SIMPLE 2017 Development Document

Responsibilities: Hans, Cyril, Dominika&Hans, Alex&Hans, Susan&Hans

RANDOM STUFF

* Pseudo symmetry refinement mode
* Make distributed flow for scale that operates on split stacks so that scaling can be done in parallel
* Random selection of images for prime2D init need to take state=0 into account
* Need to have another go at the picker (massively parallel mass-centring)
* When automask is turned on in 2D we must force the next round to search all refs, or it will go to shit
* Class for memory allocation simple\_alloc. Keep track of mem exceptions and profile memory (hash table).
* Go over all random number generations and see if we can use matrices instead
* Replace integer random number generator with intrinsic one (Guide to Fortran 2008)
* mailx -s "mail from prime2D" hans.elmlund@monash.edu < from\_prime2D
* It would be useful to write out a class-average param file that contained class average identifier and number of particles in it - then the ordering by population could be done as a visualisation thing within the gui rather than a genuine reordering of the stack
* It would be very helpful to have a sense of stage drift. Could unblur output an average value for shifts in x and y after processing an entire data set? Presumably ice movement should be somewhat random but what was left might be the stage?

STREAMING MODE (half boring)

* Relion-style input mode for substacks, one per movie (need two new programs (1) prime2Dstream in shared memory that has filetab input instead of stack and doclist, i.e. unidocs in replacement of deftab, (2) prime2Dstream in distr\_stream\_commander that monitors unidocs to identify a chunk of suitable size based on ctfres and df\_min/df\_max filter)
* unidoc: everything written to cwd
* same naming convention for picker as for others
* consider auto in name for autoboxes
* put in absolute paths whenever files are written out

HIGHRES (not so boring)

* particle update strategy: random selection of fixed number of particles to update (say 5000-20,000 depending on computer/data set size), need to save per-particle oris in binary files to be able to do this & consider mapping the weights from cavgs to ptcls
* Down-scaling/LP strategy: 10, 8, 6, 4, need distributed down-scaling workflow
* Getting the masking routine in shape (must investigate closer what Frealign/Relion are doing)
* FSC fudging (must investigate closer what Frealign/Relion are doing)
* Continuos refinement: does it move dynamically (needed for diversity)
* Continuos refinement: does it consider multiple “hot areas” & can we define these areas based on highly low-pass limited coarse grid search or should we pursue search history analysis?
* The new weighting strategy should be part of every 3D search mode (apart from the shc ones)

Refinement Considerations

Using the strategy pattern for the different modes of refinement?

When do we turn on neigh modes?

Should npeaks in refine=no/neigh modes always be at least 6?

HETEROGENEITY

* Automatic workflow het\_from\_cavgs: (1) refine=snhc (2) refine=no X 2 resolution updates, as in ini3D
* Weighted multi-resolution state sorting by fitting B-factors

NEW DEVELOPMENTS

* Probabilistic SO(3) scatter search for high-resolution refinement
* implement tilt test

EXCEPTION HANDLING

* exception handling class that makes more sensible outputs (especially when running the code in distributed mode). We need to create a database of exceptions and then have ONE control point where we check the stack for errors and report what has failed (similar to the command line dictionary)
* refs should be included in the mixed formats check
* when eo=yes is set the program should ask for fsc-file if missing & instruct better

Known bugs

None at the moment

Compilation

* port to ifort (Intel compiler)
* port to PGI (Portland group, with CUDA-FORTAN)

Documentation

Need to get a html code doc generator in place and fix the doc of every class. Will FORD source code documentation provide the solution?

<http://fortranwiki.org/fortran/show/FORD>

<https://github.com/cmacmackin/ford>

<http://jacobwilliams.github.io/json-fortran/index.html>

Books/webpages

* Structured parallel programming
* Structure and Interpretation of Computer Programs
* J-P Morrison. Flow-Based Programming: A New Approach to Application Developments. CreateSpace, 2nd ed.
* Mathematical Foundations of Imaging, Tomography and Wavefield Inversion
* Geometric Algebra for Computer Science (Revised Edition): An Object-Oriented Approach to Geometry (The Morgan Kaufmann Series in Computer Graphics) 1st Edition
* The Princeton Companion to Applied Mathematics
* Practical Machine Learning: http://www.computervisionmodels.com/
* Applied Stochastic Modelling, Second Edition (Chapman & Hall/CRC Texts in Statistical Science) 2nd Edition http://szeliski.org/Book/