CC3DSimUtils Documentation

Release 1.0

Margriet Palm

June 05, 2013

CONTENTS

Contents:

CHAPTER

ONE

PREREQUISITES

MIMBSimUtils depends on a number of python packages which must be installed.

1.1 Numpy

Numpy is a package for numerical computation in python and it provides a very powerful array object. Download numpy and install it according to the instructions. Alternatively, numpy can be installed using pip:

```
pip install numpy
```

1.2 Scipy

Scipy is a python package of scientific software, which heavily depends on numpy arrays. Download scipy and follow the installation instructions. Alternatively, scipy can be installed using pip:

```
pip install scipy
```

1.3 Python imaging library

The python imaging library (PIL) is an image processing package for python. Download PIL and install according to the instruction included in the package:

```
pip install pil
```

1.4 Mahotas and Pymorph

Mahotas and Pymorph are two packages we use for image analysis. These packages can be downloaded from here and here. Alternatively they can be installed using pip:

```
pip install mahotas
pip install pymorph
```

CHAPTER

TWO

ANALYSISUTILS

2.1 Compactness

`AnalysisUtils.`**`getCompactness`**(*sigma*, *minval=0*)

Calculate compactness of a morphology: $\frac{A_{\text{area}}}{A_{\text{convex hull}}}$.

Parameters

- **sigma** – numpy array with cell id's
- **minval** (*int*) – minimum cell id for non medium pixels

Returns compactness

`AnalysisUtils.`**`getLCC`**(*sigma*)

Find largest connected component of an image

Parameters **sigma** – numpy array with cell id's

Returns image with largest connected component

2.2 Order parameter

The order parameter describes the orientational order of a liquid crystal: $S(r) = \left\langle \cos(2\theta(\vec{X}(\sigma), r)) \right\rangle_{\sigma}$ where $\theta(\vec{X}(\sigma), r)$ denotes the angle between the cell with center of mass X and the director in a circle of radius r around X.

`AnalysisUtils.`**`getDirector`**(*com*, *r*, *sigma*, *angles*)

Find the director of the center of mass of a cell.

Parameters

- **com** – center of mass of the cell (x,y)
- **r** (*number*) – radius of neighborhood
- **sigma** – numpy array with cell id's
- **angles** – numpy array with cell angles (radians)

Returns director (radians)

`AnalysisUtils.`**`getOrderParameter`**(*sigma*, *angles*, *r*)

Calculate order parameter for a morphology using the cpm grid data. When the requested radius is larger than the maximum radius of the grid, the global order parameter is calculated with `getGlobalOrderParameter()`; otherwise the local order parameter is calculated with `getLocalOrderParameter()`.

Parameters

- **sigma** – numpy array with cell id's
- **angles** – numpy array with cell angles (radians)
- **r** (*int*) – radius of neighborhood

See Also:

`getLocalOrderParameter()`, `getGlobalOrderParameter()`

`AnalysisUtils.`**`getLocalOrderParameter`**`(`*sigma*, *angles*, *r*`)`
Calculate local order parameter.

Parameters

- **sigma** – numpy array with cell id's
- **angles** – numpy array with cell angles (radians)
- **r** (*int*) – radius of neighborhood

Returns local order parameter

See Also:

`getDirector()`

`AnalysisUtils.`**`getGlobalOrderParameter`**`(`*sigma*, *angles*`)`
Calculate global order parameter.

Parameters

- **sigma** – numpy array with cell id's
- **angles** – numpy array with cell angles (radians)

Returns global order parameter

`AnalysisUtils.`**`getRelativeDirField`**`(`*sigma*, *r*`)`
Calculate field with relative director for each pixel. The relative director is the difference to the angle of the cell at that pixel and the relative director on the pixel. Pixels with high values represent unordered areas, such as branchpoints.

Parameters

- **sigma** – numpy array with cell id's
- **r** (*int*) – radius of neighborhood

Returns field with relative director values

See Also:

`getDirector()`, `getAngleField()`

2.3 Cell clusters

Clusters of aligned cells are automatically detected using the relative director field, with the following steps:

1. Remove all pixels that have a value in the relative director field higher than a given threshold.

2. **Detect blobs in remaining image with a labeling algorith:**
   (a) an opening operation may be performed before labeling.
   (b) areas smaller than a given size are ignored;
3. **Map each blob on the CPM grid:**
   (a) at least a given fraction of the cell must be on the labeled area.
4. **Check for cells in multiple clusters:**
   (a) remove cell from all but biggest cluster;
   (b) remove cluster if it is empty after (a).

`AnalysisUtils.`**`getCellClusters`**(*field*, *sigma*, *th=15*, *minlabsize=50*, *opendisk=1*, *mincellsize=0.25*)

Get clusters for a single morphology.

Parameters

- **field** – numpy array with values on which data is seperated
- **cells** – dict with cell identifiers as keys and `ClusterCell` instances as values
- **sigma** – CPM grid
- **th** (*number*) – threshold value for step 1
- **minlabsize** (*int*) – labelled areas smaller than this value are ignored (2b)
- **opendisk** (*int*) – disk size for opening operation (2a)
- **mincellsize** (*number*) – minimal fraction of the cell that must be on the labelled area to be added to the cluster

Returns dictionary with cluster id as key and `Cluster` instances

See Also:

`Cluster`

class `AnalysisUtils.`**`Cluster`**(*id*)

Container for a cell cluster

Parameters **id** – cluster id

Variables **cells** – list of ids of the cells in the clusters

`addCell`(*cellid*)

Add cell to cluster

Parameters **cellid** – id of cell

`getClusterSize`()

Calculate number of cell in cluster

Returns number of cells in cluster

`removeCell`(*cellid*)

Remove cell from cluster

Parameters **cellid** – id of cell

class `AnalysisUtils.`**`ClusterCellTC`**(*id*)

A class that holds properties related to a cell at each measured time step. These properties are:

•cluster id and size at each time step

- long axis at each time step
- center of mass at each time step

Parameters **id** – cell id

Variables

- **id** – cell id
- **clusterId** – list of cluster id's
- **clusterSize** – list of cluster sizes
- **time** – list of time steps
- **laxis** – 2D array with long axes of the cell
- **com** – 2D array with centers of mass of the cell

`addTimeStep`(*t*, *pix*, *cid*, *csz*)

Add time step

Parameters

- **t** (*int*) – time step
- **pix** – cell coordinates ([x1,...,xn],[y1,...,yn])
- **cid** (*int*) – cluster id
- **csz** (*int*) – cluster size

2.4 Cell angles

The angle of a cell is calculated from the inertia tensor of a cell. From the intertia tensor we calucate the eigenvalues and eigenvectors; the eigenvector that corresponds with the largest eigenvalue represents the direction of the long axis of a cell. The angle between the long axis and the x-axis is the cell angle.

`AnalysisUtils.getCellInertiaTensor`(*pix*)

Get inertia tensor for a cell

Parameters **pix** – cell coordinates ([x1,...,xn],[y1,...,yn])

Returns inertia tensor [[Ixx,Ixy],[Ixy,Iyy]]

`AnalysisUtils.getCellOrientation`(*pix*)

Calculate orientation of a cell. The orientation is the eigenvector corresponding to the largest eigenvalue of the cells' inertia tensor.

Parameters **pix** – cell coordinates ([x1,...,xn],[y1,...,yn])

Returns unit vector of the cell orientation

See Also:

`getCellInertiaTensor()`

`AnalysisUtils.getCellAngle`(*pix*)

Calculate angle of a cell on interval $[0, \pi]$

Parameters **pix** – cell coordinates ([x1,...,xn],[y1,...,yn])

Returns angle in radians on interval $[0, \pi]$

See Also:

`getCellOrientation()`

`AnalysisUtils.`**`getAngleField`**`(`*sigma*`)`

Get field with the cell angles

Parameters **sigma** – numpy array with cell id's

Returns numpy array with cell angles in radians

2.5 Mean squared displacement

The mean squared displacement describes the displacement of a cell over time with respect to the initial position : $MSD = \left\langle (x(t) - x(0))^2 \right\rangle$. In a similar manner the mean squared angular displacement can be calculated : $MSD = \left\langle (\theta(t) - \theta(0))^2 \right\rangle$

`AnalysisUtils.`**`calcMSDTransForCellTC`**`(`*com*`)`

Calculate the translational MSD for a single object.

Parameters **com** – list of centers of mass at each time step

Returns list with MSD for each time step

`AnalysisUtils.`**`calcMSDRotForCellTC`**`(`*vecset*`)`

Calculate the rotational MSD for a single object.

Parameters **vecset** – list of orientation vectors for each time step

Returns list with MSD for each time step

CHAPTER

THREE

IMAGEUTILS

`ImageUtils.`**`makeImage`**`(`*id*, *inpath*, *t*, *colormap*, *timestamp=False*, *label=False*, *scale=1*, *bc=None*, *fontsize=6*, *border=True*, *gzipped=True*, *fieldname=None*, *fontpath='/usr/share/fonts/msttcore/'*`)`

Draw morphology for one timestep

Parameters

- **id** (*str*) – simulation identifier
- **inpath** (*str*) – path containing data files
- **t** (*int*) – time step
- **outpath** (*str*) – path to save images to
- **colormap** – dictionary with cell types as keys and colors (r,g,b) as values
- **timestamp** (*bool*) – add time stamp to the image
- **label** (*bool*) – add id as label to the image
- **scale** (*number*) – scaling of the image
- **bc** – color of cell boundaries (r,g,b)
- **fontsize** (*int*) – size of the fonts used for label and time stamp; font size will be multiplied by scale.
- **border** (*bool*) – cut of border pixels
- **gzipped** (*bool*) – data is gzipped
- **fieldname** (*str*) – name of chemical field
- **fontpath** (*str*) – path to freetype fonts

Returns image object

See Also:

`drawCells()`, `addTimeStamp()`, `addLabel()`

`ImageUtils.`**`drawRelDirField`**`(`*field*, *sigma*, *scale=1*`)`

Draw gray-scale image of a field representing the relative director

Parameters

- **field** – numpy array with relative director
- **sigma** – numpy array with cell id's
- **scale** (*number*) – scaling factor

Returns image object

`ImageUtils.`**`stackImages`**(*images*, *geometry*, *filename*, *label=False*, *title=None*, *fontsize=20*, *border=False*, *scale=1*, *fontpath='/usr/share/fonts/msttcore/'*)

Stack a set of images together in one image.

Parameters

- **images** – dictionary with labels as keys and image filenames as values
- **geometry** – number of rows and columns (x,y)
- **filename** (*str*) – target of the stacked image
- **label** (*bool*) – add labels to the subimages
- **title** (*str*) – overall title for image
- **fontsize** (*int*) – font size (only for freetype fonts)
- **border** (*bool*) – add border to subimages
- **scale** (*number*) – scaling factor of the created picture
- **fontpath** (*str*) – path to freetype fonts

`ImageUtils.`**`morphImages`**(*images*, *filename*, *xlabel=None*, *ylabel=None*, *xtics=None*, *ytics=None*, *fontsize=20*, *scale=1*, *border=False*, *title=None*, *bcolor=(255, 255, 255)*, *fcolor=(0, 0, 0)*, *fontpath='/usr/share/fonts/msttcore/'*, *delta=0*)

Stack a set of images together in one morphospace.

Parameters

- **images** – 2D array with image filenames
- **filename** (*str*) – target of the stacked image
- **xlabel** (*str*) – label to be plotted on x-axis
- **ylabel** – label to be plotted on y-axis
- **ylabel** – str
- **xtics** – list of labels on x-axis
- **ytics** – list of labels on y-axis
- **fontsize** (*int*) – fontsize for labels and title
- **scale** (*number*) – scaling factor of the created picture
- **border** (*bool*) – add border to subimages
- **title** (*str*) – overall title for image
- **bcolor** – background color (r,g,b)
- **fcolor** – font color (r,g,b)
- **fontpath** (*str*) – path to freetype fonts
- **delta** (*number*) – extra space between images

CHAPTER

FOUR

READERS

Collection of functions to read simulation data and other files needed by MIMBSimUtils.

`Readers.`**`readChemField`**(*simid*, *t*, *indir*, *fieldname*, *gzipped=True*, *border=True*)

Read chemical field from file.

Parameters

- **simid** (*str*) – simulation identifier
- **t** (*int*) – time step
- **indir** (*str*) – path to data files
- **fieldname** (*str*) – name of the chemical field
- **gzipped** (*bool*) – data is gzipped (gzipped data is expected to be in indir/simid/)
- **border** (*bool*) – cut of border pixels

Returns numpy array with the levels of the chemical field at each position

`Readers.`**`readColorMap`**(*filename*)

Read colormap from a file, formatted like: celltype r g b

Parameters filename (*str*) – file with the colormap

Returns dictionary with cell type as keys and colors (r,g,b) as values.

`Readers.`**`readSigma`**(*simid*, *t*, *indir*, *gzipped=True*, *border=True*)

Read cell field (sigma) from file.

Parameters

- **simid** (*str*) – simulation identifier
- **t** (*int*) – time step
- **indir** (*str*) – path to data files
- **gzipped** (*bool*) – data is gzipped (gzipped data is expected to be in indir/simid/)
- **border** (*bool*) – cut of border pixels

Returns numpy array with cell id's

`Readers.`**`readTau`**(*simid*, *t*, *indir*, *gzipped=True*, *border=True*)

Read type field (tau) from file.

Parameters

- **simid** (*str*) – simulation identifier

- **t** (*int*) – time step
- **indir** (*str*) – path to data files
- **gzipped** (*bool*) – data is gzipped (gzipped data is expected to be in indir/simid/)
- **border** (*bool*) – cut of border pixels

Returns numpy array with cell types

CHAPTER

FIVE

CC3DPIPELINE

5.1 Pre-Processing

`CC3DPipeline.`**`createPBSScripts`**(*runid*, *joblist*, *command*, *time*, *ncores=8*, *ppn=8*, *path='clusterScripts/'*)

Create a set of PBS scripts to run a simulation on a cluster. Each script starts with something like:

#PBS -S /bin/bash #PBS -lnodes=1:cores12:ppn=11 #PBS -lwalltime=12:00:00

If these commands are not correct or complete for the cluster you use, edit `createPBS()`.

For each job in joblist a single line command is added to the script:

python command jobid > log/jobid.out 2> log/jobid.err &

Parameters

- **runid** (*str*) – identifier for the scripts
- **joblist** – list of job identifiers
- **command** (*str*) – command that runs the simulation
- **time** (*str*) – requested walltime on the cluster (hh:mm:ss)
- **ncores** (*int*) – numbor of cores in the requested node
- **ppn** (*int*) – number of processers per node that will be used
- **path** (*str*) – location where pbs scripts are saved

See Also:

`createPBS()`, `addCommandToPBS()`, `finishPBS()`

`CC3DPipeline.`**`createPBS`**(*filename*, *time*, *ncores=None*, *ppn=None*)

Create a new pbs script and add initial commands and settings.

Parameters

- **filename** (*str*) – filename of the new pbs script
- **time** (*str*) – requested walltime on the cluster (hh:mm:ss)
- **ncores** (*int*) – numbor of cores in the requested node
- **ppn** (*int*) – number of processers per node that will be used

`CC3DPipeline.`**`addCommandToPBS`**(*filename*, *command*, *log*)

Add single line command to existing PBS script:

Parameters

- **filename** (*str*) – filename of the new pbs script
- **command** (*str*) – command that runs the simulation
- **log** (*str*) – name (with path) of the log files (without extension)

`CC3DPipeline.`**`finishPBS`**(*filename*)

Finish pbs file

Parameters **filename** (*str*) – filename of the new pbs script

5.2 Post-Processing

`CC3DPipeline.`**`makeImages`**(*id*, *trange*, *inpath*, *outpath*, *cm='default.ctb'*, *gzipped=False*, *timestamp=False*, *label=False*, *scale=1*, *bc=None*, *fontsize=6*, *fieldname=None*, *border=True*)

Make images for a single simulation simulation

Parameters

- **id** (*str*) – simulation identifier
- **trange** – list of time steps for which images are created
- **inpath** (*str*) – path to data
- **outpath** (*str*) – path to save images to
- **cm** (*str*) – file containing the colormap
- **gzipped** (*bool*) – data is gzipped
- **timestamp** (*bool*) – add time stamp to the image
- **label** (*bool*) – add id as label to the image
- **scale** (*number*) – scaling of the image
- **bc** – color of cell boundaries (r,g,b)
- **fontsize** (*int*) – size of the fonts used for label and time stamp; font size will be multiplied by scale.
- **fieldname** (*str*) – name of chemical field
- **border** (*bool*) – cut of border pixels

See Also:

`makeImage()`

`CC3DPipeline.`**`getCompactnessForSim`**(*id*, *trange*, *inpath*, *gzipped=False*, *border=True*, *outpath=None*)

Calculate compactness for one simulation, the compactness is in a file: outpath/id_compactness.data

Parameters

- **id** (*str*) – simulation identifier
- **trange** – list of time steps for which the compactness is calculated
- **inpath** (*str*) – path to data

- **gzipped** (*bool*) – if True, data is expected to be gzipped, and stored in inpath/id/, and the output file will be gzipped and stored in outpath/id/
- **border** (*bool*) – remove border pixels from data
- **outpath** (*str*) – path where order parameter data will be saved, if omitted outpath = inpath

See Also:

`getCompactness()`

`CC3DPipeline.`**`getOrderParameterForSim`**`(`*id*, *trange*, *inpath*, *radii*, *gzipped=False*, *border=True*, *outpath=None*`)`

Calculate orderparameters for one simulation. All order parameters are collected and saved in a file outpath/id_orderparameter.data

Parameters

- **id** (*str*) – simulation identifier
- **trange** – list of time steps for which the order parameter is calculated
- **inpath** (*str*) – path to data
- **radii** – list of radii for wich the order parameter is calculates
- **gzipped** (*bool*) – if True, data is expected to be gzipped, and stored in inpath/id/, and the output file will be gzipped and stored in outpath/id/
- **border** (*bool*) – remove border pixels from data
- **outpath** (*str*) – path where order parameter data will be saved, if omitted outpath = inpath

See Also:

`getOrderParameter()`

`CC3DPipeline.`**`getClustersForSim`**`(`*id*, *trange*, *inpath*, *r*, *th*, *minlabsize*, *opendisk*, *mincellsize*, *gzipped=False*, *border=False*, *outpath=None*`)`

Calculate clusters and mean squared displacement and rotation for each cell in a simulation. For more details on clustering see the documentation of `getCellClusters()`.

Parameters

- **id** (*str*) – simulation identifier
- **trange** – list of time steps for which the clusters are calculated
- **inpath** (*str*) – path to data
- **r** (*number*) – radius for relative director field
- **th** (*number*) – threshold value for step 1
- **minlabsize** – labelled areas smaller than this value are ignored (2b)
- **opendisk** (*int*) – disk size for opening operation (2a)
- **mincellsize** (*int*) – minimal fraction of the cell that must be on the labelled area to be added to the cluster
- **gzipped** (*bool*) – if True, data is expected to be gzipped, and stored in inpath/id/, and the output file will be gzipped and stored in outpath/id/
- **border** (*bool*) – remove border pixels from data
- **outpath** (*str*) – path where order parameter data will be saved, if omitted outpath = inpath

See Also:

`getCellClusters()`, `getRelativeDirField()`, `calcMSDTransForCellTC()`, `calcMSDRotForCellTC()`, `ClusterCellTC`

CHAPTER

SIX

EXPERIMENT

class `Experiment.`**`Experiment`**(*templatefile*)

The class Experiments holds the data for a CC3D experiment as an xml object. This objects is created from a template xml file. Elements that are already present in the template can be changed or removed and new elements can be added. When the xml is written to file all unchanged content of the template, including commented xml, is written to the new file.

`addStatistics`(*freq*, *basename*)

Edit statistics save options

Parameters

- **freq** – save frequency
- **basename** – basename for files (full path)

`deletePlugin`(*name*)

Delete plugin

Parameters **name** – plugin name

`deletePottsProperty`(*tagname*)

Remove element from the Potts element

Parameters **tagname** – name of element

`deleteSteppable`(*type*)

Delete steppable

Parameters **type** – steppable type

`setChemotaxis`(*celltype*, *towards*, *lam*)

Set chemotaxis for cell type

Parameters

- **celltype** – name of the cell type
- **towards** – cell type towards chemotaxis occurs
- **lam** – chemotactic strength

`setContact`(*_type1*, *_type2*, *J*)

Edit contact energy

Parameters

- **type1** – name of the first cell type
- **type2** – name of the second cell type

- **J** – contact energy between type1 and type2

setDebugFrequencyInMeta (*freq*)
Set debug frequency

Parameters **frequency** – frequency

setGenericPlugin (*name*, *elements=*[ ])
Set generic plugin. If the plugin is already in the template, it is updated. If not, the plugin is added to the model. This function does not support multi-level xml elements.

Parameters

- **name** – plugin name
- **elements** – list of elements described by a dictionary: {'name':name,'value':val,'attributes':{}}

setGenericSteppable (*type*, *freq=None*, *elements=*[ ])
Set generic steppable. If the steppable is already in the template, it is updated. If not, the plugin is added to the model. This function does not support multi-level xml elements.

Parameters

- **type** – steppable type
- **freq** – frequency
- **elements** – list of elements described by a dictionary with keys name, value and attributes

setMCS (*mcs*)
Set number of Monte Carlo steps

Parameters **mcs** – number of MCS (note that mcs+1 appears in the xml)

setMotility (*celltype*, *T*)
Edit motility parameters per cell type

Parameters

- **celltype** – name of the cell type
- **T** – motility

setMultiCore (*threads=1*, *cores=1*)
Set number of threads and cores for a simulation

Parameters

- **threads** – number of threads per core
- **cores** – number of cores

setPottsProperty (*tagname*, *attributes={}*, *value=None*)
General function to set a parameter in the Potts element

Parameters

- **tagname** – name of the element
- **attributes** – dictionary with attribute names as key and attribute values as values.
- **value** – element value

setSecretion (*celltype*, *s*, *solver='FastDiffusionSolver2DFE'*)
Set secretion coefficient for specific cell type

Parameters

- **celltype** – name of the cell type
- **s** – secretion coefficient
- **solver** – solver name

`setSeed`(*seed*)
Set simulation seed

Parameters **seed** – random seed

`setTemp`(*T*)
Set temperature tag

Parameters **T** – temperature

`setVolume`(*celltype*, *vol*, *lam*)
Set volume per cell type

Parameters

- **celltype** – cell type name
- **vol** – target volume
- **lam** – lambda volume

`write`(*filename*)
Save xml to file

Parameters **filename** – filename of new xml

CHAPTER

SEVEN

INDICES AND TABLES

- *genindex*
- *search*

PYTHON MODULE INDEX

INDEX