pcnaDeep

Release 1.0

Chan Kuan Yoow Group

Sep 21, 2021

CONTENTS

CHAPTER

ONE

API DOCUMENTATION

1.1 pcnaDeep

1.1.1 pcnaDeep.predictor

class pcnaDeep.predictor.**VisualizationDemo**(*cfg*, *instance_mode=ColorMode.IMAGE*, *parallel=False*)

Bases: object

run_on_image(*image*, *vis=True*)

Adapted from Facebook Detectron2 Demo. Apache 2.0 Licence.

Parameters **image** (*numpy.ndarray*) – an image of shape (H, W, C) (in BGR order). This is the format used by OpenCV.

Returns

the output of the model.

vis_output (VisImage): the visualized image output.

Return type predictions (dict)

class pcnaDeep.predictor.**AsyncPredictor**(*cfg*, *num_gpus: int = 1*)

Bases: object

Copied from Facebook Detectron2 Demo. Apache 2.0 Licence.

A predictor that runs the model asynchronously, possibly on >1 GPUs. Because rendering the visualization takes considerably amount of time, this helps improve throughput a little bit when rendering videos.

put(*image*)

get()

shutdown()

property default_buffer_size

pcnaDeep.predictor.**pred2json**(*mask*, *label_table*, *fp*)

Transform detectron2 prediction to VIA2 (VGG Image Annotator) json format.

Parameters

- **mask** (*numpy.ndarray*) – binary mask with all instance labeled with unique label.
- **label_table** (*pandas.DataFrame*) – metadata of the mask; must contain *continuous_label*, *predicted_class* and *emerging* columns.
- **fp** (*str*) – file name for this frame.

Returns json format readable by VIA2 annotator.

Return type dict

pcnaDeep.predictor.**predictFrame**(*img*, *frame_id*, *demonstrator*, *is_gray=False*, *size_flt=1000*, *edge_flt=50*)

Predict single frame and deduce meta information.

Parameters

- **img** (*numpy.ndarray*) – must be *uint8* image slice.
- **frame_id** (*int*) – index of the slice, start from 0.
- **demonstrator** (VisualizationDemo) – an detectron2 demonstrator object.
- **size_flt** (*int*) – size filter, in pixel^2.
- **is_gray** (*bool*) – whether the slice is gray. If true, will convert to 3 channels at first.
- **edge_flt** (*int*) – filter objects at the edge, whose classification may be imprecise, in pixel.

Returns labeled mask and corresponding table.

Return type tuple

1.1.2 pcnaDeep.refiner

pcnaDeep.refiner.**dist**(*x1*, *y1*, *x2*, *y2*)

Calculate distance of a set of coordinates

class pcnaDeep.refiner.**Refiner**(*track*, *smooth=5*, *maxBG=5*, *minM=10*, *mode='SVM'*, *threshold_mt_F=100*, *threshold_mt_T=25*, *search_range=10*, *sample_freq=0.2*, *model_train=''*, *mask=None*, *dilate_factor=0.5*, *aso_trh=0.5*, *dist_weight=0.8*, *svm_c=0.5*, *dt_id=None*, *test_id=None*)

Bases: object

break_mitosis()

Break mitosis tracks; iterate until no track is broken.

register_track()

Register track annotation table

render_emerging(*track*, *cov_range*)

Render emerging phase

revert(*ann*, *mt_dic*, *parentId*, *daughterId*)

Remove information of a relationship registered to ann and mt_dic

register_mitosis(*ann*, *mt_dic*, *parentId*, *daughterId*, *m_exit*, *dist_dif*, *m_entry=0*)

Register parent and dduahgter information to ann and mt_dic

getMtransition(*trackId*, *direction='entry'*, *skip=0*)

Get mitosis transition time by trackId

Parameters

- **trackId** (*int*) – track ID
- **direction** (*str*) – either 'entry'or 'exit', mitosis entry or exit
- **skip** (*int*) – escape frames from *deduce_transition* method

extract_pools(*extra_par=None*, *extra_daug=None*)

Extract potential parent and daughter pool

get_parent_mask(*p*)

Extract parent mask, begin from mitosis entry, end with parent disappearance

Parameters **p** (*int*) – parent track ID

daug_app_in_par_mask(*par*, *daug*)
Check if daughter appears in the mask of parent

Parameters

- **par** (*int*) – parent track ID
- **daug** (*int*) – daughter track ID

extract_features(*par_pool*, *daug_pool*, *remove_outlier=None*, *normalize=None*, *sample=None*)
Extract Input Features for the classifier

Parameters

- **par_pool** (*list*) – Parent pool.
- **daug_pool** (*list*) – Daughter pool.
- **remove_outlier** (*list[int]*) – Remove outlier of columns in the feature map.
- **normalize** (*bool*) – Normalize each column.
- **sample** (*numpy.ndarray*) – Training mode only, supply positive sample information, will add y as 2nd output.

plainPredict(*ipts*)
Generate cost of each potential daughter-parent pair (sample).

extract_train_from_break(*sample_id*, *ipts*, *mt_dic*)
Extract broken mitosis information to train model.

associate(*mode=None*)
Main algorithm to associate parent and daughter relationship.

update_table_with_mt()
Update tracked object table with information in self.mt_dic (mitosis lookup dict).

smooth_track()
Re-assign cell cycle classification based on smoothed confidence.

getMeanDisplace()
Calculate mean displace of each track normalized with frame.

getAsoInput(*parent*, *daughter*)
Generate SVM classifier input for track 1 & 2.

Parameters

- **parent** (*int*) – parent track ID.
- **daughter** (*int*) – daughter track ID.

Returns

- [distance_diff, frame_diff]

Some parameters are normalized with dataset specific features: - distance_diff /= ave_displace - frame_diff /= sample_freq

Return type Input vector of the classifier

get_SVM_train(*sample=None*)
Save training data for SVM classifier of this particular dataset.

Parameters **sample** (*numpy.ndarray*) – Optional matrix of shape (sample, (parent ID, daughter ID, ⋯)). If not supplied, will generate directly from mitosis-broken tracked object table. (GT with mitosis relationship)

Returns Input feature map. (numpy.ndarray): Ground truth label. (numpy.ndarray): Track ID of the corresponding feature (row).

Return type (numpy.ndarray)

`setSVMpath`(*model_train*)

`doTrackRefine`()

Perform track refinement process

Returns If run in TRH/SVM mode, will return annotation table, tracked object table and mitosis directory. If run in TRAIN mode, will only return tracked object table after smoothing, mitosis breaking. for manual inspection. After determining the training instance, generate training data through. get_SVM_train(sample).

1.1.3 pcnaDeep.resolver

`pcnaDeep.resolver.list_dist`(*a*, *b*)

Count difference between elements of two lists.

Parameters

- **a** (*list*) – classifications with method A
- **b** (*list*) – classifications with method B

`pcnaDeep.resolver.get_rsv_input_gt`(*track*, *gt_name='predicted_class'*, *G2_trh=200*, *no_cls_GT=False*)

Deduce essential input of resolver from a ground truth.

`pcnaDeep.resolver.resolve_from_gt`(*track*, *gt_name='predicted_class'*, *extra_gt=None*, *G2_trh=None*, *no_cls_GT=False*, *minG=1*, *minS=1*, *minM=1*, *minLineage=0*)

Resolve cell cycle phase from the ground truth. Wrapper of *get_rsv_input_gt()*.

Parameters

- **track** (*pandas.DataFrame*) – data frame of each object each row, must have following columns: - trackId, frame, parentTrackId, <ground truth classification column>
- **gt_name** (*str*) – refers to the column in track that corresponds to ground truth classification.
- **extra_gt** (*str*) – refers to the column in track that has G2 ground truth if *gt_name* does not. See notes below.
- **G2_trh** (*int*) – intensity threshold for classifying G2 phase (for arrest tracks only).
- **no_cls_GT** (*bool*) – Set to *true* if no classification ground truth is provided. Will resolve based on current classifications.
- **minG** (*int*) – minimum G phase frame length (default 1).
- **minS** (*int*) – minimum S phase frame length (default 1).
- **minM** (*int*) – minimum M phase frame length (default 1).
- **minLineage** (*int*) – minimum lineage frame length to resolve (default 0, resolve all tracks).

Note:

- If do not want G2 to be classified based on thresholding, rather, based on ground truth classification.

Simply leave *G2_trh=None* and the threshold will be calculated as the smallest average intensity of G2 phase in labeled tracks (outlier smaller than mena - 3*sd excluded).

- If the ground truth column does not contain *G2* instances, tell the program to look at

an extra partially G2 ground truth column like *resolved_class* to extract information. This may be useful when *predicted_class* has been corrected from the Correction Interface which only contains G1/G2 but not G2. In this case, you can also assign *resolved_class* as the ground truth classification column. Both will work.

- If *mean_intensity* or *background_mean* column is not in the table, will set the threshold to 100.
- Use at own risk if the input classification in not reliable.

`class pcnaDeep.resolver.Resolver(`*track*, *ann*, *mt_dic*, *maxBG=25*, *minS=20*, *minM=10*, *minLineage=10*, *impreciseExit=None*, *G2_trh=100*`)`

Bases: `object`

`doResolve()`
Main function of class resolver.

Returns tracked object table with additional column 'resolved_class'. pandas.DataFrame: phase table with cell cycle durations.

Return type pandas.DataFrame

`check_trans_integrity()`
Check track transition integrity. If transition other than G1->S; S->G2, G2->M, M->G1 found, do not resolve.

`getAnn()`
Add an annotation column to tracked object table The annotation format is track ID - (parentTrackId, optional) - resolved_class

`resolveArrest(`*G2_trh=None*`)`

Determine G1/G2 arrest tracks.

- **If *G2_trh* is supplied, determine G2 based on background-subtracted mean of the track** (averaged across frames).
- If *G2_trh* is not supplied, assign G1 or G2 classification according to 2-center K-mean.

Parameters `G2_trh` (*int*) – int between 1-255, above the threshold will be classified as G2.

`resolveLineage(`*lineage*, *main*`)`
Resolve all tracks in a lineage recursively main (int): the parent track ID of current search

`resolveTrack(`*trk*, *m_entry=None*, *m_exit=None*`)`
Resolve single track.

Parameters

- `trk` (*pandas.DataFrame*) – track table
- `m_entry` (*int*) – time of mitosis entry corresponding to 'frame'column in table
- `m_exit` (*int*) – time of mitosis exit corresponding to 'frame'column in table

If no m time supplied, only treat as G1/G2/S track. Arrested track not resolved, return full G1/G2 list.

Returns pandas.DataFrame table with addition column of resolved class

`doResolvePhase()`
Resolve phase durations

1.1.4 pcnaDeep.tracker

pcnaDeep.tracker.**track**(*df*, *displace=40*, *gap_fill=5*)

Track and relabel mask with trackID.

Parameters

- **df** (*pandas.DataFrame*) – Data frame with fields: - Center_of_the_object_0: x location of each object. - Center_of_the_object_1: y location of each object. - frame: time location. - BF_mean: mean intensity of bright field image. - BF_std: standard deviation of bright field image. - (other optional columns)
- **displace** (*int*) – maximum distance an object can move between frames.
- **gap_fill** (*int*) – temporal filling fo tracks.

Returns tracked object table.

Return type (pandas.DataFrame)

pcnaDeep.tracker.**track_mask**(*mask*, *displace=40*, *gap_fill=5*, *render_phase=False*, *size_min=100*, *PCNA_intensity=None*, *BF_intensity=None*)

Track binary mask objects.

Parameters

- **mask** (*numpy.ndarray*) – cell mask, can either be binary or labeled with cell cycle phases.
- **displace** (*int*) – distance restriction, see *track()*.
- **gap_fill** (*int*) – time restriction, see *track()*.
- **render_phase** (*bool*) – whether to deduce cell cycle phase from the labeled mask.
- **size_min** (*int*) – remove object smaller then some size, in case the mask labeling is not precise.
- **PCNA_intensity** (*numpy.ndarray*) – optional, if supplied, will extract fore/backgound PCNA intensity,
- **BF_intensity** (*numpy.ndarray*) – optional, if supplied, will extract bright field intensity & std for tracking. First three channels must have same length as the mask.

Returns tracked object table. (mask_lbd): mask with each frame labeled with object IDs.

Return type (pandas.DataFrame)

pcnaDeep.tracker.**track_GT_json**(*fp_json*, *height=1200*, *width=1200*, *displace=40*, *gap_fill=5*, *size_min=100*, *fp_intensity_image=None*, *fp_pcna=None*, *fp_bf=None*, *sat=None*, *gamma=None*)

Track ground truth VIA json file. Wrapper of *track_mask()*

Parameters

- **fp_json** (*str*) – file path to the json file.
- **height** (*int*) – pixel height of the mask corresponding to GT json.
- **width** (*int*) – pixel width of the mask corresponding to GT json.
- **displace** (*int*) – distance restriction, see *track()*.
- **gap_fill** (*int*) – time restriction, see *track()*.
- **size_min** (*int*) – remove object smaller then some size, in case the mask labeling is not precise.

- **fp_intensity_image** (*str*) – optional image file path, if supplied, will extract fore/backgound PCNA intensity, and bright field intensity/std for tracking. Must has the same shape as mask, so will override height and width.
- **fp_pcna** (*str*) – optional file path to PCNA channel image stack.
- **fp_bf** (*str*) – optional file path to bright field image stack.
- **sat** (*float*) – saturated pixel percentage when rescaling intensity image. If *None*, no rescaling will be done.
- **gamma** (*float*) – gamma-correction factor. If *None*, will not perform.

Returns tracked object table. (mask_lbd): mask with each frame labeled with object IDs.

Return type (pandas.DataFrame)

Note:

- If supplied with *fp_intensity_image* (composite image stack), will omit *fp_pcna* or *fp_bf*.
- *fp_pcna* and *fp_bf* must be supplied at the same time.

1.1.5 pcnaDeep.evaluate

class pcnaDeep.evaluate.**pcna_ctcEvaluator**(*root*, *dt_id*, *digit_num=3*, *t_base=0*, *path_ctc_software=None*, *init_dir=True*)

Bases: object

set_evSoft(*path_ctc_software*)
Set evaluation software path

Parameters **path_ctc_software** (*str*) – path to CTC evaluation software

generate_raw(*stack*)
Save raw images by slice

Parameters **stack** (*numpy.ndarray*) – raw image

generate_ctc(*mask*, *track*, *mode='RES'*)
Generate standard format for Cell Tracking Challenge Evaluation, for RES or GT.

Parameters

- **mask** (*numpy.ndarray*) – mask output, no need to have cell cycle labeled
- **track** (*pandas.DataFrame*) – tracked object table, can have gaped tracks
- **mode** (*str*) – either "RES"or "GT".

init_ctc_dir()
Initialize Cell Tracking Challenge directory

Directory example

>—–**0001**———- >—0001_RES—>—0001_GT—-

>—-SEG——>—-TRA——

evaluate()
Call CTC evaluation software to run ((Unix) Linux/Mac only)

1.1.6 pcnaDeep.split

pcnaDeep.split.**split_frame**(*frame*, *n=4*)
Split frame into several quadrants.

Parameters

- **frame** (*numpy.ndarray*) – single frame slice to split, shape HWC, if HW, will expand C.
- **n** (*int*) – split count, either 4 or 9.

Returns stack of split slice, order by row.

Return type numpy.ndarray

pcnaDeep.split.**join_frame**(*stack*, *n=4*, *crop_size=None*)
For each n frame in the stack, join into one complete frame (by row).

Parameters

- **stack** (*numpy.ndarray*) – tiles to join.
- **n** (*int*) – each n tiles to join, should be either 4 or 9.
- **crop_size** (*int*) – crop the square image into certain size (lower-right), default no crop.

Returns stack of joined frames.

Return type numpy.ndarray

pcnaDeep.split.**join_table**(*table*, *n=4*, *tile_width=1200*)
Join object table according to tiled frames.

Parameters

- **table** (*pandas.DataFrame*) – object table to join, essential columns: frame, Center_of_the_object_0 (x), Center_of_the_object_1 (y). The method will join frames by row.
- **n** (*int*) – each n frames form a tiled slice, either 4 or 9.
- **tile_width** (*int*) – width of each tile.

Returns object table for further processing (tracking, resolving)

Return type pandas.DataFrame

pcnaDeep.split.**relabel_seq**(*frame*, *base=1*)
Relabel single frame sequentially.

pcnaDeep.split.**register_label_to_table**(*frame*, *table*)
Register labels to the table according to centroid localization. WARNING: will round location to 2 decimal.

pcnaDeep.split.**resolve_joined_stack**(*stack*, *table*, *n=4*, *boundary_width=5*, *dilate_time=3*, *filter_edge_width=50*)
Wrapper of *resolved_joined_frame()* which resolves merged tiles by each frame. Filter imprecise objects at the edge.

pcnaDeep.split.**resolve_joined_frame**(*frame*, *table*, *n=4*, *boundary_width=5*, *dilate_time=3*, *filter_edge_width=50*)
Resolve joined frame and table of single slice.

Parameters

- **frame** (*numpy.ndarray*) – joined image slice.
- **table** (*pandas.DataFrame*) – object table with coordinate adjusted from joining.

- **n** (*int*) – tile count.
- **boundary_width** (*int*) – maximum pixel value for sealing objects at the boundary.
- **dilate_time** (*int*) – round of dilation on boundary objects to seal them.
- **filter_edge_width** (*int*) – filter objects at the edge.

Returns relabeled slice with objects at the edge joined. pandas.DataFrame: resolved object with object labels updated.

Return type numpy.ndarray

Note: Objects at the edge will be deleted, new objects due to joining will be registered.

Cell cycle phase information (prediction class and confidence) is drawn from the object of larger size.

1.1.7 pcnaDeep.correct

class pcnaDeep.correct.**Trk_obj**(*track_path*, *frame_base=1*)

Bases: object

create_or_replace(*old_id*, *frame*, *new_id=None*)

Create a new track ID or replace with some track ID after certain frame. If the old track has daughters, new track ID will be the parent.

Parameters

- **old_id** (*int*) – old track ID.
- **frame** (*int*) – frame to begin with new ID.
- **new_id** (*int*) – new track ID, only required when replacing track identity.

create_parent(*par*, *daug*)

Create parent-daughter relationship.

Parameters

- **par** (*int*) – parent track ID.
- **daug** (*int*) – daughter track ID.

del_parent(*daug*)

Remove parent-daughter relationship, for a daughter.

Parameters **daug** (*int*) – daughter track ID.

correct_cls(*trk_id*, *frame*, *cls*, *mode='to_next'*, *end_frame=None*)

Correct cell cycle classification, will also influence confidence score.

Parameters

- **trk_id** (*int*) – track ID to correct.
- **frame** (*int*) – frame to correct or begin with correction.
- **cls** (*str*) – new classification to assign.
- **mode** (*str*) – either 'to_next', 'single', or 'range'
- **end_frame** (*int*) – optional, in 'range'mode, stop correction at this frame.

delete_track(*trk_id*, *frame=None*)

Delete entire track. If frame supplied, only delete object at specified frame.

Parameters

- **trk_id** (*int*) – track ID.
- **frame** (*int*) – time frame.

save()
Save current table.

revert()
Revert to last saved version.

erase()
Erase all editing to the original file.

getAnn()
Add an annotation column to tracked object table The annotation format is track ID - (parentTrackId, optional) - resolved_class

edit_div(*par*, *daugs*, *new_frame*)
Change division time of parent and daughter to a new time location

Parameters

- **par** (*int*) – parent track ID
- **daugs** (*list*) – daughter tracks IDs
- **new_frame** (*int*) –

doCorrect()
Iteration for user command input.

1.1.8 Subpackages

1.1.8.1 pcnaDeep.data

pcnaDeep.data.annotate

pcnaDeep.data.annotate.**relabel_trackID**(*label_table*)
Relabel trackID in tracking table, starting from 1.

Parameters **label_table** (*pandas.DataFrame*) – tracked object table.

Returns tracked object table with relabeled trackID.

Return type pandas.DataFrame

pcnaDeep.data.annotate.**label_by_track**(*mask*, *label_table*)
Label objects in mask with track ID

Parameters

- **mask** (*numpy.ndarray*) – uint8 np array, output from main model.
- **label_table** (*pandas.DataFrame*) – track table.

Returns uint8/16 dtype based on track count.

Return type numpy.ndarray

pcnaDeep.data.annotate.**get_lineage_txt**(*label_table*)
Generate txt table in Cell Tracking Challenge (CTC) format.

Parameters **label_table** (*pandas.DataFrame*) – table processed, should not has gaped tracks.

Returns lineage table in .txt format that fits CTC.

Return type pandas.DataFrame

pcnaDeep.data.annotate.**break_track**(*label_table*)

Break tracks in a lineage table into single tracks, where NO gaped tracks allowed. All gaps will be transferred into parent-daughter relationship.

Parameters **label_table** (*pandas.DataFrame*) – tracked object table to process.

Algorithm:

1. Rename raw parentTrackId to mtParTrk.
2. Initiate new parentTrackId column with 0.
3. Separate all tracks individually.

Notes

In original lineage table, single track can be gaped, lineage only associates mitosis tracks, not gaped tracks.

Returns processed tracked object table.

Return type pandas.DataFrame

pcnaDeep.data.annotate.**separate**(*frame_list*, *mtPar_list*, *ori_id*, *base*)

For single gaped track, separate it into all complete tracks.

Parameters

- **frame_list** (*list*) – frames list, length equals to label table.
- **mtPar_list** (*list*) – mitosis parent list, for solving mitosis relationship.
- **ori_id** (*int*) – original track ID.
- **base** (*int*) – base track ID, will assign new track ID sequentially from base + 1.

Returns Dictionary of having following keys: frame, trackId, parentTrackId, mtParTrk.

Return type dict

pcnaDeep.data.annotate.**save_seq**(*stack*, *out_dir*, *prefix*, *dig_num=3*, *dtype='uint16'*, *base=0*, *img_format='.tif'*, *keep_chn=True*, *sep='-'*)

Save image stack and label sequentially.

Parameters

- **stack** (*numpy array*) – image stack in THW format (Time, Height, Width).
- **out_dir** (*str*) – output directory.
- **prefix** (*str*) – prefix of single slice, output will be prefix-000x.tif/png. (see sep below for separator).
- **dig_num** (*int*) – digit number (3 -> 00x) for labeling image sequentially.
- **dtype** (*numpy.dtype*) – data type to save, either 'uint8'or 'uint16'.
- **base** (*int*) – base number of the label (starting from).
- **img_format** (*str*) – image format, '.tif'or '.png', remind the dot.
- **keep_chn** (*bool*) – whether to keep full channel or not.
- **sep** (*str*) – separator between file name and id, default '-'.

pcnaDeep.data.annotate.**findM**(*gt_cls*, *direction='begin'*)

Find M exit/entry from ground truth classification.

The method assumes that all mitosis classification is continuous, therefore only suitable. for processing classification ground truth. For processing prediction, use *pcnaDeep.refiner.deduce_transition*.

Parameters

- **gt_cls** (*list*) – list of classifications.
- **direction** (*str*) – begin/end, search M from which terminal of the classification list.

Returns index of the mitosis entry/exit.

Return type int

pcnaDeep.data.annotate.**check_continuous_track**(*table*)
Check if every track is continuous (no gap). Returns trackID list that is gaped.

pcnaDeep.data.preparePCNA

pcnaDeep.data.preparePCNA.**load_PCNA_from_json**(*json_path*, *image_path*, *width=1200*, *height=1200*)
Load PCNA training data and ground truth from json.

Parameters

- **json_path** (*str*) – path to .json ground truth in VIA2 format.
- **image_path** (*str*) – path to raw image.
- **width** (*int*) – width of the image.
- **height** (*int*) – height of the image.

pcnaDeep.data.preparePCNA.**load_PCNAs_json**(*json_paths*, *image_paths*)
Load multiple training dataset.

pcnaDeep.data.utils

pcnaDeep.data.utils.**json2mask**(*ip*, *height*, *width*, *out=None*, *label_phase=False*, *mask_only=False*)
Draw mask according to VIA2 annotation and summarize information

Parameters

- **ip** (*str*) – input directory of the json file.
- **out** (*str*) – optonal, output directory of the image and summary table.
- **height** (*int*) – image height.
- **width** (*int*) – image width.
- **label_phase** (*bool*) – whether to label the mask with values corresponding to cell cycle classification or not. If true, will label as the following values: 'G1/G2' :10, 'S':50, 'M':100; If false, will output binary masks.
- **mask_only** (*bool*) – whether to suppress file output and return mask only.

Outputs: *png* files of object masks.

pcnaDeep.data.utils.**mask2json**(*in_dir*, *out_dir*, *phase_labeled=False*, *phase_dic={10: 'G1/G2', 50: 'S', 100: 'M', 200: 'E'}*, *prefix='object_info'*)
Generate VIA2-readable json file from masks

Parameters

- **in_dir** (*str*) – input directory of mask slices in .png format. Stack input is not implemented.
- **out_dir** (*str*) – output directory for .json output

- **phase_labeled** (*bool*) – whether cell cycle phase has already been labeled. If true, a phase_dic variable should be supplied to resolve phase information.
- **phase_dic** (*dic*) – lookup dictionary of cell cycle phase labeling on the mask.
- **prefix** (*str*) – prefix of .json output.

Outputs:

prefix.json in VIA2 format. Note the output is not a VIA2 project, so default image directory must be set for the first time of labeling.

pcnaDeep.data.utils.**getDetectInput**(*pcna*, *dic*, *gamma=1*, *sat=1*, *torch_gpu=False*)

Generate pcna-mScarlet and DIC channel to RGB format for detectron2 model prediction

Parameters

- **pcna** (*numpy.ndarray*) – uint16 PCNA-mScarlet image stack (T*H*W).
- **dic** (*numpy.ndarray*) – uint16 DIC or phase contrast image stack.
- **gamma** (*float*) – gamma adjustment, >0, default 0.8.
- **sat** (*float*) – percent saturation, 0~100, default 0.
- **torch_gpu** (*bool*) – use torch to speed up calculation.

Returns uint8 composite image (T*H*W*C)

Return type (numpy.ndarray)

pcnaDeep.data.utils.**retrieve**(*table*, *mask*, *image*, *rp_fields=[]*, *funcs=[]*)

Retrieve extra skimage.measure.regionprops fields of every object; Or apply customized functions to extract features form the masked object.

Parameters

- **table** (*pandas.DataFrame*) – object table tracked or untracked, should have 2 fields: 1. frame: time location; 2. continuous label: region label on mask
- **mask** (*numpy.ndarray*) – labeled mask corresponding to table
- **image** (*numpy.ndarray*) – intensity image, only the first channel allowed
- **rp_fields** (*list(str)*) – skimage.measure.regionpprops allowed fields
- **funcs** (*list(function)*) – customized function that outputs one value from an array input

Returns labeled object table with additional columns

pcnaDeep.data.utils.**mt_dic2mt_lookup**(*mt_dic*)

Convert mt_dic to mitosis lookup

Parameters **mt_dic** (*dict*) – standard mitosis info dictionary in pcnaDeep

Returns

mitosis lookup table with 3 columns: trackA (int) | trackB (int) | Mitosis? (int, 0/1)

Return type mt_lookup (pd.DataFrame)

pcnaDeep.data.utils.**get_outlier**(*array*, *col_ids=None*)

Get outlier index in an array, specify target column

Parameters

- **array** (*numpy.ndarray*) – original array
- **col_ids** (*[int]*) – target columns to remove outliers. Default all

Returns index of row containing at least one outlier

pcnaDeep.data.utils.**deduce_transition**(*l*, *tar*, *confidence*, *min_tar*, *max_res*, *escape=0*, *casual_end=True*)

Deduce mitosis exit and entry based on adaptive searching

Parameters

- **l** (*list*) – list of the target cell cycle phase
- **tar** (*str*) – target cell cycle phase
- **min_tar** (*int*) – minimum duration of an entire target phase
- **confidence** (*numpy.ndarray*) – matrix of confidence
- **max_res** (*int*) – maximum accumulative duration of unwanted phase
- **escape** (*int*) – do not consider the first n instances
- **casual_end** (*bool*) – at the end of the track, whether loosen criteria of a match

Returns two indices of the classification list corresponding to entry and exit

Return type tuple

pcnaDeep.data.utils.**find_daugs**(*track*, *track_id*)

Return list of daughters according to certain parent track ID.

Parameters

- **track** (*pandas.DataFrame*) – tracked object table.
- **track_id** (*int*) – track ID.

pcnaDeep.data.utils.**filter_edge**(*img*, *props*, *edge_flt*)

Filter objects at the edge

Parameters

- **img** (*numpy.ndarray*) – mask image with object labels.
- **props** (*pandas.DataFrame*) – part of the object table.
- **edge_flt** (*int*) – pixel width of the edge area.

pcnaDeep.data.utils.**expand_bbox**(*bbox*, *factor*, *limit*)

Expand bounding box by factor times.

Parameters

- **bbox** (*tuple*) – (x1, y1, x2, y2).
- **factor** (*float*) – positive value, expand height and width by multiplying the factor. Round if result is not integer. The output shape will be (factor + 1) ** 2 times of the original size.
- **limit** (*tuple*) – (x_max, y_max), limit values to avoid boundary crush.

Returns new bounding box (x1, y1, x2, y2).

Return type (tuple)

pcnaDeep.data.utils.**align_table_and_mask**(*table*, *mask*)

For every object in the mask, check if is consistent with the table. If no, remove the object in the mask.

Parameters

- **table** (*pandas.DataFrame*) – (tracked) object table.
- **mask** (*numpy.ndarray*) – labeled object mask, object label should be corresponding to *continuous_label* column in the table.

pcnaDeep.data.utils.**merge_obj_tables**(*a*, *b*, *col*, *mode='label'*)

Merge two object tables according to shared frame and continuous label / location identity.

Parameters

- **a** (*pandas.DataFrame*) – donor table. Record not found in acceptor will be ignored.
- **b** (*pandas.DataFrame*) – acceptor table. Record cannot be matched with donor will results in NA and warned.
- **col** (*str*) – key in both a and b that aimed to merge. Only allow one key and a time
- **mode** (*str*) – either 'label'or 'loc'.

Note:

Both a and b tables must have the keys:

- Center_of_the_object_0
- Center_of_the_object_1
- continuous label
- frame
- (key to merge)

In loc mode, location will be rounded to 3 digits before matching.

PYTHON MODULE INDEX

p