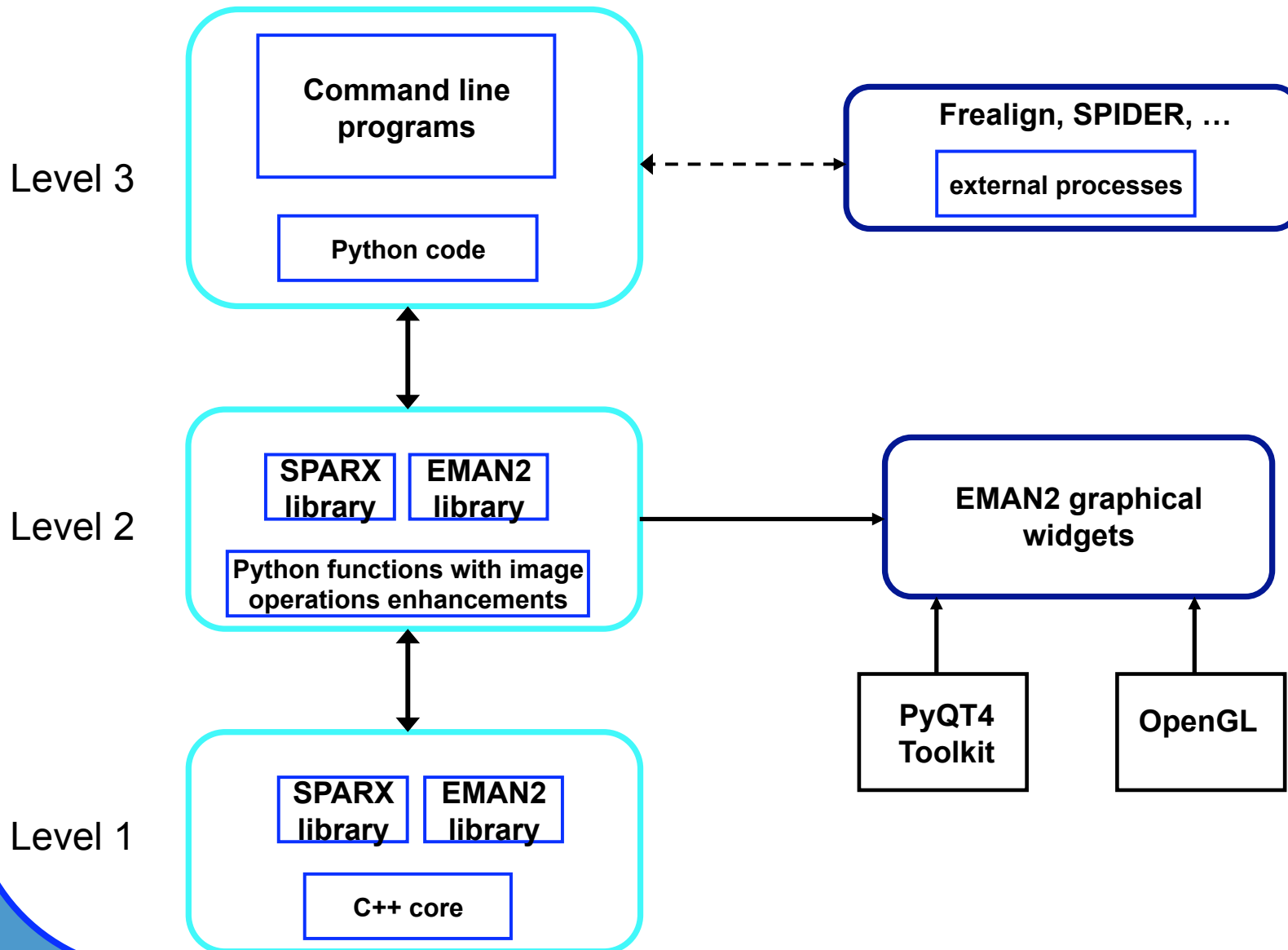SPARX

Diagram of the overall design of SPARX.
SPARX is built on top of several other toolkits including 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*

Search

Search

Titles   Text

Navigation

SparxWiki
RecentChanges
FindPage
HelpContents

Recent

info
sxali2d ra
sxali2d c
KeywordToc
SparxWiki

This Page

edit
show changes
get info
show raw text
show print view
delete cache
attach file
check spelling
show like pages
show local site map
rename page
delete page
Subscribe

More Actions

despam
subscribe user

Sparx Documentation Wiki

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

sparx book

User-editable Wiki-based documentation

http://macro-em.org/sparxwiki/SparxWiki

- Introduction
- Examples

sparx manual pages

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

How to?

- Download and install SPARX/EMAN2
- Frequently asked questions about installation
- Use SPARX
- Write/edit a manual page
- **Search**: use link http://macro-em.org/sparxwiki/FindPage
- Read and write images in SPARX/EMAN2 and how to handle image file attributes (necessary to run programs in SPARX)
- RunThroughExample
- Determine single particle structure using stain data
- Determine single particle structure using cryo data

Things To Do / Requests

- Requests

Done

Name

filt_gaussl - Gaussian low-pass Fourier filter

Usage

output = filt_gaussl(input, sigma)

Input

input
    input image (can be either real or Fourier)

sigma

    standard deviation of the Gaussian function in absolute frequency units . f_a

    Note: for sigma = 0.5 (the Nyquist frequency) the value of the filter at the maximum frequency is . $G(f_N) = \frac{1}{\sqrt{e}} = 0.61$

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 Gaussian filter:

$$G(f) = exp\left(-\frac{f^2}{2 * \sigma^2}\right)$$

Reference

Gonzalez, R. F., Woods, R. E., 2002. Digital Image Processing. Prentice Hall, Upper Saddle River, NJ.

Author / Maintainer

Pawel A. Penczek

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

Contributors

UTH, Houston
Pawel Penczek
Wei Zhang
Zhengfan Yang
Julien Bert

Baylor, Houston
Steve Ludtke
Phil Baldwin
Grant Tang
David Woolford

Charité, Berlin
Christian Spahn
Justus Loerke

Max Planck Institute, Dortmund
Stefan Raunser

What works?

❖ 2-D alignment.

❖ 2-D clustering, PCA.

❖ Multi-reference 2-D alignment.

❖ 3-D projection alignment, both coarse and fine.

❖ 3-D multi-reference alignment

❖ Real-space variance (bootstrap)

❖ Full cross-operability with EMAN2.

What is new?

❖ 2-D alignment (Yang 2008)

❖ Highly accurate 3-D projection alignment (Yang, 2008, Ultramicroscopy)

❖ Cluster analysis (Bert, in preparation)

❖ Fast 3-D reconstruction algorithm with CTF correction (Zhang, 2008, Structure)

❖ Real-space variance calculation using bootstrap (Zhang, 2008, Structure)

❖ Centering of averages

General description (1)

❖ Input image data is never modified, it resides in a database (bdb file format, hdf also supported) that also contains all CTF information and alignment parameters.

❖ Commands modify only alignment parameters:
2D alignment,
projection alignment,
3D, i.e., volume alignment.

sxheader.py bdb:data --print   --params=xform.proj
sxheader.py bdb:data --backup --params=xform.proj --suffix=_12_10_08
sxheader.py bdb:data --restore   --params=xform.proj_12_10_08

❖ It is possible to process subsets of images by setting active flag in headers

sxheader.py bdb:data --one --params="active"
sxheader.py bdb:data --params="active" --import=good_data

General description (2)

❖ Data can be processed with or without CTF correction. To use CTF correction, CTF information has to be present in headers and flag -–CTF has to be added.

❖ Most programs exist in:

1) single processor version – these are rather slow and of limited use. However, the code is transparent and easy to follow. Often, these versions lag behind (will not have most recent features).

sxali3d_d.py bdb:data output_directory

2) MPI version – recommended to run, even if machine has only one CPU.

mpirun –np 2 sxali3d_d.py bdb:data output_directory --MPI

❖ Most programs produce a logfile (name ending with a time stamp) that contain input parameters used and runtime information.

General description (3)

❖ Most programs are not entirely foolproof (automated). Not all steps can be entirely formalized or made to work for all possible datasets and needs.

To address this issue, alignment programs use an external function that prepares average for a next iteration of alignment. Examples are in user_functions.py, but user can write external functions that will reside outside of the system.

mpirun –np 2 sxali3d_d.py bdb:data output_directory --MPI --function=refi7

or

--function=[/home/justus, functions, refi7]
(i.e. refi7() defined in file /home/justus/functions.py)

   (for details please consult Justus: loerke@molgen.mpg.de)

Recommended reading

- Penczek, P.A., 2008. Single Particle Reconstruction, in: U. Shmueli, (Ed.), International Tables for Crystallography, 2008.

- Vainshtein, B.K., and P.A. Penczek, 2008. Three-dimensional reconstruction, in: U. Shmueli, (Ed.), International Tables for Crystallography, 2008.

Thank you!