PyCellID

Release 0.1

Clemente, Jose

Dec 22, 2021

CONTENTS:

CHAPTER

ONE

INSTALLATION

PyCellID can be installed using pip from PyPI.

```
$ pip install pycellid
```

1.1 Development

We welcome all kinds of contributions. You can build the development environment locally with the following commands:

```
$ git clone git@github.com:pyCellID/pyCellID.git
$ cd pyCellID
$ python3 -m venv venv
$ source venv/bin/activate
$ pip install -e .
$ pip install -r requirements/dev.txt
```

Run the tests with pytest:

```
$ pytest -v tests/
```

Or run the full checks with tox:

```
$ tox -r
```

1.2 Initialization

```
>>> import pycellid
>>> pycellid.__version__
'0.0.18'
>>> # Build or load your data, inspect your images and make plots.
>>> from pycellid.core import CellData, CellsPloter
>>> import pycellid.io as ld  # Build or load data frame.
>>> from pycellid import images  # Get a 2-D aray representing your images.
```

CHAPTER

TWO

QUICK START

2.1 Project Structure

PyCellID is an Open Source python package designed for recursively navigating a path and tracking down data tables and metadata (mapping), returning a unique CellData object. It relates easily the data with the associated images.

The power of PycellID will be maximized when working with images of time series.

file structure:

```
~folder_location
|
|--- my_experiment/
     |
     |---Position01/
     |   |---mapping.txt
     |   |---out_all.txt
     |   |
     |---Position02/
     |   |--- ...
     |   |
     |---Position03/
     |   |--- ...
     |   |
     |---Position32/
     |   |--- ...
     |   |
     |---channel-x_Position01_time01.tif
     |---channel-x_Position01_time01.tif.out.tif
     |--- ...
     |---channel-z_PositionW_timeY.tif
```

2.1.1 Resource helpers

[1]:
```
# Build or load your data, inspect your images and make plots.
from pycellid.core import CellData, CellsPloter
# Build or load data frame.
import pycellid.io as ld
# Get a 2-D aray representing your images.
from pycellid import images
import matplotlib.pyplot as plt
```

2.2 Load your data

2.2.1 Load your data frame

Instantiate a CellData from your data table and the path to your images.

2.2.2 From Cell-ID output

[2]:
```
df = CellData.from_csv('samples_cellid')
```

[3]:
```
display(df)
```

```
       pos  t_frame          ucid  cellID  time  xpos  ypos  a_tot  num_pix  \
0        1        0  100000000000       0     0    29   725  527.0      527
1        1        1  100000000000       0     0    31   725  614.0      614
2        1        2  100000000000       0     0    31   724  639.0      639
3        1        3  100000000000       0     0    31   725  688.0      688
4        1        4  100000000000       0     0    24   721  457.0      457
...    ...      ...           ...     ...   ...   ...   ...    ...      ...
18184    3       12  300000000720     720     0   802   736  147.0      147
18185    3       12  300000000721     721     0   840   707  391.0      391
18186    3       12  300000000722     722     0   915   827  587.0      587
18187    3       12  300000000723     723     0   986   336  119.0      119
18188    3       12  300000000724     724     0  1014    84  120.0      120

       fft_stat  ...  f_nucl_tag6_tfp  f_nucl_tag6_yfp  f_local_bg_cfp  \
0      0.387982  ...        1675334.0          54200.0        378.6183
1      0.443545  ...        1473843.0          54020.0        383.6143
2      0.500724  ...        1451150.0          46463.0        379.6064
3      0.591332  ...        1566434.0          47871.0        390.2697
4      0.085026  ...        1204067.0          52761.0        372.6058
...         ...  ...              ...              ...             ...
18184  0.556508  ...        1237009.0          39586.0        520.8205
18185  0.040324  ...        1053566.0          44035.0        477.8000
18186  0.672861  ...        1534440.0          58636.0        508.6136
18187  0.494793  ...         469596.0          36213.0        530.8571
18188  0.827304  ...         526434.0         127674.0        453.5542

       f_local_bg_rfp  f_local_bg_tfp  f_local_bg_yfp  f_local2_bg_cfp  \
0            240.4300        12571.83        312.5942         372.9697
1            240.4676        12212.17        319.5286         377.4458
2            241.2868        11988.62        322.5431         376.9560
3            436.6727        11646.41        330.3378         391.3493
4            241.3909        11593.46        330.6237         367.9105
...               ...             ...             ...              ...
18184        254.4151        10049.78        612.9800         522.1818
18185        248.6977        11320.68        576.1446         463.5887
18186        250.6716         9132.60        578.7262         509.0000
18187        250.7143        11290.47        423.6049         536.5789
18188        249.6765        10899.60        716.8197         460.7500

       f_local2_bg_rfp  f_local2_bg_tfp  f_local2_bg_yfp
0             241.2194         12523.050         310.0760
1             240.1235         12138.300         317.3976
2             242.0784         11993.090         323.3418
```

(continues on next page)

(continued from previous page)

```
3             437.9252        11549.440         331.3429
4             242.5352        11503.960         329.0977
...                ...              ...              ...
18184         254.9767        10533.790         621.8302
18185         250.3025        11368.400         571.9272
18186         250.9851         9129.724         572.0000
18187         251.0638        11238.870         423.4595
18188         248.6575        11796.740         903.9592

[18189 rows x 125 columns]
```

2.3 Inspect your data

The development team decided to use pandas library as backend because of its syntax and its extensive documentation. The idea is to make you feel you are working with a pandas object, but with the flexibility of having access to your experimental images.

You would be able to choose from different backends in future versions.

[4]:
```
df.describe()
```

[4]:
```
                pos       t_frame          ucid        cellID     time  \
count  18189.000000  18189.000000  1.818900e+04  18189.000000  18189.0
mean       1.966078      6.216944  1.966078e+11    317.438947      0.0
std        0.793782      3.726506  7.937819e+10    204.508054      0.0
min        1.000000      0.000000  1.000000e+11      0.000000      0.0
25%        1.000000      3.000000  1.000000e+11    150.000000      0.0
50%        2.000000      6.000000  2.000000e+11    297.000000      0.0
75%        3.000000      9.000000  3.000000e+11    452.000000      0.0
max        3.000000     12.000000  3.000000e+11    966.000000      0.0

               xpos          ypos         a_tot       num_pix      fft_stat  \
count  18189.000000  18189.000000  18189.000000  18189.000000  18189.000000
mean     683.000935    529.324097    441.458959    441.460718      0.265314
std      396.587704    289.357639    241.039776    241.038580      0.225960
min        9.000000      9.000000     98.000000     98.000000      0.018116
25%      327.000000    285.000000    277.000000    277.000000      0.068958
50%      681.000000    538.000000    384.000000    384.000000      0.202481
75%     1025.000000    779.000000    555.000000    555.000000      0.420594
max     1384.000000   1032.000000   1499.000000   1499.000000      2.200194

       ...  f_nucl_tag6_tfp  f_nucl_tag6_yfp  f_local_bg_cfp  f_local_bg_rfp  \
count  ...     1.818900e+04     1.818900e+04    18189.000000    18189.000000
mean   ...     1.258965e+06     1.087345e+05      462.553463      258.523334
std    ...     4.217068e+05     1.837672e+05       42.853657       50.494524
min    ...     2.100890e+05     7.107000e+03      314.227900      232.300000
25%    ...     9.154950e+05     2.818300e+04      439.835400      244.754400
50%    ...     1.177204e+06     4.325000e+04      463.130400      248.077800
75%    ...     1.592472e+06     1.144470e+05      486.517200      251.027400
max    ...     2.342580e+06     2.358444e+06      734.365400      615.073200

       f_local_bg_tfp  f_local_bg_yfp  f_local2_bg_cfp  f_local2_bg_rfp  \
count    18189.000000    18189.000000     18189.000000     18189.000000
mean     11753.645404      502.869092       459.888134       258.526309
std       1378.053509      349.180950        44.225006        50.631219
```

(continues on next page)

(continued from previous page)

```
min       6984.529000        0.000000       316.946400       231.608700
25%      10809.180000      350.870100       435.564800       244.753600
50%      11783.600000      397.611100       458.925400       248.076900
75%      12781.720000      511.681500       484.095600       250.980100
max      15438.730000     6654.856000       971.200000       618.813700

       f_local2_bg_tfp  f_local2_bg_yfp
count     18189.000000     18189.000000
mean      11921.632917       489.371202
std        1297.764306       308.290503
min           0.000000         0.000000
25%       11063.320000       351.651900
50%       11939.820000       397.530600
75%       12887.200000       501.397500
max       15294.890000      7044.286000

[8 rows x 125 columns]
```

2.3.1 Using Cell-ID features

2.4 Cell-ID together with PyCellID provide 5 categories of calculated variables:

1. General measurments.

```
pos,
cellID,
ucid,
t_frame,
time,
xpos,
ypos,
f_tot,
a_tot,
fft_stat,
perim,
maj_axis,
min_axis,
flag,
rot_vol,
con_vol,
a_vacuole,
f_bg
```

2. To calculate membrane proximal fluorescence (for relocalization experiment).

```
f_tot_p1_channels,
a_tot_p1,
f_tot_m1_channels,
a_tot_m1,x
f_tot_m2_channels,
a_tot_m2,
```

(continues on next page)

(continued from previous page)

```
f_tot_m3_channels,
a_tot_m3
```

3. Information obtained from "nuclear image" type (Variables containing the area and fluorescence of concentric disks of user-defined radius).

```
f_nucl_channels,
f_nucl1_channels to f_nucl6_channels,
a_nucl1 to a_nucl6,
f_nucl_tag1_channels to f_nucl_tag6_channels
```

4. More background information.

```
f_local_bg_channels,
a_local_bg,
a_local,
f_local2_bg_channels,
a_local2_bg,
a_local2
```

5. More volume measurments.

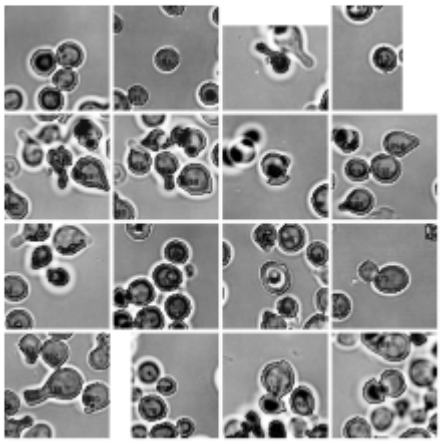
```
a_surf, sphere_vol,
```

the final tag `_channels` *indicates that the vatiable will be repeated for each illumination type.*

For detailed information, reading Cell-ID documentation is recomended.

[5]:
```
df.plot()
```

[5]:
```
<AxesSubplot:>
```



2.4.1 View your images

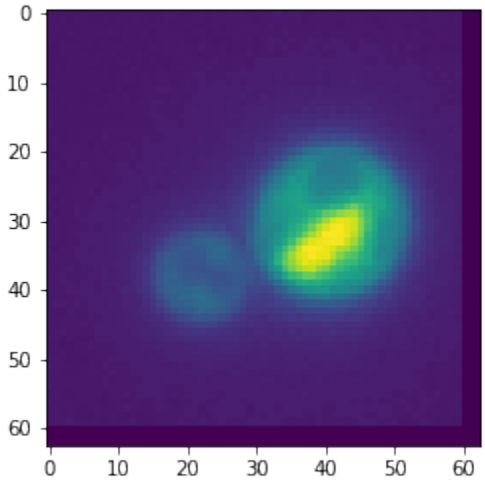
Obtain a numpy array representation of an image. Make a crop, operate o simply plot it.

[6]:
```
img = plt.imread("samples_cellid/YFP_Position01_time01.tif")

array = images.box_img(im=img, x_pos=640, y_pos=560, radius=30, mark_center=False)

array
```

[6]:
```
array([[371., 348., 330., ...,   0.,   0.,   0.],
       [368., 351., 368., ...,   0.,   0.,   0.],
       [370., 363., 351., ...,   0.,   0.,   0.],
       ...,
       [  0.,   0.,   0., ...,   0.,   0.,   0.],
       [  0.,   0.,   0., ...,   0.,   0.,   0.],
       [  0.,   0.,   0., ...,   0.,   0.,   0.]])
```

[7]:
```
plt.imshow(array)
```

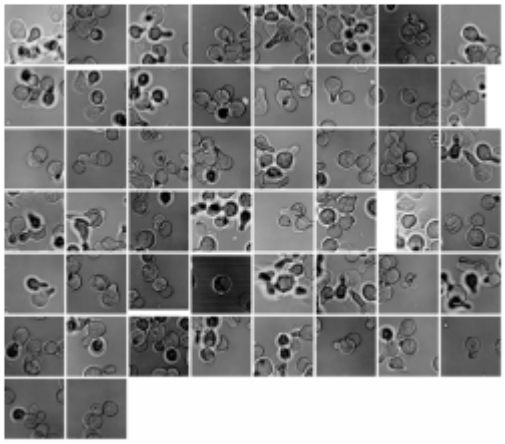
[7]:
```
<matplotlib.image.AxesImage at 0xb50faf0>
```



You can use PycellID accessor to inspect images. + Use data from your dataframe for finding what you are looking for.

[8]:
```
# Specify your values.
df.plot(array_img_kws={"channel":"tfp", "n":50, "criteria":{"a_tot":[0, 500]}})
```

[8]:
```
<AxesSubplot:>
```

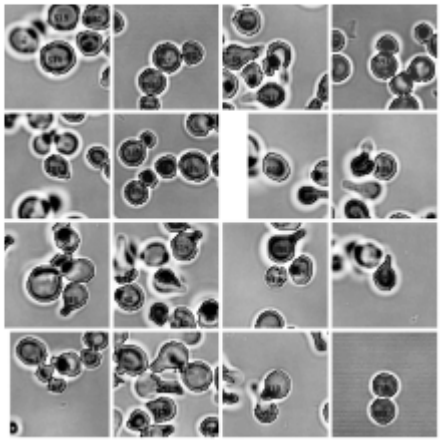


2.4.2 Use CellsPloter to inspect images

[9]:
```
cells = CellsPloter(df)
```

[10]:
```
cells.cells_image()
```

[10]:
```
<AxesSubplot:>
```



CHAPTER

THREE

BASIC EXAMPLES

3.1 Prerequisites

1. Clone the repository runing the following command:

```
$ git clone https://github.com/pyCellID/pyCellID
$ cd pyCellID
```

2. create a jupyter notebook called 'example.ipynb'

The data used in the following example can be downloaded from the 'samples_cellid' folder of the PycellID repository.

[1]:
```
import os
import matplotlib.pyplot as plt
import pandas as pd
import pycellid as ld
```

3.2 CellData object instantiation

A CellData object can be instantiated by declaring the parameters:

- path: directory that contains the images segmented by Cell-ID.
- df: a Dataframe containing the variables measured for each cell.

[2]:
```
base_dir = os.path.join(".","samples_cellid")
data_dir = os.path.join(base_dir,"pydata","df.csv")
data = pd.read_csv(data_dir)
```

[3]:
```
cells = ld.CellData(path=base_dir,df=data)
```

3.3 Inspection of data structure

Information about the way the data is structured can be obtained through intuitive commands.

- Number of rows times number of columns of the table containing Cell-ID data

[4]:
```
cells.size
```

[4]:
```
2255436
```

- Number of rows and columns of the table

[5]:
```
cells.shape
```

[5]:
```
(18189, 124)
```

- List of the first 15 parameters of data table

[6]:
```
cells.columns[0:15].to_list()
```

[6]:
```
['pos',
 't_frame',
 'ucid',
 'cellID',
 'time',
 'xpos',
 'ypos',
 'a_tot',
 'num_pix',
 'fft_stat',
 'perim',
 'maj_axis',
 'min_axis',
 'a_nucl',
 'rot_vol']
```

- Description of the statistics of parameters

[7]:
```
cells["maj_axis"].describe()
```

[7]:
```
count    18189.000000
mean        28.362692
std         10.433144
min          3.343855
25%         20.955810
50%         25.280250
75%         34.729460
max         92.609380
Name: maj_axis, dtype: float64
```

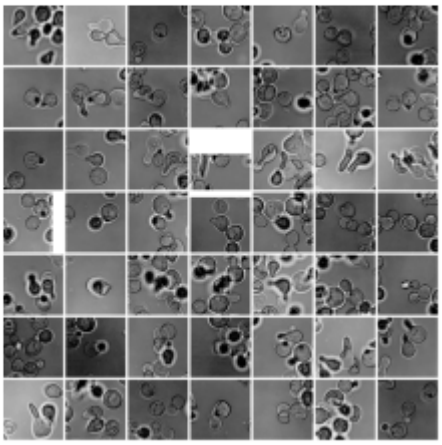
3.4 Filtering data

You would want to look at your images to rule out out-of-focus cells, find dead cells, or remove those cells that biology tells you to be outliers.

It is posible to present exploratory plots with randomly chosen cells satisfying certain criteria.

[8]:
```
# Exploratory plot presenting n = 49 randomly chosen cells
# with an area smaller than 500 pixels.
# Out of focus cells, dead cells, clusters and other
# defects can be seen among the images displayed.
cells.plot(array_img_kws={"channel":"tfp", "n":49, "criteria":{"a_tot":[0, 500]}})
```

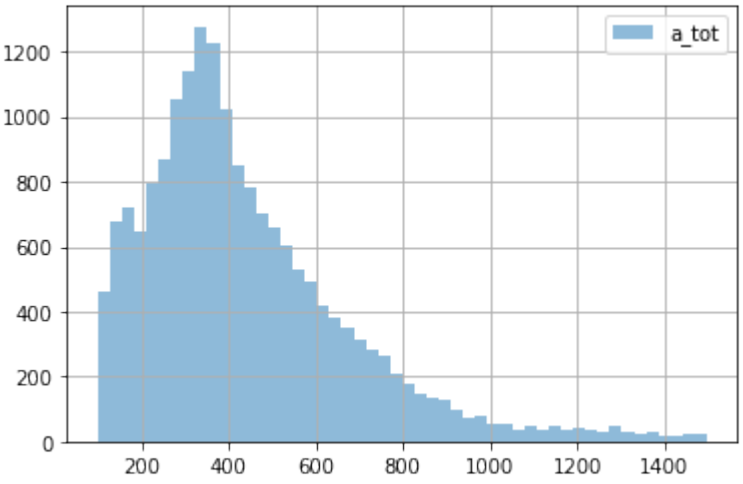
[8]:
```
<AxesSubplot:>
```



Data can also be presented using histograms and other types of plots.

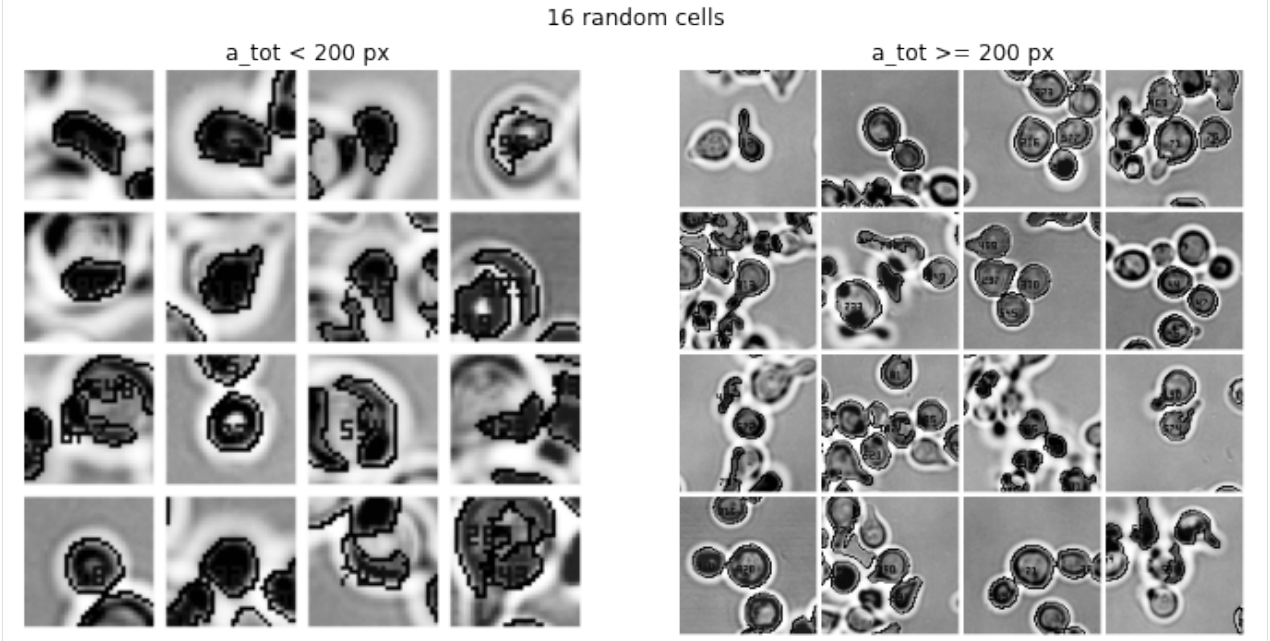
[9]:
```
# Histogram of cells in the experiment based on their total area.
cells.a_tot.hist(bins=50, alpha=0.5, legend=True)
```

[9]:
```
<AxesSubplot:>
```



Data can be filtered applying thresholds to the measured parameters. In this case we want to differentiate cells with areas greater and smaller than 200 pixels and show examples of them in two plots.

[10]:
```
# initialize your figure
fig, axs = plt.subplots(1, 2)
fig.set_size_inches(10, 5)

# Filtering your data
cells[cells["a_tot"]<200].plot(ax=axs[0])
cells[cells["a_tot"]>=200].plot(ax=axs[1])

# titles
plt.suptitle('16 random cells')
axs[0].set_title('a_tot < 200 px')
axs[1].set_title('a_tot >= 200 px')

# customize your output
fig.tight_layout()
```



3.5 CellsPloter

PyCellID owns the CellsPloter accessor, in charge of rendering the images. You can use the axes provided by CellPloter with the library of your choice.

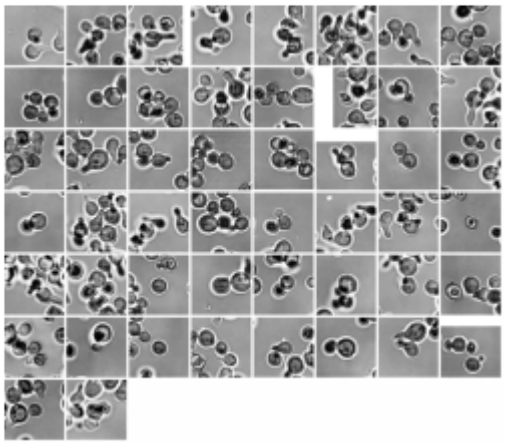
CellPloter defines two useful methods: + cells_image + cimage

[11]:
```
# Defining the accessor
ploter = ld.CellsPloter(cells)
```

cells_image method allows the user to plot a custom number of cells.

[12]:
```
ploter.cells_image({"n":50})
```

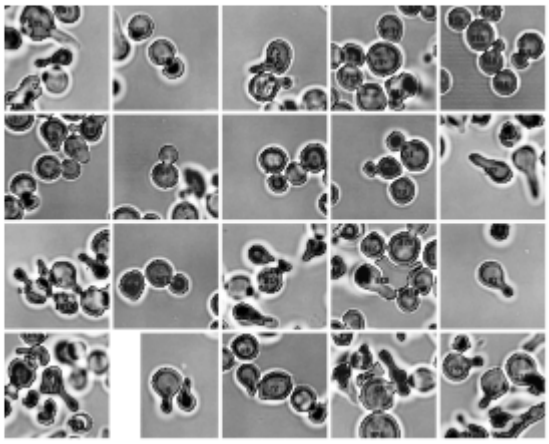
[12]:
```
<AxesSubplot:>
```



It also let the user to define filtering criteria.

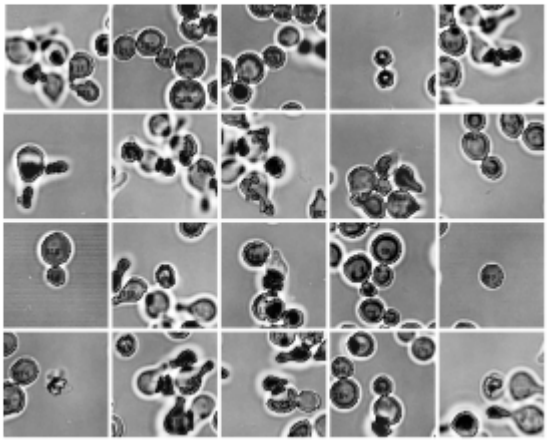
[13]:
```
ploter.cells_image({"n":20,"criteria":{"maj_axis":[30,40]}})
```

[13]:
```
<AxesSubplot:>
```



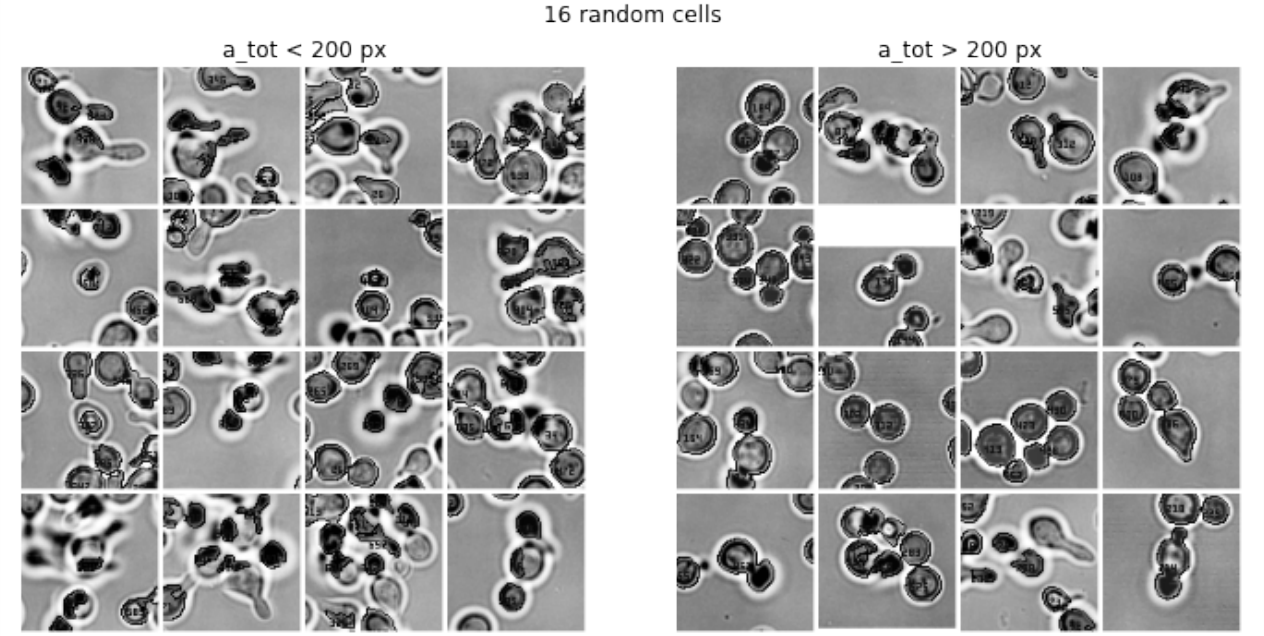
[14]:
```
ploter.cells_image({"n":20,"criteria":{"maj_axis":[10,20]}})
```

[14]:
```
<AxesSubplot:>
```



Data filtering performed in the previous section can be also carried out using method *cells_image* from CellsPloter.

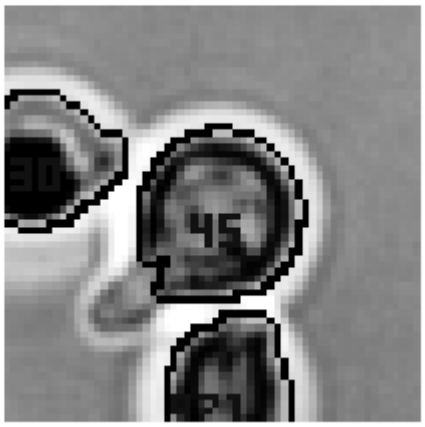
[15]:
```
# initialize your figure
fig, axs = plt.subplots(1, 2)
fig.set_size_inches(10, 5)

a_max = cells['a_tot'].max()

# Filtering your data
ploter.cells_image(array_img_kws={'n':16,'criteria':{'a_tot':[0.0,150]}},ax=axs[0])
ploter.cells_image(array_img_kws={'n':16,'criteria':{'a_tot':[150,a_max]}},ax=axs[1])

# titles
plt.suptitle('16 random cells')
axs[0].set_title('a_tot < 200 px')
axs[1].set_title('a_tot > 200 px')

# customize your output
fig.tight_layout()
```



On the other hand, *cimage* method is used to plot a single cell in a fluorescence channel based on its unique cell identifier (ucid) and the time stamp (t_frame) corresponding to the instant of interest.

[16]:
```
indentifier = { "channel":"BF", "ucid":100000000045, "t_frame":5}
box_img_kws = {"radius":30}
ploter.cimage( indentifier,box_img_kws = box_img_kws)
```

[16]:
```
<AxesSubplot:>
```



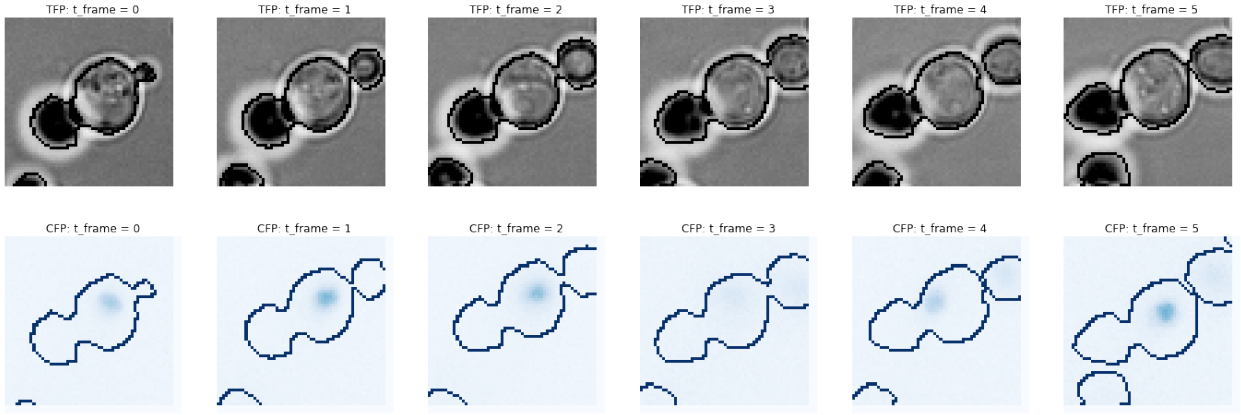
It can be used to show the time evolution of an individual cell. As an example, a single cell is presented at different times in TFP and CFP channels.

[17]:
```
ucid = 100000000047
box_img_kws = {"radius":30}

# initialize your figure
fig, axs = plt.subplots(2, 6)
fig.set_size_inches(24, 8)
```

(continues on next page)

(continued from previous page)

```
channels = ["TFP", "CFP"]
cmaps = ["Greys", "Blues"]

for j, c in enumerate(channels):
    imshow_kws = None
    imshow_kws = {"cmap": cmaps[j]}
    for i in range(6):
        indentifier = { "channel":c, "ucid":ucid, "t_frame":i}
        ploter.cimage( indentifier, box_img_kws = box_img_kws, imshow_kws=imshow_kws,␣
↪ax=axs[j,i])
        titles = f'{c}: t_frame = {i}'
        axs[j,i].set_title(titles)
```



CHAPTER

FOUR

LICENSE

CHAPTER

FIVE

PYCELLID CHANGELOG

CHAPTER

SIX

PYCELLID

6.1 pycellid package

6.1.1 Submodules

6.1.2 pycellid.core module

Merge and analyze tables of characteristics of cells and find images.

class pycellid.core.**CellData**(*path*, *df*)

Bases: object

Collapse your data into a single data frame.

Recursively inspect the path, create a unique identifier per cell, and inspect related images.

Parameters

- **_path** (*str*) – global path to output CellID tables.
- **_df** (pandas.DataFrame) – Dataframe (output of CellID) containing all the measured parameters of each cell.

Returns

- An instance of CellData containing all the information of
- *microscopy experiment.*

Examples

```
>>> from pycellid.core import CellData
>>> df = CellData(
    path = '../my_experiment',
    df = my_dataframe
)
```

classmethod from_csv(*path*, ***kwargs*)

Build a data frame from csv files contained in path.

A CellData class will be instantiated.

get_dataframe()

Return a copy of the internal _df.

property plot
Represent set of `cells_image` or numerical data.

For `cimage` method you must specify an identifier `id={}`.

class pycellid.core.**CellsPloter**(*cells*)
Bases: `object`

Accessor to plotter class.

Create a representation of each cell within a grid, inspect an entire image or create a snippet of a single cell. Provide a wrapper of pandas methods for plotting. Return axes to plot.

cells
An instance of `CellData` containing all the information of microscopy experiment.

Type *CellData*

cells_image(*array_img_kws=None*, *imshow_kws=None*, *ax=None*)
Display a random selection of cells on a square grid.

By default it represents a 4×4 matrix chosen at random.

Returns

Return type ax to plot or figure.

Other Parameters

- **array_img_kws** (*dict*) – Set the `pycellid.images.img_array` parameters.

  `n` : number of cells.

  `channels` : "`TFP`" or another that you have encoded.
- **imshow_kws** (*dict*) – If you use matplotlib set equal to `plt.imshow`.
- **ax** – Use your axes to plot.

cimage(*identifier*, *box_img_kws=None*, *imshow_kws=None*, *ax=None*)
Show a sigle cell or complete image.

'`identifier`' param is required. Reference to a valid image or position. By default, an image with a size of `(1392 X 1040)px` will be rendered.

Parameters **identifier** (*path or dict*) – path to an image file `dict = { "channel":str, "UCID":int, t_frame":int }`.

Returns

Return type ax to plot or figure.

Other Parameters

- **box_img_kws** (*dict*) – Set the `pycellid.images.box_img` parameters.

  im [numpy.array] A full fluorescence microscopy image.

  x_pos [int] x-coordinate of the center of the cell of interest.

  y_pos [int] y-coordinate of the center of the cell of interest.

  radius [int] lenght (in px) between the center of the image and each edge. Default = `90`.

  mark_center [`bool`] mark a black point. Default = `False`.
- **imshow_kws** (*dict*) – If you use matplotlib set equal to `plt.imshow`.

- **ax** – Use your axes to plot.

6.1.3 pycellid.images module

Images for PyCellID.

Attention! This module will not provide images. This module provides matrix representations for your analysis or to work with your favorite framework.

`pycellid.images.`**`array_img`**(*data*, *path*, *channel='BF'*, *n=16*, *criteria=None*)

Create a grid of images containing cells which satisfy given criteria.

Resulting image has 'n' instances ordered in a grid of shape 'shape'. Each instance corresponds to a image centered in a cell satisfying provided criteria.

Parameters

- **`data`** (`pandas.DataFrame`) – Dataframe (output of `CellID`) containing all the measured parameters of each cell.
- **`path`** (*`str`*) – Path to the directory containing the images asociated to `data`.
- **`channel`** (*`str`*) – Fluorescence channel of the image. The allowed values are `'BF'`, `'CFP'`, `'RFP'` or `'YFP'`.
- **`n`** (*`int`*) – Number of instances composing the grid.
- **`criteria`** (*`dict`*) – Dictionay containing the criteria of selection of cells.

Returns **iarray** – A grid of `n` images of cells satisfying given criteria.

Return type numpy.array

Raises **`ValueError`** – If the number of cells satisfying the selection criteria is less than the number of cells to be shown.

`pycellid.images.`**`box_img`**(*im*, *x_pos*, *y_pos*, *radius=90*, *mark_center=False*)

Create a single image contatinig an individualised cell.

The resulting image posses a mark in the center of the individualised cell and a pair of delimiters in the right and bottom edges.

Parameters

- **`im`** (*`numpy.array`*) – A full fluorescence microscopy image.
- **`x_pos`** (*`int`*) – x-coordinate of the center of the cell of interest.
- **`y_pos`** (*`int`*) – y-coordinate of the center of the cell of interest.
- **`radius`** (*`int`*) – lenght (in pixels) between the center of the image and each edge.
- **`mark_center`** (`bool`) – mark a black point, defoult = `False`.

Returns **iarray** – An array (image) containing an individualised, center-pinned, cell.

Return type numpy.array

`pycellid.images.`**`img_name`**(*path*, *ucid*, *channel*, *t_frame=None*, *fmt='.tif.out.tif'*)

Construct the name of an image according to output format of CellID.

The returned string contains the path and name of the image.

Parameters

- **`path`** (*`str`*) – Path to the directory containing images asociated to 'data'.

- **ucid** (*int*) – Unique traking number.
- **channel** (*str*) – Fluorescence channel of the image. The allowed values are `'BF'`, `'CFP'`, `'RFP'` or `'YFP'`.
- **t_frame** (*int*) – Time tag of the image.
- **fmt** (*str*) – Format of the image to be read.

Returns **name** – Name and path of an image according to the output format of `CellID`.

Return type str

6.1.4 pycellid.io module

in-out implementations for pyCellID.

`pycellid.io.`**make_df**(*path_file*)

Make a dataframe with number tracking 'ucid' and 'position'.

Parameters **path_file** (*str*) – Path to CellID's `outall` data files.

Returns **df** – Dataframe with `df['ucid']` unique cell identifier.

Return type pandas.DataFrame

`pycellid.io.`**merge_tables**(*path*, *n_data='out_all'*, *n_mdata='*mapping'*)

Concatenate tables in the path with pandas method.

Transforms the identifying index of each cell from each data table into a temporal index UCID (Unique Cell Identifier) Disaggregate the columns of morphological measurements into columns by fluorescence channel. It uses the mapping present in the metadata file (mapping).

Parameters

- **path** (*str*) – Global path to output 'cellID' tables.
- **n_data** (*str*) – File name to find each data table.
- **n_mdata** (*str*) – File name to find metadata tables or mapping_tags.

Returns **df** – Dataframe containing 'cellID' data.

Return type pandas.DataFrame

Examples

```
>>> import pycellid.io as ld
>>> df=ld.cellid_table(
    path = '../my_experiment',
    n_data ='out_all',
    n_mdata ='mapping'
)
```

`pycellid.io.`**read_df**(*path_file*)

Read files with data of fluorescence microscopy experiments.

Create a dataframe with the data and rewrite headers format.

Parameters **path_file** (*str*) – Path to files to be read.

Returns **df** – Dataframe with data of fluorescence microscopy experiments.

Return type pandas.DataFrame

6.1.5 Module contents

pyCellID.

An extension that analyze Cell-ID single-cell.

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX