# CellOrganizer for Docker Tutorial

V2.10, May 13, 2023

## What CellOrganizer does

- CellOrganizer is an open source system for using microscope images to learn statistical models of the structure of cell components and of how those components are organized relative to each other.
- Since these models can be used to synthesize new images reflecting what the model learned, they are referred to as generative models.
- For example, CellOrganizer can take fluorescence microscope images of many cells expressing a GFP-tagged lysosomal protein and construct a model of how the cell and nuclear shapes vary among those cells, how many lysosomes are typically found per cell, how they vary in size, and where they are typically located relative to the cell and nuclear boundaries.
- Publications describing the development and use of various software components of CellOrganizer are located here. An overview of the principles behind CellOrganizer can be found here.
- CellOrganizer can learn many different types of models that are appropriate for different cell properties.

## About the tutorial

- You will need to install Docker and pull the CellOrganizer Docker image.
- The tutorials highlight different capabilities of CellOrganizer, using provided example image datasets and existing models.
- If you would like to build CellOrganizer models with your own images, please see the text under Tutorial 1 below about appropriate organization and file formats.
- **The MMBioS CellOrganizer team is available to answer questions, help diagnose problems, receive requests for new features, and generally try to ensure the success of your projects using CellOrganizer for your projects. You can contact us by emailing cellorganizer-dev@compbio.cmu.edu.**

## File housekeeping

- When you run the CellOrganizer Docker container, it connects to a local folder that will appear as *local* in the main Jupyter window. After you have run the *Download_files* notebook (during installation), three folders will have been created within this local folder: *models, images,* and *notebooks*. This tutorial uses the files in the *workshop_demos* folder in the *notebooks* folder.
- Output files from most notebooks are created in a *results* folder (which will automatically be created by the first notebook that uses it). Inside that folder will be folders corresponding to each tutorial Module.

## The CellOrganizer API

- There are four main CellOrganizer functions. *img2slml* makes models from images. *slml2img* synthesizes images from models. *slml2info* describes a model. *slml2report* compares two different models. There are also auxiliary functions for working directly with object shape representations.

## Tutorial Modules

# Tutorial Module 1. Working with images

This tutorial covers how to organize your own images for use with CellOrganizer. Typically, telling CellOrganizer what files to use to construct a model is done using wildcards (e.g., "Expt1Sample*Nuclear.tif"), and it is helpful to organize your files to facilitate this.

*You can do this module later if you first want to learn how to use CellOrganizer using provided images.*

**CellOrganizer requires that its input be single cells.** There are three major options for how to do this. The first is to have a separate file for each cell. The second is to have multiple cells in each image (the way they were presumably acquired) and provide an additional file containing a "cell mask" to specify where a cell is located within the image; this file is a binary image in which the on pixels are to be included in the cell. The third is to have multiple cells in an OME-TIFF file and include Regions Of Interest (ROIs) to for each cell.

**CellOrganizer also needs to know which fluorescence channel should be used to construct which parts of the model.** There are again two basic ways to do this. The first is to have separate files for each channel, and include into the appropriate argument when calling CellOrganizer. The second is to have all channels in the same file and specify the channel number as part of the filenames. The second approach requires that you use OME-TIFF files.

*Doing cell segmentation and saving images into OME-TIFF files is covered in Tutorial Module 5.*

- Module 1A shows how to provide input images in the three different ways and builds a simple model to verify that the input images are provided properly

  The notebook has example code for reading files into CellOrganizer the three different ways: files contain single channel/single cell, files contain single channel/multiple cells, and files contain multiple channels/multiple cells. It is initially set to use the first method, but you can comment out that cell and uncomment a different cell to use one of the other methods. You can also modify the notebook to use your own images and run the notebook to see if they are being read properly.

- Module 1B builds a more accurate model of cell and nuclear shape (using the SPHARM-RPDM method)

SPHARM-RPDM models provide a much more accurate representation of cell and/or nuclear shape. They take longer to calculate and therefore this Module downsamples the images before building the model so that it takes less time.

*The point of this module is to illustrate how you can change the type of model being built. Note that changing the type of model may change which options are used to fine-tune the model learning process. See XXX for a list of the different types of models and which options apply to each.*

## Tutorial Module 2. Working with shape models

Tutorial 2 notebooks contain example code which can be run to create shape models using *img2slml* and then illustrates some things that can be done with those models using *slml2info, slml2report* and *slml2img.*

- Module 2A uses *img2slml* to train **two** cell and nuclear shape models for later comparison
  (Input: single cell dataset; output: two models, Module2A1 and Module2A2)

- Module 2B uses *slml2info* to generate and customize reports
  (Input: a model file; output: index.html and various PNG files containing plots)

- Module 2C uses *slml2report* to compare cell shape models
  (Input: two model files; output: index.html and various PNG files containing plots)

- Module 2D uses *slml2img* to generate a synthetic cell
  (Input: a model file; output: multi-image TIFF files containing cell and nuclear masks)

  The notebook creates a folder with the same name as the model that will contain a folder name *img* containing a folder named *cell1*. This will contain files *cell1.tif* and *nucleus.tif* which will contain 2D images corresponding to consecutive slices of the 3D cell.

- Module 2E uses *slml2info* to generate a movie showing one cell shape "evolving" into another cell shape
  (Input: a model file; output: AVI file showing one cell morphing into another)

## Tutorial Module 3. Working with basic organelle models

Tutorial 3 notebooks introduce organelle models. The properties of the models can be visualized through *slml2info* and *slml2report.*

- Module 3A Builds a basic cell, nuclear and organelle model

- Module 3B Generate reports on an organelle model with slml2info (input: model)

- Module 3C Compare organelle models with slml2report (input: two or more existing models)

- Module 3D Generates a synthetic cell image from a model

  This notebook creates a folder with the same name as the model that will contain a folder name *img* containing a folder named *cell1*. This will contain files *cell1.tif*, *nucleus.tif* and which will contain 2D images corresponding to consecutive slices of the 3D cell.

## Tutorial Module 4. Working with spharm-obj models

Tutorial 4 introduces *spharm-obj* models, which provide a highly-detailed representation of the size, shape and subcellular distribution of organelles and other subcellular structures. They use the same SPHARM-RPDM capability that is used for modeling cell and nuclear shape. *spharm-obj* models are described in a recent *Bioinformatics paper* which uses them to compare real with synthetic cell images created using deep learning. Note that CellOrganizer does not yet (as of version 2.10) support synthesizing images from this model type – coming soon!

- Module 4A creates a *spharm-obj* organelle model from just one image (to minimize compute time). More generalizable models can be created by modifying the notebook so that is uses a larger collection of images.

- Module 4B uses *slml2report* to compare *spharm-obj* models of two different organelles. The comparison includes comparing the organelle shapes and their subcellular distributions. The models used are from the *Bioinformatics paper*.

## Tutorial Module 5. Working with your own images

As discussed in Tutorial 1, CellOrganizer needs segmented single cells in order to build models. Tutorial 1 shows various ways to do this for the case where each image contains a single cell. An alternative is to allow images to contain more than one cell but to provide a set of "Regions-Of-Interest" (ROIs) defining which pixels/voxels of the image belong to each cell.

- Module 5 shows how to create an OME-TIFF image file with ROIs out of an original image and a segmentation "mask" image. The mask image can be created with any of a number of cell segmentation packages, such as CellPose and DeepCell. The mask is an "indexed image" that is the same dimensions as the original image for which the value of each pixel/voxel is the number of the cell to which that that pixel belongs. The Module uses example images from the provided images folder.

## Tutorial Module 6. Working with cell parameterizations

Tutorial 6 shows how to read the contents of a CellOrganizer Matlab binary file (MAT-file) into a python dictionary. The contents include documentation of some of the model creation settings and, most importantly, the parameters describing the nuclear, cell and/or organelle models.

- Module 6A extracts information from a cell shape model.
  The notebook reads a .mat file into a python dictionary and illustrates the keys used to store various parts of the model. It then shows how to reconstruct a full SPHARM descriptor from the lower-dimensional shape embedding. Lastly, it creates a histogram of the cell sizes.

- Module 6B extracts information from an organelle model.
  This notebook is similar to Module 6A except that the .mat file read contains a protein (organelle) model instead of a cell shape model and some additional analyses are include. After reconstructing a full SPHARM descriptor, it displays parts of the organelle subcellular position model. It then shows how to use the parameterizations of shape and position to determine whether organelle shape is correlated with organelle position. It finishes by showing how to retrieve the original image of each of the organelles and makes a histogram of the Hausdorff distance between each organelle and its shape representation (a stringent measure of how good the model is).

## Tutorial Module 7. Simulating cell biochemistry with different geometries

Module 7 is an **advanced** module that illustrates using synthetic cell geometries produced from CellOrganizer models in combination with a biochemical model to simulate cell biochemistry in in varying realistic geometries.

- Module 7A takes as input a CellOrganizer 3D model and a Virtual Cell biochemistry specification and produces five Virtual Cell markup files that combine the biochemistry with a synthetic cell geometry generated from the CellOrganizer model. These files can be imported into Virtual Cell to do the simulations. See
  W Running Virtual Cell simulations using CellOrganizer generated files.docx for instructions.
  To adapt the notebook to your work, you can:
  - Change the CellOrganizer geometry model by setting `model_name`.
  - Change the number of geometries and simulations generated by setting `numberOfSynthesizedImages`.
  - Change the reaction network by setting `vcml_name` to the name of your VCML file and setting `vcml_gist_url = None`.
  - Change the simulation time, accuracy, and related parameters by setting `default_time_step`, `end_time`, etc.

- Module 7B takes as input a CellOrganizer 3D model and produces files containing a synthetic cell geometry that can be imported into the visualization program Blender (https://www.blender.org).

- Module 7C takes as input a CellOrganizer 3D model and produces a synthetic cell geometry file in SBML spatial format (see https://doi.org/10.1515/jib-2022-0054 for a description of the standard). These files are readable by a number of simulation tools.

## Tutorial Module 8. Building object shape representations directly

Module 8 is an **advanced** module that illustrates how to directly generate a shape descriptor (using spherical harmonics) for a shape that you provide, without having to construct a CellOrganizer model using *img2slml*.

- Module 8A converts a 3D image into a spherical harmonic representation using CellOrganizer SPHARM-RPDM functionality. The module has three parts corresponding to the 3 main functions:
  • Image2SPHARMparameterization takes an image and converts it into a structure that contains the spherical harmonic parameters along with additional metadata information.
  • SPHARMparameterization2Mesh takes a generated structure of spherical harmonic parameters and converts it into a mesh structure that contains vertices and faces.
  • SPHARMparameterization2Image takes a generated structure of spherical harmonic parameters and converts it into a 3D array or image.

- Module 8B creates a "movie" of continuous shape changes by interpolating between two shapes using principal component reduction of the full shape descriptors. Similar to Module 6, it illustrates how to extract shape descriptors from a CellOrganizer model and manipulate them to show the evolution of one shape into another.

### Some closing technical information

- When running these notebooks, the output from python appears in the notebook cell, while the output from the compiled CellOrganizer Matlab executables appears in the terminal window that was used to start the Docker container. Matlab errors listed in the terminal window will include a trace of the Matlab module and line numbers that generated the error. For those interested in debugging a problem, the line numbers can be referenced against the Matlab code, which can be found at

**https://github.com/murphygroup/cellorganizer**

- Inside the container, the CellOrganizer python library can be found at

**/opt/conda/lib/python3.7/site-packages/cellorganizer**

- As mentioned above, questions, help requests and/or new feature suggestions can be sent to
cellorganizer@compbio.cmu.edu

- All files created by the modules in this tutorial can be found at

https://cmu.box.com/s/1kzsye9u7m8o8grvj7b1t0vn9xitz87e