General guidelines for beta-release of RELION-3.0

August 8, 2018

Contents

1 Introduction

1.1 READ THIS!

These release notes accompany the beta-release of RELION-3.0, which includes many new features as listed below. This stage of beta-testing is intended to provide feedback and stability-testing to the gain of RELION and its users. Therefore, please read section 3 on how to provide us with your feedback. In section 2 we describe how you can access, install, *update* and use RELION-3.0_beta. It is important that you make sure to **update to the most recent version before reporting issues**, as they will get fixed and amended continually. RELION-3.0 comes with a new tutorial. Searching for '3.0' in this document is an efficient way to learn how to use it's new features. You can download the tutorial from our FTP server.

1.2 New features on the GUI

- **Motion correction:** Takanori Nakane at MRC-LMB has made our own implementation of the MotionCor2 algorithm, which is currently CPU-based only, but runs at comparable speed to the UCSF implementation for 4kx4k movies on many systems where disk access is limiting. We have removed the UNBLUR wrapper from the GUI. The jobtype now writes out a logfile.pdf with useful figures.

- **CTF estimation:** The jobtype now writes out a logfile.pdf with useful figures.

- **Auto-picking:** We have implemented a GUI-based option to provide a 3D reference to generate 2D templates for the auto-picking. We have also implemented a new reference-free auto-picking algorithm based on Laplacian-of-Gaussian filters (which are closely related to the Difference-of-Gaussian filters in DoG-picker). We have also implemented an amyloid-specific picker on the 'Helix' tab. The jobtype now writes out a logfile.pdf with useful figures.

- **Particle extraction:** One can re-extract refined particles using a user-defined new X,Y,Z-center in the 3D reference. Thereby, one can for example re-extract particles that are centered on a specific domain for focused refinements with partial image subtraction. This functionality overlaps with the scripts for localised reconstruction by Juha Huiskonen. Similar functionality has also been implemented in the `relion_star_handler` program mentioned below.

- **Subset selection:** A new 'Subsets' tab on the GUI allows selection of subsets based on metadata, on image statistics and to split the data into (random or ordered) subsets. There is now also an option to discard duplicate particles.

- **2D/3D classification:** On the 'Optimisation tab, there is a new option called 'Use fast subsets?'. This implements a procedure to use fewer images in the first few iterations of the classification process, which we basically copied from Tim Grant, Alexis Rohou and Niko Grigorieff's cisTEM.

- **3D initial model:** We have completely re-written this. It now follows much more closely the algorithm implemented in Ali Punjani and Marcus Brubaker's cryoSPARC. This has greatly improved it's robustness compared to RELION-2.1, and the results now closely match those from cryoSPARC.

- **3D multi-body:** This is a new, automated procedure for simultaneous focused refinement with partial image subtraction on multiple independently moving parts of your large complex. The details have been published in this eLife paper by Nakane et al. Please note that the minimum size of a body is probably above 150 kDa, and that multi-body refinement is probably not much better than careful (repeated) image subtraction by an experienced user. It does however offer a

convenient and easy-to-use method for focused refinement, and produces movies of the principal motions encountered in the data set.

- **CTF refinement:** This was done by Jasenko Zivanov at MRC-LMB. The program refines the defocus values of individual particles by comparing them to high-resolution reference projections. Note the options to refine astigmatism will require *very* strong data. The same program can also be used to estimate the amount of beamtilt as an average over all particles in input STAR file. We find that defocus refinement and/or beam-tilt correction can improve the resolutions of many data sets.

- **Bayesian polishing:** Jasenko Zivanov has also implemented a completely new way of estimating the trajectories of particle motion and the B-factors that model radiation damage. His method associates with each hypothetical set of particle trajectories a prior likelihood that favors spatially coherent and temporally smooth motion without imposing any hard constraints. The dip in B-factors for the first few movie frames that was often observed in previous versions of RELION disappears using this method. Therefore, we strongly recommend against throwing away those first few movie frames (or the later ones). As it outperform the old movie refinement and particle polishing methods in all cases tested, Bayesian polishing replaces these methods on the GUI. It can directly work with compressed TIFFs (doing gain-correction on the fly) and no longer requires writing out aligned movie stacks or movie particles. *Note that one has to run the MotionCorr job in* RELION*-3.0 before one can do Bayesian polishing. Jobs run in* RELION*-2.1, or through custom scripts will not be compatible with Bayesian polishing, as the new MotionCor2 wrapper hands over a lot more metadata than the old one.* We are currently writing up a paper about the Bayesian polishing, and we will shortly put it on bioRxiv.

- **Mask creation:** This has now been parallelised using (OpenMP) threads.

1.3 New features on the command line

- **relion_reconstruct:** This program has finally been parallelised using MPI. Also, it now implements Ewald sphere correction, as proposed by Chris Russo and Richard Henderson. Make sure you also try the reverse curvature when testing this. For 300kV data, Ewald sphere correction is only useful at relatively high resolutions and for large particles.

- **relion_star_handler:** This new program implements a range of useful utilities for dealing with STAR files. New functions include adding or removing columns, setting certain columns to certain values, perform operations on columns etc.

- **relion_align_symmetry:** This new program aligns the symmetry axes of an input 3D map in an arbitrary orientation, e.g. as obtained from de novo initial model generation in C1.

- **relion_it.py:** This python script can be found in the `scripts` directory of your installation. It can be used for fully-automated on-the-fly image processing. It depends on a new command-line interface to scheduling and executing relion jobs inside the pipeline. We envision that these scripts are relatively easy to modify to the individual needs of many groups. More detailed documentation for the current script is still pending, but the script already writes the most useful instructions to the screen.

- **bfactor_plot.py:** This python scripts uses the same functionality to run multiple 3D auto-refinement and post-processing jobs to automatically generate a PDF file with the Rosenthal and Henderson $\ln(N)$ vs $1/d^2$ B-factor plot

1.4 CPU acceleration

GPU-acceleration was introduced into RELION-2.0 by writing fast functions in CUDA. For RELION-3.0, the acceleration of `relion_refine` has been generalised to a dual CPU/GPU format, so that the program runs the same parallelisation scheme on both architectures. Especially the more recent generations of CPUs make efficient use of vectorised execution, performing many instructions in parallel. In its current form, you can choose to have either of the three combinations when executing this program:

- Legacy CPU-execution
- Legacy CPU-execution + GPU-acceleration
- Legacy CPU-execution + CPU-acceleration

We realize that the most appealing combination is dual acceleration, however this is easier said than done for all compilers. We are working to unify the compilation, and will shortly provide a version which has all three in one compilation. For now, however, CPU-acceleration and GPU-acceleration needs to be compiled separately. To activate the accelerated version, use the flag `--cpu` together with a CPU-accelerated binary, just like you would use `--gpu` together with a GPU-accelerated binary.

More detailed descriptions of **how to compile**, and what configurations are supported, are given below in section 2.4

2 Download and install

2.1 Getting the code

We temporarily store the RELION-3.0_beta code on `bitbucket.org`. We will not describe the use of bitbucket and git, as you will not need more than very basic features. Below we outline the few commands needed on a UNIX-system, please refer to general git descriptions and tutorials to suit your system. To get the code, you must "clone" the repository from bitbucket. Use the following command-line to download the code:

```
git clone https://bitbucket.org/scheres/relion-3.0_beta.git
```

This will create a local git repository. All subsequent git-commands should be run inside this directory.

2.2 Updating

The code will be intermittently updated to amend issues detected during beta-development. To incorporate these changes, inside your local repository (the source-code directory downloaded) use the command:

```
git pull
```

If you have changed the code in some way, this will force you to commit a local merge. You are free to do so, but we will assume you have not changed the code. Refer to external instructions regarding git and merging so-called conflicts if you have changed the code and need to keep those changes.

2.3 Installing

RELION-3.0 has an installation procedure which relies on `cmake`. Only minor changes have been made to the installation procedure since the previous RELION-2.1 version. You will need to have `cmake` installed, but most UNIX-systems have this by default. You will need to make a build-directory in which the code will be compiled. This can be placed inside the repository:

```
cd relion-3.0_beta
mkdir build
cd build
```

You then invoke cmake inside the build-directory, but point to the source-directory to configure the installation. This will *not* install RELION, just configure the build:

```
cmake ..
```

As before, check the output to make sure CUDA, FFTW, FLTK, MPI and libtiff are all detected as expected. This should all be very similar to installing RELION-2.1, though the requirement on libtiff is new in RELION-3.0.

2.4 Compiling the CPU-accelerated code

In order to run as fast as possible, we want to make maximal use of the hardware. If you compile the program for your specific hardware, you will thus get the best performance. If you are using Intel CPUs, you will get better performance by compiling with the Intel(R) Compiler (icc), and depending on which generation CPUs you have, you will also benefit from specifying which type of vectorisation (parallelisation) that CPU can use.

To build generic support for CPU-accelerated kernels in addition to the original CPU version, build by setting 'ALTCPU=ON':

```
cd build
rm -r *
cmake -DALTCPU=ON ..
make -j4
make install
```

This will require the TBB (Threading Building Blocks) library. RELION will look for TBB and fetch+install it if it cannot find it on your system. You can force this behaviour (and make sure you are using the latest version) by adding:

```
-DFORCE_OWN_TBB=ON
```

to your cmake configuration command. In addition, you can make use of the Intel(R) Math Kernel Library (Intel(R) MKL) when using the icc-compiler. This is optional (but will scale better with increased threads), and can be added through:

```
-DMKLFFT=ON
```

2.4.1 Supported configurations

GCC versions later than 4.9 (support for lambda-expressions is required)
ICC versions with 2018 update 3 and later

2.4.2 Building with the Intel(R) Compiler

The CPU-accelerated code seems to run fastest using the Intel(R) Compiler. Build it this way as follows:

```
source /opt/intel/impi/<version>/intel64/bin/mpivars.sh intel64cd build
source /opt/intel/compilers_and_libraries_<version>/linux/bin/compilervars.sh intel64
source /opt/intel/compilers_and_libraries_<version>/linux/mkl/bin/mklvars.sh intel64
cd build
rm -r *
CC=mpiicc CXX=mpiicpc cmake -DALTCPU=ON -DMKLFFT=ON ..
make -j4
make install
```

2.4.3 Hardware-specific vectorisation using the Intel(R) Compiler

If you are compiling for a cluster, or just want to get the most out of your hardware, you can specify that the binary be limited to a specific type of hardware (vectorisation), in order to get the best possible performance.

Best performance on **AVX2** platforms has been measured using the following cmake-configuration flags:

```
-DALTCPU=ON -DMKLFFT=ON
-DCMAKE_C_FLAGS="-O3 -ip -g -debug inline-debug-info -xCORE-AVX2 -restrict "
-DCMAKE_CXX_FLAGS="-O3 -ip -g -debug inline-debug-info -xCORE-AVX2 -restrict "
```

Note that this binary will only run on AVX2 platforms.

Best performance on **AVX512** platforms has been measured using using these flags:

```
-DALTCPU=ON -DMKLFFT=ON
-DCMAKE_C_FLAGS="-O3 -ip -g -debug inline-debug-info -xCORE-AVX512 -qopt-zmm-usage=high -restrict "
-DCMAKE_CXX_FLAGS="-O3 -ip -g -debug inline-debug-info -xCORE-AVX512 -qopt-zmm-usage=high -restrict
```

Note that the resulting binary will only work on AVX512 platforms.

Best performance on Intel **Xeon Phi** Processors (x700 Family) has been measured using these flags:

```
-DALTCPU=ON -DMKLFFT=ON
-DCMAKE_C_FLAGS="-O3 -ip -g -debug inline-debug-info -xMIC-AVX512 -restrict "
-DCMAKE_CXX_FLAGS="-O3 -ip -g -debug inline-debug-info -xMIC-AVX512 -restrict "
```

Note that this binary will only run on platforms of the Xeon Phi processor family.

3 Providing feedback

3.1 Reporting bugs

We have tested this code extensively at MRC-LMB and in the SciLife lab, and it has now also been tested by several other labs. This however does not mean there are no remaining issues, as many different users may use the program in many different ways, and some changes have still been made recently.

If you find what you think is a bug, please report it though the bitbucket issue-tracker through this link. In that case, please do tell us about you operating system, CUDA version, compiler, CMake flags if you suspect things might be wrong with the compilation; and/or run.out, run.err and the exact command-line argument to the relion program of interest if something is wrong with a specific job.

3.2 Asking questions

As the beta-release is now open to everyone, please do use the CCPEM mailing list to ask questions about how to use RELION-3.0.