (bool, default: false)** If set to true, the parameters that are passed to a OpenCL compute kernel are printed to the log prior to every kernel call.

**debug.displayProjectionRMS (bool, default: false)** If set to true, the root-mean-square error of every residual image will be calculated and written to the log.

**debug.writeProjections (bool, default: false)** If set to true, the measured projections will be written to disc prior to processing. After reading a projection, it will be written again in TIFF format (32 bit float) in a directory <current_path>/Debug. Files will be named indicating the iteration number and projection number.

**debug.writeProjectionsInFourierSpace (bool, default: false)** If set to true, the measured projections will additionally be transferred to Fourier space written to disc. The output involved two files, on for the real part of the Fourier transform and one for the imaginary part.

**debug.writeVirtualProjections (bool, default: false)** If set to true, the virtual projections will be written to disc immediately after the forward projection.

**debug.writeVirtualProjectionsInFourierSpace (bool, default: false)** If set to true, the virtual  projections will additionally be transferred to Fourier space written to disc.

**debug.writeResiduals (bool, default: false)** If set to true, the residual images will be written to disc immediately after computation.

**debug.writeRayLength (bool, default: false)** If set to true, the an image showing the ray length for every pixel will be written immediately after each forward projection.

**debug.writeIntermediateVolumes (bool, default: false)** If set to true, after the processing of each back projection, an intermediate volume will be written to disc. The file format is MRC stack at 32 bit floating point precision.

**debug.writeIntermediateVolumesAfterIterations (bool, default: false)** If set to true, after each iteration (i.e. after executing the back projection once for every projection), an intermediate volume will be written to disc. The file format is MRC stack at 32 bit floating point precision.

**debug.infoPath (string, default: "debug")** This parameter specifies in which directory debug images and volumes will be written. The path is used for the output generated by setting one of the following paramters: writeProjections, writeProjectionsInFourierSpace, writeResiduals, etc...

**debug.logProgressInfo (bool, default: false)** If set to true, status information on the algorithm progress will be continuously written to the log. The system will output a report line after every projection and every iteration.

**debug.profilingLevel ("none", "overall", "normal", "detail", default: overall)** This parameter controls the granularity at which OpenCL kernel profiling information is recorded. Kernel profiling works only in debug mode.

**debug.performanceReportFile (filepath, default: performance.csv)** This parameter controls, where the OpenCL profiling information will be written.