Download and Installation

Download a release from the CTS github page. Unzip the archive into your matlab/toolbox folder, and add the resulting CTS folder to your Matlab path. Currently, adding subfolders is irrelevant as CTS has none. IMOD must also be installed to run simulations and reconstructions.

CTS requires a matlab version 2019b or later, and the image processing and statistics & machine learning toolbox. These are standard with most matlab licenses. CTS also requires the [EMIOD toolbox](https://github.com/rbehrouzi/emtoolbox) for matlab, installed in the same way.

Download at least one structure file from a database such as the RCSB for testing. Cif files are preferred as they have a few very useful features, but pdb files are perfectly serviceable.

Minimal commands for a basic workflow

WARNING: PASTING INTO MATLAB MIGHT BREAK CHARACTERS. ‘ and “ (single or double quotes) may be converted to improper special characters. If it’s red, it’s broken. If the argument is purple, it’s working. Replace the broken instance with a manually typed one when necessary.

1. set up the model parameters. This command readies parameters, with most using the defaults but specifying a pixel size of 12 (required input), and a single layer of particles (optional, defaults to 1). A GUI will appear to select the input structures for each layer. This command can be run inside the next command as an argument, or you can provide the arguments as a cell array (in curly brackets). Creating a variable makes it fast and easy to run many model generations with the same parameters programmatically, and avoids needing to use a GUI to select files repeatedly. Estimated runtime: <1 minute (will be longer if large numbers of structures are input).

[parammodel] = param\_model(12,'layers',1);

help param\_model for a manual of model parameter arguments and options.

1. Generate the model with the provided parameters. The model size is 300x400x50 voxels, and is output with the suffix \_model\_1. The files are in the /tomosim directory, in a folder named with the input structure names following a timestamp. The second command demonstrates how to include the model parameters inside the same argument, if you do not wish to store them as a variable. Estimated runtime: <2 minutes

[cts] = cts\_model(zeros(300,400,50),parammodel,'suffix','model\_1');

Or [cts] = cts\_model(zeros(300,400,50),{12});

1. View the model if desired. The default model parameters include a grid, so the edge of a carbon hole should be visible along the left side. Subcomponents of the model are also viewable, stored inside the cts struct variable. The particle types are stored across a number of models in the splitmodel field, according to their filename. cts.splitmodel run in the command window will display its contents for viewing with the second example command.

sliceViewer(cts.vol);

sliceViewer(cts.splitmodel.X);

1. Set up simulation parameters. This functions similarly to the param\_model function, but has no required inputs – all options have a default value. The following command is the easiest to use, as it provides a GUI to input all the parameters with default values already filled in. To skip the GUI, name-value pairs are used to change parameters from default values. If you do not want to change any parameters, you can simply not provide this argument to the simulator function – it will automatically supply defaults.

paramsim = param\_simulate('gui');

1. Run the simulation. This uses input (or default) parameters to project an initial tiltseries, detect electron scattering, CTF convolve, and reconstruct a tomogram, along with generating an atlas of object identities. The input model is selected with a GUI (change ‘gui’ to a full path to use a file programmatically) – you can use either a model .mrc or .mat file, but the .mat file is required to generate the atlas. Estimated runtime: <1 minute

cts\_simulate('gui',paramsim,'suffix','tutorial1');

cts\_simulate('gui','suffix','tutorial1'); %using default parameters only

The simulation steps will be output into a subfolder inside the model folder that was used, and in this example the simulation folder will have \_tutorial1 as a suffix. The easiest way to view all the steps in series is with IMOD’s 3dmod command in that folder, with 3dmod \*.mrc. the 5\_recon\_X is the reconstructed tomogram, and atlas\_X is the atlas of particle classes.

For more detailed information on each command, including a list of all input/output options, use matlab’s built-in help functionality. The following is the syntax for retrieving documentation:

help command