ETSimulations User Manual

Kyung Min Shin

Ver 0.1 - 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_run.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_run.py -i configs.yaml
```

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

2 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_run.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 and inverted version. The normal stack is the default output of the TEM-simulator, which contains densities as whiter. The inverted version has densities as darker, as is more commonly used with other cryo-EM software.

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"))
```