Image analysis software

https://github.com/albagranados/cellviewer/tree/master/imageanalysis

Alba Granados

Jérôme Solon's laboratory

Center for Genomic Regulation, Barcelona, Spain

October 10, 2018

Outline

Workflow 0: read dataset

```
file_dirs = ['/home/alba/ownCloud/postdoc_CRG/coding/github/
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣cellviewer/data/melike/Histones/crops/noTSA/',
              '/home/alba/ownCloud/postdoc_CRG/coding/github/
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣cellviewer/data/melike/Histones/crops/TSA/']
is_dataset = 0
file_name = 'hFb.2_driftcorrectedlist_85120_105135'
fileExt = '.txt'
is_storm = 0
run = dict(image_processing=1, image_analysis=1)
```

file_dirs contains one or more paths to the experiments
is_dataset is 0 if we only want to analyze 1 cell/window
file_name, fill if is_dataset=0,
is_storm is 1 if the file contains all output columns from typical STORM output. It is 0 if the .txt file is two-columns with x-corrected and y-corrected (e.g., crop window from entire STORM image)

Workflow 0: bin2txt and crop

Input parameters:

```
dict_inputfile = dict(file_dir=file_dir, file_name=file_name,
                      fileExt=fileExt, is_storm=is_storm,
                      out_channel='all',
                  ispp=1, compute_ROI=0, crop=1,
                        crop_range=[85, 120, 105, 135],
                      pixelate=0,
                      tessellate=1,
                  original_pixel_size=160, photonconv=0.14,
                      resolution=1)
```

out_channel matters if **is_storm**=1. Values: 'all', [1,2], [0],....
ispp is 1 if input is a point pattern, like STORM output, and not a regular image.
compute_ROI is 1 if we want to compute a smaller area. Then, if **crop** is 1, we can select the **crop_range** as $[x_0, x_1, y_0, y_1]$.
pixelate is 1 if the image is generated by regular pixellation, or **tessellate** is 1 if we do that via Voronoï tessellation.
original_pixel_size is STORM pixel size (nm)

Outline

Workflow 1: Point pattern to image

Input parameters:

```
analysis_pixel_size = 10
scale_pixel_size = float(analysis_pixel_size)/
                        dict_inputfile.get('original_pixel_size')
dict_image = dict(scale_pixel_size=scale_pixel_size,
                  original_pixel_size=dict_inputfile.
                                    get('original_pixel_size'),
                  interpolate_method='linear',
                  detect_densitytransform='log',
                  descr_densitytransform='linear')
```

analysis_pixel_size is the pixel size (regular grid) of the analysis density-based image (point(STORM) to image).
interpolate_method is the method of density interpolation from irregular grid (Voronoï) to regular grid.
The Voronoï-based density map (image) is transformed for the feature detection algorithm (**detect_densitytranform**) and the feature descriptor algorithm (**descr_densitytransform**); go to slide 8

Workflow 2: Cluster detection

Input parameters:

```
print '\n________IMAGE_PROCESSING_______'
dict_sift = dict(scale_pixel_size=scale_pixel_size, resolution=1,
            original_pixel_size=dict_inputfile.
                                    get('original_pixel_size'),

            # feature detection
            t=10, feature_name='blob', # 0.8, scale_ini=80
            thresholding=1, threshold_percent=0.8,
            num_features='all', scale_range_is='nm', scale_ini=40,
            scale_end=200,
            scale_spacing='odd', nscales=150,
            scale_resolution=dict_inputfile.get('resolution'),
            max_filter_width=7, max_filter_depth=7,

            [...]
            )
```

t is a single scale, but typically we'll use a range (see below)
thresholding can be activated. **threshold_percent** corresponds to the % of the largest values of the Laplacian.
num_features is the maximum number of features to be detected. Can be 'all'
scale_ini and **scale_end** defines the limits of the scale range search. If 'odd' is **scale_spacing**, then $t_1, \ldots, t_n$ s.t. $3\sigma = 3\sqrt{t_i} = d_i$ for $d_i = 2n + 1$ diameters (odd number of pixels), and **nscales** is neglected.
Filter for local maxima has dimensions **max_filter_width** (space) and max_filter_depth (scale)

Workflow 2: Cluster detection

detect_densitytranform

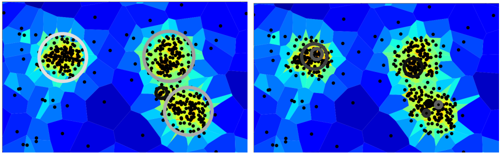
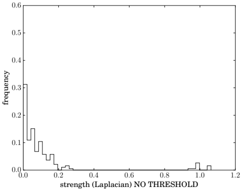


Figure: Detection with a logarithmic (left) and linear (right) density map. Logarithmic is shown in both cases for clarity reasons.

threshold_percent needed?



Noise is discernible from circular clusters in terms of the Laplacian.

Workflow 3: Cluster description

SIFT input parameters:

```
dict_sift = dict(

            [...]

            # feature description [main orientation(s)]
            compute_orientation=1, n_bins_ori=36, peak_ratio=0.7,
            sigma_ori_times=1.5, window_ori_radtimes=1,
            smooth_cycles=2,

            # feature description [histograms]
            compute_sift_descr=1,
            sigma_descr_times=2, window_descr_radtimes=1,
            n_hist=16, n_bins_descr=8, threshold_sat=0.2,
            plot_graphics=0)
```

n_bins_ori is the number of bins of the histogram of gradients arguments to define the main orientations.
peak_ratio is the portion of the largest bin (largest orientation) above which all maxima are considered main orientations.
smooth_cycles is the number of cycles to smooth the histogram.
sigma_ori_times and **window_ori_radtimes** define the window of the local patch to compute the gradients. Recall $\sigma = \sqrt{t}$ (scale).
sigma_descr_times and **window_descr_radtimes** is the size of the local patch to compute the n-dimensional SIFT descriptor, where $n =$**n_hist**$\cdot$**n_bins_descr**.
With **threshold_sat** we reduce the influence of large gradient magnitudes by thresholding the values in the unit feature vector to each be no larger than 0.2, and then renormalizing to unit length.

Outline

Workflow 4: Cluster-based classification

```
print '\n ---------- IMAGE ANALYSIS -------'; ini_time = time.time()
init = 'k-means++'
k0 = 2; kn = 5; sserror = []  # build bag of words
for k in range(k0, kn+1):
        [...]
```

k0 and **kn** define the range of number of clusters/words we want to analyze