ETSimulations User Manual

Kyung Min Shin

Ver 0.2 - March 2020

Contents

1 Getting started

The ETSimulations package is designed to be a tool to facilitate generation and processing of simulated cryo-electron tomography data. As of this preliminary version, only the generation of simulated tilt stacks is well-supported and covered by this manual.

External dependencies

The following external software is used by ETSimulations and should thus be installed first:

- Chimera - https://www.cgl.ucsf.edu/chimera/
- TEM-simulator - http://tem-simulator.sourceforge.net/
- Python 3.3 or greater

Setting up ETSimulations

To begin, navigate to a directory in which you want to install the package and run the following commands:

git clone https://github.com/kmshin1397/ETSimulations.git

cd ETSimulations

python3 -m venv env

source env/bin/activate

pip install -r requirements.txt

This will have downloaded the ETSimulations code and set up the necessary Python packages and environment.

Running a simulation

Most all general parameter set up is done through a YAML file which is passed in as an argument to the main **ets_generate_data.py** program. An example such YAML file is provided in the ETSimulations directory as **configs.yaml**.

Detailed set-up with regards to the characteristics of the particles simulated are controlled through custom **"Assembler"** Python classes which can define a series of Chimera commands to open, manipulate, and combine one or more source maps, i.e. various proteins, into a fake particle source. (This is done for the T4SS simulations in the src/assemblers/t4ss_assembler.py file) Custom arguments to pass

along to your custom particle Assemblers can also be defined in the configuration YAML file, as shown in the example.

To actually run a set of simulations, run (assuming you've activated the virtual environment as indicated above - the **source ...** command):

python src/ets_generate_data.py -i configs.yaml

More details on the parameters available through the configurations can be found in the next section.

Processing a simulated dataset

Another program called **ets_process_data.py** is provided to facilitate processing of the raw datasets generated by the simulations. Here as well, we provide input via a configuration YAML file. This time parameters are given to modules called **"Processors"**. Each Processor module implements a function to take the inputted configurations and automatically set up tomogram reconstruction and sub-tomogram averaging. For example, the EMAN2 processor (the only one implemented so far) will automatically create an EMAN2 project and generate a Python script which will run all desired EMAN2 processing steps (i.e. e2import.py, e2tomogram.py, etc.) in order.

The YAML file provided to the ets_process_data.py run should have:

- root: The path to the folder which contains the raw_data and sim_metadata.json files created by ets_generate_data.py
- name: The project name - should be the same one used for the simulations
- processors: A list of processor objects which have a "name" and "args" arguments. The specifics to what goes into these fields can be found in more details in the section of the manual particular to that processor.

To run the processor set up, run:

python src/ets_process_data.py -i processor_configs.yaml

2 Simulation configuration parameters

- tem_simulator_executable : The file path to the TEM-simulator executable
- chimera_exec_path : The file path to your Chimera installation

- model : The path to the main particle source file (Doesn't actually matter for T4SS simulations because I'm bypassing this and putting together a bunch of different source maps)
- root: The project root directory in which to generate simulations
- config: The TEM-simulator configuration text file to apply to each simulation. An example is provided in the templates folder.
- coord: TheTEM-simulator particle coordinates text file to use as a reference for placing particles in each generated tiltseries. An example for this is also provided in the templates folder.
- num_stacks: The number of tilt stacks to generate
- name: A name for the project
- num_cores: The number of parallel cores to utilize
- apix: The pixel size to give to generated stacks, in nm
- num_chimera_windows: The number of Chimera instances to spawn to drive particle assembly. Creating more windows will clutter your display more, but can make simulations run faster. If your particle assembly does not use a lot of Chimera commands/spend a lot of time running Chimera commands, then having multiple Chimera windows may not be necessary
- bead_map : The MRC map representing fake gold beads to scatter throughout the tilt stacks. An example is provided in the templates folder
- email : An email address to send completion notifications to

3 Dataset generation outputs

Running the **ets_generate_data.py** program as discussed in section 1.3 will result in a **raw_data** folder being created in the project directory specified in the configurations. In the **raw_data** folder, each tiltseries will get its own sub-directory titled {name}_{stack number}. In each sub-directory, you will find a no-noise version of the stack and a normal noisy version.

The other important output to note is the sim_metadata.json file. This is a JSON file containing metadata for each tiltseries generated, including custom metadata that can be saved from your custom Assembler. For example, the T4SS Assembler saves the random orientations and random shifts/angles away from the centered/perpendicular positions for each component of the simulated particle which were generated during the run. An easy way to interact with and retrieve this information is the Python json module which can load this json as a Python dictionary, i.e.:

```
import json
metadata = json.load(open("sim_metadata.json", "r"))
```

4 The EMAN2 Processor

The EMAN2 Processor, found in processors/eman2_processor.py, is implemented to automatically set up cryo-ET processing of generated simulated data. It works by taking a pre-written Python template script, which lists the EMAN2 programs needed to run to take a series of raw tilt stacks to a subtomogram average, and filling in the desired parameters to those programs based on configurations provided in the YAML input file. This template script can be found at templates/eman2/eman2_process.py. Any argument which can be given to the EMAN2 programs called can be given in the YAML file as part of the parameters object for the program. Note the example configurations below.

```
processors: [
  {name: "eman2",
   args:
  {
     particle_coordinates_file: "/Users/kshin/Documents/repositories/ETSimulations
/templates/eman2/T4SS_coords_3by3.txt",
     steps_to_run: ["import", "reconstruct"],
     e2import_parameters : {
      "import_tiltseries": "enable",
      "importation": "copy",
      "apix": 2.83,
      "boxsize": 64
    },
    e2tomogram_parameters : {
      "tltstep": 3,
      "tltax": -90,
      "npk": 10,
      "tltkeep": 0.9,
      "outsize": "1k",
      "niter": "2,1,1,1",
      "pkkeep": 0.9,
      "bxsz": 64,
      "pk_mindist": 0.125,
      "filterto": 0.45,
      "rmbeadthr": 10.0,
      "threads": 48,
      "clipz": 350,
      "notmp": "enable"
    }
}]
```

Like the example above, you can include EMAN2 processing in your processing run by adding a processor argument to the "processors" list with the name "eman2" and suitable args.

steps_to_run

In the "args" field, the "steps_to_run" lists the processing steps which will be run when you eventually run the script generated by the processor (Note that the functions will still exist in the script to run steps not included in the list, they just won't be executed unless you manually enable them back). The available steps taken by the steps_to_run are "import", "reconstruct", "extract", "initial_model", and "average". Note also that the order matters in this list (i.e. reconstructing before importing tiltseries will result errors).

particles_coordinates_file

The particles_coordinates_file argument is used to define a text file containing the 3D coordinates of the particles within the simulated tomograms. This will be used during the "extract" processing step to transfer particle coordinates to the EMAN2 project as if picked in Boxer so that the sub-volumes can be extracted for sub-tomogram averaging. The coordinates in this file should be in pixels, based on the EMAN2 conventions (with the origin being in the center of the axis instead of the lower left, for example). An example such file is provided at templates/eman2/T4SS_coords_3by3.txt.

Parameters objects

Each EMAN2 program is given its own *_parameters section in the "args" field, listing all the command line arguments that would be passed in if calling these programs normally. For example, arguments to e2import.py would be listed in the e2import_parameters field as shown. Arguments which are just flags instead of taking a value, such as the "–help" option available in all these programs, should be put in to the configuration section with the special value of "enable" as can be seen in the example above. The full list of EMAN2 programs exposed by the EMAN2 Processor, and thus able to take their own *_parameters section is:

- e2import.py : e2import_parameters
- e2tomogram.py : e2tomogram_parameters
- e2spt_extract.py : e2spt_extract_parameters
- e2spt_buildsets.py : e2spt_buildsets_parameters
- e2spt_sgd.py : e2spt_sgd_parameters
- e2spt_refine.py : e2spt_refine_parameters