Short Manual of the Software

This is an public repository containing the Iterative Scanner Software introduced in *Link to Paper*

Dependencies

The Script is fully written in Python 3 (v 3.9), used of mostly using standard libraries and OpenCV (v 4.5) it requires the following packages in order to be ran:

- *PIL*
- *hyperspy*
- *numpy*
- *matplotlib*
- glob (python core library)
- *scipy*
- ast (python core library)
- csv (python core library)
- os (python core library)
- shutil (python core library)
- *openCV*

Installation and Using the program

Assuming Python3 and the above mentioned libraries are installed, the only installation step needed should be downloading the code provided by this repository without changing the local folder structure. The program can then be executed in the terminal by using:

```
cd *parent folder of direction of this folder*
python3 iterative_particle_analyzer.py
```

Guides and Explanation of Input Parameters

This program is a command line tool. Upon activation, it will ask if you want to read an existing input file. If yes is selected, it will ask you for its location. If no is selected, you enter "on the fly" mode, where the program asks you to provide all necessary input parameters. A sample input file is provided in the folder "Examples". You can download and edit the parameters to your needs in any text editor. The meaning of each input parameter is provided down below. In general, it is possible to analyze multiple images at once. However, all images will then use identical input parameters. Therefore, we recommend analyzing data with different imaging conditions separately from each other.

File Management

- Program Version: Miscellanous Information about the Version used.

- Home Directory: File Location of the Data set you wish to analyze. Make sure that your directory ends on a separator (such as / for Linux). This applies to all instances asking for a path.
- Relative Working Directory: Folder of a data Subset
- Name of Results Folder: Name of the folder, where each analysis results is saved.
- Pixel size and Pixel unit: Value and Unit of square pixel.
- File Format: Format of your data for analysis. Our code can read the image formats "png", "jpeg", "tif" through PIL as well as raw data formats "dm4" and "hdf5" via hyperspy. Other formats can still be possible but have not been tested.

Our code assumes the following organization:

```
Home Path/
    Data Set 1/
        image1.ext
        image2.ext
        ...
    Data Set 2/
        image1.ext
        image2.ext
        ....
    ....
```

In this example, choosing Data Set 1 as Relative working directory will analyze all images within Data Set 1, assuming 'ext' was chosen as File Format. The analysis result can be found under

```
Home Path/
    Results Folder/
        Data Set 1/
            *Analysis Results*
```

Pre-Processing Parameters

- Standard Deviation of Gaussian Blurring: Optional Gaussian Blurring of Images for Despiking. Default value: 2
- Data Bar Length: Some image format have a data bar the bottom, which belongs to the image. This parameter cuts the bottom rows from all images. Default value: 0
- Use Contrast inversion: Our particle analysis assumes bright particles on dark background by default. If the opposite is true, enable this option by entering 1 in "on the fly" mode or changing True to False in the input parameter file. Default: 0/True.
- Use Bottom/ Top Thresholding: You can apply optional image intensity thresholding. Selecting 1 will enable it, 0 will disable it. Default: 0/False

- Top Threshold / Bottom Threshold: If any thresholding option is selected, the program will ask you to set its value. You can either insert a float value or an integer. If a float value is entered, the program will analyze the gray values of each image and selects the specific quantile, 0.975 means the threshold is set to the 97.5 % quantile. If an integer is entered, the value is treated as a flat gray value number. It is important to note, that our Software normalizes the images to 8 bit integer before applying any other pre-processing steps, as this format is required by OpenCV. Therefore you have to select the value with this in mind. If the use of a specific threshold was not selected previously, the program will automatically set the values to 0.0 and 1.0 respectively.

The code executes its pre-processing in the following order:

```
Loading image
Save a Backup Copy of the Original Image
Cutting Data Bar (optional)
Gaussian Blurring (Optional)
Conversion to 8-bit Integer (Maximum of current image: 255, Minimum of current image: 0)
Thresholding (Optional)
```

Detection Parameters

Our code uses the following detection Parameters:

- Canny Edge Detection Threshold p_1: Upper Gradient Limit for Edge Detection. Necessary for Circular Hough Transform
- Max Radius, Min Radius, Radius Interval size: These parameters control the iteration over different radius intervals. We use monotonous radius intervals and start from largest to smallest radius (example: 50 to 48, 48 to 46, 46 to 44, ...). Choosing an interval size equal to or larger than the difference between Max Radius and Min Radius effectively disables radius iteration
- Particle Overlap Factor P: This is a proportionality factor that controls how much the particles are allowed to overlap by determining the minimum distance between 2 neighboring particles for each sub-interval. The minimum neighbor distance for each sub-interval is calculated by $\Delta d_i = P \cdot r^i_{min}$. Default: 1.5
- G , a , b , c, d : Function Parameters of the logistic scaling function $S(r)$ function (Eq. 2 in Paper). Default Values: $G = 3$, $a = 1$, $b = 2$, $c = 0.1$, $d = 12$. The scaling function $S(r)$ can be disabled by choosing large values of $d \to \infty$.
- Linear Voting Threshold Factors c_1, c_2 : These factors are the proportionality factors between sub-interval voting threshold V_i and the effective particle perimeter $\frac{2\pi r^i_{min}}{S(r)}$ for each interval. Default: $c_1 = 0.5$, $c_2 = 0.15$.
- use intensity criteria (bool): Use of the smart intensity criteria introduced in our paper. They are enabled by default, but can be disabled by setting

this variable to False.
- Particle Intensity Threshold: Gray Value Threshold used by our smart intensity criteria (8 bit gray, float allowed). If Intensity criteria are disabled, this value is automatically set to 0.

About voting threshold scaling We have talked about it in our paper but can provide more information about our thought process regarding the choices here. Without any consideration of resolution effects. The contour of a circle should be proportional to its perimeter in the form of $V = const \cdot 2\pi r$. In practice however, poor resolution affect the perceived circularity of the image object. At low resolution, only the dominant circular shape can be resolved. Minor deviations remain unseen. This results in the particle to be perceived very circular. At high resolution, these details are resolved and the perceived circularity should therefore be reduced. Under these considerations, we wanted a non-linear increase between V and r with a decreasing but still positive slope. The easiest way to realize such a relationship was to use a modified equation in the form of $V = const\frac{2\pi r}{S(r)}$, with $S(r)$ as a positive function with monotonous growth. A function that fulfills these criteria is the logistic function

$$S(r) = \frac{G}{a + b\exp(-c(r-d))}$$

With $a = 1$, $b = G - a$, the function is capped at G towards $r \to \infty$, whereas $S(d) = 1$. For $r < d$, we enforced $S(r) = 1$. This scaling function is continuous, monotonous, non-negative and has no problems with "exploding at high values", therefore fulfilling every desirable property for a stable scaling function. Furthermore, the parameters of the function can be easily adapted, G stretches the function in the y-axis, d shifts the function on the x-axis, and c streches the function on the x-axis around its effective center d. Changing $a = 1$, $b = G - a$ is not recommended as this results in a discontinous scaling function at $r = d$.

Furthermore, our routine tries to consider additional effects on the circularity of particles unrelated to resolution (e.g. growth related effects). For this reason, we wanted to allow additional variation in V for each sub-interval. This is realized by variation of the proportionality constant between V and r. The parameter c_1 is therefore a starting guess, whereas c_2 determines a bottom threshold in V_{min}. However, we also allowed for an early exit of the variation in V, if $V < 2 \cdot V_{min}$ does not result in any positive detection result.

Output files

Our Program provides the following output files:

- Canny Edge Detection after each iteration step in each image
- Binary masks after each iteration step in each image

- A particle overlay for each - the detected particles are drawn onto the original image as red circles. Also, the number of particles in the image is provided in the title of the image. Therefore this program is also fit to "just count" particles for you with ease.
- A diameter histogram, showing the distribution of particle diameters across all analyzed images
- A CSV file listing the position and radius of each detected circle and which image it was found in.
- A text file saving all input parameters. This file is written as compatible input parameter file for the programs auto-parameter-reading function.

Fix for potential Issue

The code was written in Linux. When you concat multiple path strings, our software assumes / as separator between different sub-folders by default. if you have a different separator (like for Windows), open the script iterative_scanner.py in an editor and change the / in line 30 to the separator of your local system.

License

Citation

Please cite our software by citing *Link to paper*, if you are using it in any publication.