SPyAD (Streamlined Python Automatic Detection) for Nanoparticle Detection and Unit Conversion in CANS and CCEM pictures

Rory Gao, 11 August 2020

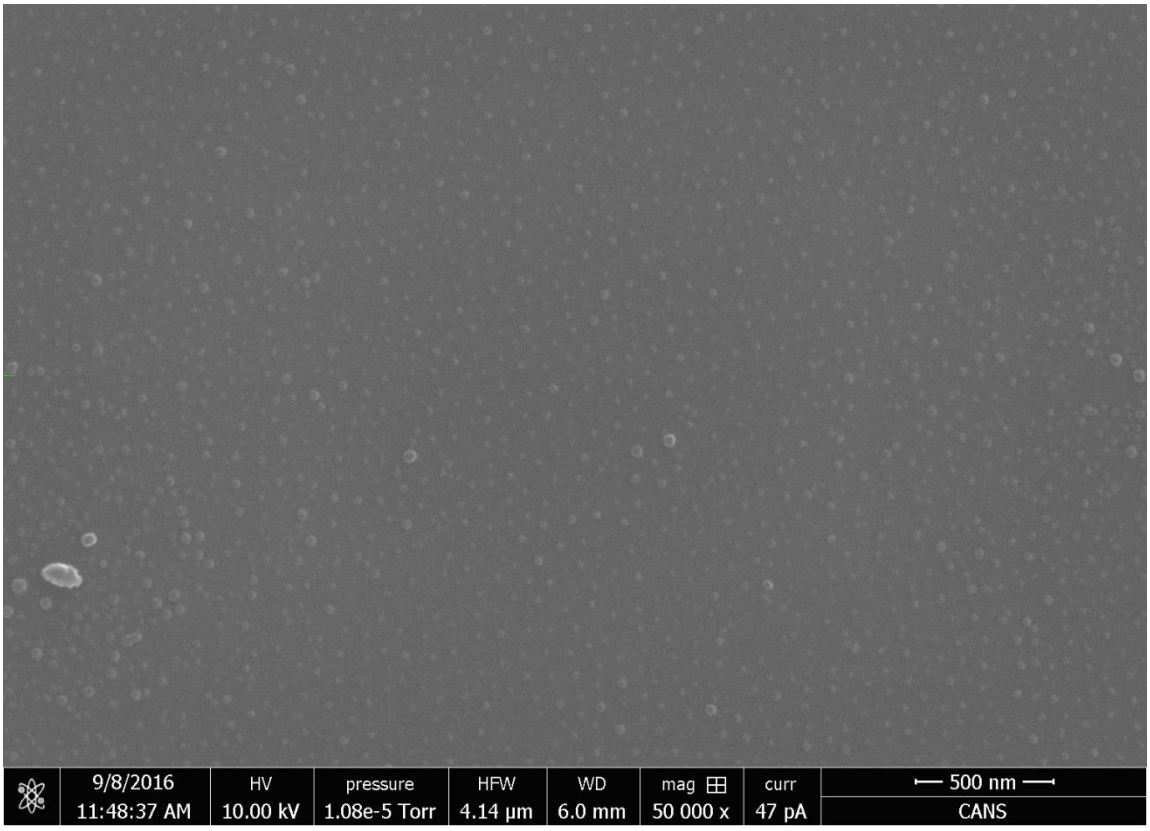

Figure 1. Sample CANS picture

Image Particle Coordinate extraction is a crucial step in the overall goal of quantifying order in nanoparticle distributions. Any particle detection program must able to detect nanoparticles in a picture to a reasonable accuracy and extract their x/y coordinates for further analysis in disLocate. Many tools already exist to develop images in a simple process: Take an image as input, detect all nanoparticles (or at least attempt to at the best of its ability), and output the coordinates of each particle centre. This newly developed program aims to enhance our current suite of tools, namely by allowing simplified bulk processing, command-line interfaces, and greater control over image processing parameters. It is important to note that SPyAD and existing tools (such as ImageJ) have their errors and inconsistencies – ImageJ may detect particles that SPyAD doesn't, and SPyAD may detect particles that ImageJ doesn't. Further explanation of program errors is in the Limitations section, but ultimately it should be recognized that SPyAD does not serve to replace ImageJ and instead is another powerful tool to speed up a previously tedious task.

Definitions:

ImageJ:

ImageJ is an image processing program with a built-in Graphical User Interface. Developed at the National Institutes of Health in 1997[1], it is currently the main program used with the Turak Research Group to handle difficult-to-process SEM images and AFM images.

Python:

Python is a high level, object-oriented programming language released in 1991. SPyAD is written completely in Python, and the latest version of Python needs to be installed to use SPyAD. It can be found here: https://www.python.org/downloads/

A brief outline of SPyAD operations:

SPyAD's operation works in two main steps. The input CANS/CCEMS image with nanoparticles is put through a processing pipeline, where a variety of filters, blurs, and adjustments are automatically applied to best highlight particles. Note that SPyAD operates on the basis that particles are discernible due to their higher apparent brightness compared to their background.

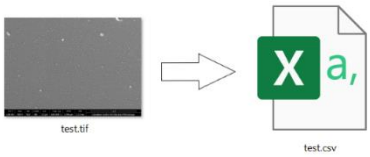

Figure 2. SPyAD Input and Output files

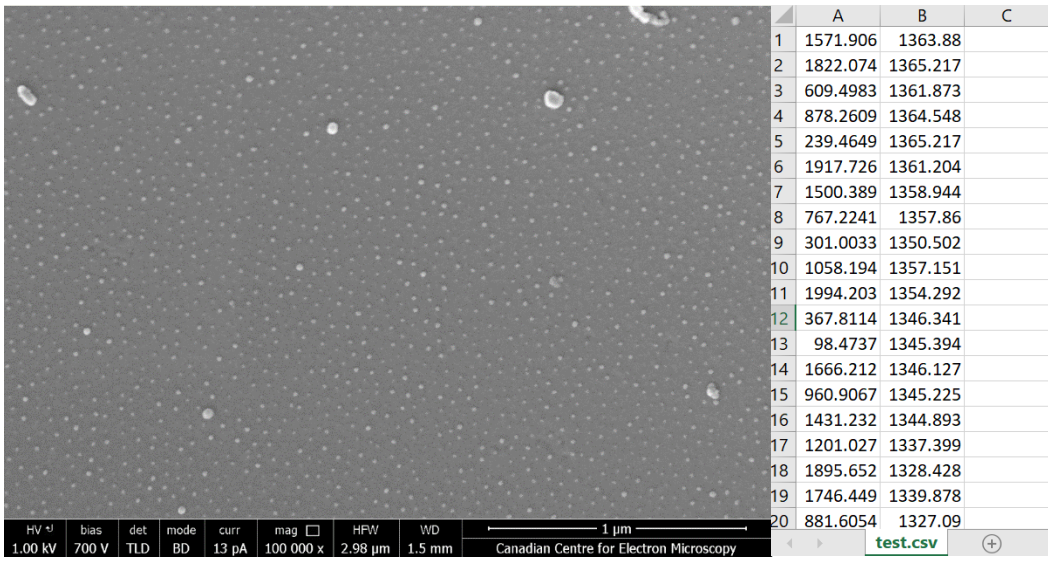

Figure 3. Preview of Input CCEM image and Output Coordinates

[1] (Schneider, Rasband and Eliceiri 2017)

Centre of Particle Convention:

Once the image has been processed to make particles "pop out", contours are then drawn around each potential particle. These contours are filtered by size, and particles smaller than the default size are treated as image noise and dropped. Each contour is then bounded by its minimum enclosing circle. (see https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=minenclosingcircle#minenclosingcircle for exact documentation) The coordinates of the centres of said circles are then saved in a .csv file.

Scale and Unit Detection:

A second feature SPyAD offers is automatic scale and unit detection with CANS and CCEM images. SPyAD reads the distance of the scale bar in pixels and parses the scale legend to convert pixel coordinates into nanometer distance units. This reduces the work to be done in disLocate later on.

Figure 4. Box Containing Scale and Unit in CANS and CCEM images

Setup:

SPyAD requires several dependencies, installed onto the local filesystem. These are:

1. Python - pip
2. OpenCV2
3. Tesseract OCR/ pytesseract
4. Numpy

Python – pip is a package manager for Python and is needed to install all the subsequent libraries.

OpenCV2 is a library of functions for the Python Language that allows computer vision operations to be performed on Nanoparticle Images.

Tesseract OCR is an open-source optical character recognition software. It is required by SPyAD's scale and unit detection functionality. **Pytesseract** allows python commands to interact with Tesseract OCR and needs to be installed separately.

Numpy is a python library used to support operations with large arrays and matrices. This is especially useful when working with images, as OpenCV treats an image as a matrix.

Installing Dependencies:

1. **Pip:** Pip is automatically installed in Python versions 2.7.9+ and 3.4+. If for some reason pip is not installed, on the command line, perform: (Windows)

   "**curl https://bootstrap.pypa.io/get-pip.py -o get pip. py**"

   ```
   >curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
   ```

   This downloads the pip package from an online repository. Then run:

   "**python get-pip.py**" to install it.

   ```
   >python get-pip.py
   ```

Verify that you have installed successfully by typing "**pip -V**".

If on Linux, installation is easier. Run:

"**sudo apt install python-pip**"

2. **OpenCV2:** On the command line, run

   "**pip install opencv-python**"

   ```
   >pip install opencv-python
   ```

3. **Tesseract OCR:** Download Tesseract Engine from:

   https://github.com/UB-Mannheim/tesseract/wiki

   Make sure that the correct version is installed depending on the host operating system. (32 bit vs 64 bit). Follow through with install instructions. NOTE: **make sure** that it is installed in C:/Program Files.

   If on Linux, simply run:

   "**sudo apt install tesseract-ocr**".

   After the Tesseract engine is installed (for Windows and Linux Users), run:

   "**pip install pytesseract**" to install pytesseract.

4. **Numpy:** Install with:

   "**pip install numpy**"

SPyAD command usage:

```
python SPyAD.py [-i/-b] subfolder/image name [-p] [-t] custom
threshold value [-c] custom contour size value
```

- [-b] Provide the name of subfolder with images to process in bulk
- [-i] Provide the name of image file to process. Mutually exclusive with [-b]
- [-p] Toggle Image – by – image preview
- [-t] Set custom Threshold value
- [-c] Set custom particle size for filtering

**Notes: Must navigate to the SPyAD directory to execute the program. The image must be from either CANS or CCEM.

SPyAD will also output two values, which will be useful for troubleshooting:

“Pixel to Distance Conversion factor”:

- By default, OpenCV gathers coordinates of particles in pixel units. The pixel to distance conversion factor (derived from unit and scale detection) is multiplied to the pixel coordinates to give distance values in nanometers.

“Particles Detected”:

- SPyAD displays a total count of all particles detected which were greater than the minimum particle size.

[-b] Processing Images in bulk:

Command:

```
python SPyAD.py -b foldername
```

Where:

"`foldername`" is the name of the folder with images to be processed. SPyAD will process every single image file and output individual .csv files with particle XY coordinates in the folder.

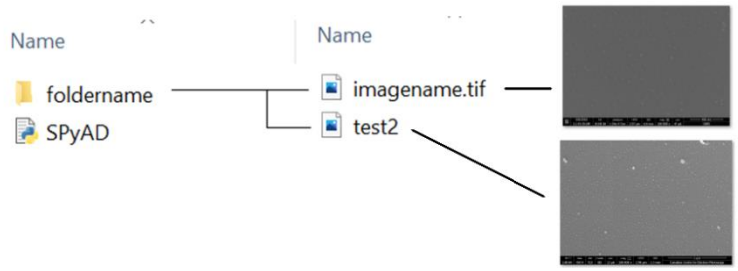

Figure 5. Example directory showing "foldername" with two image files to be processed.

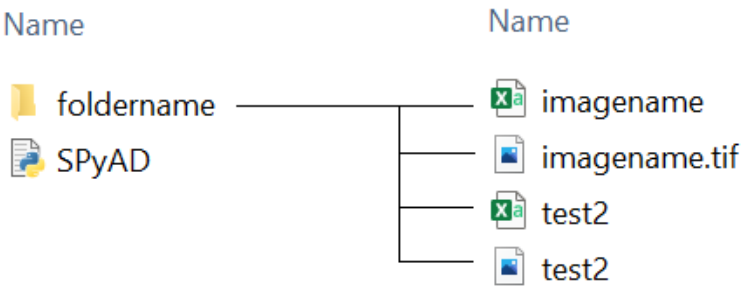

Figure 6. Example output after running SPyAD on previous two images in foldername.

[-i] Processing Single Image:

Command:

```
python SPyAD.py -i imagename
```

Where:

"`imagename`" is the name of the image to be processed. SPyAD will process the input image and output .csv file with the same name in the current working directory.

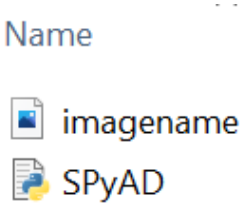

Figure 7. Example directory showing "imagename" as image to be processed.

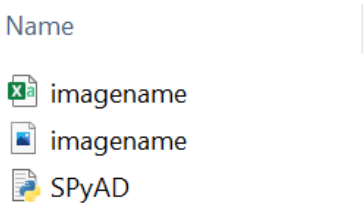

Figure 8. Example output, imagename.csv

[-p] Toggling Image Preview:

Command:

```
python SPyAD.py [-p] [-i/-b] imagename/foldername
```

Where:

`[-p]` must be called before `[-i]` or `[-b]`. Calling `[-p]` will display all processed images with all detected particles highlighted. For bulk processes, SPyAD will wait for a keypress (any key) before previewing the next image in the folder.

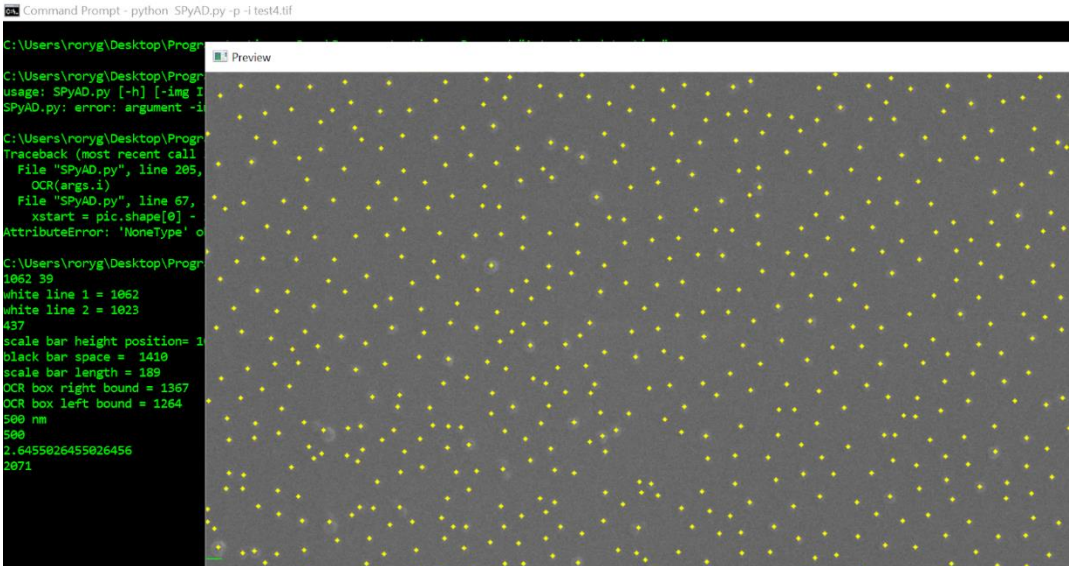

Figure 9. Image Preview highlighting centres of detected nanoparticles – SpyAD is awaiting key press to continue to next image.

[-t] Setting Custom Threshold Value:

Although SPyAD has been configured to work with the majority of the images we receive, there may be some cases where you will want to play around with the processing settings.

Command:

```
python SPyAD.py [-p] [-i/-b] imagename/foldername [-t] t-value
```

Where:

`[-p]` is optional and compatible with `[-t]`

"`t-value`" is the custom threshold setting value. The default value is 50, SPyAD subtracts this value from every pixel's intensity value before thresholding. Setting a higher value will spare less bright particles from being filtered out but will pick more noise in the final product, increasing the total count. Lowering the custom value will let only the brightest particles through the filter, decreasing the total count.

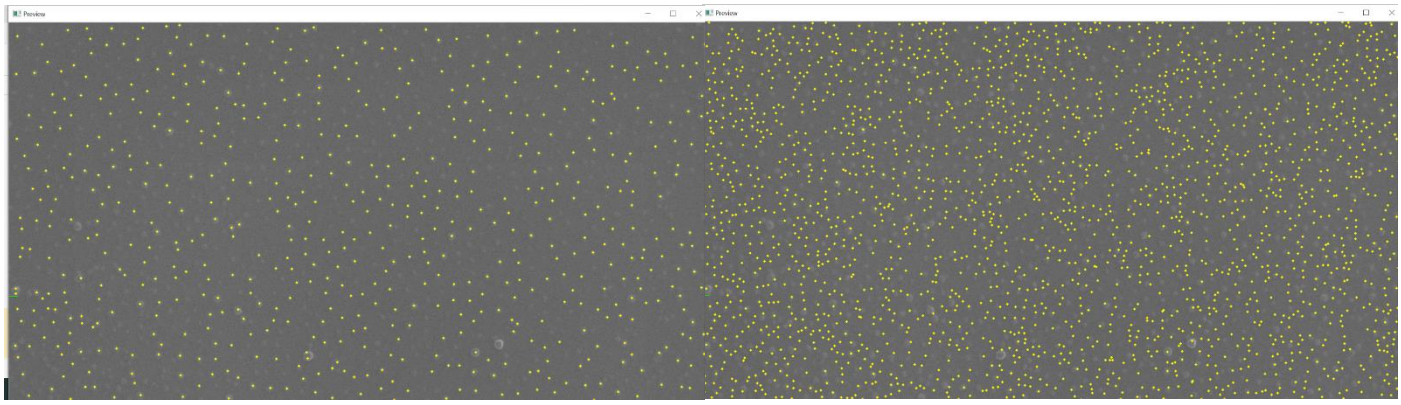

Figure 10. Side by side comparison of two different threshold values on the same input image. ***Left Threshold = -100, Right Threshold = 0****. Note that a lower value picks up more overall particles but may not be accurate as most of it is noise.*

[-c] Setting Custom Minimum Particle Size:

There may be cases where adjusting the threshold is necessary yet yields inaccurate results as lots of noise is let through. SPyAD has a "safety net" for this issue as it filters out all particles that do not meet a minimum size requirement. This assumes that unwanted noise is usually present as small, single-pixel dots and significantly smaller than actual particles. Some experimentation may be required to determine the optimal threshold and size values for an image.

Command:

```
python SPyAD.py [-p] [-i/-b] image/folder [-t] t-value [-c] c-value
```

Where:

`[-p]` is optional and compatible with `[-t]` and `[-c]`

`[-t]` is optional and compatible with `[-c]`

"`c-value`" is the custom particle size setting. The default value is 4 (pixels in area). SPyAD scans all detected particles for their area, and if a particle does not meet the minimum size requirement it is

dropped from the final count and its coordinates are not recorded in the .csv file. Increasing the c-value would increase the minimum size requirement and drop some particles that are smaller than the others, therefore decreasing the total count. Decreasing the c-value would do to the opposite and increase the final count.

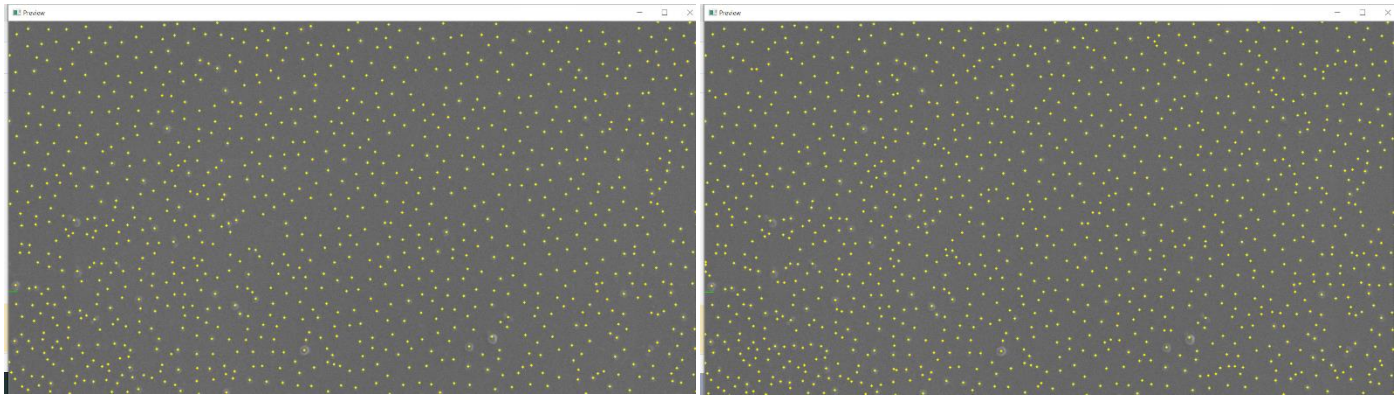

Figure 11. Side by side comparison of two different size requirements on the same input image. Left Value = 10 px, Right Value = 1 px. Note that the differences are usually miniscule but may be useful in conjunction with changing threshold value.

Limitations of SPyAD:

The SEM images presented in this report are all examples of very clear, easy to process images. Particles in those images are all very discernible and SPyAD has no problem processing those. However, not all images received from CANS or CCEM are as clear and well-defined as those, so SPyAD will inevitably encounter difficulties processing some. This section mentions the most common types of images that **do not** work with SPyAD and should be analyzed with another tool, namely ImageJ.

- Non-CANS or CCEM images
- Particles with high aspect ratios/ irregular shapes (see Figure 12.)
    - Coordinates of the centre of a particle may not be accurate, coordinates may even be outside of particle for unusual particle shapes.
- Images with inconsistent backgrounds (see Figure 13.)
- Very low contrast/ very blurry images (see Figure 14.)
    - As SPyAD works on the assumption that particles will be brighter than their background, images with very blurry particle boundaries, and/or boundaries not decided by brightness cannot be analyzed.

It should be noted that this list is not exhaustive and only representative of the most common image problems.

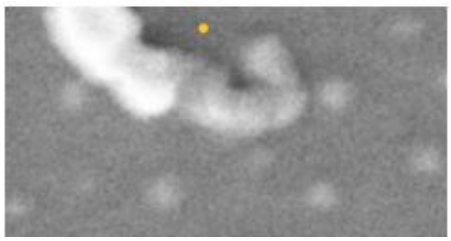

Figure 12. Particle with high aspect ratio and irregular shape in SEM image. Notice how detected centre is outside of particle.

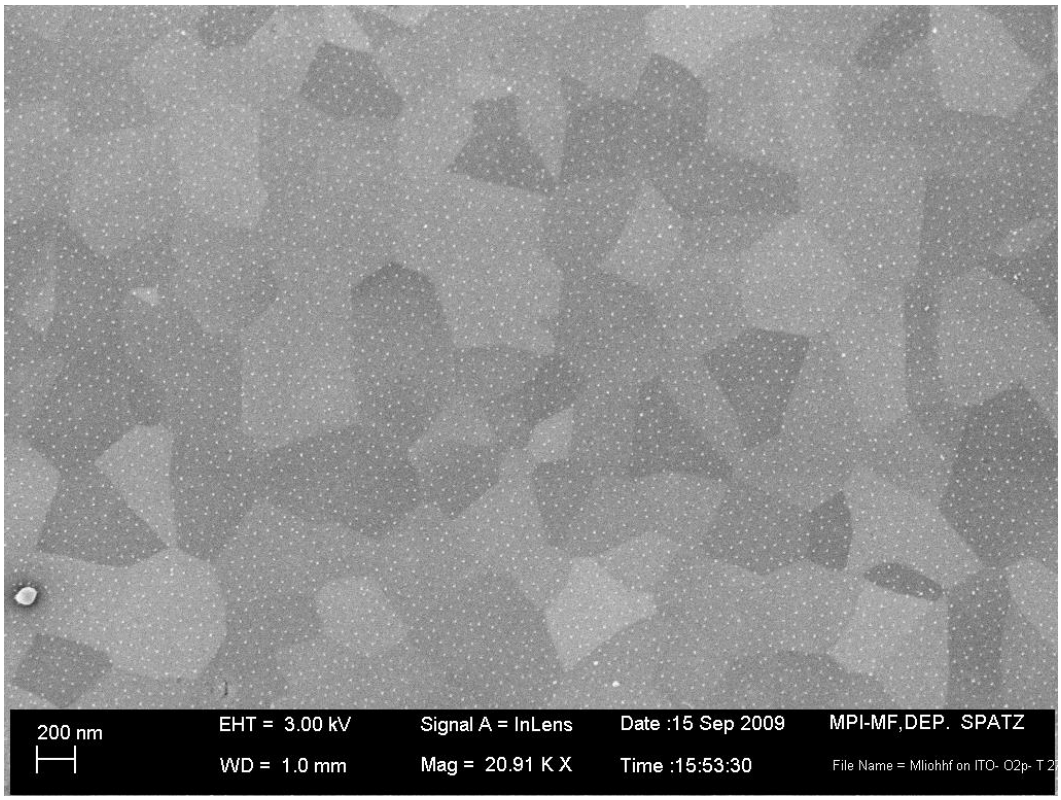

Figure 13. Non-CANS/CCEM image with an inconsistent background. This background brightness variation interferes with SPyAD's particle detection. Default to ImageJ for these images.

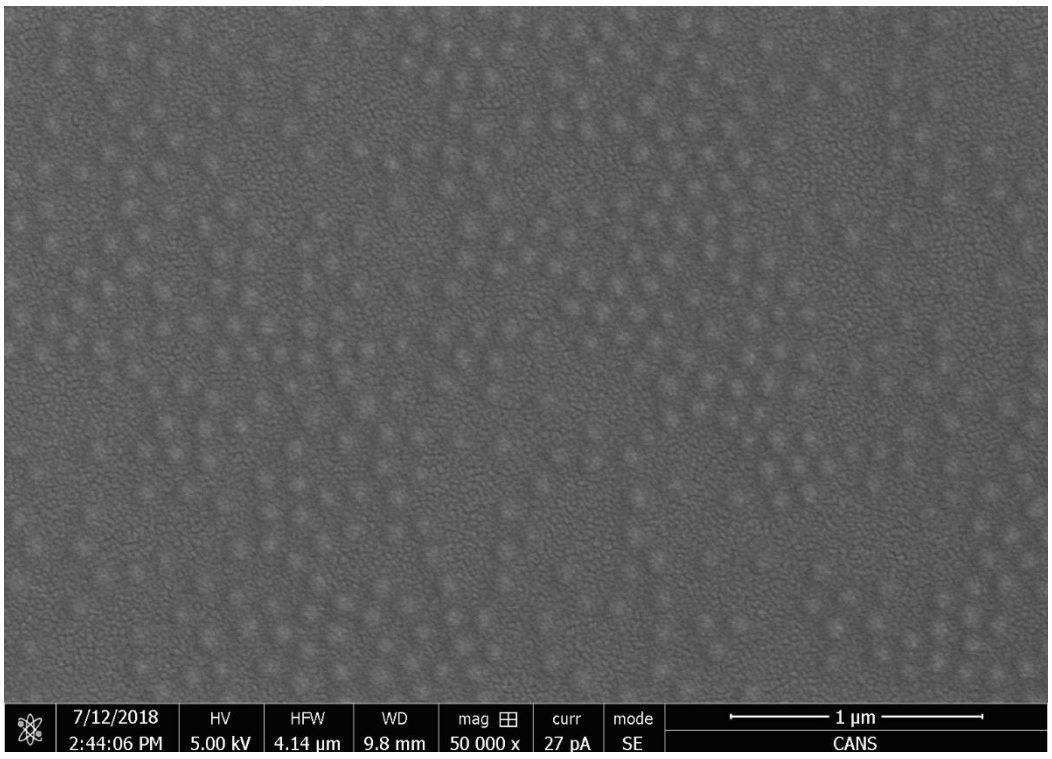

Figure 14. Low contrast image with unusual particle demarcation. Particles in most SEM images are discernible due to their higher relative brightness to the background, but particles in this image look more like "bumps" due to their boundary gradient.

Conclusion:

SPyAD was developed to hasten SEM image processing, meaning extra attention was paid to simplifying user input to streamline bulk operations. The trade-off is that SPyAD does not have as many functionalities as other programs. However, SPyAD's features such as command-line interface and support for bulk processing puts it into a quite different category than graphical tools such as ImageJ. Therefore, further development for this program should be focused on what it was designed to do: Allow for fast, intuitive processing of lots of easy-to-analyze SEM images, with potential expansion into harder-to-analyze pictures. As well, the overall goal should not be overlooked – gathering coordinates from image files has never been an easy task, and any simplification is a step towards improvement. From a programmer's point-of-view, the end goal of automating image processing would be a program that could take any nanoparticle image as input and automatically perform spatial statistical analysis (for example disLocate operations) on it. That may require more advanced concepts such as machine learning to handle the large variance in image quality, type, and brightness.

Patches and updates to SPyAD may or may not be added in the future*, depending on the program's effectiveness in the field. Possible areas of improvement include: Adding the ability for users to manually create/delete points, support for non-CANS/CCEM images, and automatic image sorting. There is still a lot of room for improvement, and SPyAD may or may not be the first step in the answer to full automation.

References

Schneider, Caroline A., Wayne S. Rasband, and Kevin W. Eliceiri. 2017. "NIH Image to ImageJ: 25 years of Image Analysis." *nature methods* 9(7): 671-675. doi:10.1038/nmeth.2089.

*Correspondence should be made to *rory.gao@mail.utoronto.ca* for any urgent issues with SPyAD.