The SIMPLE 3.0 Manual

Jan 24, 2017

Contributors:
cyril.reboul@monash.edu
dominika.elmlund@monash.edu
hans.elmlund@monash.edu
Adress:
Dept. Biochemistry and Molecular Biology
School of Biomedical Sciences
Monash University, Bldg. 77
Clayton, VIC, Australia, 3800
Webpage:
www.simplecryoem.com
Contact:
`http://simplecryoem.com/contact.html`
dominika@simplecryoem.com

"Keep it SIMPLE stupid"
(*Kelly Johnson*; lead engineer at the Lockheed Skunk Works, coined the famous KISS principle stating that systems work best if they are kept simple rather than made complex. Therefore, simplicity should be a key goal in design and unnecessary complexity should be avoided.)

"Everything should be made as SIMPLE as possible, but no SIMPLEr"
(*Albert Einstein*)

"Complex theories do not work, SIMPLE algorithms do"
(*Vladimir N. Vapnik*; author of *The Nature of Statistical Learning Theory*)

Contents

1 About SIMPLE **5**
1.1 What is new in SIMPLE release 3.0? . . . 5

2 File Formats **5**
2.1 Image File Formats . . . 5
2.2 SIMPLE Parameter File Format . . . 5
2.3 Parameter File Conversions . . . 6

3 Installation **6**
3.1 System requirements . . . 6
3.1.1 Hardware . . . 6
3.1.2 Software . . . 6
3.2 Compiling SIMPLE . . . 6
3.3 Compilation troubleshooting . . . 8
3.3.1 FFTW . . . 8
3.4 Installation on a Linux Cluster . . . 8
3.5 Testing the SIMPLE Installation . . . 9
3.6 SIMPLE Help Tools . . . 9

4 Workflows **10**
4.1 Distributed Execution of SIMPLE on Multi-socket Workstations and Clusters . . . 10
4.2 From Movies to Structure . . . 12
4.3 Unresolved Bugs . . . 12

5 Command Line Dictionary **14**

6 SIMPLE Programs **17**
6.1 Program: automask2D . . . 17
6.2 Program: automask3D . . . 18
6.3 Program: binarise . . . 18
6.4 Program: boxconvs . . . 18
6.5 Program: cavgassemble . . . 19
6.6 Program: cenvol . . . 19
6.7 Program: check2D_conv . . . 19
6.8 Program: check3D_conv . . . 20
6.9 Program: check_box . . . 20
6.10 Program: check_nptcls . . . 21
6.11 Program: cluster_oris . . . 21
6.12 Program: cluster_smat . . . 21
6.13 Program: comlin_smat . . . 21
6.14 Program: cont3D . . . 22
6.15 Program: convert . . . 22
6.16 Program: corrcompare . . . 23
6.17 Program: ctffind . . . 23
6.18 Program: ctfops . . . 23
6.19 Program: eo_volassemble . . . 24
6.20 Program: extract . . . 24
6.21 Program: filter . . . 25
6.22 Program: find_nnimgs . . . 25
6.23 Program: het_init . . . 25
6.24 Program: image_smat . . . 26
6.25 Program: iminfo . . . 26
6.26 Program: integrate_movies . . . 26

6.27 Program: makedeftab . . . 27
6.28 Program: makeoris . . . 27
6.29 Program: map2ptcls . . . 28
6.30 Program: mask . . . 28
6.31 Program: masscen . . . 29
6.32 Program: merge_algndocs . . . 29
6.33 Program: merge_nnmat . . . 29
6.34 Program: merge_shellweights . . . 30
6.35 Program: merge_similarities . . . 30
6.36 Program: multiptcl_init . . . 30
6.37 Program: noiseimgs . . . 31
6.38 Program: norm . . . 31
6.39 Program: npeaks . . . 31
6.40 Program: nspace . . . 31
6.41 Program: orisops . . . 32
6.42 Program: oristats . . . 33
6.43 Program: pick . . . 33
6.44 Program: postproc_vol . . . 33
6.45 Program: powerspecs . . . 34
6.46 Program: prime2D . . . 34
6.47 Program: prime2D_init . . . 35
6.48 Program: prime3D . . . 35
6.49 Program: prime3D_init . . . 36
6.50 Program: print_cmd_dict . . . 37
6.51 Program: print_dose_weights . . . 37
6.52 Program: print_fsc . . . 37
6.53 Program: projvol . . . 38
6.54 Program: rank_cavgs . . . 38
6.55 Program: recvol . . . 39
6.56 Program: res . . . 39
6.57 Program: respimg . . . 39
6.58 Program: resrange . . . 40
6.59 Program: rotmats2oris . . . 40
6.60 Program: scale . . . 41
6.61 Program: select . . . 41
6.62 Program: select_frames . . . 41
6.63 Program: shellweight3D . . . 42
6.64 Program: shift . . . 42
6.65 Program: simimgs . . . 42
6.66 Program: simmovie . . . 43
6.67 Program: simsubtomo . . . 44
6.68 Program: split . . . 44
6.69 Program: split_pairs . . . 44
6.70 Program: stack . . . 45
6.71 Program: stackops . . . 45
6.72 Program: symsrch . . . 46
6.73 Program: tseries_split . . . 46
6.74 Program: unblur_movies . . . 47
6.75 Program: volassemble . . . 47
6.76 Program: volaverager . . . 48
6.77 Program: volops . . . 48
6.78 Program: volume_smat . . . 48

7 Distributed SIMPLE Workflows **49**
7.1 Program: ctffind . . . 49
7.2 Program: find_nnimgs . . . 49
7.3 Program: ini3D_from_cavgs . . . 49
7.4 Program: prime2D . . . 50
7.5 Program: prime2D_init . . . 51
7.6 Program: prime3D . . . 51
7.7 Program: prime3D_init . . . 52
7.8 Program: recvol . . . 53
7.9 Program: shellweight3D . . . 54
7.10 Program: unblur_movies . . . 54
7.11 Program: unblur_tomo_movies . . . 55

1 About SIMPLE

Single-particle **IM**age **P**rocessing **L**inux **E**ngine (SIMPLE) is a program package for cryo-EM image processing, focusing on *ab initio* 3D reconstruction of low-symmetry single-particles. The SIMPLE back-end consists of an object-oriented numerical library written in modern Fortran. The SIMPLE front-end consists of a few standalone, interoperable components developed according to the "Unix toolkit philosophy".

SIMPLE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the license, or (at your option) any later version. SIMPLE is distributed with the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

1.1 What is new in SIMPLE release 3.0?

- A new DDD movie pre-processing program `unblur_movies` that implements motion correction and automatic correlation-based frame weighting.
- A new Fourier shell-weighting scheme for resolution-dependent data weighting that when used in conjunction with Wiener restoration in `prime2D` and `prime3D` enables near-atomic resolution *ab initio* 3D reconstruction.
- High-level workflows for 2D analysis and initial 3D model production.

2 File Formats

2.1 Image File Formats

SIMPLE supports SPIDER (`*.spi`) and MRC (`*.mrc`) formats for image stacks and volumes. The MRC file handling classes are shared with the the FREALIX program for helical reconstruction (Rohou and Grigorieff, 2014). RELION (Scheres, 2012) uses the convention that MRC stacks have the suffix `*.mrcs` and volumes the suffix `*.mrc`. This is to overcome the annoyance that it is not possible to tell from an MRC file header whether a MRC file is a volume or a stack. With SIMPLE you can select to use either the `*.mrcs` or `*.mrc` suffix for stacks. The way that we keep track of whether a file is a volume or stack is via the command line key value. The key-value pairs `vol1=rec.mrc` and `vol2=rec2.mrc` refer to volumes whereas the key-value pairs `stk=ptcls.mrc` and `stk2=ptcls2.mrc` refer to stacks.

2.2 SIMPLE Parameter File Format

The SIMPLE text-files used for parameter input/output use a `key=value` syntax of the form

```
e1=80. e2=100. e3=5.5 x=1.23 y=4.25 dfx=2.56 dfy=2.54 angast=30.5 state=1
```

to represent per-particle information. Internally, the orientation information is stored in a dynamic hash data structure, which gives the file format high flexibility. Therefore, writing conversion scripts to allow interchange of parameters between SIMPLE and other packages is easy. SIMPLE uses the same conventions as FREALIGN (Grigorieff, 2007) to represent orientations and CTF parameters. The CTF parameterisation obtained by CTFFIND (Mindell and Grigorieff, 2003) can be directly plugged into SIMPLE, for example by creating a file `deftab.txt`, looking like

```
kv=300 cs=2.7 fraca=0.07 dfx=2.56 dfy=2.76 angast=30.5
kv=300 cs=2.7 fraca=0.07 dfx=3.50 dfy=3.33 angast=60.0
kv=300 cs=2.7 fraca=0.07 dfx=1.98 dfy=2.02 angast=120.5
...
```

and adding `deftab=deftab.txt` and `ctf=yes` to the PRIME command line (if the images are phase-flipped, this should be indicated by `ctf=flip`). Note that we now include `kv`, `cs`, and `fraca` in the document listing the CTF parameters. Having all the CTF parameters listed per particle allows easy merging of data sets from different microscopes. SIMPLE now implements a wrapper program for CTFFIND version 4.1.X and newer, producing a SIMPLE conforming document by executing CTFFIND in parallel (via `simple_distr_exec prg=ctffind`). If you have obtained CTF parameters with CTFFIND by other means (via RELION, for example) you can provide a plain text file with the two defocus values and the angle of astigmatism to the SIMPLE program `makedeftab` and it will take care of the formatting and unit conversions for you.

2.3 Parameter File Conversions

The `SIMPLE/scripts` folder contains a perl-script (`convert_frealign2simple.pl`) to convert a Frealing parameter file to a SIMPLE parameter file. This is easy, since both software internally use the **Spider Euler angle convention**. Other packages may use other conventions. There's also a `convert_relion2simple.pl` for extracting CTF parameters from RELION `*.star` files and a `relion2emanbox.pl` script for converting box files obtained with RELION to the EMAN `*.box` format used by SIMPLE.

3 Installation

3.1 System requirements

3.1.1 Hardware

CPU
- Linux
- MacOSX (Yosemite and El Capitan, *i.e.* 10.10 and above)

3.1.2 Software

CPU
- GNU tool chain 4.9 and above.
- FFTW-3 (The Fastest Fourier Transform in the West library)

3.2 Compiling SIMPLE

To check the compiler version, execute

```
@!#> gfortran --version
GNU Fortran (GCC) 4.9.1
Copyright (C) 2014 Free Software Foundation, Inc.
```

To check whether the FFTW library is installed, open the package manager installed on our system (in our case `Synaptic` but other systems may have others, such as `YaST`, `apt` etc.) and search for `fftw` and see that the `libfftw3-single-3` is installed on the system. We `cd` to the directory where we have our software installed (`<my software location>`) and copy the tar ball there

```
@!#> cd <my software location>
@!#> cp ~/Downloads/simple.tar.gz .
```

Next, we unpack the tar ball and `cd` to the simple directory

```
@!#> gunzip simple.tar.gz
@!#> tar -xvf simple.tar
@!#> cd simple/
```

We open the `simple_user_input.pm` file in our favourite text editor. There are a few lines that needs to be edited:

```
...
26  # enter the SIMPLE root path
27  our$SIMPLE_PATH="/Users/hael/src/fortran/simple";
...
```

Line 27 `our$SIMPLE_PATH = "/Users/hael/src/fortran/simple/"` is replaced with `our$SIMPLE_PATH="<my software location>/simple/"`. This is the path in which the software will be installed.

```
...
46  our$FCOMPILER = "gfortran-5";
...
```

Line 46 `our$FCOMPILER = "gfortran-5";` is the command that executes the `gfortran` compiler. On a mac, using the fink package manager, this command would be `/sw/bin/gfortran`.

```
...
49  # enter the fftw lib default: /usr/lib/x86_64-linux-gnu for [linux]
50  #                             /usr/local/lib for [MacOSX]
51  our$FFTW_LIB="/usr/lib/x86_64-linux-gnu";
52  # on clusters we need extra path after module load fftw/3.3.4-gcc
53  our$FFTW_INC="/usr/include/";
...
```

Lines 51 and 53 specify the FFTW library and include directories. On a mac, using the fink package manager, these locations would be `our$FFTW_LIB="/sw/lib";` and `our$FFTW_INC="/sw/include/";`, respectively. We save and close the configuration file and execute:

```
@!#> ./compileSIMPLE.pl
*********************************************************
* Checking and printing the input directories...        *
*********************************************************
SIMPLE_PATH          : /Users/hael/src/fortran/simple
SIMPLE_SRC_PATH      : /Users/hael/src/fortran/simple/src/simple_main
SIMPLE_PROD_PATH     : /Users/hael/src/fortran/simple/production
SIMPLE_TEST_PROD_PATH: /Users/hael/src/fortran/simple/production/simple_tests
SIMPLE_SCRIPTS_PATH  : /Users/hael/src/fortran/simple/scripts
*********************************************************
Moving to dir: /Users/hael/src/fortran/simple/src/simple_main
Executing simple_args_generator.pl in dir: /Users/hael/src/fortran/simple...
Moving to dir: /Users/hael/src/fortran/simple
Generating Makefile_macros: /Users/hael/src/fortran/simple
/sw/bin/gfortran -c -fimplicit-none -fall-intrinsics -ffree-form -cpp -fpic...
...
darwin, Platform = 0
Architecture: darwin-thread-multi-2level

Moving to dir: /Users/hael/src/fortran/simple/production
Generating compile_and_link: /Users/hael/src/fortran/simple/production
Moving to dir: /Users/hael/src/fortran/simple
Compiling production codes:
>>> COMPILING & LINKING: simple_test_shelliter
>>> COMPILING & LINKING: simple_test_autoscale
```

```
>>> COMPILING & LINKING: simple_test_scatsrch
>>> COMPILING & LINKING: simple_test_ft_expanded
>>> COMPILING & LINKING: simple_exec
>>> COMPILING & LINKING: simple_test_volpft_srch
>>> COMPILING & LINKING: simple_test_srch
>>> COMPILING & LINKING: simple_test_units
>>> COMPILING & LINKING: simple_test_cartcorr_sanity
>>> COMPILING & LINKING: simple_test_imgfile
>>> COMPILING & LINKING: simple_distr_exec
Compilation of SIMPLE completed in dir: /Users/hael/src/fortran/simple
```

Compilation was succesful and we see that a new directory `/bin` has been created in the simple directory in addition to two text files `add2.bashrc` and `add2.tcshrc`

```
@!#> cat add2.bashrc
export SIMPLE_EMAIL="my.name@uni.edu"
export SIMPLE_QSYS="local"
export SIMPLE_PATH=/Users/hael/src/fortran/simple
export PATH=${SIMPLE_PATH}/scripts:${SIMPLE_PATH}/bin:$PATH
@!#> cat add2.tcshrc
setenv SIMPLE_EMAIL="my.name@uni.edu"
setenv SIMPLE_QSYS="local"
setenv SIMPLE_PATH /Users/hael/src/fortran/simple
set path=(${SIMPLE_PATH}/scripts ${SIMPLE_PATH}/bin $path)
```

On this machine we use the bash shell, so we add the lines in `add2.bashrc` to the `.bashrc` file. This is done by using a text editor, such as `gedit`, to edit the hidden file `.bashrc` located in your home folder.

```
@!#> cd
@!#> gedit .bashrc
```

3.3 Compilation troubleshooting

3.3.1 FFTW

If not already installed, the FFTW-3 library needs to be installed. Most Linux package managers `YaST`, `Synaptic`, `apt-get` etc. provide the FFTW-3 library. On a Mac system the `Fink` and `Macports` package managers provide the FFTW-3 library or it can be obtained from: `http://www.fftw.org/install/mac.html`. SIMPLE relies on the single-precision FFTW library. Typically we will need to

```
@!#> ./configure --enable-floats --enable-threads
@!#> make
@!#> sudo make install
```

The `-enable floats -enable-threads` directives are critical as SIMPLE uses multi-threaded single-precision Fourier transforms. We check that we have in the lib folder (typically: `/usr/local/lib/`):

```
libfftw3.a libfftw3.la
libfftw3f.a libfftw3f.la
libfftw3f_threads.a libfftw3f_threads.la
```

3.4 Installation on a Linux Cluster

Installation on a Linux cluster is essentially the same as on a Linux workstation with the exception that the appropriate modules need to be loaded before installation and execution. On a typical SLURM cluster

```
@!#> module load fftw/3.3.4-gcc
@!#> module load gcc/4.9.1
@!#> module load lapack/3.4.2
```

The instructions for how to execute SIMPLE in distributed environments (clusters or workstations with more than one CPU socket) are described below .

3.5 Testing the SIMPLE Installation

<2 be inputted>

3.6 SIMPLE Help Tools

In attempt to reduce the dependency on the manual, we have packaged a lot of the documentation in the software itself. For example

```
@!#> simple_exec prg=list
automask2D
automask3D
binarise
boxconvs
cavgassemble
cenvol
check2D_conv
check3D_conv
...
```

lists all programs executed with `simple_exec` (shared-memory parallelisation) and

```
@!#> simple_distr_exec prg=list
ctffind
find_nnimgs
ini3D_from_cavgs
prime2D
prime2D_init
prime3D
prime3D_init
recvol
shellweight3D
unblur_movies
unblur_tomo_movies
```

lists all distributed workflows executed with `simple_distr_exec`. If you don't quite remember which program you are looking for but remember that it was called `prime` something, you could execute

```
@!#> simple_distr_exec prg=list | grep prime
prime2D
prime2D_init
prime3D
prime3D_init
```

If you want a description for a particular program, for example `prime2D`, execute

```
@!#> simple_distr_exec prg=prime2D describe=yes
is a reference-free 2D alignment/clustering algorithm adopted from the
prime3D probabilistic ab initio 3D reconstruction algorithm
```

To obtain a description of the what command line options are available, execute

```
@!#> simple_distr_exec prg=prime2D
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
stk    = particle stack with all images(ptcls.ext)
smpd   = sampling distance, same as EMANs apix(in A)
msk    = mask radius(in pixels)
ncls   = nr of clusters
ctf    = ctf flag(yes|no|flip)
nparts = nr of partitions in distributed exection
nthr   = nr of OpenMP threads{1}

OPTIONAL
ncunits   = number of computing units, can be < nparts {nparts}
deftab    = text file with CTF info(*.txt/*.asc)
refine    = refinement mode(no|shc|neigh|shcneigh|qcont|qcontneigh|shift){no}
refs      = initial2Dreferences.ext
oritab    = table (text file) of orientations(*.asc/*.txt)
hp        = high-pass limit(in A)
lp        = low-pass limit(in A)
cenlp     = low-pass limit for binarisation in centering
trs       = maximum halfwidth shift(in pixels)
automsk   = envelope masking(yes|no|cavg){no}
amsklp    = low-pass limit for envelope mask generation(in A)
inner     = inner mask radius(in pixels)
width     = falloff of inner mask(in pixels){10}
startit   = start iterating from here
maxits    = maximum number of iterations{500}
filwidth  = width of filament (in A)
srch_inpl = search in-plane degrees of freedom(yes|no){yes}
nnn       = number of nearest neighbors{500}
minp      = minimum cluster population
```

4 Workflows

4.1 Distributed Execution of SIMPLE on Multi-socket Workstations and Clusters

SIMPLE is executed either via `simple_exec`, which implements all individual SIMPLE programs and runs in shared-memory parallelisation mode, or via `simple_distr_exec` that implements higher-level workflows intended for distributed execution on workstations and clusters using a hybrid parallelisation model (distributed *and* shared memory). In cluster environments using a job scheduler (PBS and SLURM are supported by SIMPLE) the file `simple_distr_config.env` in the current working directory controls the execution.

```
bash-3.2$ cat simple_distr_config.env
# CONFIGURATION FILE FOR DISTRIBUTED SIMPLE EXECUTION

# ABSOLUTE PATH TO SIMPLE ROOT DIRECTORY
simple_path           = /scratch/m3earlyadopters/simple/simple_devel/

# ESTIMATED TIME PER IMAGE (IN SECONDS)
```

```
time_per_image          = 400

# USER DETAILS
user_account            =
user_email              = hans.elmlund@monash.edu
user_project            =

# QSYS DETAILS (qsys_name=<local|slurm|pbs>)
qsys_name               = slurm
qsys_partition          = m3a
qsys_qos                =
qsys_reservation        = simple

# JOB DETAILS
job_ntasks              = 1
job_memory_per_task     = 32000
job_name                = dummy
job_ntasks_per_socket   = 1
```

If you need help configuring distributed SIMPLE execution, please file a help ticket on the webpage. The two most important parameters for distributed execution is the number of partitions `nparts` and the number of shared-memory CPU threads `nthr`. To check the number of processors on a linux system, execute `nproc` in the terminal. To obtain maximum performance, we need to select sensible values for the `nthr` and `nparts` parameters, which requires some knowledge about the computer architecture that we are running SIMPLE on. Consider a heterogeneous cluster with N nodes, two CPU sockets per node and six CPUs per socket. It rarely pays off

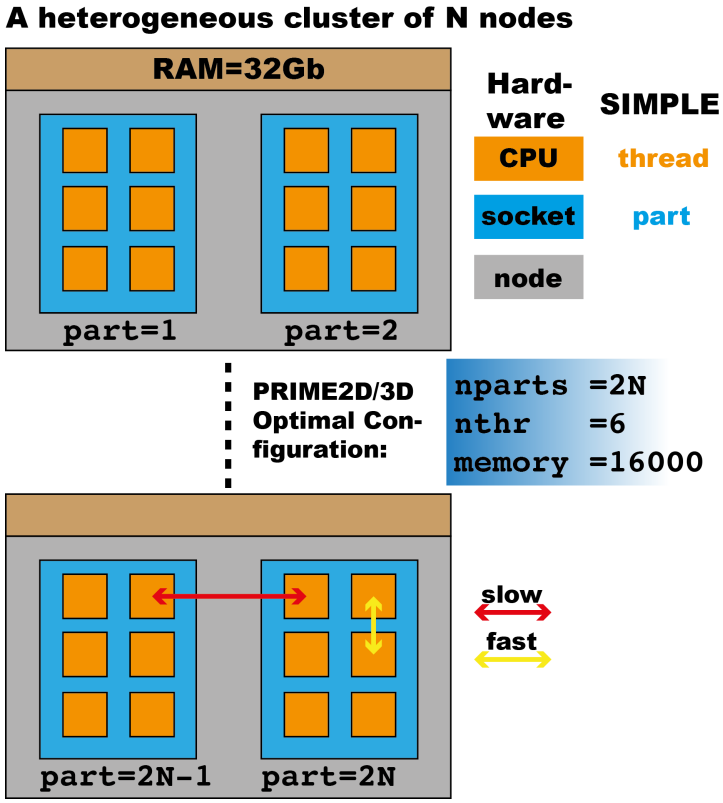

Figure 1: Configuration of the parallel PRIME-2D/3D execution on a heterogeneous cluster. We here represent the nodes in a heterogeneous cluster by two sockets with six CPUs each and 32Gb RAM/node. The best performance of PRIME–2D/3D is going to be obtained by partitioning the jobs into `npart=2N` partitions, where `N` is the number of nodes. Each partition will then execute six threads `nthr=6` and these six threads will get access to half the RAM on the node (`memory=16000`) because we have two sockets per node that need to share the RAM between them

to increase the number of shared-memory CPU threads `nthr` above 10 because the overhead for thread creation with respect to the gain in parallelism is unfavourable. Therefore, if you have a system with say 20 CPUs per socket, then execute 4N partitions instead of increasing `nthr` to 20. This is a bit technical, but it matters for performance so if you are unsure how to configure your SIMPLE execution please file a help ticket.

We normally let simple_distr_exec run in the background on the login node of our cluster. An example of how to distribute prime2D using ten nodes is provided below.

```
@!#> nohup simple_distr_exec prg=prime2D stk=ptcls.mrc smpd=1.77 msk=100
ncls=600 nthr=8 nparts=10 >> PRIME2DOUT &
```

Another option available on clusters that use the SLURM scheduler is to use the `srun` command for simple_distr_exec via

```
@!#> srun --ntasks=1 --ntasks-per-socket=1 --cpus-per-task=1 --mem=32000
--time=2-0:0:0 --output=PRIME2DOUT.%j --error=PRIME2DERR.%j
simple_distr_exec prg=prime2D stk=ptcls.mrc smpd=1.77 msk=100
ncls=600 nthr=8 nparts=10 &
```

However, beta testers have reported that srun job sometimes dies with no warning, possibly because of the low tolerance for network errors. A more robust route may be to use `sbatch` as follows

```
@!#>  sbatch -p MYCLUSTER --wrap="simple_distr_exec prg=prime2D stk=ptcls.mrc
smpd=1.77 msk=100 ncls=600 nthr=8 nparts=10 >> PRIME2DOUT"
```

where the `-wrap` flag automatically generates a bash script for the given command.

4.2 From Movies to Structure

These steps describe a typical SIMPLE workflow.

1. DDD (Direct Detector Device) movie alignment and frame-weighting using SIMPLE program unblur_movies
2. CTF parameter identification with the SIMPLE program ctffind, wrapping CTFFIND4 (Rohou and Grigorieff, 2015)
3. Particle identification using EMAN2 (Tang et al., 2007) to generate `*.box` files
4. Particle extraction with SIMPLE program extract
5. 2D analysis using SIMPLE program prime2D
6. *Ab initio* 3D reconstruction from class averages using the SIMPLE ini3D_from_cavgs high-level workflow
7. Mapping of class average selection and 3D class orientations to the particles using SIMPLE program map2ptcls
8. Reconstruction of a 3D map from the individual particle images with SIMPLE program recvol
9. Map refinement using SIMPLE program prime3D

For hands-on descriptions for how to execute the individual steps, please refer to the documentation of each program (below) and the SIMPLE workshop material, available at (INSERTREF).

4.3 Unresolved Bugs

On certain systems the file-system and/or the master node becomes overloaded by the volassemble step when executing prime3D and gives odd errors. For example, it fails to open the submit script files for writing and then goes on trying to submit non-existing scripts. We have tried to add a linux `sync` command in the `qsys_cleanup` subroutine and we are flushing all the file-handles every iteration, but the error nevertheless occurs. The problem has been nailed down to a being volassemble-related, as if the volassemble step is taken out and the current map is simply copied to the `vol_iter` name, the problem never occurs. The error looks as follows

```
System error        4096  for command: echo DISTRIBUTED MODE :: submitting scripts:ls -1
distr_simple_script_*REPLACEWRITEsimple_qsys_ctrl :: submit_scripts#!/bin/bashdistr_
simple_script_simple_qsys_ctrl :: gen_qsys_script; Error when opening file for writing:  ;
qsys_namelocalyescd cmdstat /= 0, command could not be executed: echo DISTRIBUTED
MODE :: submitting scripts:ls -1 distr_simple_script_*REPLACEWRITEsimple_qsys_ctrl ::
submit_scripts#!/bin/bashdistr_simple_script_simple_qsys_ctrl :: gen_qsys_script; Error
when opening file for writing:  ; qsys_namelocalyescd
Termination status of the command-language interpreter cannot be obtained
System error        4096  for command: ls -1 distr_simple_script_*REPLACEWRITE
simple_qsys_ctrl :: submit_scripts#!/bin/bashdistr_simple_script_simple_qsys_ctrl ::
gen_qsys_script; Error when opening file for writing:  ; qsys_namelocalyescd   > OUT
simple_qsys_scripts :: gen_qsys_scri cmdstat /= 0, command could not be executed:
ls -1 distr_simple_script_*REPLACEWRITEsimple_qsys_ctrl :: submit_scripts
#!/bin/bashdistr_simple_script_simple_qsys_ctrl :: gen_qsys_script; Error when opening
file for writing:  ; qsys_namelocalyescd   > OUTsimple_qsys_scripts :: gen_qsys_scri
Termination status of the command-language interpreter cannot be obtained
```

and when it occurs, you need to restart prime3D from the last completed iteration.

5 Command Line Dictionary

`acf`	calculate autocorrelation function(yes|no){no}
`amsklp`	low-pass limit for envelope mask generation(in Å)
`angastunit`	angle of astigmatism unit (radians|degrees){degrees}
`angerr`	angular error(in degrees){0}
`append`	append in context of files(yes|no){no}
`astigerr`	astigmatism error(in microns)
`astigstep`	step size for astigamtism search(in microns)
`athres`	angular threshold(in degrees)
`automsk`	envelope masking(yes|no|cavg){no}
`avg`	calculate average(yes|no)
`bfac`	bfactor for sharpening/low-pass filtering(in Å**2){200.}
`bfacerr`	bfactor error in simulated images(in Å**2){0}
`bin`	binarise image(yes|no){no}
`binwidth`	binary layers grown for molecular envelope(in pixels){1}
`box`	square image size(in pixels)
`boxconvsz`	size of box used for box-convolution(in pixels)
`boxtab`	table (text file) of files with EMAN particle coordinates(*.txt/*.asc)
`cenlp`	low-pass limit for binarisation in centering
`center`	center image(s)(yes|no){no}
`chunksz`	number of images/orientations in chunk
`class`	cluster identity
`clip`	clipped image box size(in pixels)
`clustvalid`	validate clustering(yes|homo|no){no}
`comlindoc`	shc_clustering_nclsX.txt
`compare`	do comparison(yes|no){no}
`corner`	corner size(in pixels){0}
`countvox`	count the number of voxels(yes|no){no}
`cs`	spherical aberration constant(in mm){2.7}
`ctf`	ctf flag(yes|no|flip)
`ctfsq`	apply ctf**2 to the images(yes|no){no}
`ctfsqspec`	filename of ctf**2 spectrum{ctfsqspec_state01.bin}
`ctfstats`	calculate ctf statistics(yes|no){no}
`cube`	side size(in pixels){0}
`dcrit_rel`	critical distance relative to box(0-1){0.5}
`defocus`	defocus(in microns){3.}
`deftab`	text file with CTF info(*.txt/*.asc)
`dferr`	defocus error(in microns){1.0}
`dfmax`	maximum expected defocus(in microns)
`dfmin`	minimum expected defocus(in microns)
`dfunit`	defocus unit (A|microns){microns}
`dir`	directory
`dir_reject`	move rejected files to here{rejected}
`dir_select`	move selected files to here{selected}
`discrete`	discrete(yes|no){no}
`diverse`	diverse or not flag (yes|no){no}
`doclist`	list of oritabs for different states
`dose_rate`	dose rate(in e/A2/s)
`dynlp`	automatic resolution limit update(yes|no){yes}
`e1`	1st Euler(in degrees){0}
`e2`	2nd Euler(in degrees){0}
`e3`	3d Euler(in degrees){0}
`edge`	edge size for softening molecular envelope(in pixels)
`endian`	endiannesss of files(big|little|native){native}

eo	use FSC for filtering and low-pass limit update(yes|no){no}
errify	introduce error(yes|no){no}
even	calculate even eo-pair(yes|no){no}
exp_doc	specifying exp_time and dose_rate per tomogram
exp_time	expusure time(in s)
expastig	expected astigmatism(in microns)
fbody	file body
filetab	list of files(*.txt/*.asc)
filwidth	width of filament (in Å)
find	Fourier index
fix_gpu	fix GPU execution to one device(yes|no){no}
fname	file name
frac	fraction of ptcls(0-1){1}
fraca	fraction of amplitude contrast used for fitting CTF{0.07}
fracdeadhot	fraction of dead or hot pixels{0.01}
frameavg	nr of frames to average{0}
fromf	start frame index
fromp	start ptcl index
fsc	binary file with FSC info{fsc_state01.bin}
ft2img	convert Fourier transform to real image of power(yes|no){no}
grow	number of binary layers to grow(in pixels)
guinier	calculate Guinier plot(yes|no){no}
hfun	function used for normalization(sigm|tanh|lin){sigm}
hist	give variable for histogram plot
hp	high-pass limit(in Å)
iares	integer angular resolution{10}
infile	table (text file) of inputs(*.asc/*.txt)
inner	inner mask radius(in pixels)
jumpsz	size of contigous segment
kv	acceleration voltage(in kV){300.}
label	discrete label(class|state){class}
lp	low-pass limit(in Å)
lpstart	start low-pass limit{15}
lpstop	stop low-pass limit{8}
masscen	center using binarisation and mass centering(yes|no){no}
maxits	maximum number of iterations{500}
minp	minimum cluster population
mirr	mirror(no|x|y){no}
moldiam	molecular diameter(in Å)
msk	mask radius(in pixels)
mskfile	maskfile.ext
msktype	type of mask(hard|soft){soft}
mul	origin shift multiplication factor{1}
mw	molecular weight(in kD)
ncls	nr of clusters
ncunits	number of computing units, can be $<$ nparts {nparts}
ndiscrete	nr of discrete orientations
ndocs	nr of documents
neg	invert contrast of images(yes|no)
newbox	new box size(in pixels)
nframes	number of frames{30}
nnn	number of nearest neighbors{500}
noise	noise initialisation(yes|no){no}

noise_norm	normalise based on sdev of background(yes|no){no}
norec	do not reconstruct volume(s)(yes|no){no}
norm	do statistical normalisation avg
nparts	nr of partitions in distributed exection
npeaks	nr of nonzero orientation weights{1}
npix	number of pixles/voxels in binary representation
nptcls	nr of images in stk/nr of orientations in oritab
nran	number of random images to select
nrepeats	nr of times to restart workflow{1}
nspace	nr of projection directions
nstates	nr of states to reconstruct
nthr	nr of OpenMP threads{1}
nthr_master	nr of OpenMP threads on master node{1}
numlen	length of number string
nvox	number of voxels{0}
odd	calculate odd eo-pair(yes|no){no}
offset	pxels offset{3}
opt	optimiser (powell|simplex|oasis|bforce|pso|de){simplex}
order	order ptcls according to correlation(yes|no){no}
oritab	table (text file) of orientations(*.asc/*.txt)
oritab2	2nd table (text file) of orientations(*.asc/*.txt)
outer	outer mask radius(in pixels)
outfile	output document
outside	only extract boxes within the micrograph boundaries(yes|no){yes}
outstk	output image stack
outvol	output volume{outvol.ext}
paramtab	per-micrograph parameters to transfer (CTF/movie shifts)
pgrp	point-group symmetry(cn|dn|t|o|i)
phaseplate	images obtained with phaseplate(yes|no){no}
phrand	phase randomize(yes|no){no}
plaintexttab	plain text file of input parameters
plot	make plot(yes|no){no}
prg	SIMPLE program to execute
pspecsz	size of power spectrum(in pixels)
refine	refinement mode(no|shc|neigh|shcneigh|qcont|qcontneigh|shift){no}
refs	initial2Dreferences.ext
remap	adjust the number of clusters by splitting(yes|no){no}
rnd	random(yes|no){no}
scale	image scale factor{1}
set_gpu	index of GPU device used(1-MAX_N_GPU){0}
shalgn	do 2D shift alignment(yes|no){no}
shbarrier	use shift search barrier constraint(yes|no){yes}
shell_norm	normalise based on power spectrum (yes|no){no}
shellw	shell-weight reconstruction (yes|no)
sherr	shift error(in pixels){2}
shrink	shrink factor
single	simulate a single image(yes|no){no}
smpd	sampling distance, same as EMANs apix(in Å)
snr	signal-to-noise ratio
soften	soften envelope with cosine edge(yes|no){no}
speckind	power spectrum kind(amp|square|phase|real|log|sqrt){sqrt}
split_mode	mode of splitting for distributed execution(even|chunk){even}
srch_inpl	search in-plane degrees of freedom(yes|no){yes}

startit	start iterating from here
state	state to extract
state2split	state group to split
stats	provide statistics(yes|no){yes}
stk	particle stack with all images(ptcls.ext)
stk2	2nd stack(in map2ptcls/select: selected(cavgs).ext)
stk3	3d stack (in map2ptcls/select: (cavgs)2selectfrom.ext)
thres	threshold (binarisation: 0-1; distance filer: in pixels)
tof	stop frame index
tomo	tomography mode(yes|no){no}
tomoseries	filetable of filetables of tomograms
top	stop particle index
trs	maximum halfwidth shift(in pixels)
trsstats	provide origin shift statistics(yes|no){no}
use_gpu	execute SIMPLE on GPU(yes|no){no}
verbose	verbose output(yes|no)
vis	visualise(yes|no)
vol1	input volume no1(invol1.ext)
vol2	input volume no2(invol2.ext)
vollist	table (text file) of volume files(*.txt/*.asc)
voltab	table (text file) of volume files(*.txt/*.asc)
voltab2	2nd table (text file) of volume files(*.txt/*.asc)
which_iter	iteration nr
width	falloff of inner mask(in pixels){10}
wiener	Wiener restoration mode(full|highres){highres}
xdim	x dimension(in pixles)
xfel	images are XFEL diffraction patterns(yes|no){no}
xsh	x shift(in pixels){0}
ydim	y dimension(in pixles)
ysh	y shift(in pixels){0}
zero	zeroing(yes|no){no}
zsh	z shift(in pixels){0}

6 SIMPLE Programs

6.1 Program: automask2D

automask2D is a program for solvent flattening of class averages. The algorithm for background removal is based on low-pass filtering and binarization. First, the class averages are low-pass filtered to amsklp. Binary representatives are then generated by assigning foreground pixels using sortmeans. A cosine function softens the edge of the binary mask before it is multiplied with the unmasked input averages to accomplish flattening

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
stk  = particle stack with all images(ptcls.ext)
smpd = sampling distance, same as EMANs apix(in A)
msk  = mask radius(in pixels)

OPTIONAL
nthr   = nr of OpenMP threads{1}
amsklp = low-pass limit for envelope mask generation(in A)
edge   = edge size for softening molecular envelope(in pixels)
```

6.2 Program: automask3D

automask3D is a program for solvent flattening of a volume. The algorithm for background removal is based on low-pass filtering and binarization. First, the volume is low-pass filtered to amsklp. A binary volume is then generated by assigning foreground pixels (=1) based on the volume calculated from the molecular weight. A cosine function softens the edge of the binary mask before it is multiplied with the unmasked input to generate the flattened map

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
vol1 = input volume no1(invol1.ext)
smpd = sampling distance, same as EMANs apix(in A)
msk  = mask radius(in pixels)
mw   = molecular weight(in kD)

OPTIONAL
nthr     = nr of OpenMP threads{1}
vol2     = input volume no2(invol2.ext)
amsklp   = low-pass limit for envelope mask generation(in A)
edge     = edge size for softening molecular envelope(in pixels)
binwidth = binary layers grown for molecular envelope(in pixels){1}
```

6.3 Program: binarise

binarise is a program for binarisation of stacks and volumes

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

OPTIONAL
nthr   = nr of OpenMP threads{1}
stk    = particle stack with all images(ptcls.ext)
vol1   = input volume no1(invol1.ext)
thres  = threshold (binarisation: 0-1; distance filer: in pixels)
npix   = number of pixles/voxels in binary representation
grow   = number of binary layers to grow(in pixels)
edge   = edge size for softening molecular envelope(in pixels)
neg    = invert contrast of images(yes|no)
outvol = output volume{outvol.ext}
outstk = output image stack
```

6.4 Program: boxconvs

boxconvs is a program for averaging overlapping boxes across a micrograph in order to check if gain correction was appropriately done

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
fbody = file body

OPTIONAL
```

```
stk       = particle stack with all images(ptcls.ext)
filetab   = list of files(*.txt/*.asc)
boxconvsz = size of box used for box-convolution(in pixels)
startit   = start iterating from here
```

6.5 Program: cavgassemble

cavgassemble is a program that assembles class averages when the clustering program (prime2D) has been executed in distributed mode

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
stk    = particle stack with all images(ptcls.ext)
smpd   = sampling distance, same as EMANs apix(in A)
ncls   = nr of clusters
oritab = table (text file) of orientations(*.asc/*.txt)
nparts = nr of partitions in distributed exection
ctf    = ctf flag(yes|no|flip)

OPTIONAL
nthr       = nr of OpenMP threads{1}
deftab     = text file with CTF info(*.txt/*.asc)
inner      = inner mask radius(in pixels)
width      = falloff of inner mask(in pixels){10}
which_iter = iteration nr
```

6.6 Program: cenvol

cenvol is a program for centering a volume and mapping the shift parameters back to the particle images

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
vol1 = input volume no1(invol1.ext)
smpd = sampling distance, same as EMANs apix(in A)

OPTIONAL
oritab  = table (text file) of orientations(*.asc/*.txt)
outfile = output document
cenlp   = low-pass limit for binarisation in centering
```

6.7 Program: check2D_conv

check2D_conv is a program for checking if a PRIME2D run has converged. The statistics outputted include (1) the overlap between the distribution of parameters for succesive runs. (2) The percentage of search space scanned, i.e. how many reference images are evaluated on average. (3) The average correlation between the images and their corresponding best matching reference section. If convergence to a local optimum is achieved, the fraction increases. Convergence is achieved if the parameter distribution overlap is larger than 0.95 and more than 99% of the reference sections need to be searched to find an improving solution

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
smpd   = sampling distance, same as EMANs apix(in A)
box    = square image size(in pixels)
oritab = table (text file) of orientations(*.asc/*.txt)
nptcls = nr of images in stk/nr of orientations in oritab

OPTIONAL
lp = low-pass limit(in A)
```

6.8 Program: check3D_conv

check3D_conv is a program for checking if a PRIME3D run has converged. The statistics outputted include (1) angle of feasible region, which is proportional to the angular resolution of the set of discrete projection directions being searched. (2) The average angular distance between orientations in the present and previous iteration. In the early iterations, the distance is large because a diverse set of orientations is explored. If convergence to a local optimum is achieved, the distance decreases. (3) The percentage of search space scanned, i.e. how many reference images are evaluated on average. (4) The average correlation between the images and their corresponding best matching reference sections. (5) The average standard deviation of the Euler angles. Convergence is achieved if the angular distance between the orientations in successive iterations falls significantly below the angular resolution of the search space and more than 99% of the reference sections need to be matched on average

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
smpd   = sampling distance, same as EMANs apix(in A)
box    = square image size(in pixels)
oritab = table (text file) of orientations(*.asc/*.txt)
nptcls = nr of images in stk/nr of orientations in oritab
pgrp   = point-group symmetry(cn|dn|t|o|i)

OPTIONAL
lp      = low-pass limit(in A)
nstates = nr of states to reconstruct
eo      = use FSC for filtering and low-pass limit update(yes|no){no}
nspace  = nr of projection directions
find    = Fourier index
refine  = refinement mode(no|shc|neigh|shcneigh|qcont|qcontneigh|shift){no}
```

6.9 Program: check_box

check_box is a program for checking the image dimensions of MRC and SPIDER stacks and volumes

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

OPTIONAL
stk  = particle stack with all images(ptcls.ext)
vol1 = input volume no1(invol1.ext)
```

6.10 Program: check_nptcls

check_nptcls is a program for checking the number of images in MRC and SPIDER stacks

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
stk = particle stack with all images(ptcls.ext)
```

6.11 Program: cluster_oris

cluster_oris is a program for clustering orientations based on geodesic distance

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
oritab = table (text file) of orientations(*.asc/*.txt)
ncls   = nr of clusters

OPTIONAL
nthr = nr of OpenMP threads{1}
```

6.12 Program: cluster_smat

cluster_smat is a program for clustering a similarity matrix and use an combined cluster validation index to assess the quality of the clustering based on the number of clusters

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
nptcls = nr of images in stk/nr of orientations in oritab
fname  = file name
ncls   = nr of clusters
label  = discrete label(class|state){class}

OPTIONAL
nthr = nr of OpenMP threads{1}
```

6.13 Program: comlin_smat

comlin_smat is a program for creating a similarity matrix based on common line correlation. The idea being that it should be possible to cluster images based on their 3D similarity witout having a 3D model by only operating on class averages and find averages that fit well together in 3D

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
stk  = particle stack with all images(ptcls.ext)
smpd = sampling distance, same as EMANs apix(in A)
lp   = low-pass limit(in A)
msk  = mask radius(in pixels)
```

```
OPTIONAL
hp  = high-pass limit(in A)
trs = maximum halfwidth shift(in pixels)
```

6.14 Program: cont3D

cont3D is a continuous refinement code under development

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
stk    = particle stack with all images(ptcls.ext)
vol1   = input volume no1(invol1.ext)
smpd   = sampling distance, same as EMANs apix(in A)
msk    = mask radius(in pixels)
oritab = table (text file) of orientations(*.asc/*.txt)
trs    = maximum halfwidth shift(in pixels)
ctf    = ctf flag(yes|no|flip)
pgrp   = point-group symmetry(cn|dn|t|o|i)

OPTIONAL
nthr    = nr of OpenMP threads{1}
deftab  = text file with CTF info(*.txt/*.asc)
frac    = fraction of ptcls(0-1){1}
automsk = envelope masking(yes|no|cavg){no}
mw      = molecular weight(in kD)
amsklp  = low-pass limit for envelope mask generation(in A)
edge    = edge size for softening molecular envelope(in pixels)
inner   = inner mask radius(in pixels)
width   = falloff of inner mask(in pixels){10}
hp      = high-pass limit(in A)
lpstart = start low-pass limit{15}
lpstop  = stop low-pass limit{8}
startit = start iterating from here
maxits  = maximum number of iterations{500}
xfel    = images are XFEL diffraction patterns(yes|no){no}
shellw  = shell-weight reconstruction (yes|no)
```

6.15 Program: convert

convert is a program for converting between SPIDER and MRC formats

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

OPTIONAL
stk    = particle stack with all images(ptcls.ext)
vol1   = input volume no1(invol1.ext)
outstk = output image stack
outvol = output volume{outvol.ext}
```

6.16 Program: corrcompare

corrcompare is a program for comparing stacked images using real-space and Fourier-based approaches

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
stk  = particle stack with all images(ptcls.ext)
stk2 = 2nd stack(in map2ptcls/select: selected(cavgs).ext)

OPTIONAL
msk   = mask radius(in pixels)
stats = provide statistics(yes|no){yes}
lp    = low-pass limit(in A)
smpd  = sampling distance, same as EMANs apix(in A)
```

6.17 Program: ctffind

ctffind is a wrapper program for CTFFIND4 (Grigorieff lab)

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
filetab = list of files(*.txt/*.asc)
smpd    = sampling distance, same as EMANs apix(in A)
kv      = acceleration voltage(in kV){300.}
cs      = spherical aberration constant(in mm){2.7}
fraca   = fraction of amplitude contrast used for fitting CTF{0.07}

OPTIONAL
pspecsz    = size of power spectrum(in pixels)
hp         = high-pass limit(in A)
lp         = low-pass limit(in A)
dfmin      = minimum expected defocus(in microns)
dfmax      = maximum expected defocus(in microns)
astigstep  = step size for astigamtism search(in microns)
expastig   = expected astigmatism(in microns)
phaseplate = images obtained with phaseplate(yes|no){no}
```

6.18 Program: ctfops

ctfops is a program for applying CTF to stacked images

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
smpd = sampling distance, same as EMANs apix(in A)
ctf  = ctf flag(yes|no|flip)

OPTIONAL
stk    = particle stack with all images(ptcls.ext)
```

```
outstk = output image stack
neg    = invert contrast of images(yes|no)
oritab = table (text file) of orientations(*.asc/*.txt)
deftab = text file with CTF info(*.txt/*.asc)
bfac   = bfactor for sharpening/low-pass filtering(in A**2){200.}
```

6.19 Program: eo_volassemble

eo_volassemble is a program that assembles volume(s) when the reconstruction program (recvol with eo=yes) has been executed in distributed mode. inner applies a soft-edged inner mask. An inner mask is used for icosahedral virus reconstruction, because the DNA or RNA core is often unordered and if not removed it may negatively impact the alignment. The width parameter controls the fall-off of the edge of the inner mask

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
stk    = particle stack with all images(ptcls.ext)
nparts = nr of partitions in distributed exection
smpd   = sampling distance, same as EMANs apix(in A)
msk    = mask radius(in pixels)
oritab = table (text file) of orientations(*.asc/*.txt)
ctf    = ctf flag(yes|no|flip)

OPTIONAL
nthr    = nr of OpenMP threads{1}
deftab  = text file with CTF info(*.txt/*.asc)
mul     = origin shift multiplication factor{1}
state   = state to extract
nstates = nr of states to reconstruct
```

6.20 Program: extract

extract is a program that extracts particle images from DDD movies or integrated movies. Box-files are assumed to be in EMAN format but we provide a conversion script (relion2emanbox.pl) for *.star files containing particle coordinates obtained with Relion. The program creates one stack per movie frame as well as a stack of corrected framesums. In addition to single-particle image stacks, the program produces a parameter file extract_params.txt that can be used in conjunction with other SIMPLE programs. We obtain CTF parameters with CTFFIND4

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
filetab = list of files(*.txt/*.asc)
boxtab  = table (text file) of files with EMAN particle coordinates(*.txt/*.asc)
smpd    = sampling distance, same as EMANs apix(in A)

OPTIONAL
msk        = mask radius(in pixels)
paramtab   = per-micrograph parameters to transfer (CTF/movie shifts)
neg        = invert contrast of images(yes|no)
box        = square image size(in pixels)
noise_norm = normalise based on sdev of background(yes|no){no}
```

```
outside    = only extract boxes within the micrograph boundaries(yes|no){yes}
```

6.21 Program: filter

filter is a program for filtering stacks/volumes

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
smpd = sampling distance, same as EMANs apix(in A)

OPTIONAL
stk    = particle stack with all images(ptcls.ext)
vol1   = input volume no1(invol1.ext)
outstk = output image stack
outvol = output volume{outvol.ext}
lp     = low-pass limit(in A)
hp     = high-pass limit(in A)
phrand = phase randomize(yes|no){no}
bfac   = bfactor for sharpening/low-pass filtering(in A**2){200.}
```

6.22 Program: find_nnimgs

find_nnimgs is a program for cidentifying the nnn nearest neighbor images for each image in the inputted stack

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
stk  = particle stack with all images(ptcls.ext)
smpd = sampling distance, same as EMANs apix(in A)
msk  = mask radius(in pixels)
nnn  = number of nearest neighbors{500}

OPTIONAL
lp = low-pass limit(in A)
hp = high-pass limit(in A)
```

6.23 Program: het_init

het_init

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
stk     = particle stack with all images(ptcls.ext)
smpd    = sampling distance, same as EMANs apix(in A)
oritab  = table (text file) of orientations(*.asc/*.txt)
nstates = nr of states to reconstruct
ctf     = ctf flag(yes|no|flip)

OPTIONAL
```

```
nthr   = nr of OpenMP threads{1}
deftab = text file with CTF info(*.txt/*.asc)
msk    = mask radius(in pixels)
inner  = inner mask radius(in pixels)
width  = falloff of inner mask(in pixels){10}
lp     = low-pass limit(in A)
eo     = use FSC for filtering and low-pass limit update(yes|no){no}
frac   = fraction of ptcls(0-1){1}
```

6.24 Program: image_smat

image_smat is a program for creating a similarity matrix based on common line correlation. The idea being that it should be possible to cluster images based on their 3D similarity witout having a 3D model by only operating on class averages and find averages that fit well together in 3D

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
stk  = particle stack with all images(ptcls.ext)
smpd = sampling distance, same as EMANs apix(in A)

OPTIONAL
lp   = low-pass limit(in A)
msk  = mask radius(in pixels)
hp   = high-pass limit(in A)
nthr = nr of OpenMP threads{1}
```

6.25 Program: iminfo

iminfo is a program for printing header information in MRC and SPIDER stacks and volumes

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
fname = file name

OPTIONAL
box    = square image size(in pixels)
smpd   = sampling distance, same as EMANs apix(in A)
stats  = provide statistics(yes|no){yes}
endian = endiannesss of files(big|little|native){native}
```

6.26 Program: integrate_movies

integrate_movies is a program for integrating DDD movies

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
filetab = list of files(*.txt/*.asc)
fbody   = file body
smpd    = sampling distance, same as EMANs apix(in A)
```

```
OPTIONAL
pspecsz = size of power spectrum(in pixels)
scale   = image scale factor{1}
```

6.27 Program: makedeftab

makedeftab is a program for creating a SIMPLE conformant file of CTF parameter values (deftab). Input is either an earlier SIMPLE deftab/oritab. The purpose is to get the kv, cs, and fraca parameters as part of the CTF input doc as that is the new convention. The other alternative is to input a plain text file with CTF parameters dfx, dfy, angast according to the Frealign convention. Unit conversions are dealt with using optional variables. The units refer to the units in the inputted document

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
kv      = acceleration voltage(in kV){300.}
cs      = spherical aberration constant(in mm){2.7}
fraca   = fraction of amplitude contrast used for fitting CTF{0.07}
outfile = output document

OPTIONAL
plaintexttab = plain text file of input parameters
oritab       = table (text file) of orientations(*.asc/*.txt)
deftab       = text file with CTF info(*.txt/*.asc)
dfunit       = defocus unit (A|microns){microns}
angastunit   = angle of astigmatism unit (radians|degrees){degrees}
```

6.28 Program: makeoris

makeoris is a program for making SIMPLE orientation/parameter files (text files containing input parameters and/or parameters estimated by prime2D or prime3D). The program generates random Euler angles e1.in.[0,360], e2.in.[0,180], and e3.in.[0,360] and random origin shifts x.in.[-trs,yrs] and y.in.[-trs,yrs]. If ndiscrete is set to an integer number > 0, the orientations produced are randomly sampled from the set of ndiscrete quasi-even projection directions, and the in-plane parameters are assigned randomly. If even=yes, then all nptcls orientations are assigned quasi-even projection directions,and random in-plane parameters. If nstates is set to some integer number > 0, then states are assigned randomly .in.[1,nstates]. If zero=yes in this mode of execution, the projection directions are zeroed and only the in-plane parameters are kept intact. If errify=yes and astigerr is defined, then uniform random astigmatism errors are introduced .in.[-astigerr,astigerr]

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
nptcls = nr of images in stk/nr of orientations in oritab

OPTIONAL
nthr      = nr of OpenMP threads{1}
minp      = minimum cluster population
ncls      = nr of clusters
outfile   = output document
```

```
trs       = maximum halfwidth shift(in pixels)
nstates   = nr of states to reconstruct
pgrp      = point-group symmetry(cn|dn|t|o|i)
defocus   = defocus(in microns){3.}
angerr    = angular error(in degrees){0}
sherr     = shift error(in pixels){2}
dferr     = defocus error(in microns){1.0}
even      = calculate even eo-pair(yes|no){no}
zero      = zeroing(yes|no){no}
discrete  = discrete(yes|no){no}
ndiscrete = nr of discrete orientations
diverse   = diverse or not flag (yes|no){no}
state     = state to extract
nspace    = nr of projection directions
iares     = integer angular resolution{10}
```

6.29 Program: map2ptcls

map2ptcls is a program for mapping parameters that have been obtained using class averages to the individual particle images

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
stk    = particle stack with all images(ptcls.ext)
stk2   = 2nd stack(in map2ptcls/select: selected(cavgs).ext)
stk3   = 3d stack (in map2ptcls/select: (cavgs)2selectfrom.ext)
oritab = table (text file) of orientations(*.asc/*.txt)

OPTIONAL
oritab2   = 2nd table (text file) of orientations(*.asc/*.txt)
comlindoc = shc_clustering_nclsX.txt
doclist   = list of oritabs for different states
deftab    = text file with CTF info(*.txt/*.asc)
outfile   = output document
mul       = origin shift multiplication factor{1}
nthr      = nr of OpenMP threads{1}
```

6.30 Program: mask

mask is a program for masking images and volumes. If you want to mask your images with a spherical mask with a soft falloff, set msk to the radius in pixels

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
msk = mask radius(in pixels)

OPTIONAL
stk      = particle stack with all images(ptcls.ext)
vol1     = input volume no1(invol1.ext)
msktype  = type of mask(hard|soft){soft}
inner    = inner mask radius(in pixels)
```

```
width    = falloff of inner mask(in pixels){10}
outer    = outer mask radius(in pixels)
nthr     = nr of OpenMP threads{1}
mw       = molecular weight(in kD)
edge     = edge size for softening molecular envelope(in pixels)
binwidth = binary layers grown for molecular envelope(in pixels){1}
amsklp   = low-pass limit for envelope mask generation(in A)
automsk  = envelope masking(yes|no|cavg){no}
smpd     = sampling distance, same as EMANs apix(in A)
```

6.31 Program: masscen

masscen is a program for centering images acccording to their centre of mass

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
stk  = particle stack with all images(ptcls.ext)
smpd = sampling distance, same as EMANs apix(in A)
lp   = low-pass limit(in A)

OPTIONAL
msk    = mask radius(in pixels)
neg    = invert contrast of images(yes|no)
thres  = threshold (binarisation: 0-1; distance filer: in pixels)
outstk = output image stack
```

6.32 Program: merge_algndocs

merge_algndocs is a program for merging alignment documents from SIMPLE runs in distributed mode

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
fbody   = file body
nptcls  = nr of images in stk/nr of orientations in oritab
ndocs   = nr of documents
outfile = output document
```

6.33 Program: merge_nnmat

merge_nnmat is a program for merging partial nearest neighbour matrices calculated in distributed mode

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
nptcls = nr of images in stk/nr of orientations in oritab
nparts = nr of partitions in distributed exection
nnn    = number of nearest neighbors{500}
```

6.34 Program: merge_shellweights

merge_shellweights is a program for merging partial shellweight matrices calculated in distributed mode

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
stk    = particle stack with all images(ptcls.ext)
nparts = nr of partitions in distributed exection
```

6.35 Program: merge_similarities

merge_similarities is a program for merging similarities calculated between pairs of objects into a similarity matrix that can be inputted to cluster_smat

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
nptcls = nr of images in stk/nr of orientations in oritab

OPTIONAL
nparts = nr of partitions in distributed exection
```

6.36 Program: multiptcl_init

multiptcl_init is a program for generating random initial models for initialisation of PRIME3D when run in multiparticle mode

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
stk    = particle stack with all images(ptcls.ext)
smpd   = sampling distance, same as EMANs apix(in A)
oritab = table (text file) of orientations(*.asc/*.txt)
ctf    = ctf flag(yes|no|flip)

OPTIONAL
nthr        = nr of OpenMP threads{1}
deftab      = text file with CTF info(*.txt/*.asc)
nstates     = nr of states to reconstruct
msk         = mask radius(in pixels)
inner       = inner mask radius(in pixels)
width       = falloff of inner mask(in pixels){10}
lp          = low-pass limit(in A)
eo          = use FSC for filtering and low-pass limit update(yes|no){no}
frac        = fraction of ptcls(0-1){1}
state2split = state group to split
norec       = do not reconstruct volume(s)(yes|no){no}
mul         = origin shift multiplication factor{1}
zero        = zeroing(yes|no){no}
```

6.37 Program: noiseimgs

noiseimgs is a program for generating pure noise images

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
box    = square image size(in pixels)
nptcls = nr of images in stk/nr of orientations in oritab
```

6.38 Program: norm

norm is a program for normalization of MRC or SPIDER stacks and volumes. If you want to normalise your images inputted with stk, set norm=yes. hfun (e.g. hfun=sigm) controls the normalisation function. If you want to perform noise normalisation of the images set noise_norm=yes given a mask radius msk (pixels). If you want to normalise your images or volume (vol1) with respect to their power spectrum set shell_norm=yes

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

OPTIONAL
stk        = particle stack with all images(ptcls.ext)
msk        = mask radius(in pixels)
norm       = do statistical normalisation avg
noise_norm = normalise based on sdev of background(yes|no){no}
shell_norm = normalise based on power spectrum (yes|no){no}
hfun       = function used for normalization(sigm|tanh|lin){sigm}
nthr       = nr of OpenMP threads{1}
```

6.39 Program: npeaks

npeaks is a program for checking the number of nonzero orientation weights (number of correlation peaks included in the weighted reconstruction)

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
smpd = sampling distance, same as EMANs apix(in A)
box  = square image size(in pixels)
lp   = low-pass limit(in A)

OPTIONAL
nspace  = nr of projection directions
moldiam = molecular diameter(in A)
pgrp    = point-group symmetry(cn|dn|t|o|i)
```

6.40 Program: nspace

nspace is a program for calculating the expected resolution obtainable with different values of nspace (number of discrete projection directions used for discrete search)

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...
```

```
REQUIRED
moldiam = molecular diameter(in A)
```

6.41 Program: orisops

orisops is a program for analyzing SIMPLE orientation/parameter files (text files containing input parameters and/or parameters estimated by prime2D or prime3D). If only oritab is inputted, there are a few options available. If errify=yes, then the program introduces uniform random angular errors .in.[-angerr,angerr], and uniform origin shift errors .in.[-sherr,sherr], and uniform random defocus errors .in.[-dferr,dferr]. If nstates > 1 then random states are assigned .in.[1,nstates]. If mirr=2d, then the Euler angles in oritab are mirrored according to the relation e1=e1, e2=180.+e2, e3=-e3. If mirr=3d, then the Euler angles in oritab are mirrored according to the relation R=M(M*R), where R is the rotation matrix calculated from the Euler angle triplet and M is a 3D reflection matrix (like a unit matrix but with the 3,3-component sign swapped). If e1, e2, or e3 is inputted, the orientations in oritab are rotated correspondingly. If you input state as well, you rotate only the orientations assigned to state state. If mul is defined, you multiply the origin shifts with mul. If zero=yes, then the shifts are zeroed. If none of the above described parameter are defined, and oritab is still defined, the program projects the 3D orientation into the xy-plane and plots the resulting vector (this is useful for checking orientation coverage)

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

OPTIONAL
oritab    = table (text file) of orientations(*.asc/*.txt)
nptcls    = nr of images in stk/nr of orientations in oritab
plot      = make plot(yes|no){no}
outfile   = output document
e1        = 1st Euler(in degrees){0}
e2        = 2nd Euler(in degrees){0}
e3        = 3d Euler(in degrees){0}
trs       = maximum halfwidth shift(in pixels)
nstates   = nr of states to reconstruct
pgrp      = point-group symmetry(cn|dn|t|o|i)
defocus   = defocus(in microns){3.}
deftab    = text file with CTF info(*.txt/*.asc)
angerr    = angular error(in degrees){0}
sherr     = shift error(in pixels){2}
dferr     = defocus error(in microns){1.0}
zero      = zeroing(yes|no){no}
discrete  = discrete(yes|no){no}
ndiscrete = nr of discrete orientations
state     = state to extract
errify    = introduce error(yes|no){no}
mul       = origin shift multiplication factor{1}
mirr      = mirror(no|x|y){no}
xsh       = x shift(in pixels){0}
ysh       = y shift(in pixels){0}
zsh       = z shift(in pixels){0}
```

6.42 Program: oristats

oristats is a program for analyzing SIMPLE orientation/parameter files (text files containing input parameters and/or parameters estimated by prime2D or prime3D). If two orientation tables (oritab and oritab2) are inputted, the program provides statistics of the distances between the orientations in the two documents. These statistics include the sum of angular distances between the orientations, the average angular distance between the orientations, the standard deviation of angular distances, the minimum angular distance, and the maximum angular distance

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
oritab = table (text file) of orientations(*.asc/*.txt)

OPTIONAL
nptcls     = nr of images in stk/nr of orientations in oritab
oritab2    = 2nd table (text file) of orientations(*.asc/*.txt)
outfile    = output document
nstates    = nr of states to reconstruct
state      = state to extract
ctfstats   = calculate ctf statistics(yes|no){no}
trsstats   = provide origin shift statistics(yes|no){no}
hist       = give variable for histogram plot
ncls       = nr of clusters
minp       = minimum cluster population
clustvalid = validate clustering(yes|homo|no){no}
thres      = threshold (binarisation: 0-1; distance filer: in pixels)
```

6.43 Program: pick

pick is a template-based picker program under development

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
filetab = list of files(*.txt/*.asc)
refs    = initial2Dreferences.ext
smpd    = sampling distance, same as EMANs apix(in A)
msk     = mask radius(in pixels)

OPTIONAL
nthr      = nr of OpenMP threads{1}
lp        = low-pass limit(in A)
offset    = pxels offset{3}
shrink    = shrink factor
thres     = threshold (binarisation: 0-1; distance filer: in pixels)
dcrit_rel = critical distance relative to box(0-1){0.5}
```

6.44 Program: postproc_vol

postproc_vol is a program for post-processing of volumes

```
USAGE:
```

```
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
vol1 = input volume no1(invol1.ext)
smpd = sampling distance, same as EMANs apix(in A)
msk  = mask radius(in pixels)

OPTIONAL
fsc      = binary file with FSC info{fsc_state01.bin}
lp       = low-pass limit(in A)
mw       = molecular weight(in kD)
bfac     = bfactor for sharpening/low-pass filtering(in A**2){200.}
automsk  = envelope masking(yes|no|cavg){no}
amsklp   = low-pass limit for envelope mask generation(in A)
edge     = edge size for softening molecular envelope(in pixels)
binwidth = binary layers grown for molecular envelope(in pixels){1}
```

6.45 Program: powerspecs

powerspecs is a program for generating powerspectra from a stack or filetable

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
smpd  = sampling distance, same as EMANs apix(in A)
fbody = file body

OPTIONAL
nthr     = nr of OpenMP threads{1}
stk      = particle stack with all images(ptcls.ext)
filetab  = list of files(*.txt/*.asc)
pspecsz  = size of power spectrum(in pixels)
speckind = power spectrum kind(amp|square|phase|real|log|sqrt){sqrt}
startit  = start iterating from here
lp       = low-pass limit(in A)
clip     = clipped image box size(in pixels)
```

6.46 Program: prime2D

prime2D is a reference-free 2D alignment/clustering algorithm adopted from the prime3D probabilistic ab initio 3D reconstruction algorithm

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
stk  = particle stack with all images(ptcls.ext)
smpd = sampling distance, same as EMANs apix(in A)
msk  = mask radius(in pixels)
ncls = nr of clusters
refs = initial2Dreferences.ext
ctf  = ctf flag(yes|no|flip)

OPTIONAL
```

```
nthr      = nr of OpenMP threads{1}
deftab    = text file with CTF info(*.txt/*.asc)
oritab    = table (text file) of orientations(*.asc/*.txt)
hp        = high-pass limit(in A)
lp        = low-pass limit(in A)
cenlp     = low-pass limit for binarisation in centering
trs       = maximum halfwidth shift(in pixels)
automsk   = envelope masking(yes|no|cavg){no}
amsklp    = low-pass limit for envelope mask generation(in A)
inner     = inner mask radius(in pixels)
width     = falloff of inner mask(in pixels){10}
startit   = start iterating from here
maxits    = maximum number of iterations{500}
srch_inpl = search in-plane degrees of freedom(yes|no){yes}
```

6.47 Program: prime2D_init

prime2D_init is used to produce the initial random references for prime2D execution. The random clustering and in-plane alignment is printed in the file prime2D_startdoc.txt produced by the program. This file is used together with the initial references (startcavgs.ext) to execute prime2D

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
stk  = particle stack with all images(ptcls.ext)
smpd = sampling distance, same as EMANs apix(in A)
ncls = nr of clusters
ctf  = ctf flag(yes|no|flip)

OPTIONAL
nthr      = nr of OpenMP threads{1}
deftab    = text file with CTF info(*.txt/*.asc)
oritab    = table (text file) of orientations(*.asc/*.txt)
filwidth  = width of filament (in A)
mul       = origin shift multiplication factor{1}
srch_inpl = search in-plane degrees of freedom(yes|no){yes}
```

6.48 Program: prime3D

prime3D is an ab inito reconstruction/refinement program based on probabilistic projection matching. PRIME is short for PRobabilistic Initial 3D Model generation for Single- particle cryo-Electron microscopy. Do not search the origin shifts initially, when the model is of very low quality. If your images are far off centre, use stackops with option shalgn=yes instead to shiftalign the images beforehand (the algorithm implemented is the same as EMANs cenalignint program). We recommend running the first round of PRIME with the default dynamic resolution stepping dynlp=yes. The dynlp option implements a heuristic resolution weighting/update scheme. The initial low-pass limit is set so that each image receives ten nonzero orientation weights. When quasi-convergence has been reached, the limit is updated one Fourier index at the time until PRIME reaches the condition where six nonzero orientation weights are assigned to each image. FSC-based filtering is unfortunately not possible to do in the ab initio reconstruction step, because when the orientations are mostly random, the FSC overestimates the resolution. Once the initial model has converged, we recommend start searching the shifts (by

setting trs to some nonzero value) and applying the FSC for resolution- weighting (by setting eo=yes)

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
stk  = particle stack with all images(ptcls.ext)
vol1 = input volume no1(invol1.ext)
smpd = sampling distance, same as EMANs apix(in A)
msk  = mask radius(in pixels)
ctf  = ctf flag(yes|no|flip)
pgrp = point-group symmetry(cn|dn|t|o|i)

OPTIONAL
nthr      = nr of OpenMP threads{1}
vol2      = input volume no2(invol2.ext)
oritab    = table (text file) of orientations(*.asc/*.txt)
deftab    = text file with CTF info(*.txt/*.asc)
trs       = maximum halfwidth shift(in pixels)
hp        = high-pass limit(in A)
lp        = low-pass limit(in A)
cenlp     = low-pass limit for binarisation in centering
dynlp     = automatic resolution limit update(yes|no){yes}
lpstart   = start low-pass limit{15}
lpstop    = stop low-pass limit{8}
eo        = use FSC for filtering and low-pass limit update(yes|no){no}
refine    = refinement mode(no|shc|neigh|shcneigh|qcont|qcontneigh|shift){no}
frac      = fraction of ptcls(0-1){1}
automsk   = envelope masking(yes|no|cavg){no}
mw        = molecular weight(in kD)
amsklp    = low-pass limit for envelope mask generation(in A)
edge      = edge size for softening molecular envelope(in pixels)
binwidth  = binary layers grown for molecular envelope(in pixels){1}
inner     = inner mask radius(in pixels)
width     = falloff of inner mask(in pixels){10}
nspace    = nr of projection directions
nstates   = nr of states to reconstruct
npeaks    = nr of nonzero orientation weights{1}
startit   = start iterating from here
maxits    = maximum number of iterations{500}
shbarrier = use shift search barrier constraint(yes|no){yes}
noise     = noise initialisation(yes|no){no}
xfel      = images are XFEL diffraction patterns(yes|no){no}
nnn       = number of nearest neighbors{500}
shellw    = shell-weight reconstruction (yes|no)
```

6.49 Program: prime3D_init

prime3D_init is a program for generating a random initial model for initialisation of PRIME3D. If the data set is large (>5000 images), generating a random model can be slow. To speedup, set nran to some smaller number, resulting in nran images selected randomly for reconstruction

```
USAGE:
```

```
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
stk  = particle stack with all images(ptcls.ext)
smpd = sampling distance, same as EMANs apix(in A)
msk  = mask radius(in pixels)
ctf  = ctf flag(yes|no|flip)
pgrp = point-group symmetry(cn|dn|t|o|i)

OPTIONAL
nthr   = nr of OpenMP threads{1}
deftab = text file with CTF info(*.txt/*.asc)
lp     = low-pass limit(in A)
inner  = inner mask radius(in pixels)
width  = falloff of inner mask(in pixels){10}
nspace = nr of projection directions
nran   = number of random images to select
npeaks = nr of nonzero orientation weights{1}
xfel   = images are XFEL diffraction patterns(yes|no){no}
```

6.50 Program: print_cmd_dict

print_cmd_dict is a program for printing the command line key dictonary

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

OPTIONAL
outfile = output document
```

6.51 Program: print_dose_weights

print_dose_weights is a program for printing the dose weights applied to individual frames

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
nframes   = number of frames{30}
exp_time  = expusure time(in s)
dose_rate = dose rate(in e/A2/s)
box       = square image size(in pixels)
smpd      = sampling distance, same as EMANs apix(in A)

OPTIONAL
kv = acceleration voltage(in kV){300.}
```

6.52 Program: print_fsc

print_fsc is a program for printing the binary FSC files produced by PRIME3D

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
```

```
smpd = sampling distance, same as EMANs apix(in A)
box  = square image size(in pixels)
fsc  = binary file with FSC info{fsc_state01.bin}
```

6.53 Program: projvol

projvol is a program for projecting a volume using interpolation in Fourier space. Input is a SPIDER or MRC volume. Output is a stack of projection images of the same format as the inputted volume. Projections are generated by extraction of central sections from the Fourier volume and back transformation of the 2D FTs. nspace controls the number of projection images generated with quasi-even projection directions. The oritab parameter allows you to input the orientations that you wish to have your volume projected in. If rnd=yes, random rather than quasi-even projections are generated, trs then controls the halfwidth of the random origin shift. Less commonly used parameters are pgrp, which controls the point-group symmetry c (rotational), d (dihedral), t (tetrahedral), o (octahedral) or i (icosahedral). The point-group symmetry is used to restrict the set of projections to within the asymmetric unit. neg inverts the contrast of the projections. mirr=yes mirrors the projection by modifying the Euler angles. If mirr=x or mirr=y the projection is physically mirrored after it has been generated

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
vol1 = input volume no1(invol1.ext)
smpd = sampling distance, same as EMANs apix(in A)

OPTIONAL
nspace = nr of projection directions
outstk = output image stack
oritab = table (text file) of orientations(*.asc/*.txt)
nthr   = nr of OpenMP threads{1}
rnd    = random(yes|no){no}
trs    = maximum halfwidth shift(in pixels)
pgrp   = point-group symmetry(cn|dn|t|o|i)
neg    = invert contrast of images(yes|no)
mirr   = mirror(no|x|y){no}
top    = stop particle index
xfel   = images are XFEL diffraction patterns(yes|no){no}
```

6.54 Program: rank_cavgs

rank_cavgs is a program for ranking class averages by decreasing population, given the stack of class averages (stk argument) and the 2D orientations document (oritab) generated by prime2D

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
stk    = particle stack with all images(ptcls.ext)
oritab = table (text file) of orientations(*.asc/*.txt)

OPTIONAL
outstk = output image stack
```

6.55 Program: recvol

recvol is a program for reconstructing volumes from MRC and SPIDER stacks, given input orientations and state assignments. The algorithm is based on direct Fourier inversion with a Kaiser-Bessel (KB) interpolation kernel. This window function reduces the real-space ripple artifacts associated with direct moving windowed-sinc interpolation. The feature sought when implementing this algorithm was to enable quick, reliable reconstruction from aligned individual particle images. mul is used to scale the origin shifts if down-sampled were used for alignment and the original images are used for reconstruction. ctf=yes or ctf=flip turns on the Wiener restoration. If the images were phase-flipped set ctf=flip. amsklp, mw, and edge control the solvent mask: the low-pass limit used to generate the envelope; the molecular weight of the molecule (protein assumed but it works reasonably well also for RNA; slight modification of mw might be needed). The inner parameter controls the radius of the soft-edged mask used to remove the unordered DNA/RNA core of spherical icosahedral viruses

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
stk    = particle stack with all images(ptcls.ext)
smpd   = sampling distance, same as EMANs apix(in A)
oritab = table (text file) of orientations(*.asc/*.txt)
msk    = mask radius(in pixels)
ctf    = ctf flag(yes|no|flip)
pgrp   = point-group symmetry(cn|dn|t|o|i)

OPTIONAL
nthr   = nr of OpenMP threads{1}
eo     = use FSC for filtering and low-pass limit update(yes|no){no}
deftab = text file with CTF info(*.txt/*.asc)
frac   = fraction of ptcls(0-1){1}
mw     = molecular weight(in kD)
mul    = origin shift multiplication factor{1}
state  = state to extract
```

6.56 Program: res

res is a program for checking the low-pass resolution limit for a given Fourier index

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
smpd = sampling distance, same as EMANs apix(in A)
find = Fourier index
box  = square image size(in pixels)
```

6.57 Program: respimg

respimg

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
```

```
stk    = particle stack with all images(ptcls.ext)
smpd   = sampling distance, same as EMANs apix(in A)
msk    = mask radius(in pixels)
oritab = table (text file) of orientations(*.asc/*.txt)
ctf    = ctf flag(yes|no|flip)

OPTIONAL
outstk = output image stack
deftab = text file with CTF info(*.txt/*.asc)
nthr   = nr of OpenMP threads{1}
```

6.58 Program: resrange

resrange is a program for estimating the resolution range used in the heuristic resolution-stepping scheme in the PRIME3D initial model production procedure. The initial low-pass limit is set so that each image receives ten nonzero orientation weights. When quasi-convergence has been reached, the limit is updated one Fourier index at the time, until PRIME reaches the condition where six nonzero orientation weights are assigned to each image. FSC-based filtering is unfortunately not possible to do in the ab initio 3D reconstruction step, because when the orientations are mostly random, the FSC overestimates the resolution. This program is used internally when executing PRIME in distributed mode. We advise you to check the starting and stopping low-pass limits before executing PRIME3D using this program. The resolution range estimate depends on the molecular diameter, which is estimated based on the box size. If you want to override this estimate, set moldiam to the desired value (in Å). This may be necessary if your images have a lot of background padding. However, for starting model generation it is probably better to clip the images snugly around the particle, because smaller images equal less computation

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
smpd = sampling distance, same as EMANs apix(in A)

OPTIONAL
nthr    = nr of OpenMP threads{1}
nspace  = nr of projection directions
pgrp    = point-group symmetry(cn|dn|t|o|i)
box     = square image size(in pixels)
moldiam = molecular diameter(in A)
```

6.59 Program: rotmats2oris

rotmats2oris converts a text file (9 records per line) describing rotation matrices into a SIMPLE oritab

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
infile = table (text file) of inputs(*.asc/*.txt)

OPTIONAL
outfile = output document
```

6.60 Program: scale

scale is a program that provides re-scaling and clipping routines for MRC or SPIDER stacks and volumes

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
smpd = sampling distance, same as EMANs apix(in A)

OPTIONAL
stk     = particle stack with all images(ptcls.ext)
vol1    = input volume no1(invol1.ext)
filetab = list of files(*.txt/*.asc)
msk     = mask radius(in pixels)
newbox  = new box size(in pixels)
scale   = image scale factor{1}
clip    = clipped image box size(in pixels)
outvol  = output volume{outvol.ext}
outstk  = output image stack
```

6.61 Program: select

select is a program for selecting files based on image correlation matching

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
stk  = particle stack with all images(ptcls.ext)
stk2 = 2nd stack(in map2ptcls/select: selected(cavgs).ext)

OPTIONAL
nthr       = nr of OpenMP threads{1}
stk3       = 3d stack (in map2ptcls/select: (cavgs)2selectfrom.ext)
filetab    = list of files(*.txt/*.asc)
outfile    = output document
outstk     = output image stack
dir_select = move selected files to here{selected}
dir_reject = move rejected files to here{rejected}
```

6.62 Program: select_frames

select_frames is a program for selecting contiguous segments of frames from DDD movies

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
filetab = list of files(*.txt/*.asc)
fbody   = file body
fromf   = start frame index
tof     = stop frame index
smpd    = sampling distance, same as EMANs apix(in A)
```

```
OPTIONAL
startit = start iterating from here
```

6.63 Program: shellweight3D

shellweight3D is a program for calculating the shell-by-shell resolution weights in a global sense, so that particles that do contribute with higher resolution information (as measured by the FRC) are given the appropriate weight

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
stk    = particle stack with all images(ptcls.ext)
vol1   = input volume no1(invol1.ext)
smpd   = sampling distance, same as EMANs apix(in A)
msk    = mask radius(in pixels)
oritab = table (text file) of orientations(*.asc/*.txt)
ctf    = ctf flag(yes|no|flip)

OPTIONAL
deftab   = text file with CTF info(*.txt/*.asc)
automsk  = envelope masking(yes|no|cavg){no}
mw       = molecular weight(in kD)
amsklp   = low-pass limit for envelope mask generation(in A)
edge     = edge size for softening molecular envelope(in pixels)
binwidth = binary layers grown for molecular envelope(in pixels){1}
inner    = inner mask radius(in pixels)
width    = falloff of inner mask(in pixels){10}
```

6.64 Program: shift

shift is a program for shifting a stack according to shifts in oritab

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
stk    = particle stack with all images(ptcls.ext)
smpd   = sampling distance, same as EMANs apix(in A)
oritab = table (text file) of orientations(*.asc/*.txt)

OPTIONAL
outstk = output image stack
mul    = origin shift multiplication factor{1}
```

6.65 Program: simimgs

simimgs is a program for simulating cryo-EM images. It is not a very sophisticated simulator, but it is nevertheless useful for testing purposes. It does not do any multi-slice simulation and it cannot be used for simulating molecules containing heavy atoms. It does not even accept a PDB file as an input. Input is a cryo-EM map, which we usually generate from a PDB file using EMANs program pdb2mrc. simimgs then projects the volume using Fourier interpolation, applies 20% of the total noise to the images (pink noise), Fourier transforms them, and multiplies

them with astigmatic CTF and B-factor. The images are inverse FTed before the remaining 80% of the noise (white noise) is added

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
vol1   = input volume no1(invol1.ext)
smpd   = sampling distance, same as EMANs apix(in A)
msk    = mask radius(in pixels)
nptcls = nr of images in stk/nr of orientations in oritab
snr    = signal-to-noise ratio

OPTIONAL
nthr      = nr of OpenMP threads{1}
xsh       = x shift(in pixels){0}
ysh       = y shift(in pixels){0}
sherr     = shift error(in pixels){2}
oritab    = table (text file) of orientations(*.asc/*.txt)
outfile   = output document
outstk    = output image stack
single    = simulate a single image(yes|no){no}
ndiscrete = nr of discrete orientations
diverse   = diverse or not flag (yes|no){no}
pgrp      = point-group symmetry(cn|dn|t|o|i)
ctf       = ctf flag(yes|no|flip)
kv        = acceleration voltage(in kV){300.}
cs        = spherical aberration constant(in mm){2.7}
fraca     = fraction of amplitude contrast used for fitting CTF{0.07}
deftab    = text file with CTF info(*.txt/*.asc)
defocus   = defocus(in microns){3.}
dferr     = defocus error(in microns){1.0}
astigerr  = astigmatism error(in microns)
bfac      = bfactor for sharpening/low-pass filtering(in A**2){200.}
bfacerr   = bfactor error in simulated images(in A**2){0}
```

6.66 Program: simmovie

simmovie is a program for crude simulation of a DDD movie. Input is a set of projection images to place. Movie frames are then generated related by randomly shifting the base image and applying two different noise sources: shot and detector noise

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
stk  = particle stack with all images(ptcls.ext)
smpd = sampling distance, same as EMANs apix(in A)
msk  = mask radius(in pixels)
xdim = x dimension(in pixles)
ydim = y dimension(in pixles)
snr  = signal-to-noise ratio

OPTIONAL
```

```
nthr    = nr of OpenMP threads{1}
nframes = number of frames{30}
trs     = maximum halfwidth shift(in pixels)
vis     = visualise(yes|no)
kv      = acceleration voltage(in kV){300.}
cs      = spherical aberration constant(in mm){2.7}
fraca   = fraction of amplitude contrast used for fitting CTF{0.07}
deftab  = text file with CTF info(*.txt/*.asc)
defocus = defocus(in microns){3.}
bfac    = bfactor for sharpening/low-pass filtering(in A**2){200.}
```

6.67 Program: simsubtomo

simsubtomo is a program for crude simulation of a subtomograms

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
vol1   = input volume no1(invol1.ext)
smpd   = sampling distance, same as EMANs apix(in A)
nptcls = nr of images in stk/nr of orientations in oritab
snr    = signal-to-noise ratio

OPTIONAL
nthr = nr of OpenMP threads{1}
```

6.68 Program: split

split is a program for splitting of image stacks into partitions for parallel execution. This is done to reduce I/O latency

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
stk = particle stack with all images(ptcls.ext)

OPTIONAL
split_mode = mode of splitting for distributed execution(even|chunk){even}
nparts     = nr of partitions in distributed exection
neg        = invert contrast of images(yes|no)
ncls       = nr of clusters
nspace     = nr of projection directions
```

6.69 Program: split_pairs

split_pairs is a program for splitting calculations between pairs of objects into balanced partitions

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
nptcls = nr of images in stk/nr of orientations in oritab
nparts = nr of partitions in distributed exection
```

6.70 Program: stack

stack is a program for stacking individual images or multiple stacks into one

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
filetab = list of files(*.txt/*.asc)
outstk  = output image stack

OPTIONAL
lp      = low-pass limit(in A)
smpd    = sampling distance, same as EMANs apix(in A)
clip    = clipped image box size(in pixels)
nframes = number of frames{30}
fbody   = file body
numlen  = length of number string
xdim    = x dimension(in pixles)
ydim    = y dimension(in pixles)
endian  = endiannesss of files(big|little|native){native}
```

6.71 Program: stackops

stackops is a program that provides standard single-particle image processing routines that are applied to MRC or SPIDER stacks. If you want to extract a particular state, give an alignment document (oritab) and set state to the state that you want to extract. If you want to select the fraction of best particles (according to the goal function), input an alignment doc (oritab) and set frac. You can combine the state and frac options. If you want to apply noise to images, give the desired signal-to-noise ratio via snr. If you want to calculate the autocorrelation function of your images set acf=yes. If you want to extract a contiguous subset of particle images from the stack, set fromp and top. If you want to fish out a number of particle images from your stack at random, set nran to some nonzero integer number less than nptcls. With avg=yes the global average of the inputted stack is calculated. If you define frameavg to some integer number larger than one averages with chunk sizes of frameavg are produced, which may be useful for analysis of dose-fractionated image series. neg inverts the contrast of the images

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
stk = particle stack with all images(ptcls.ext)

OPTIONAL
smpd     = sampling distance, same as EMANs apix(in A)
oritab   = table (text file) of orientations(*.asc/*.txt)
outstk   = output image stack
mirr     = mirror(no|x|y){no}
nran     = number of random images to select
frac     = fraction of ptcls(0-1){1}
state    = state to extract
class    = cluster identity
neg      = invert contrast of images(yes|no)
acf      = calculate autocorrelation function(yes|no){no}
avg      = calculate average(yes|no)
```

```
frameavg = nr of frames to average{0}
vis      = visualise(yes|no)
snr      = signal-to-noise ratio
fromp    = start ptcl index
top      = stop particle index
stk2     = 2nd stack(in map2ptcls/select: selected(cavgs).ext)
nptcls   = nr of images in stk/nr of orientations in oritab
append   = append in context of files(yes|no){no}
order    = order ptcls according to correlation(yes|no){no}
outfile  = output document
```

6.72 Program: symsrch

symsrch is a program for searching for the principal symmetry axis of a volume reconstructed without assuming any point-group symmetry or assessing the degree of symmetry of class averages or individual particles of higher pointgroups (dn,t,o,i). The program takes as input an asymmetrical reconstruction or stack of class averages/individual particles. For volumes, the alignment document for all the particle images that have gone into the 3D reconstruction and the desired point-group symmetry needs to be inputted. The 3D reconstruction is then projected in 150 (default option) even directions, common lines-based optimisation is used to identify the principal symmetry axis, the rotational transformation is applied to the inputted orientations, and a new alignment document is produced. Input this document to recvol together with the images and the point-group symmetry to generate a symmetrised map. If you are unsure about the point-group, you should use the compare=yes mode and input the highest conceivable point-group. The program then calculates probabilities for all lower groups inclusive. The class average/particle option operates in an equivalent fashion but with individual images. The output is then a per-image correlation value that informs about how well the image conforms to to inputted point-group. The state parameter allows you to apply symmetry for the given state.

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
smpd    = sampling distance, same as EMANs apix(in A)
msk     = mask radius(in pixels)
pgrp    = point-group symmetry(cn|dn|t|o|i)
outfile = output document
lp      = low-pass limit(in A)

OPTIONAL
nthr    = nr of OpenMP threads{1}
vol1    = input volume no1(invol1.ext)
stk     = particle stack with all images(ptcls.ext)
oritab  = table (text file) of orientations(*.asc/*.txt)
cenlp   = low-pass limit for binarisation in centering
hp      = high-pass limit(in A)
nspace  = nr of projection directions
compare = do comparison(yes|no){no}
```

6.73 Program: tseries_split

tseries_split is a program for splitting time series

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...
```

```
REQUIRED
stk     = particle stack with all images(ptcls.ext)
oritab  = table (text file) of orientations(*.asc/*.txt)
chunksz = number of images/orientations in chunk
jumpsz  = size of contigous segment
```

6.74 Program: unblur_movies

unblur_movies is a program for movie alignment or unblurring. Input is a textfile with absolute paths to movie files in addition to a few obvious input parameters

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
filetab = list of files(*.txt/*.asc)
smpd    = sampling distance, same as EMANs apix(in A)

OPTIONAL
nthr      = nr of OpenMP threads{1}
fbody     = file body
lpstart   = start low-pass limit{15}
lpstop    = stop low-pass limit{8}
trs       = maximum halfwidth shift(in pixels)
pspecsz   = size of power spectrum(in pixels)
numlen    = length of number string
startit   = start iterating from here
frameavg  = nr of frames to average{0}
tomo      = tomography mode(yes|no){no}
```

6.75 Program: volassemble

volassemble is a program that assembles volume(s) when the reconstruction program (recvol) has been executed in distributed mode. odd is used to assemble the odd reconstruction, even is used to assemble the even reconstruction, eo is used to assemble both the even and the odd reconstruction and state is used to assemble the inputted state. Normally, you do not fiddle with these parameters. They are used internally

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
stk    = particle stack with all images(ptcls.ext)
nparts = nr of partitions in distributed exection
smpd   = sampling distance, same as EMANs apix(in A)
oritab = table (text file) of orientations(*.asc/*.txt)

OPTIONAL
nthr  = nr of OpenMP threads{1}
even  = calculate even eo-pair(yes|no){no}
odd   = calculate odd eo-pair(yes|no){no}
eo    = use FSC for filtering and low-pass limit update(yes|no){no}
state = state to extract
xfel  = images are XFEL diffraction patterns(yes|no){no}
```

6.76 Program: volaverager

volaverager is a program for averaging volumes according to state label in oritab

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
vollist = table (text file) of volume files(*.txt/*.asc)
oritab  = table (text file) of orientations(*.asc/*.txt)

OPTIONAL
nthr = nr of OpenMP threads{1}
```

6.77 Program: volops

volops provides standard single-particle image processing routines that are applied to MRC or SPIDER volumes

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
vol1 = input volume no1(invol1.ext)

OPTIONAL
nthr    = nr of OpenMP threads{1}
guinier = calculate Guinier plot(yes|no){no}
smpd    = sampling distance, same as EMANs apix(in A)
hp      = high-pass limit(in A)
lp      = low-pass limit(in A)
neg     = invert contrast of images(yes|no)
snr     = signal-to-noise ratio
mirr    = mirror(no|x|y){no}
outvol  = output volume{outvol.ext}
```

6.78 Program: volume_smat

volume_smat is a program for creating a similarity matrix based on volume2volume correlation

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
vollist = table (text file) of volume files(*.txt/*.asc)
smpd    = sampling distance, same as EMANs apix(in A)

OPTIONAL
nthr = nr of OpenMP threads{1}
hp   = high-pass limit(in A)
lp   = low-pass limit(in A)
msk  = mask radius(in pixels)
```

7 Distributed SIMPLE Workflows

7.1 Program: ctffind

ctffind is a wrapper program for CTFFIND4 (Grigorieff lab)

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
filetab = list of files(*.txt/*.asc)
smpd    = sampling distance, same as EMANs apix(in A)
kv      = acceleration voltage(in kV){300.}
cs      = spherical aberration constant(in mm){2.7}
fraca   = fraction of amplitude contrast used for fitting CTF{0.07}
nthr    = nr of OpenMP threads{1}
nparts  = nr of partitions in distributed exection

OPTIONAL
ncunits   = number of computing units, can be < nparts {nparts}
pspecsz   = size of power spectrum(in pixels)
hp        = high-pass limit(in A)
lp        = low-pass limit(in A)
dfmin     = minimum expected defocus(in microns)
dfmax     = maximum expected defocus(in microns)
astigstep = step size for astigamtism search(in microns)
expastig  = expected astigmatism(in microns)
```

7.2 Program: find_nnimgs

find_nnimgs is a program for cidentifying the nnn nearest neighbor images for each image in the inputted stack

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
stk    = particle stack with all images(ptcls.ext)
smpd   = sampling distance, same as EMANs apix(in A)
msk    = mask radius(in pixels)
nthr   = nr of OpenMP threads{1}
nparts = nr of partitions in distributed exection

OPTIONAL
ncunits = number of computing units, can be < nparts {nparts}
nnn     = number of nearest neighbors{500}
lp      = low-pass limit(in A)
hp      = high-pass limit(in A)
```

7.3 Program: ini3D_from_cavgs

ini3D_from_cavgs is a program for generating an initial 3D model from class averages obtained with prime2D

```
USAGE:
```

```
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
stk    = particle stack with all images(ptcls.ext)
smpd   = sampling distance, same as EMANs apix(in A)
msk    = mask radius(in pixels)
pgrp   = point-group symmetry(cn|dn|t|o|i)
nthr   = nr of OpenMP threads{1}
nparts = nr of partitions in distributed exection

OPTIONAL
ncunits     = number of computing units, can be < nparts {nparts}
nthr_master = nr of OpenMP threads on master node{1}
hp          = high-pass limit(in A)
lp          = low-pass limit(in A)
frac        = fraction of ptcls(0-1){1}
automsk     = envelope masking(yes|no|cavg){no}
mw          = molecular weight(in kD)
amsklp      = low-pass limit for envelope mask generation(in A)
edge        = edge size for softening molecular envelope(in pixels)
binwidth    = binary layers grown for molecular envelope(in pixels){1}
inner       = inner mask radius(in pixels)
width       = falloff of inner mask(in pixels){10}
nspace      = nr of projection directions
shbarrier   = use shift search barrier constraint(yes|no){yes}
```

7.4 Program: prime2D

prime2D is a reference-free 2D alignment/clustering algorithm adopted from the prime3D probabilistic ab initio 3D reconstruction algorithm

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
stk    = particle stack with all images(ptcls.ext)
smpd   = sampling distance, same as EMANs apix(in A)
msk    = mask radius(in pixels)
ncls   = nr of clusters
ctf    = ctf flag(yes|no|flip)
nparts = nr of partitions in distributed exection
nthr   = nr of OpenMP threads{1}

OPTIONAL
ncunits   = number of computing units, can be < nparts {nparts}
deftab    = text file with CTF info(*.txt/*.asc)
refine    = refinement mode(no|shc|neigh|shcneigh|qcont|qcontneigh|shift){no}
refs      = initial2Dreferences.ext
oritab    = table (text file) of orientations(*.asc/*.txt)
hp        = high-pass limit(in A)
lp        = low-pass limit(in A)
cenlp     = low-pass limit for binarisation in centering
trs       = maximum halfwidth shift(in pixels)
automsk   = envelope masking(yes|no|cavg){no}
```

```
amsklp    = low-pass limit for envelope mask generation(in A)
inner     = inner mask radius(in pixels)
width     = falloff of inner mask(in pixels){10}
startit   = start iterating from here
maxits    = maximum number of iterations{500}
filwidth  = width of filament (in A)
srch_inpl = search in-plane degrees of freedom(yes|no){yes}
nnn       = number of nearest neighbors{500}
minp      = minimum cluster population
```

7.5 Program: prime2D_init

prime2D_init is used to produce the initial random references for prime2D execution. The random clustering and in-plane alignment is printed in the file prime2D_startdoc.txt produced by the program. This file is used together with the initial references (startcavgs.ext) to execute prime2D

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
stk    = particle stack with all images(ptcls.ext)
smpd   = sampling distance, same as EMANs apix(in A)
ncls   = nr of clusters
ctf    = ctf flag(yes|no|flip)
nthr   = nr of OpenMP threads{1}
nparts = nr of partitions in distributed exection

OPTIONAL
ncunits   = number of computing units, can be < nparts {nparts}
deftab    = text file with CTF info(*.txt/*.asc)
oritab    = table (text file) of orientations(*.asc/*.txt)
filwidth  = width of filament (in A)
mul       = origin shift multiplication factor{1}
srch_inpl = search in-plane degrees of freedom(yes|no){yes}
```

7.6 Program: prime3D

prime3D is an ab inito reconstruction/refinement program based on probabilistic projection matching. PRIME is short for PRobabilistic Initial 3D Model generation for Single- particle cryo-Electron microscopy. Do not search the origin shifts initially, when the model is of very low quality. If your images are far off centre, use stackops with option shalgn=yes instead to shiftalign the images beforehand (the algorithm implemented is the same as EMANs cenalignint program). We recommend running the first round of PRIME with the default dynamic resolution stepping dynlp=yes. The dynlp option implements a heuristic resolution weighting/update scheme. The initial low-pass limit is set so that each image receives ten nonzero orientation weights. When quasi-convergence has been reached, the limit is updated one Fourier index at the time until PRIME reaches the condition where six nonzero orientation weights are assigned to each image. FSC-based filtering is unfortunately not possible to do in the ab initio reconstruction step, because when the orientations are mostly random, the FSC overestimates the resolution. Once the initial model has converged, we recommend start searching the shifts (by setting trs to some nonzero value) and applying the FSC for resolution- weighting (by setting eo=yes)

```
USAGE:
```

```
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
stk    = particle stack with all images(ptcls.ext)
smpd   = sampling distance, same as EMANs apix(in A)
msk    = mask radius(in pixels)
ctf    = ctf flag(yes|no|flip)
pgrp   = point-group symmetry(cn|dn|t|o|i)
nthr   = nr of OpenMP threads{1}
nparts = nr of partitions in distributed exection

OPTIONAL
ncunits   = number of computing units, can be < nparts {nparts}
deftab    = text file with CTF info(*.txt/*.asc)
vol1      = input volume no1(invol1.ext)
oritab    = table (text file) of orientations(*.asc/*.txt)
trs       = maximum halfwidth shift(in pixels)
hp        = high-pass limit(in A)
lp        = low-pass limit(in A)
cenlp     = low-pass limit for binarisation in centering
dynlp     = automatic resolution limit update(yes|no){yes}
lpstart   = start low-pass limit{15}
lpstop    = stop low-pass limit{8}
eo        = use FSC for filtering and low-pass limit update(yes|no){no}
refine    = refinement mode(no|shc|neigh|shcneigh|qcont|qcontneigh|shift){no}
frac      = fraction of ptcls(0-1){1}
automsk   = envelope masking(yes|no|cavg){no}
mw        = molecular weight(in kD)
amsklp    = low-pass limit for envelope mask generation(in A)
edge      = edge size for softening molecular envelope(in pixels)
binwidth  = binary layers grown for molecular envelope(in pixels){1}
inner     = inner mask radius(in pixels)
width     = falloff of inner mask(in pixels){10}
nspace    = nr of projection directions
nstates   = nr of states to reconstruct
npeaks    = nr of nonzero orientation weights{1}
startit   = start iterating from here
maxits    = maximum number of iterations{500}
shbarrier = use shift search barrier constraint(yes|no){yes}
noise     = noise initialisation(yes|no){no}
xfel      = images are XFEL diffraction patterns(yes|no){no}
nnn       = number of nearest neighbors{500}
shellw    = shell-weight reconstruction (yes|no)
```

7.7 Program: prime3D_init

prime3D_init is a program for generating a random initial model for initialisation of PRIME3D

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
stk    = particle stack with all images(ptcls.ext)
smpd   = sampling distance, same as EMANs apix(in A)
```

```
msk    = mask radius(in pixels)
ctf    = ctf flag(yes|no|flip)
pgrp   = point-group symmetry(cn|dn|t|o|i)
nthr   = nr of OpenMP threads{1}
nparts = nr of partitions in distributed exection

OPTIONAL
ncunits = number of computing units, can be < nparts {nparts}
deftab  = text file with CTF info(*.txt/*.asc)
lp      = low-pass limit(in A)
inner   = inner mask radius(in pixels)
width   = falloff of inner mask(in pixels){10}
nspace  = nr of projection directions
nran    = number of random images to select
npeaks  = nr of nonzero orientation weights{1}
xfel    = images are XFEL diffraction patterns(yes|no){no}
```

7.8 Program: recvol

recvol is a program for reconstructing volumes from MRC and SPIDER stacks, given input orientations and state assignments. The algorithm is based on direct Fourier inversion with a Kaiser-Bessel (KB) interpolation kernel. This window function reduces the real-space ripple artifacts associated with direct moving windowed-sinc interpolation. The feature sought when implementing this algorithm was to enable quick, reliable reconstruction from aligned individual particle images. mul is used to scale the origin shifts if down-sampled were used for alignment and the original images are used for reconstruction. ctf=yes or ctf=flip turns on the Wiener restoration. If the images were phase-flipped set ctf=flip. amsklp, mw, and edge control the solvent mask: the low-pass limit used to generate the envelope; the molecular weight of the molecule (protein assumed but it works reasonably well also for RNA; slight modification of mw might be needed). The inner parameter controls the radius of the soft-edged mask used to remove the unordered DNA/RNA core of spherical icosahedral viruses

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
stk    = particle stack with all images(ptcls.ext)
smpd   = sampling distance, same as EMANs apix(in A)
oritab = table (text file) of orientations(*.asc/*.txt)
msk    = mask radius(in pixels)
ctf    = ctf flag(yes|no|flip)
pgrp   = point-group symmetry(cn|dn|t|o|i)
nthr   = nr of OpenMP threads{1}
nparts = nr of partitions in distributed exection

OPTIONAL
ncunits = number of computing units, can be < nparts {nparts}
eo      = use FSC for filtering and low-pass limit update(yes|no){no}
deftab  = text file with CTF info(*.txt/*.asc)
frac    = fraction of ptcls(0-1){1}
mw      = molecular weight(in kD)
mul     = origin shift multiplication factor{1}
state   = state to extract
shellw  = shell-weight reconstruction (yes|no)
```

```
vol1    = input volume no1(invol1.ext)
```

7.9 Program: shellweight3D

shellweight3D is a program for calculating the shell-by-shell resolution weights in a global sense, so that particles that do contribute with higher resolution information (as measure by the FRC) are given the appropriate weight

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
stk    = particle stack with all images(ptcls.ext)
vol1   = input volume no1(invol1.ext)
smpd   = sampling distance, same as EMANs apix(in A)
msk    = mask radius(in pixels)
oritab = table (text file) of orientations(*.asc/*.txt)
ctf    = ctf flag(yes|no|flip)
nthr   = nr of OpenMP threads{1}
nparts = nr of partitions in distributed exection

OPTIONAL
ncunits  = number of computing units, can be < nparts {nparts}
deftab   = text file with CTF info(*.txt/*.asc)
automsk  = envelope masking(yes|no|cavg){no}
mw       = molecular weight(in kD)
amsklp   = low-pass limit for envelope mask generation(in A)
edge     = edge size for softening molecular envelope(in pixels)
binwidth = binary layers grown for molecular envelope(in pixels){1}
inner    = inner mask radius(in pixels)
width    = falloff of inner mask(in pixels){10}
```

7.10 Program: unblur_movies

unblur_movies is a program for movie alignment or unblurring. Input is a textfile with absolute paths to movie files in addition to a few obvious input parameters

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
filetab = list of files(*.txt/*.asc)
smpd    = sampling distance, same as EMANs apix(in A)
nthr    = nr of OpenMP threads{1}
nparts  = nr of partitions in distributed exection

OPTIONAL
ncunits   = number of computing units, can be < nparts {nparts}
fbody     = file body
lpstart   = start low-pass limit{15}
lpstop    = stop low-pass limit{8}
trs       = maximum halfwidth shift(in pixels)
pspecsz   = size of power spectrum(in pixels)
numlen    = length of number string
startit   = start iterating from here
```

```
frameavg  = nr of frames to average{0}
```

7.11 Program: unblur_tomo_movies

unblur_tomo_movies is a program for movie alignment or unblurring of tomographic movies. Input is a textfile with absolute paths to movie files in addition to a few obvious input parameters

```
USAGE:
bash-3.2$ simple_exec prg=simple_program key1=val1 key2=val2 ...

REQUIRED
tomoseries = filetable of filetables of tomograms
exp_doc    = specifying exp_time and dose_rate per tomogram
smpd       = sampling distance, same as EMANs apix(in A)
nthr       = nr of OpenMP threads{1}
nparts     = nr of partitions in distributed exection

OPTIONAL
ncunits = number of computing units, can be < nparts {nparts}
lpstart = start low-pass limit{15}
lpstop  = stop low-pass limit{8}
trs     = maximum halfwidth shift(in pixels)
kv      = acceleration voltage(in kV){300.}
pspecsz = size of power spectrum(in pixels)
numlen  = length of number string
startit = start iterating from here
scale   = image scale factor{1}
```

References

Grigorieff, N., Jan 2007. Frealign: high-resolution refinement of single particle structures. J Struct Biol 157 (1), 117–25.

Mindell, J. A., Grigorieff, N., Jun 2003. Accurate determination of local defocus and specimen tilt in electron microscopy. J Struct Biol 142 (3), 334–47.

Rohou, A., Grigorieff, N., May 2014. Frealix: model-based refinement of helical filament structures from electron micrographs. J Struct Biol 186 (2), 234–44.

Rohou, A., Grigorieff, N., 2015. Ctffind4: Fast and accurate defocus estimation from electron micrographs. Journal of Structural Biology 192 (2), 216–221.

Scheres, S. H. W., Dec 2012. Relion: implementation of a bayesian approach to cryo-em structure determination. J Struct Biol 180 (3), 519–30.

Tang, G., Peng, L., Baldwin, P. R., Mann, D. S., Jiang, W., Rees, I., Ludtke, S. J., Jan 2007. Eman2: an extensible image processing suite for electron microscopy. J Struct Biol 157 (1), 38–46.