PyFSRec.py - Python code for focal series reconstruction

Purpose

PyFSRec is a focal series reconstruction program that runs from the command line. The program requires an input parameter file, here it is called 'PRM' (for instance config.fsr), that details the storage place for input files, the reconstruction parameters and output options.

Example call

```
$> python PyFSRec.py --prm PRM
```

Help

To get information on the command line options, pass -h or --help to the program

```
$> python PyFSRec.py --help
```

Usage

```
$> python PyFSRec.py [-h] --prm PRM [--debug] [--showalignment] [--
showarea] [--prealignonly]  [--liniter] [--nonlinear] [--useprevious] [-
-stepanalysis] [--autofocus]

Mandatory arguments

--prm PRM             Full path to the input configuration file PRM.

Optional arguments

 --help, -h           Show this help message and exit.
 --debug              Enter debugging mode.
 --showalignment      Enter debugging mode for alignment.
 --showarea           Export the first image with the reference area
                      marked.
 --prealignonly       Exit program after prealignment.
 --boxcar BOXCAR      Save boxcar sum images after pre-alignment or
                      loading previous alignment data.
 --liniter LINITER    Number of linear iterations.
 --nonlinear          Nonlinear refinement (EXPERIMENTAL).
 --useprevious        Continue from the output of a previous iteration.
 --stepanalysis       Focus step analysis for a previous output.
 --autofocus          Determine the minimum contrast focus of the wave
                      function for a previous output.
```

Format of the input parameter file PRM

An example of an input parameter file is given below. It uses the syntax of .ini files, comment lines can be added with a preceding # or ; on an unindented line.

Example for a parameter input file (e.g. with the name ‘config,fsr’):

```
#
# PyFSRec input file
#

[INPUT/OUTPUT]
INPUTPATH=F:/Dropbox (Weizmann Institute)/ImagesEM/test/ #I nput files directory..
INPUTFILENAME=basename_ # Image file basename, digits 000, 001 are added by the program.
NImages=298 # Number of images.
OUTPUTPATH=F:/Dropbox (Weizmann Institute)/ImagesEM/results/  #Output files directory.
OUTPUTFILENAME=outname # Basename of output files.

[IMAGEDATA]
Dimensions=[512, 512] # Pixel dimensions of input images.
Sampling=[0.0429873,0.0429873] # Input image sampling in nm per pixel
Offset=[0,0] # Alignment area offset from the center location of the images
AREA=[256,256] # Area from alignment within the images.
Range=[[180,297]] # Range of images from the input directory used for the reconstruction.
Fixhotpix=No # Option to fix outlier and dead pixel values

[ALIGNMENT]
Comkernel=3 # Radius of the center-of-mass support in the correlation image in pixels. [3]
Maxprealign=5 # Maximum count of pre-alignment iterations. [5]
Raverage=5 # Range length for boxcar averaging during pre-alignment, used when ‘Commonframe’ is on. [5]
Commonframe=True # Do an initial pre-alignment over running averages of images, averaging length is specified by ‘Raverage’. [False]
Methprealign=sequential-com # Method used for pre-alignment, so far sequential center of mass correlation only. [sequential-com]
Invertprealign=No # Invert contrast of images during pre-alignment following the sign of the imaginary part of the contrast transfer function. [No]

[OPTICS]
Voltage=300 # Microscope HT in kV.
C3=2.7E6 #  Spherical aberration constant in nm. [few um] for an aberration corrected instrument, [few mm] for a non-corrected instrument
C5=0. #  Spherical aberration constant higher order in nm.
Semiconvergence=0.05 #  Semi-convergence angle in mrad..[0.05] – [0.2]
Focusspread=6 #  Focus spread in nm. [3]-[10]
Imagespread=20. #  Image spread parameter in pm. [20]
Foci=[0 ,200.0] # Start and end value of image defocus values, starting from the defocus value of the first image in the reconstruction.

[RECONSTRUCTION]
Limit=6 # Reconstruction limit in reciprocal nm. [Microscope information limit]
Filtercutoff=0.1  # Lower cut-off for the reconstruction filter. To Avoid divide by zero scaling error for small filter function values. [0.1]
Prealignment=On # Toggle pre-alignment of input images. [On]
Envelopes=On # Toggle for including partial coherence envelopes in the reconstruction.[On]
Alignmentfilter=[0.5,4] # Alignment tophat filter parameters in reciprocal nm.
Alignmentprecision=0.1 # Target precision for refined subpixel alignment, in pixel. [ 0.1]
LinearIterations=15 # Number of reconstruction iterations. [20].
Savealigned=No  # Optional output of aligned images. [No]
Saveintermediaterec=No # Optional output of intermediate reconstructions. [No]
Outputfilter=[0.8,6] # Optional: output of a filtered wavefunction using these tophat filter  parameters. [Lower reconstruction limit, Microscope information limit].

[STEPREFINEMENT]
Rangemultiplicator=1.5 # Search range for focus refinement in units of the input range from the first to the last image. [1.5]
Stepmultiplicator=0.3 # Search increment for the focus in units of the input focus step. [0.3]
Refinementfilter=[0.8,6] # Optional: step refinement and autofocus are done using these tophat filter  parameters. [Lower reconstruction limit, Reconstruction limit]

[AUTOCORRECTION]
Autofocusrange=180 # Half-range of focus search in nm. A value of 50 means that a range between +-50 nm is tested. [50]
Autofocusfilter=[1.1,5.] # Tophat filter parameters for the focus search. [1,5]

#
```

```
# End
#
```

Format of the image input files

Known image format is raw binary data, 32bit floating point. The filename suffix has to be .raw and filenames are numbered with three digits, starting from 000. The first filename is then basename000.raw, the second basename001.raw etc.

Parameters

- 'Foci' under the section '[OPTICS]'

  The defocus values can be given as start and end value of a linear ramp or as the full list. In case of a full list the number of list values has to fit the subset of images selected for reconstruction (see 'RANGE') field below.

  Syntax example for start and end values:

  Foci=[-275.71, -228.47]

  and for a list:

  Foci=[-275.71,-269.62,-262.00,-255.90,-249.81,-242.19,-234.57,-228.47]

- 'Range' under the section '[IMAGEDATA]'

  Determines the subset of images that is taken for reconstruction. 'Range' is an array of index ranges.

  Syntax examples for the configuration file :
  # take images 5 to 10
  Range=[[5,10]]
  # take images 5 and 10
  Range=[5,10]
  # take images 5 to 10, 13, 15-20
  Range=[[5,10],13, [15,20]]

- 'AREA' under the section '[IMAGEDATA]'

  Specifies the size of the reference frame e.g. for correlation or rms calculation. If not specified then the size in x and y is the dimension divided by two.

  Syntax example:
  # take a frame size of 384 x 384 pixels
  AREA=[384,384]

- 'Offset' under the section '[IMAGEDATA]'

Specifies the displacement of the reference frame with respect to the center of the input images. If not specified the offset is zero.

Syntax example:
# take a frame size of 384 x 384 pixels
AREA=[384,384]

- 'Fixhotpix' under '[IMAGEDATA]'

  Option to fix outlier and dead pixel values.
  A local median filter is applied to fix outliers beyond 10 times the standard deviation, NAN pixels are also fixed.

  Syntax example:
  # switch hot or dead pixel correction on
  Fixhotpix=Yes

- 'Commonframe' under the section '[ALIGNMENT]'

  The field specifies that images will be aligned to a (noise reduced) common frame.
  The width for the calculation of running averages is the field 'Raverage' under '[ALIGNMENT]'.
  This feature is experimental, it is supposed to help with low dose images.

  Syntax examples for the configuration file :
  # switch on the pre-alignment to an averaged common frame
  Commonframe=Yes

- 'Raverage' under the section '[ALIGNMENT]'

  The field specifies the number of images that are averaged for pre-alignment
  It works in conjunction with 'Commonframe', and is supposed to help with low dose images.
  Raverage=0 means no running averages

  Syntax example:
  # average over 5 images
  Raverage=5

- 'Invertprealign' under the section '[ALIGNMENT]'
  The optional boolean field specifies whether there is an attempt to compensate for contrast inversions of the phase contrast CTF during pre-alignment

  Syntax example:
  # switch compensation on
  Invertprealign=Yes

Standard output of the program

When finished PyFSRec saves a reconstructed image wave function, a filtered wavefunction, the relative displacements for the set or subset of images that were included in the reconstruction

The data is stored in the following files:

<outname>.wav — wave function, complex valued, 2x16bit complex data, single precision
<outname>_filt.wav — filtered wave function, complex valued, 2x16bit complex data, single precision
<outname>_dxy.csv — csv file with image number, displacement in x, displacement in y as columns.

<outname> corresponds to the Outputfilename specified in the parameter file. The files are saved in the Outputpath specified in the parameter file.

How to continue a previous reconstruction

PyFSRec can use results stored in the standard output files to continue a previous reconstruction, e.g. for the purpose of further refinement of displacements or foci.

When activating the command-line option –useprevious:

- The program loads <basename>_dxy.csv as initial displacements. The pre-alignment of images is skipped and the program starts directly with the reconstruction loops. The program halts with an error message if <basename>_dxy.csv is not found or if there is a mismatch between displacement data and number of images.
  Alternatively, the program can continue with zero shifts at this point when the pre-alignment is switched off in the input parameter file (check 'Prealignment' under the section '[Reconstruction]')
- The program loads <basename>_foci.csv as initial focus values. The focus data in the parameter file will be overwritten. However, the program will use the settings in the parameter file if <basename>_foci.csv is not found or if there is a mismatch between focus data and number of images.
- The program loads <basename>.wav as initial wave function. The program halts with an error message if <basename>.wav is not found.

Example on the command line:

```
$> python PyFSRec.py --prm=config.fsr  --stepanalysis  --useprevious --liniter=4
```

The above command redo the focus step analysis, using the refined focus values in the previous reconstruction output. It skips the pre-alignment and it will do four linear reconstruction iterations.

How to use the autofocus

PyFSRec can focus the wavefunction obtained in a previous reconstruction

When activating the command-line option --autofocus:

- The program loads the previously obtained wavefunction <outname>.wav, calculates the propagation in the search range declared under '[AUTOCORRECTION]' in the parameter file and looks for the minimum in the real part of the propagated wavefunction.
- The program will override the previous output <outname>.wav and <outname>_filt.wav.
- For subsequent reconstructions use the focus values 'Focus values in input format' that the program displays, paste these values into the parameter file as they contain the focus correction. Subsequent reconstructions then do not require a correction or only a minor refinement.
- Use this function only once! You'll have to redo the reconstruction after using autofocus in order to obtain convergence.

Example on the command line:

```
$> python PyFSRec.py --prm=config.fsr  --autofocus
```

How to use the focus step refinement

PyFSRec can refine the focus steps using the wavefunction obtained in a previous reconstruction. It is good practice to redo the reconstruction in case you used the autofocus function before, since the procedure relies on a consistent set of parameter file and wavefunction.

When activating the command-line option --stepanalysis:

- The program loads the previously obtained wavefunction <outname>.wav and image displacements <outname>_dxy.csv, calculates the propagation in the search range declared under '[STEPREFINEMENT]' in the parameter file and looks for the best match between input images and images obtained from the propagated wavefunction.
- For subsequent reconstructions use the focus values 'Focus values in input format' that the program displays, paste these values into the parameter file as they contain the focus correction. Subsequent reconstructions then do not require a correction or only a minor refinement.
- The program will create a file <outname>_foci.csv with refined defocus values.
- For subsequent reconstructions you may use the focus values 'Focus values in input format' that the program displays, paste these values into the parameter file as they contain the focus correction.

Example on the command line:

```
$> python PyFSRec.py --prm=config.fsr  --stepanalysis
```

How to save boxcar sum images

PyFSRec can save aligned and summed images for a subsequent reconstruction. The images are summed after pre-alignment, alignment is done either done by the program call or through loading previous alignment data.

When activating the command-line option –boxcar=m:

- If the --useprevious is chosen the program loads the previously obtained displacements <outname>_dxy.csv, align images and sums m nearest neighbour images. If the --useprevious is not chosen then the program does the pre-alignment first and then sums m nearest neighbour images.
- Summed images will be saved under <outname>_boxcar_m_float32_xxx.raw where xxx is a three-digit running number from 0 to Floor(N/m)-1

- For subsequent reconstructions with the summed images you'll need to create an appropriate parameter input file.

Example on the command line:

```
$> python PyFSRec.py --prm=config.fsr  --useprevious --boxcar=5
```