SPARX

SPARX

- SPARX in an add-on to EMAN2
-

SPARX

- SPARX in an add-on to EMAN2
- EMAN2 is an add-on to SPARX

SPARX

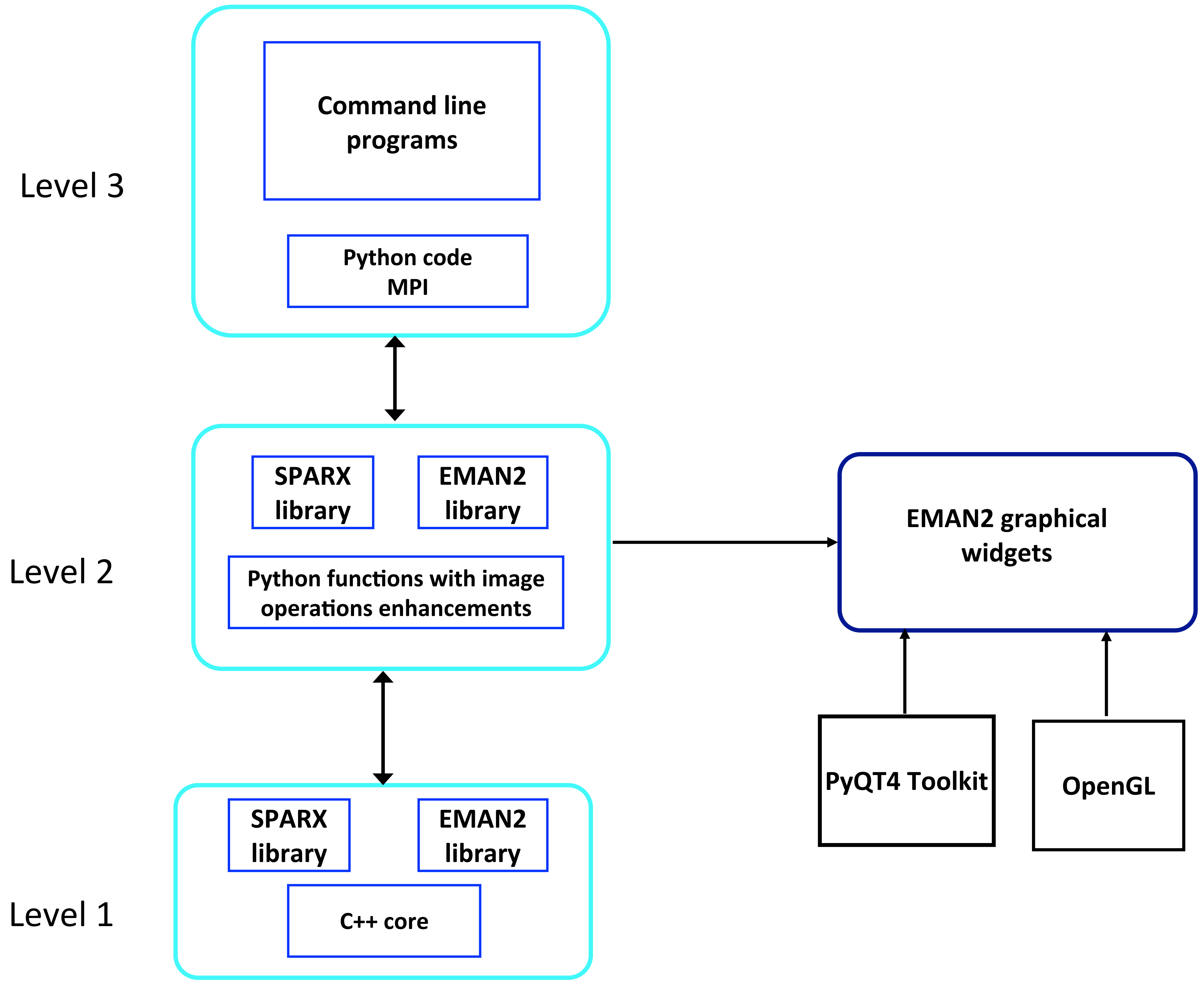
- SPARX in installed together with EMAN2
- Command line interface:
  > sparx
- sparx relies heavily on MPI parallelization, which has to be installed separately (see instructions).

SPARX

- SPARX in installed together with EMAN2
- Similarly, one can load sparx commands into EMAN2 command line interface:

```
> e2.py
  from sparx import *
```

SPARX is built on top of several other toolkits including its essential dependency EMAN2



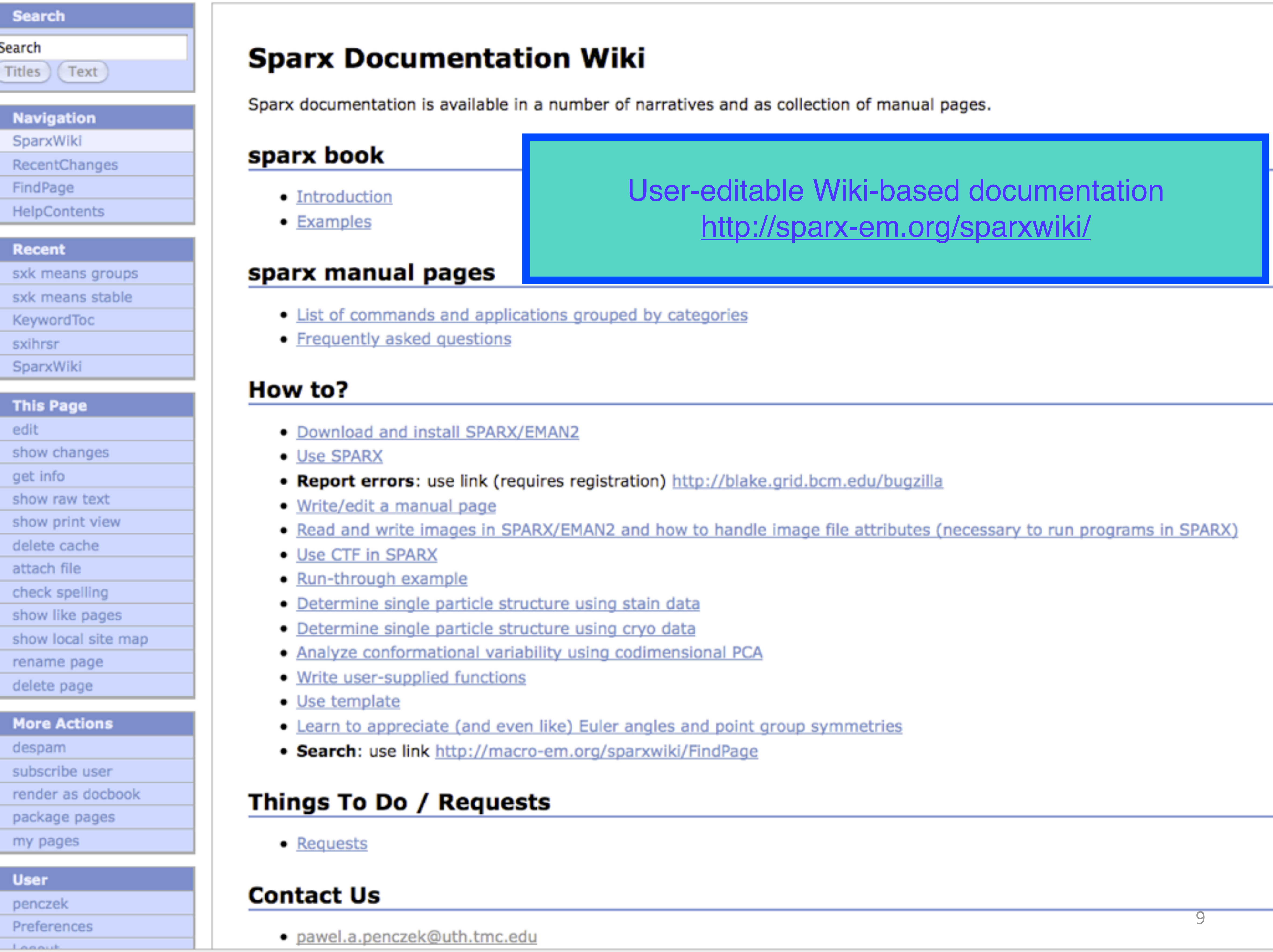
Relation between EMAN2 and SPARX

each package implements unique single particle strategies

communication through shared format of header attributes and file formats (bdb and hdf)

e2______.py	sx_____.py	*programs*
e2.py	sparx	*interactive session*
??	MPI on python level	*parallelization*
GUI driven	System command lines	*project done using*

User-editable Wiki-based documentation
http://sparx-em.org/sparxwiki/



Sparx Documentation Wiki

Sparx documentation is available in a number of narratives and as collection of manual pages.

sparx book

- Introduction
- Examples

sparx manual pages

- List of commands and applications grouped by categories
- Frequently asked questions

How to?

- Download and install SPARX/EMAN2
- Use SPARX
- **Report errors**: use link (requires registration) http://blake.grid.bcm.edu/bugzilla
- Write/edit a manual page
- Read and write images in SPARX/EMAN2 and how to handle image file attributes (necessary to run programs in SPARX)
- Use CTF in SPARX
- Run-through example
- Determine single particle structure using stain data
- Determine single particle structure using cryo data
- Analyze conformational variability using codimensional PCA
- Write user-supplied functions
- Use template
- Learn to appreciate (and even like) Euler angles and point group symmetries
- **Search**: use link http://macro-em.org/sparxwiki/FindPage

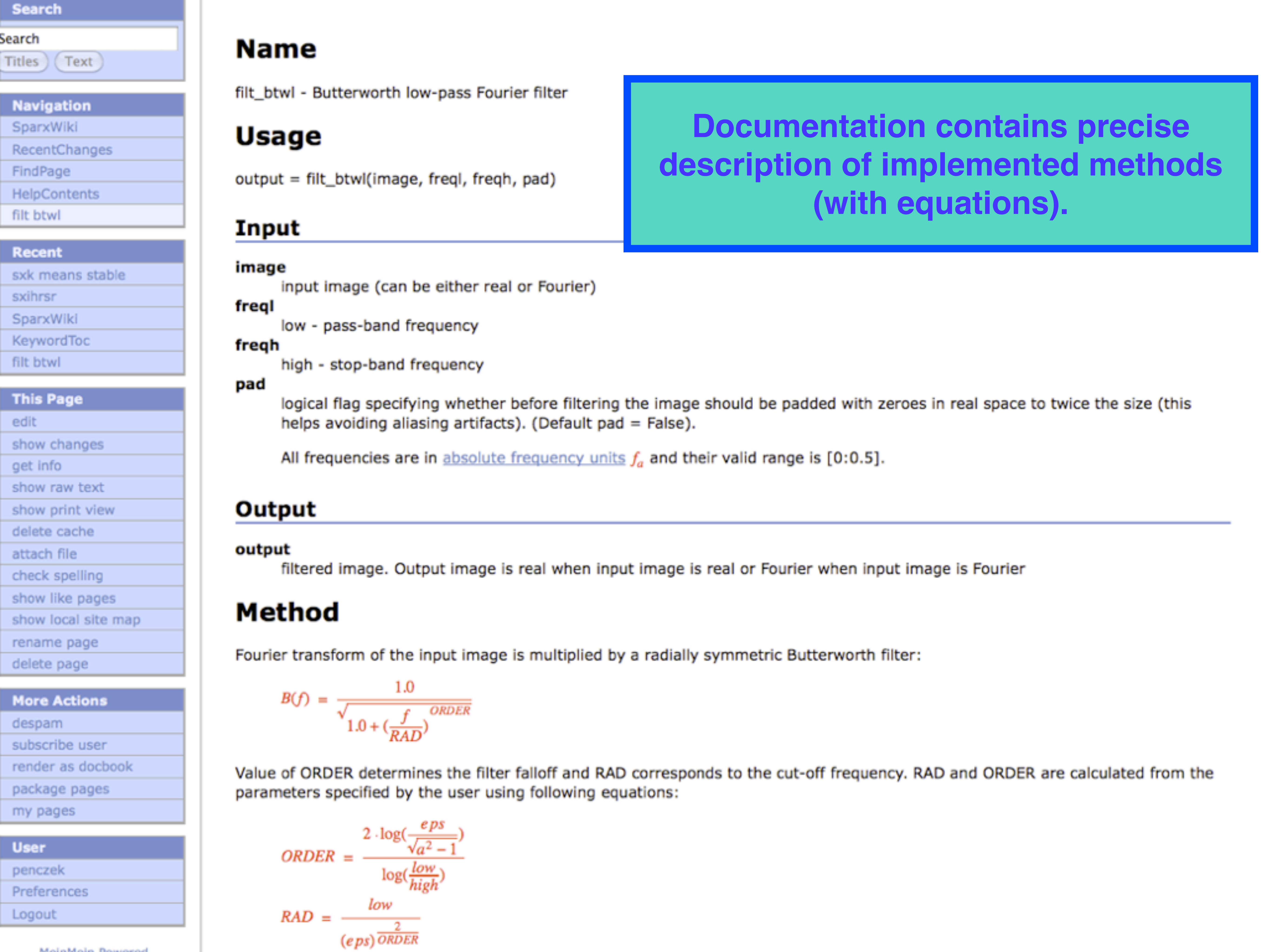
Things To Do / Requests

- Requests

Contact Us

- pawel.a.penczek@uth.tmc.edu

Documentation contains precise description of implemented methods (with equations).



Name

filt_btwl - Butterworth low-pass Fourier filter

Usage

```
output = filt_btwl(image, freql, freqh, pad)
```

Input

image
input image (can be either real or Fourier)

freql
low - pass-band frequency

freqh
high - stop-band frequency

pad
logical flag specifying whether before filtering the image should be padded with zeroes in real space to twice the size (this helps avoiding aliasing artifacts). (Default pad = False).

All frequencies are in absolute frequency units f_a and their valid range is [0:0.5].

Output

output
filtered image. Output image is real when input image is real or Fourier when input image is Fourier

Method

Fourier transform of the input image is multiplied by a radially symmetric Butterworth filter:

$$B(f) = \frac{1.0}{\sqrt{1.0 + \left(\frac{f}{RAD}\right)^{ORDER}}}$$

Value of ORDER determines the filter falloff and RAD corresponds to the cut-off frequency. RAD and ORDER are calculated from the parameters specified by the user using following equations:

$$ORDER = \frac{2 \cdot \log\left(\frac{eps}{\sqrt{a^2 - 1}}\right)}{\log\left(\frac{low}{high}\right)}$$

$$RAD = \frac{low}{(eps)^{\frac{2}{ORDER}}}$$

SPARX provide human-friendly interface to image functions...

- "Object oriented" style (EMAN2)

```
im2 = im1.process("filter.lowpass.gauss",{"cutoff_abs":0.25})
```

- "Math style" (SPARX)

```
im2 = filt_gaussl(im1, 0.25)
```

and a full set of single particle structure determination programs!

Data files

We use two file formats:

1. ***hdf*** (high density file) mainly to exchange information with other systems. For example, output 3D structures are generally stored in hdf format, as then they can be examined in *chimera*.

2. ***bdb*** (Berkeley data base) is the main file format for EMAN2/SPARX. It is not a file format, but an actual data base and as such it cannot be easily transferred between computers.

Generally, bdb files will be stored in a directory where the program started and in a subdirectory called EMAN2DB. One should not delete or copy anything in this directory. Programs *sxheader.py* and *e2bdb.py* provides basic functionality for bdb files.

Both bdb and hdf store information in image headers in a very flexible manner. Results are stored in image headers and required for subsequent commands to run properly.

Programs

sx*program*.py

Simple instructions

```
sxprogram.py   -h
```

General syntax

```
sxprogram.py   <input file/stack>  <output file>  <mask file>  --parameter=[number | text]
```

Examples:

Copy two stacks of image into a third one with a change of format:

```
sxcpy.py   bdb:set1  bdb:set2  bothsets.hdf
```

Reset to zero 3D alignment parameters

```
sxheader.py   bdb:segments  --zero --params=xform.projection
```

Recommended reading

1. Penczek, P.A., Frank, J.: Resolution in Electron Tomography, in J. Frank (Ed.), Electron Tomography: Methods for Three-Dimensional Visualization of Structures in the Cell, 2 edn., Springer, Berlin, 307-330, 2006.

2. Vainshtein, B.K., Penczek, P.A.: Three-dimensional reconstruction, in U. Shmueli (Ed.), International Tables for Crystallography 3 edn., vol. B Reciprocal Space, 2008.

3. Penczek, P.A.: Single Particle Reconstruction, in U. Shmueli (Ed.), International Tables for Crystallography 3 edn., vol. B Reciprocal Space, 2008.

4. Penczek, P.A.: Fundamentals of three-dimensional reconstruction from projections. Methods Enzymol 2010, 482, 1-33.

5. Penczek, P.A.: Image restoration in cryo-electron microscopy. Methods Enzymol 2010, 482, 35-72.

6. Penczek, P.A.: Resolution measures in molecular electron microscopy. Methods Enzymol 2010, 482, 73-100.

Iterative Helical Real Space Refinement (IHRSR)

The problem of helical reconstruction, i.e., determination of orientation parameters of EM projection images of helical filaments, is cast as a single particle structure determination problem. Ed Egelman used the "3D projection refinement" concept developed in early 90s and added to it simultaneous determination (or refinement) of helical symmetry parameters.

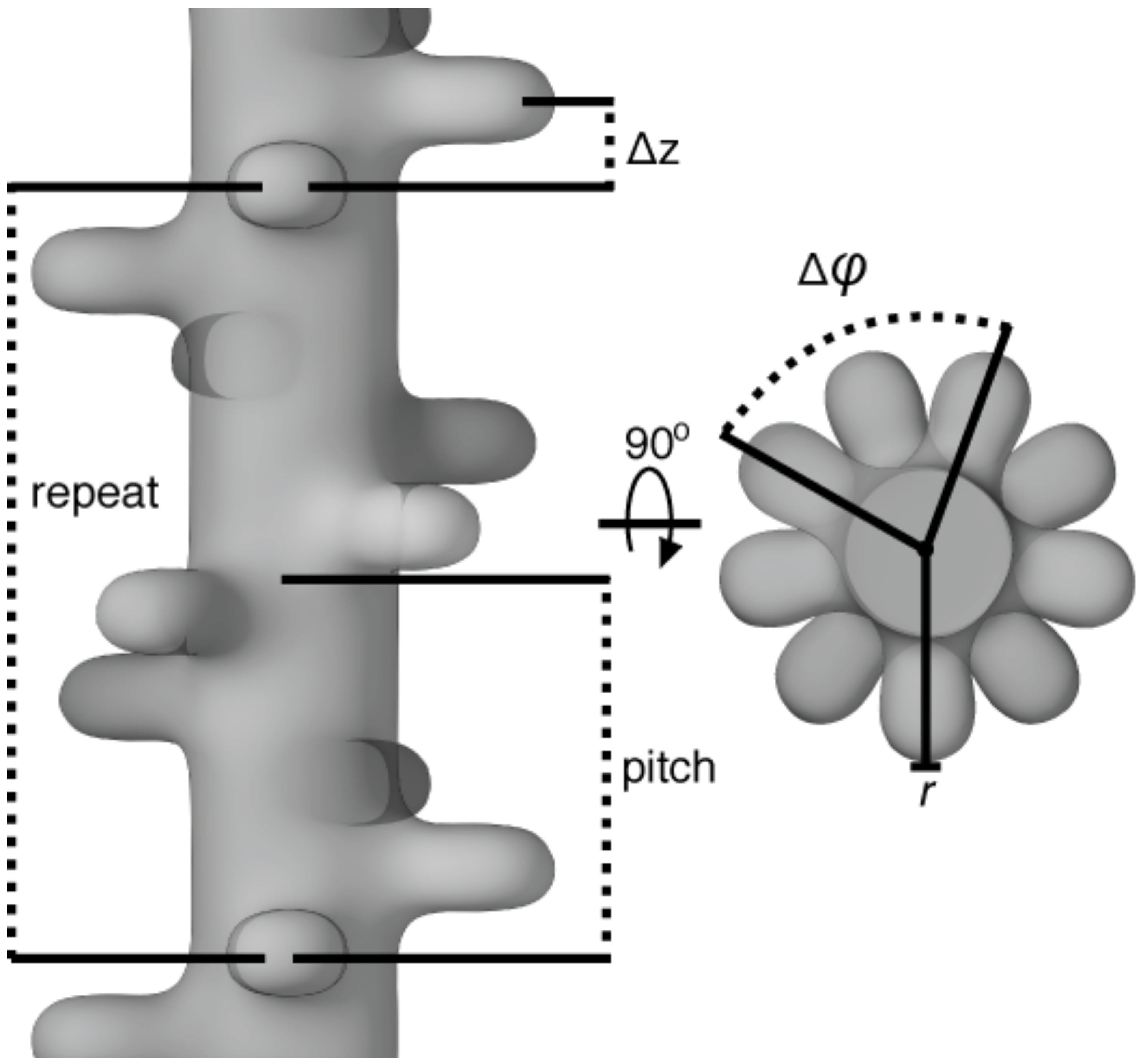
IHRSR - Iterative Helical Real Space Refinement

Egelman, E.H. (2000). A robust algorithm for the reconstruction of helical filaments using single-particle methods. Ultramicroscopy *85*, 225-234.

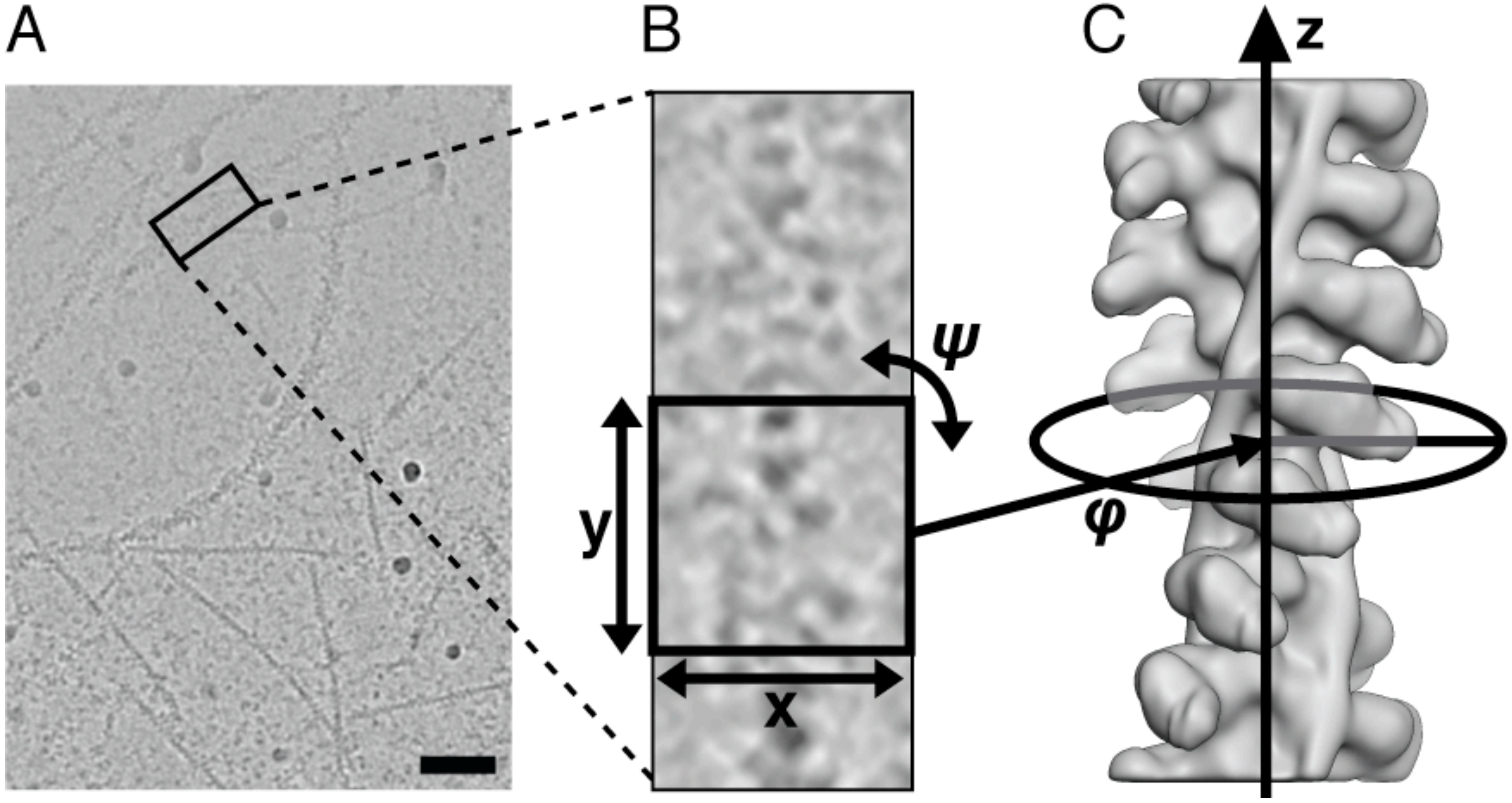
Penczek, P.A., Grassucci, R.A., and Frank, J. (1994). The ribosome at improved resolution: new techniques for merging and orientation refinement in 3D cryo-electron microscopy of biological particles. Ultramicroscopy *53*, 251-270.

Helical symmetry is given, in cylindrical coordinates, by two parameters: *azimuthal rotation per subunit* Δφ *and axial subunit translation* (*rise*) Δ*z*

$$f(r,\varphi,z) = f(r,\varphi+\Delta\varphi,z+\Delta z)$$



If filaments were perfectly flat within the ice layer, all EM projection images would constitute orthoaxial projections of the filament and the problem would be to find three orientation parameters for each segment: angles ***phi*** and ***psi*** (***theta***=90) and translation along the main axis **z**



Ranges of orientation parameters:

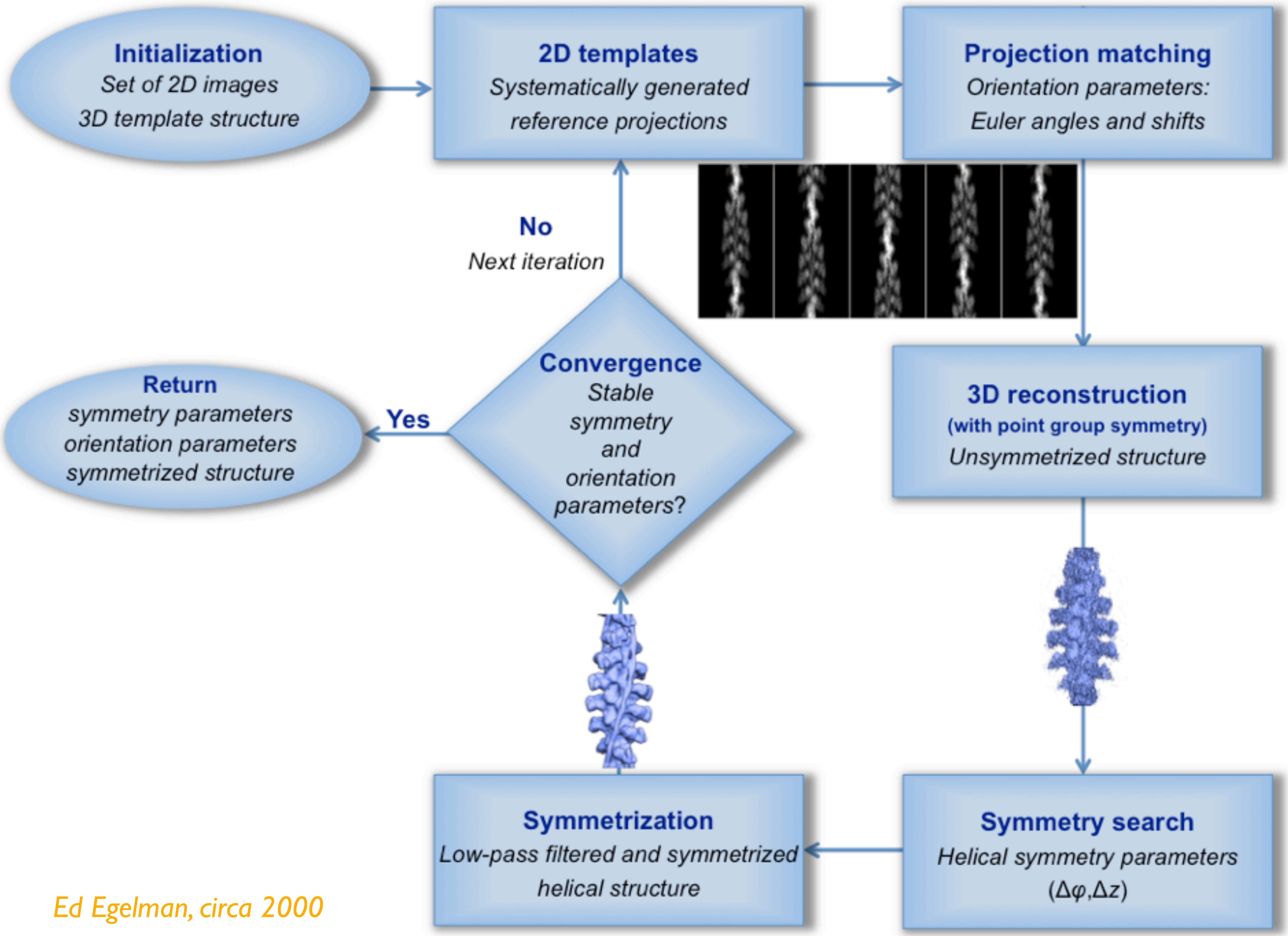
$$0 \le \varphi < 180$$

$$-\frac{\Delta z}{2} \le z < \frac{\Delta z}{2}$$

(mirror handled by the code)

$$-\Delta\psi < \psi < \Delta\psi$$

IHRSR



Ed Egelman, circa 2000

IHRSR implementation in SPARX

The new implementation differs from Ed's original one in SPIDER in many important ways:

1. new code offers more flexibility,
2. orientation searches are done in a sensible way,
3. point-group symmetries of helical filaments (Cn, Dn)

Δz

IHRSR implementation in SPARX

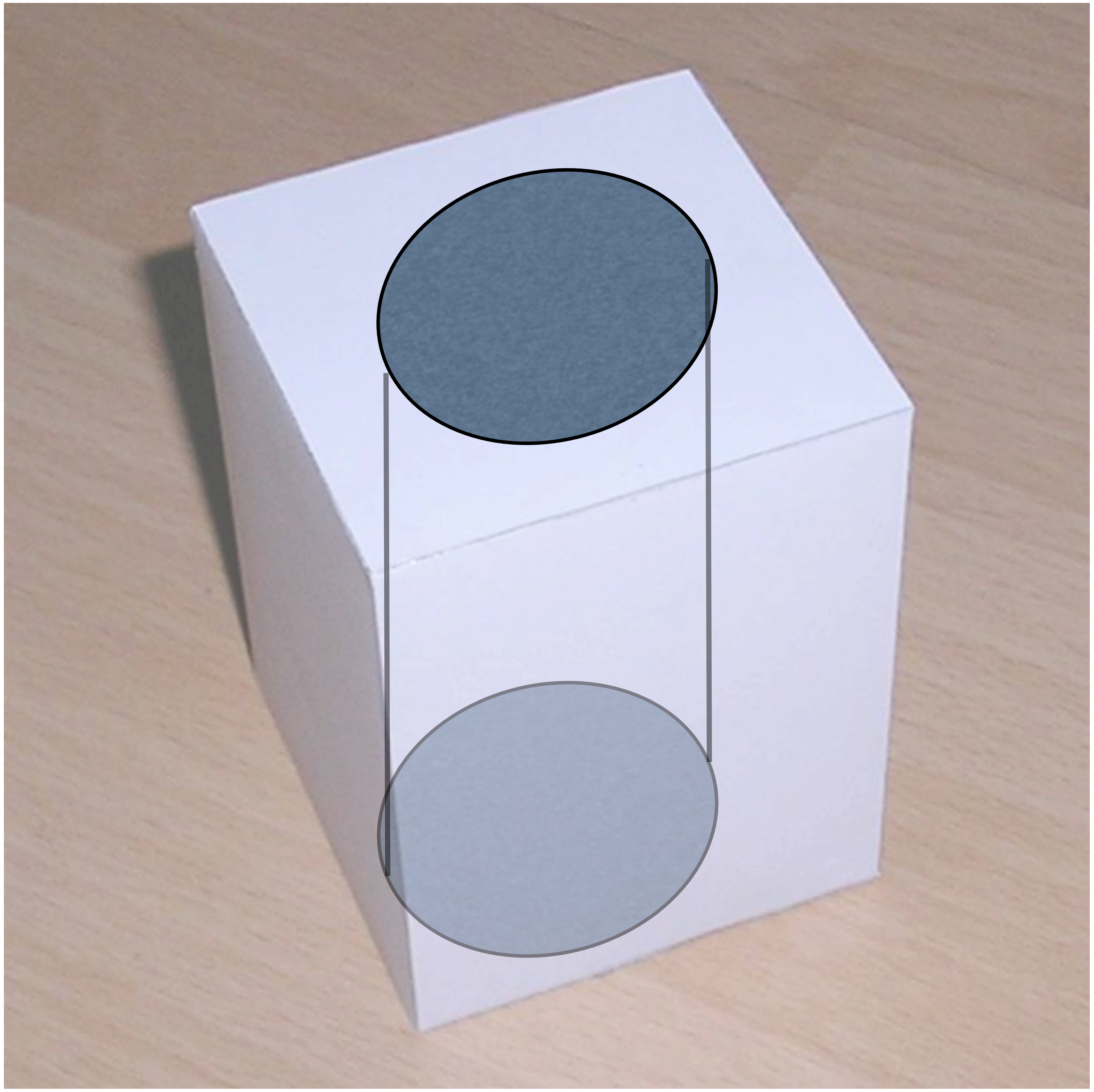
The new implementation differs from Ed's original one in SPIDER in many important ways:

1. new code offers more flexibility,
2. orientation searches are done in a sensible way,
3. point-group symmetries of helical filaments (Cn, Dn)

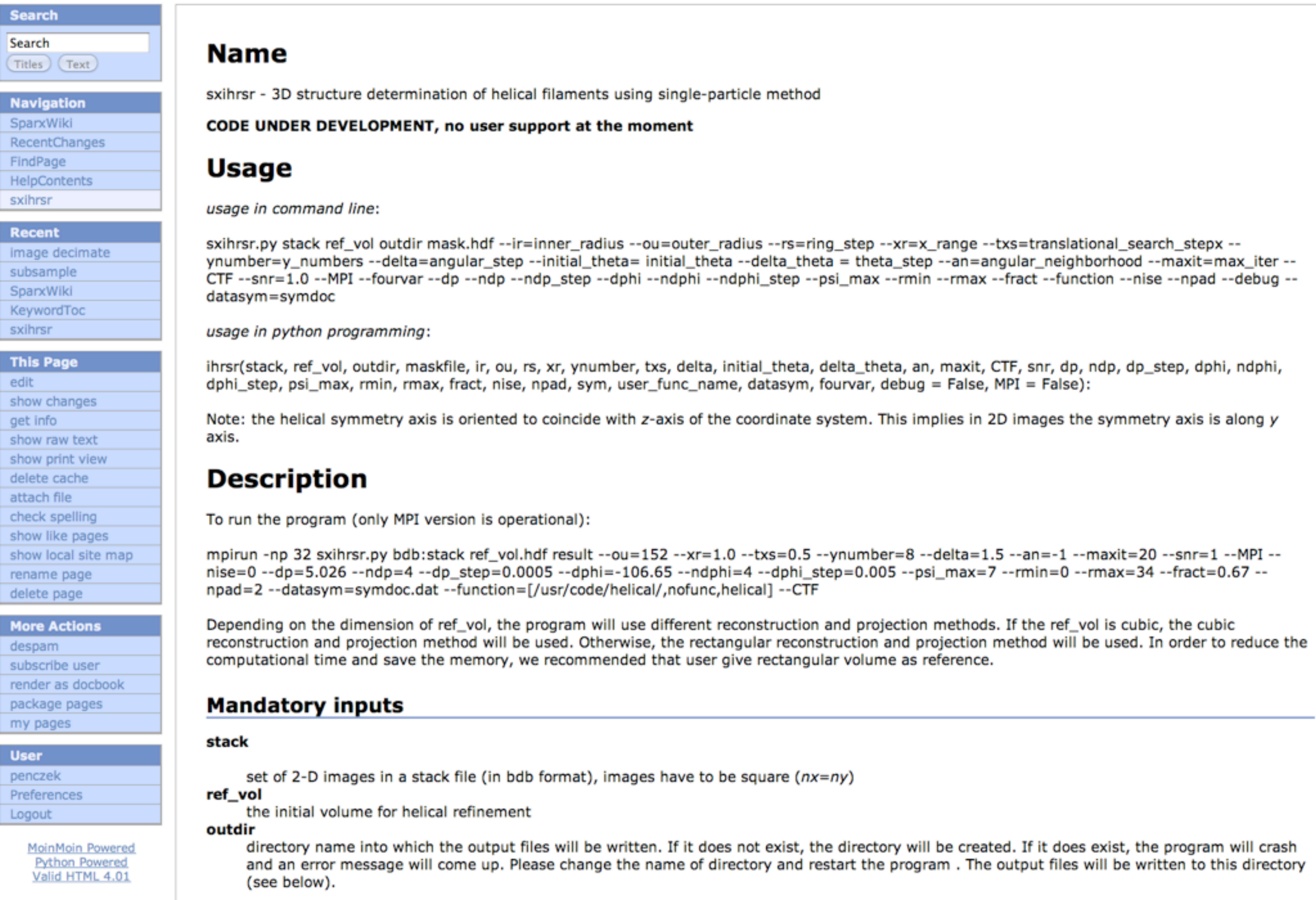
New features:

1. parallelization using python-level MPI makes it possible to execute the refinement rapidly on large clusters
2. restricted (constrained) search for in-plane rotation (psi) make the procedure more robust (segments are pre-aligned along z-axis)
3. search for translation restricted to axial rise Δz
4. search for helical symmetry implemented under MPI (it tends to be time consuming)
5. search for translation adapts itself to the current axial rise
6. out-of-plane tilt (theta <> 90) implemented!
7. 3D reconstruction and reprojections done within rectangular prism

Rectangular prism geometry saves computer memory and time of calculations.



Documentation:



Name

sxihrsr - 3D structure determination of helical filaments using single-particle method

CODE UNDER DEVELOPMENT, no user support at the moment

Usage

usage in command line:

```
sxihrsr.py stack ref_vol outdir mask.hdf --ir=inner_radius --ou=outer_radius --rs=ring_step --xr=x_range --txs=translational_search_stepx --ynumber=y_numbers --delta=angular_step --initial_theta= initial_theta --delta_theta = theta_step --an=angular_neighborhood --maxit=max_iter --CTF --snr=1.0 --MPI --fourvar --dp --ndp --ndp_step --dphi --ndphi --ndphi_step --psi_max --rmin --rmax --fract --function --nise --npad --debug --datasym=symdoc
```

usage in python programming:

```
ihrsr(stack, ref_vol, outdir, maskfile, ir, ou, rs, xr, ynumber, txs, delta, initial_theta, delta_theta, an, maxit, CTF, snr, dp, ndp, dp_step, dphi, ndphi, dphi_step, psi_max, rmin, rmax, fract, nise, npad, sym, user_func_name, datasym, fourvar, debug = False, MPI = False):
```

Note: the helical symmetry axis is oriented to coincide with *z*-axis of the coordinate system. This implies in 2D images the symmetry axis is along *y* axis.

Description

To run the program (only MPI version is operational):

```
mpirun -np 32 sxihrsr.py bdb:stack ref_vol.hdf result --ou=152 --xr=1.0 --txs=0.5 --ynumber=8 --delta=1.5 --an=-1 --maxit=20 --snr=1 --MPI --nise=0 --dp=5.026 --ndp=4 --dp_step=0.0005 --dphi=-106.65 --ndphi=4 --dphi_step=0.005 --psi_max=7 --rmin=0 --rmax=34 --fract=0.67 --npad=2 --datasym=symdoc.dat --function=[/usr/code/helical/,nofunc,helical] --CTF
```

Depending on the dimension of ref_vol, the program will use different reconstruction and projection methods. If the ref_vol is cubic, the cubic reconstruction and projection method will be used. Otherwise, the rectangular reconstruction and projection method will be used. In order to reduce the computational time and save the memory, we recommended that user give rectangular volume as reference.

Mandatory inputs

stack
: set of 2-D images in a stack file (in bdb format), images have to be square (*nx=ny*)

ref_vol
: the initial volume for helical refinement

outdir
: directory name into which the output files will be written. If it does not exist, the directory will be created. If it does exist, the program will crash and an error message will come up. Please change the name of directory and restart the program . The output files will be written to this directory (see below).

Windowing of filament segments from micrographs
http://blake.bcm.edu/emanwiki/EMAN2/Programs/e2helixboxer

Pawel Penczek Settings Logout

EMANWiki: EMAN2/ Programs/ e2helixboxer

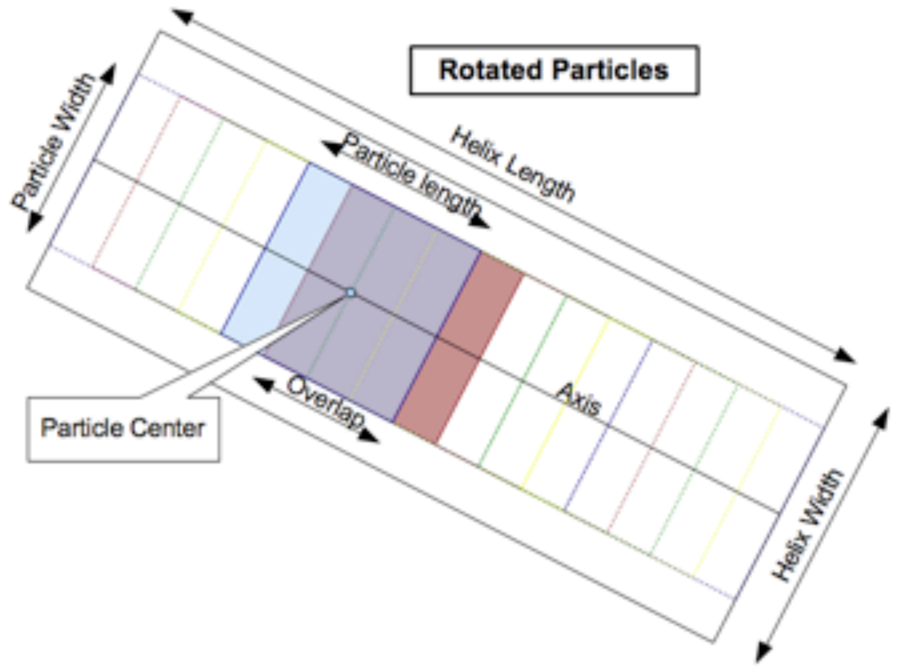
Ws2011/Agenda » COMPILE_EMAN2_MAC_OS_X » EMAN2/Programs/e2ctf » EMAN2/Programs » e2helixboxer

RecentChanges | FindPage | HelpContents | EMAN | EMAN1 | EMAN2 | EMEN2 | e2helixboxer

Edit Info Subscribe Add Link Attachments More Actions:

e2helixboxer.py: Overview

e2helixboxer.py is used to select rectangular 2D projections of helices from a micrograph, and extract overlapping particles from the boxed helices. The boxing must be done in GUI mode, but extracting particles from boxed regions may be do



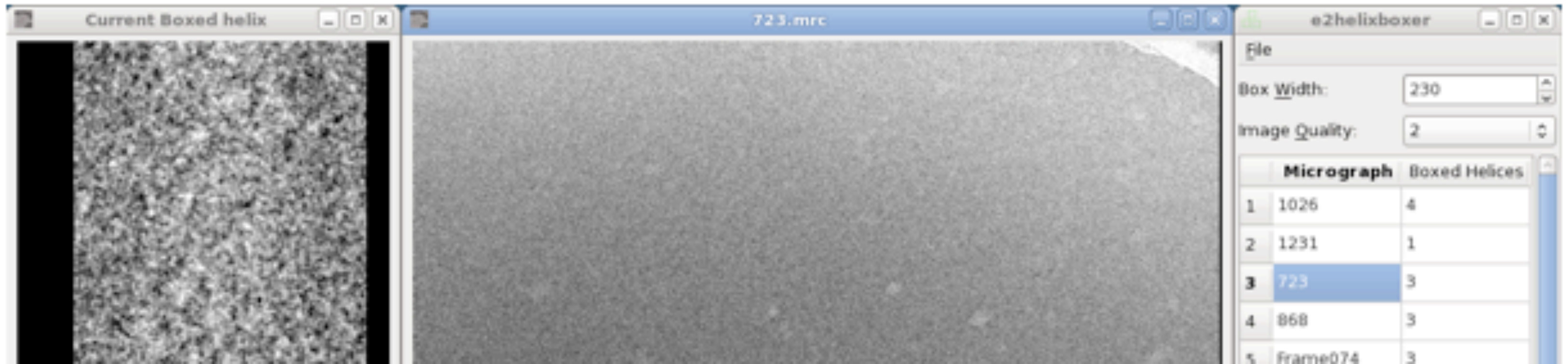
GUI mode

To start the program's graphic user interface, use the "--gui" option. You can follow this with a micrograph filename, a list of micrograph filenames, or nothing.

```
$ e2helixboxer.py --gui <micrograph1> <micrograph2> <...>
```

```
$ e2helixboxer.py --gui
$ e2helixboxer.py --gui 101.mrc
$ e2helixboxer.py --gui *.mrc micrograph.hdf *.img abc.dm3
```

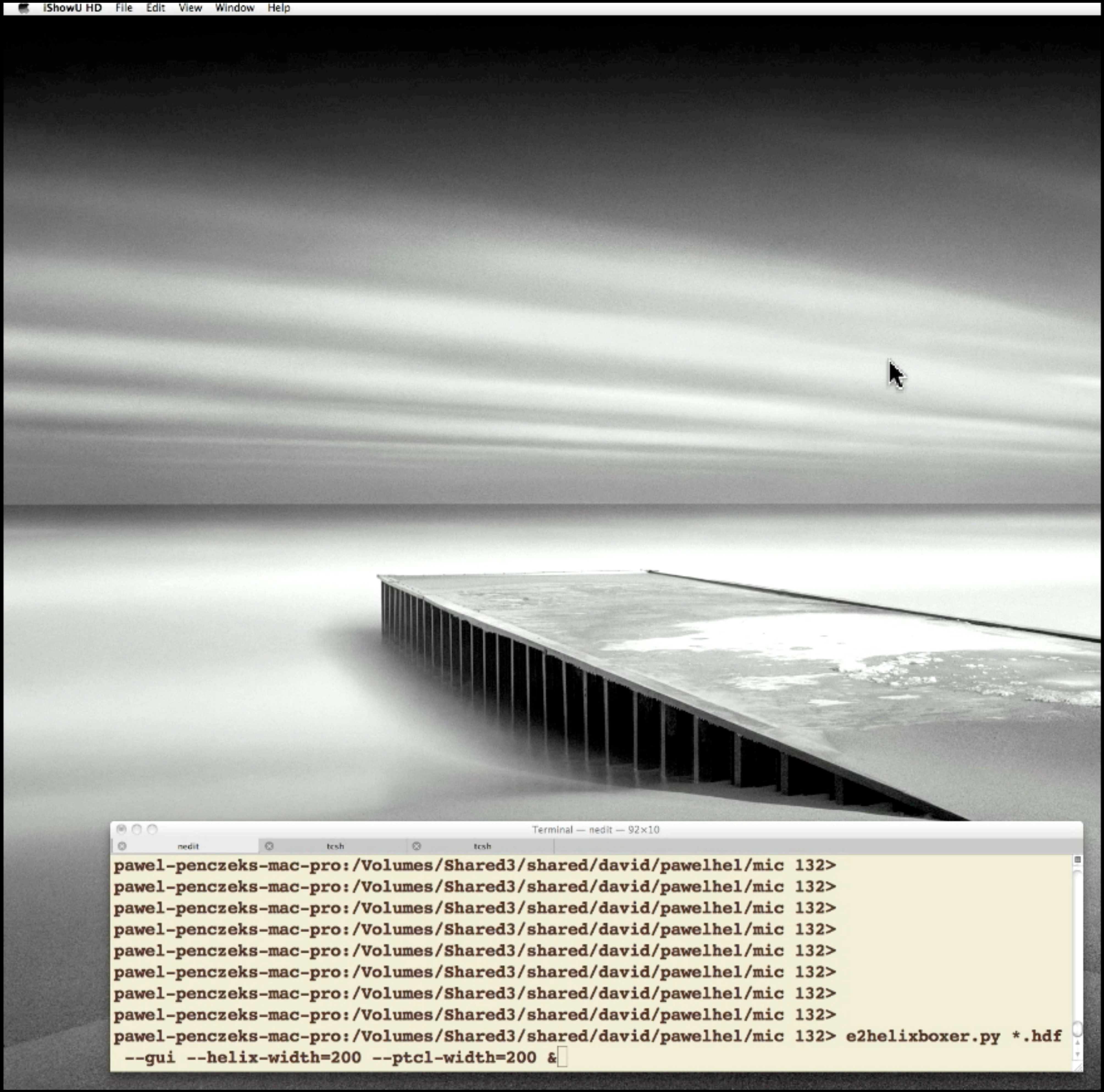
The left window is the helix viewer, the middle is the micrograph viewer, and the right is the main window.



Box Width: 230

Image Quality: 2

	Micrograph	Boxed Helices
1	1026	4
2	1231	1
3	723	3
4	868	3
5	Frame074	3



```
pawel-penczeks-mac-pro:/Volumes/Shared3/shared/david/pawelhel/mic 132>
pawel-penczeks-mac-pro:/Volumes/Shared3/shared/david/pawelhel/mic 132>
pawel-penczeks-mac-pro:/Volumes/Shared3/shared/david/pawelhel/mic 132>
pawel-penczeks-mac-pro:/Volumes/Shared3/shared/david/pawelhel/mic 132>
pawel-penczeks-mac-pro:/Volumes/Shared3/shared/david/pawelhel/mic 132>
pawel-penczeks-mac-pro:/Volumes/Shared3/shared/david/pawelhel/mic 132>
pawel-penczeks-mac-pro:/Volumes/Shared3/shared/david/pawelhel/mic 132>
pawel-penczeks-mac-pro:/Volumes/Shared3/shared/david/pawelhel/mic 132>
pawel-penczeks-mac-pro:/Volumes/Shared3/shared/david/pawelhel/mic 132> e2helixboxer.py *.hdf
 --gui --helix-width=200 --ptcl-width=200 &
```