

How to use the algorithm

Part 1 : Overview of the procedure

Part 2 : Step-by step procedures for operating provided Python codes

2.1 Overlay of DFM image and FM image obtained by a dual-mode microscope

2.2 Overlay of DFM image and FM image obtained by a DFM microscope and a fluorescence microscope

2.3 Analysis of aggregation states of SNAs with colorimetry-based classification algorithm

Part 3 : Data examples for analyzing classification of intracellular SNAs

3.1 Data examples

3.2 Important steps for optimizing algorithm parameter

Part 1 : Overview of the procedure

We are not very good at coding. The codes we provided may be not well documented. We suggest to optimize them according to your specific studies if possible. For those wanting to try it out:

System Requirements:

1. Hardware Requirements

Personal computer, desktop/laptop (Operating system: Windows or Mac, we have not tried Linux).

2. Software Requirements

All scripts are written in Python 2. It is recommended to download the latest version of Python 2.7. The following Python libraries are also required:

Pillow (6.1.0)

Opencv-python (4.1.1.26)

numpy (1.16.5)

scipy (0.13.0b1)

matplotlib (2.2.4)

Atom (<https://atom.io>) provides a way to quickly edit Python scripts. If you install Atom, you can edit and run codes in Atom.

Installation guide:

Download and install Python 2.7 and Atom.

Install package mentioned with pip (e.g. '[pip install matplotlib](#)')

Download scripts from file folder '*python_scripts*'. We provided five '.py' files in total (see Table 1 below).

Typical install time - up to one hour.

Table 1: The list of manual programs we provide.

File name	Function	Procedure
" <i>pic_overlay_dual mode.py</i> "	To merge the images of DFM and FM obtained by a dual-mode microscope	Section 2.1
" <i>pic_overlay_DFM_FM.py</i> "	To merge the images of DFM and FM obtained by a DFM microscope and a fluorescence microscope	Section 2.2
" <i>pic_overlay_DFM_FM2.py</i> "		
" <i>darkfield_analysis_protocols.py</i> "	To analyze the aggregation states of SNAs with colorimetry-based classification algorithm	Section 2.3
" <i>read_pixel_protocols.py</i> "		

Part 2: Step-by step procedures for operating provided programs

2.1 Overlay of DFM image and FM image obtained by a dual-mode microscope

The demo INPUT DFM image and FM image are provided in the folder “test_data/dual_mode”, and the expected OUTPUT image is provided in the same folder with a file name “Expect_result.tif”. We use a mac laptop to run our programs. Expected run time is 20~30 seconds.

! CAUTION The parameters (X_shift and Y_shift) applied in the provided Python code are optimized for our own experimental setup and conditions. Further optimization may be needed for particular studies.

1. Open the file *pic_overlay_dual mode.py* with python text editor (for example, Atom, <https://atom.io>).
2. Change “File Path” and “File name” to your own file name in lines 2, 3 and 6.
3. Change “Y_shift” and “X_shift” to your own shift in lines 4.
4. Run the edited code.

2.2 Overlay of DFM image and FM image obtained by a DFM microscope and a fluorescence microscope

The demo INPUT DFM image and FM image are provided in the folder “test_data/DFM_FM”, and the expected OUTPUT image is provided in the same folder with a file name “Expect_result.tif”. We use a mac laptop to run our programs. Expected run time is 20~30 seconds.

! CAUTION The parameters (X_shift ,Y_shift and rotation angles) applied in the provided Python code are optimized for our own experimental setup and conditions. Further optimization may be needed for particular studies.

1. Open the file *pic_overlay_DFM_FM.py* with python text editor (for example, Atom, <https://atom.io>).
2. Change the “Ratio” to your own in line 8.
3. Change the “Rotate Angle” to your own in line 9.
4. Change the name of “Input file (DFM image and confocal image)” to your own file name in line 15 and 16, and the name of “output file” to your own file name in line 17.
5. Change “Y_shift” and “X_shift” to your own shift in lines 17.
6. Open the output file, evaluate the performance of this program. If not right, adjust the parameters in step 3 and 5, until the large spots in DFM and FM image colocalize with each other well.


(Optional) If you want to change the fake color of FM image, or change the color of DFM image into a fake color, do the following process.

7. Open the file *pic_overlay_DFM_FM2.py* with python text editor (for example, Atom, <https://atom.io>).
8. Repeat steps 2-5. (Be aware that the location of “Ratio”, “Rotate Angle”, “Input file”, “Output file” “Y_shift” and “X_shift” have been changed).
9. Change the color to your required fake color, e.g. r represents red in FM image, rG represents

red channel in DFM image.

2.3 Analysis of aggregation states of SNAs with colorimetry-based classification algorithm (Figure H1)

1. Put two py files, *darkfield_analysis_protocols.py* and *read_pixel_protocols.py*, in the same file folder.
2. Open the file *darkfield_analysis_protocols.py* with python text editor (for example, Atom, <https://atom.io>).
3. Change the “input file”, “output file”, “name rule”, “size of output image”, “the color, shape and size of output spot”, “name of output image” and “name of output file” in line 74, 75, 83, 97, 126, 144 and 145 to your own, respectively. (In order to find these lines easily, we make notes in the responding lines, e.g. *# input file* in line 74). (The location of lines was marked by red square, and the “#” were marked by blue square).
4. Change the range of x and y in line 64. The number represent the size of intensity spot which may be SNAs.
5. Open the file *read_pixel_protocols.py* with python text editor (for example, Atom, <https://atom.io>).
6. Change the “H threshold” and “S threshold”, “I threshold” in line 38, 106, and 119 to your own, respectively. (In order to find these lines easily, we make notes in the responding lines, e.g. *# the H threshold* in line 38).
7. Change the boundary criteria to your own in function reference. (In order to find the function easily, we make a note above the function, *“““the boundary of domains””””, which marked by orange square*).
8. Save this edit in file *read_pixel_protocols.py*.
9. Run the file *darkfield_analysis_protocols.py*.



```
72
73
74 target_path = '/Users/liumengmeng/Desktop/darkfield/dna_2h'# input file
75 save_path = '/Users/liumengmeng/Desktop/darkfield/dna_2h'output file
76 filelist = os.listdir(target_path)
```

Figure H1. Schematic of the Atom interface. Here, major information was highlighted. The location of lines was marked by red square, “#” was marked by blue square and “““ “””” was marked by orange square.

Part 3: Data example for analyzing classification of intracellular

SNAs

3.1 Data examples

Here we describe the procedure by taking DFM image in **Fig.7** as the example.

The INPUT DFM image is provided in the folder “test_data/dna2hours”, with a file name “20.jpg”, and the expected OUTPUT image is provided in the same with a file name “expect_result.jpg”.

! CAUTION The parameters we applied in this example is shown in Table S1. The parameter setting procedure was appended in Supplementary Information. We use a mac to run our programs.

1. Open the file *darkfield_analysis.py* in “python_scripts” file folder with python text editor (for example, Atom, <https://atom.io>).
2. Change the name of “input file” and “output file” into “test_data/dna2hours” in line 74,75.

```
74 target_path = '/Users/liumengmeng/Desktop/darkfield/'# input file
75 save_path = '/Users/liumengmeng/Desktop/darkfield/'#output file
```

Figure H2. Code lines 74-75

3. Change the “name rule” of image to “d+.jpg” in line 83.
4. Change the “size of output image” to “size [1]/50/1.54, size [0]/50/1.54” in line 97.

! CAUTION We have tried for many times and found that these values match the size of original images best.

5. In line 125, 129, 133, and 138, we determine the markerfacecolor, markersize and markershape of a single particle, small cluster, and large cluster as blue circle (10 pixels), green square (10 pixels) and red triangle (10 pixels), respectively.

```
124 if (Pointcolor_below[i] == 'green'):
125     plt.plot(Intensity[i][1],Intensity[i][0], 'o', markerfacecolor = 'mediumblue',\
126             markersize = 10, markeredgecolor = 'mediumblue')# the color, shape and size of output spot
127     list_green.append(Intensity[i])
128 elif (Pointcolor_below[i] == 'yellow'):
129     plt.plot(Intensity[i][1],Intensity[i][0], 's', markerfacecolor = 'forestgreen',\
130             markersize = 10, markeredgecolor = 'forestgreen')
131     list_yellow.append(Intensity[i])
132 elif (Pointcolor_below[i] == 'brightyellow'):
133     plt.plot(Intensity[i][1],Intensity[i][0], '<', markerfacecolor = 'firebrick',\
134             markersize = 10, markeredgecolor = 'firebrick')
135     list_brightyellow.append(Intensity[i])
136 elif (Pointcolor_below[i] == 'red'):
137     list_red.append(Intensity[i])
138     plt.plot(Intensity[i][1],Intensity[i][0], 'r<', markersize = 10,\
139             markeredgecolor = 'r')
```

Figure H3. The contents of lines 124-139

6. In line 144 and 145, we determine the “name of output image” and “name of output file” as “.png” and “.csv”.
7. In line 64, we determine the x and y range as $1 < x < 20$ & $1 < y < 20$.
8. Open the file *read_pixel_protocols.py* with python text editor in the same file folder.
9. We determine the “H threshold” and “S threshold”, “I threshold” in line 38, 106, and 119 as 120, 0.33 and 240.
10. Determine the boundary of domain in lines 9-27 according to Table S1 in Supplementary Information.
11. Press “command + s” in mac os.
- ! CAUTION** ctrl + shift + B in windows
12. Switch to *darkfield_analysis.py*, and press “command + i” in mac os.
13. Procedure

3.2 The procedures for optimizing parameter for your own experimental conditions.

<code>"darkfield_analysis_protocols.py"</code>
--

<code>"read_pixel_protocols.py"</code>
--

1. The region of interest (ROI) in the original dark-field image was extracted by ImageJ.
2. Load a selected dark-field image into Python program `"darkfield_analysis_protocols.py"`. The dark-field image was converted to gray mode in code line 18-19. In our case, the single particles with weak scattering intensity are observed as dark-green spots in DFM image, so we convert the original dark-field image of RGB mode to a green-channel image of gray mode.

```
18     image1 = cv.imread(fileitem,3)
19     image2 = image1[:, :, 1]
```

Figure H4. Code lines 18-19.

3. Set brightness threshold (T_L) in the green channel to depict most of SNAs' signal from background in code line 20-21 (`darkfield_analysis_protocols.py`, default 80, 160). Analyze at least 10 randomly selected SNAs of weak brightness, to obtain a proper value of (T_L) for recognizing image spots of weak signal from background. Analyze at least 10 randomly selected SNAs of strong brightness, to obtain a proper value of (T_U) to efficiently separate two neighboring image spots of strong signal. By typing `"imshow contours_below"` and `"imshow contours_upper"` below code line 21, two binary images are generated as temporary files. Based on these two images, the values of brightness threshold (T_L) and (T_U) can be evaluated.

! CAUTION Delete `"imshow contours_below"` and `"imshow contours_upper"` under code line 21, before continue with program running.

```
20     thershold_below = 80
21     thershold_upper = 160
```

Figure H5. Code lines 20-21.

4. Set the value of segment volume (V) threshold in code line 64 (`darkfield_analysis_protocols.py`, default $1 < \text{threshold} < 20$). The value of (V) is quickly determined by counting the number of pixels in non-target image spots. In our case, the observed pixel number of loose assembly of SNAs, dust or defects on glass slides are generally bigger than 20, and the ones of noises are smaller than 2, therefore, we set $1 < (V) < 20$ in either x or y axis.

```
64     if (1 <= len(B['x']) <= 20) & (1 <= len(B['y']) <= 20):
```

Figure H6. Code line 64

5. Set the value of Hue thresholdin (H) in code lines38-40 & 110-117 (*read_pixel_protocol.py*, default 120,4). Specifically, this threshold is sensitive to experimental design. In details, the spots of gold nanoparticles (50 nm in diameter) generally exhibit green-to-red color under DFM imaging, therefore, we use Hue threshold (blue color) to exclude the off-focus image spots. If other plasmonic particles are used, such as silver or copper exhibit blue color under DFM imaging, this threshold has to be excluded from the program.

```
38  if h > 120: # the H threshold
39      count_blue += 1
40      return count_blue
```

Figure H7. Code lines 38-40

```
110  if len(Intensity ) > 5:
111      if count_blue > 4:
112          pointcolor = 'blue point'
113          return pointcolor
114  if len(Intensity )<= 5:
115      if count_blue > 1:
116          pointcolor = 'blue point'
117          return pointcolor
```

Figure H8. Code lines 110-117

6. Set the value of saturation threshold (S) in code lines 119-124 (*read_pixel_protocol.py*). Specifically, this threshold is sensitive to experimental design. The values are dependent on cellular background under DFM imaging. For cellular organelles, the averaged value of saturation is obtained based on the correlation of HSB mode and RGB mode in the color map. In our case, (S) is commonly set as 0.33.

CRITICAL STEP For image spots of cellular organelles under DFM imaging, the averaged value of saturation is highly sensitive to cellular background. It is necessary to adjust (S) for each cellular experiment.

! CAUTION If the cellular background is too strong, set (S) to 0 in code line 119 and 122 to skip this step.

```
119  if ((Max_intensity < 220)&(saturation < 0.33)): # the S threshold
120      pointcolor = 'noise'
121      return pointcolor
122  elif (220 <= Max_intensity <= 240)&(saturation < 0.2):
123      pointcolor = 'rednoise'
124      return pointcolor
```

Figure H9. Code lines 119-124

7. Set the maximum intensity threshold (I) in code lines 106-107 (*read_pixel_protocol.py*). Randomly select at least 10 bright yellow spots, calculate intensity of each pixel in these bright yellow spots according to Equation1¹ to obtain the value of maximum intensity. The value of (I) is set according to the averaged value of these maximum intensity. In our case, the value of (I) is set

as 240.

$$I = R * 0.114 + G * 0.5876 + B * 0.299$$

Equation 1

! CAUTION The bright yellow spots representing large clusters can be directly recognized at this step, without need to continue with the program.

```
106         if (intensity <= 240): # the I threshold
107             saturace = max(colorhsb_s(loc[2],loc[1],loc[0]),saturace)
```

Figure H10. Code lines 106-107

8. Set the color domain boundaries in the CIE map in code lines 12-24 (*read_pixel_protocol.py*) based on the correlation of spectra profile and clustering states of your plasmonic nanoprobe. In our case, three domains are set in the CIE map, which are domain I for single particles (maximum scattering wavelength between 530 and 570 nm), domain II for small clusters (maximum scattering wavelength between 570 and 580 nm), and domain III for large clusters (maximum scattering wavelength between 580 and 620 nm).

```
12     x530 = 0.1547
13     y530 = 0.8059
14
15     x570 = 0.4441
16     y570 = 0.5547
17
18     x580 = 0.5125
19     y580 = 0.4866
20
21     slope_530 = (y530 - 0.33) / (x530 - 0.33)
22     slope_570 = (y570 - 0.33) / (x570 - 0.33)
23     slope_620 = (y580 - 0.33) / (x580 - 0.33)
24     return slope_530, slope_570, slope_620
--
```

Figure H11. Code lines 12-24

Reference:

1. Jing, C., et al., New Insights into Electrocatalysis Based on Plasmon Resonance for the Real-Time Monitoring of Catalytic Events on Single Gold Nanorods. *Anal Chem*, **86**, 5513-5518 (2014).