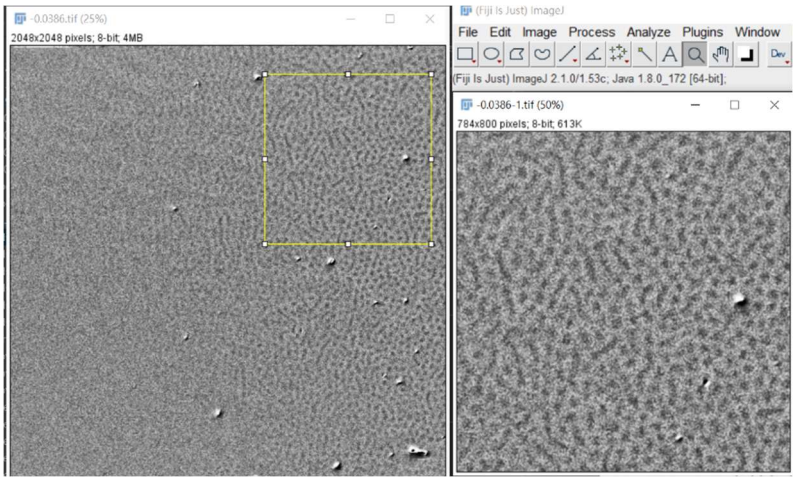
Comprehensive report on MOKE image processing

Madeline Chan

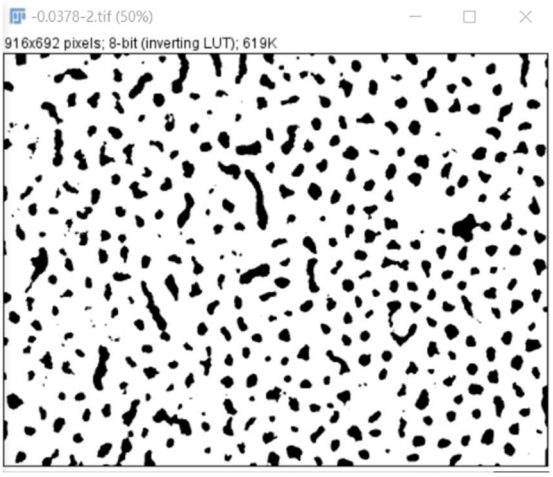
1. Image measurement

1.1 Measure Skyrmion area

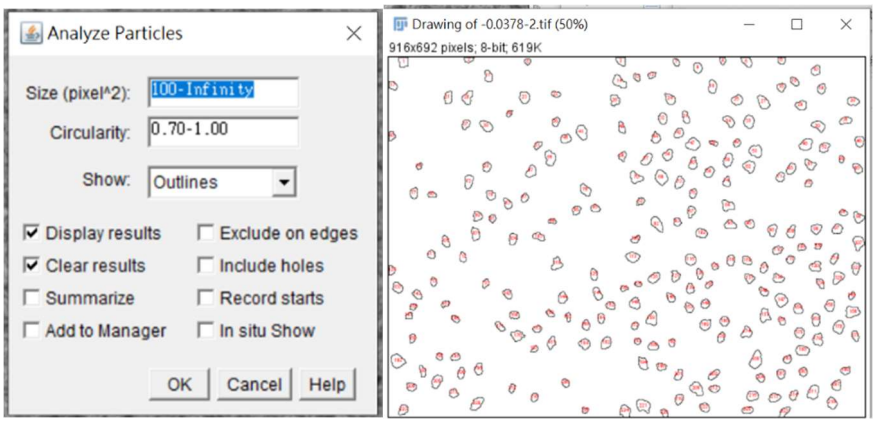
Step1. Use Area selection tools to select the region of interest. Choose *Image -> Crop* to cut the region of interest



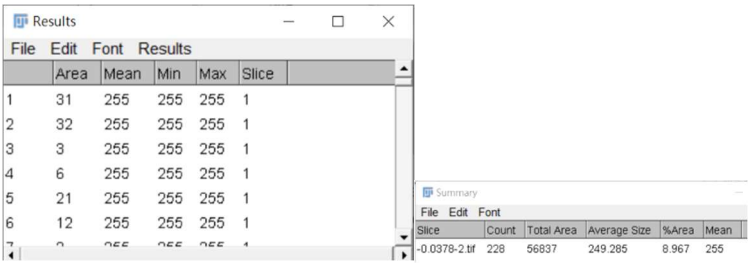
Step2. Choose *Image -> Adjust -> Threshold*. Take threshold to filter the skyrmion



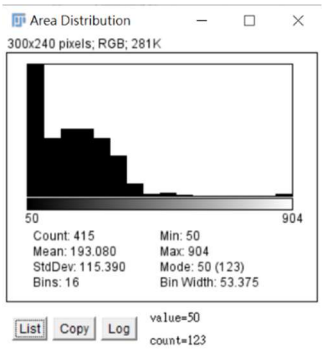
Step 3. Choose *Analyze -> Analyze Particles*. It is a tool to calculate skyrmion size. Set suitable parameters to filter out unwanted shapes. (the red labeling are the index)



Step 4. Check the "*Display results*" and "*Summarize*" box. The "Display results" box shows each skyrmion size (numbering index with respect to the red labeling) and "Summarize" box calculates the average size (unit: pixel^2). Results can be exported into csv files by choosing *File -> Save As*.

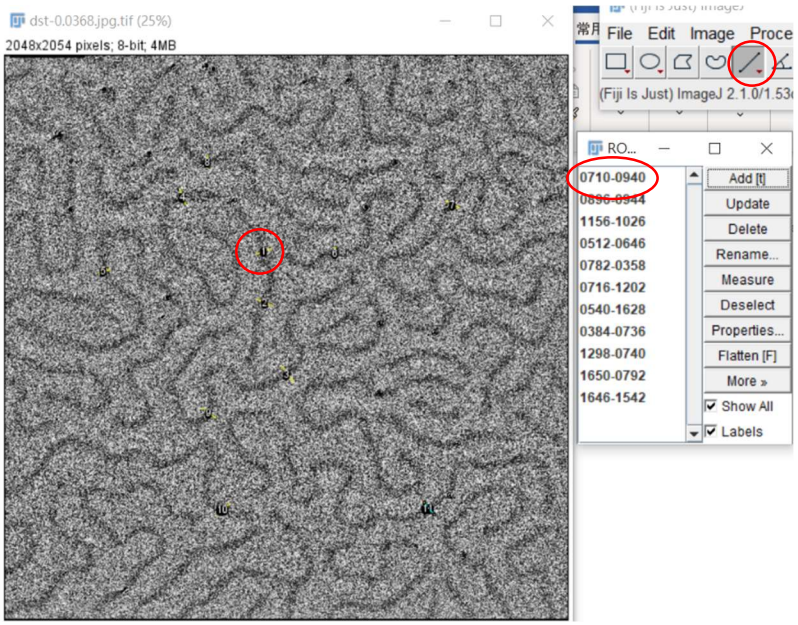


Step 5. Choose *Results -> Distribution* to obtain the area distribution histogram



1.2 Measure Domain wall width

Step 1. Open *Analyze -> Tools -> ROI Manager*. Use the segment tool to highlight the wall and press the "Add" button to add the line to ROI Manager. Check the "Show All" box if you want to see all added lines.



Step 2. Press "*Measure*" button to calculate each thickness and average. Results can be exported into csv files by choosing *File -> Save As*.

Results

File Edit Font Results

	Label	StdDev	Angle	Slice	MinThr	MaxThr	Length
1		60.330	19.654	1	0	255	59.464
2		72.394	-23.199	1	0	255	60.926
3		60.839	-51.843	1	0	255	71.218
4		61.572	-70.346	1	0	255	59.464
5		68.817	15.255	1	0	255	45.607
6		65.423	-84.289	1	0	255	40.200
7		65.225	-26.565	1	0	255	53.666
8		82.249	-90.000	1	0	255	48.000
9		67.745	-36.870	1	0	255	60.000
10		61.375	42.510	1	0	255	65.115
11		63.636	-50.711	1	0	255	56.851
12	Mean	66.328	-32.400	1	0	255	56.410
13	SD	6.488	43.468	0	0	0	8.972
14	Min	60.330	-90.000	1	0	255	40.200
15	Max	82.249	42.510	1	0	255	71.218

For more ImageJ details, please visit:

https://imagej.nih.gov/ij/.

More documentation on ImageJ Basic:

https://imagej.nih.gov/ij/docs/pdfs/ImageJ.pdf

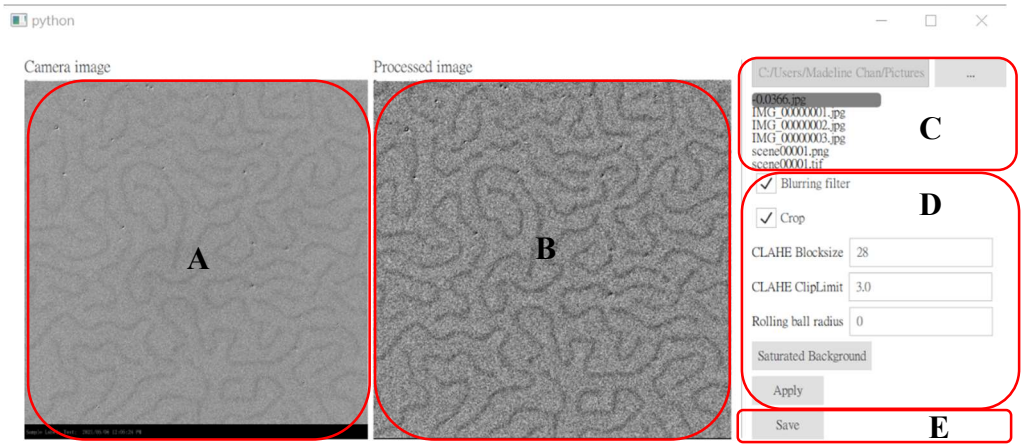
More documentation on ImageJ Analyze:

https://imagej.nih.gov/ij/docs/menus/analyze.html

2. GUI interface

2.1 GUI interface to do image processing

2.1.1 Components



A: raw input image, video

B: processed image, video

C: File directory

- Choose a directory
- A list to view images store in the directory
- Two modes: manual image selection by clicking image in the list
- Automatic processing when new image is created

D: image operations

- Blurring filter (3x3 averaging)
- Crop image (remove the black label below)
- CLAHE specification (block size and clip limit)
- Rolling ball radius
- Saturated background subtraction

E: saving

2.2 Image operations

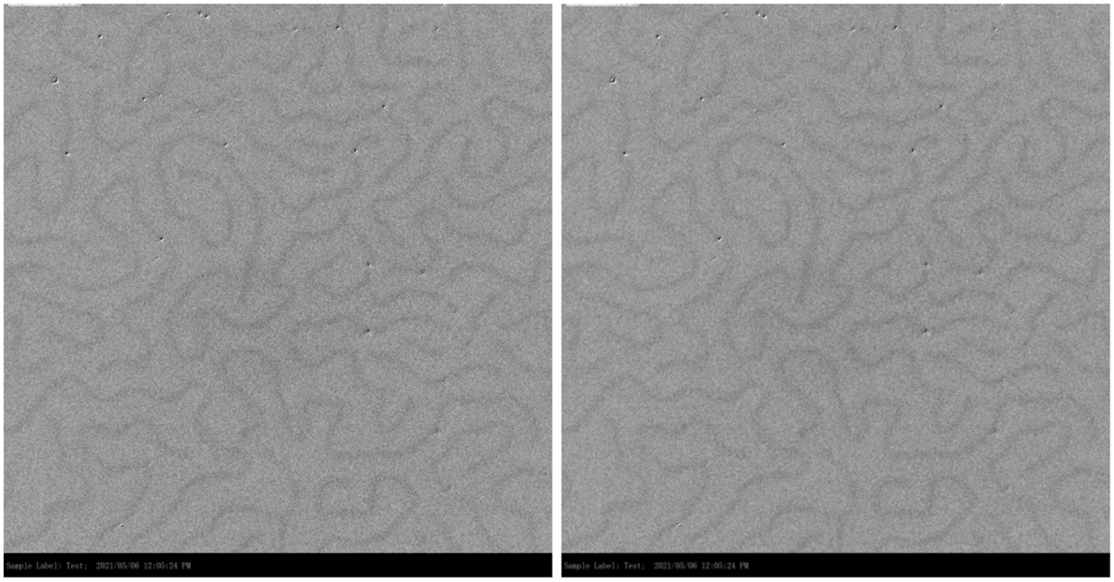
1. Blurring

A blurring filter is used to smoothing and denoising. When a suitable blurring filter is applied, it can remove high frequency noise from the image without blurring the important features in the image.
Averaging filter will be a suitable filter in this case. It is a 3x3

normalized box filter:

$$K = \frac{1}{9}\begin{matrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{matrix}$$

Example on 2021 5.6 skyrmion +to-/-0.0366.jpg:

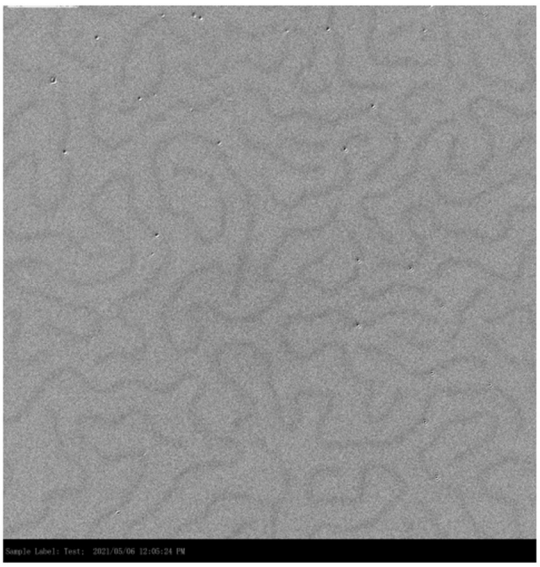


Raw image Blurred image

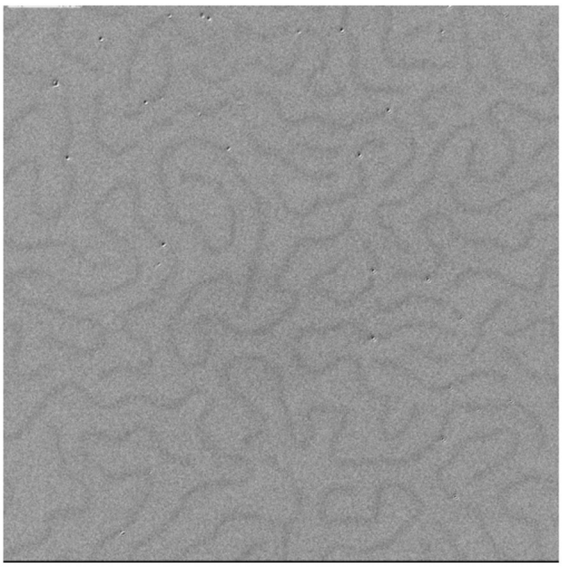
2. Crop image

This cropping operation is specified for removing the black label. The raw image size is 2138:2048. Size after cropping is 2054:2048.

Example on 2021 5.6 skyrmion +to-/-0.0366.jpg:



Raw image



Cropped image

3. CLAHE (Contrasted limited adaptive histogram equalization)

It is an operation to improve contrast in images. It originates from histogram equalization.
Histogram equalization is a method that evenly distribute the intensity on full range.
First step, the pdf (possibility density function) is defined as below.

$$p = \frac{number\ of\ pixels\ with\ intensity\ n}{total\ number\ of\ pixels} \qquad n = 0, 1, \dots, L-1$$

Then, the cdf(cumulative density function) is defined as below.

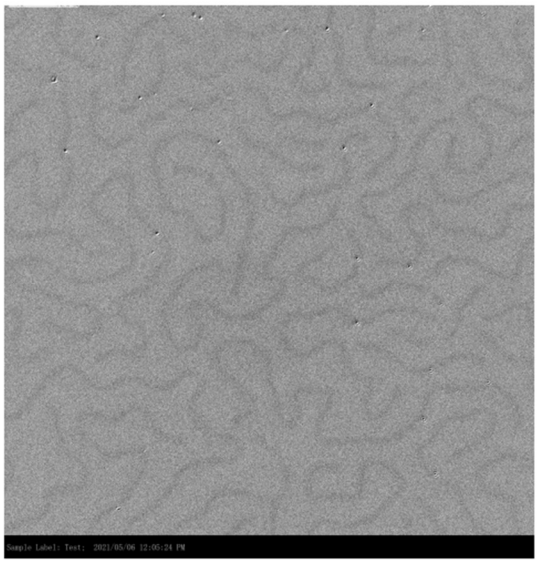
$$c = \sum_{j}^{i} p(x)$$

The cdf is then round down to the nearest integer.
And it will transform to get a new image to map the probability back to pixel values, k = L-1.

CLAHE is different from normal histogram equalization in two ways. It is an adaptive approach that it computes several histograms, each corresponding to a distinct section of the image. Therefore, it can improve the local contrast. However, it will overamplify noise in relatively homogeneous region. Therefore, clipping value is needed to control the contrast enhancement.

Therefore, users must use suitable block size and clipping value to achieve a nicely processed image. Block size of 28 and clipping value of 3.0 is recommended.

Example on 2021 5.6 skyrmion +to-/-0.0366.jpg:



Raw image

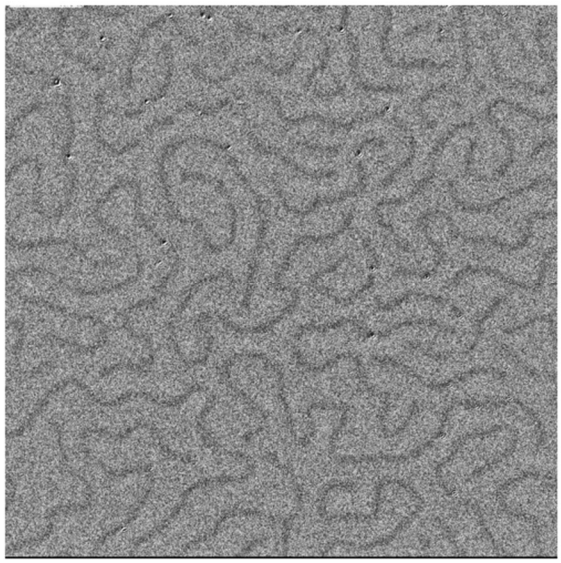


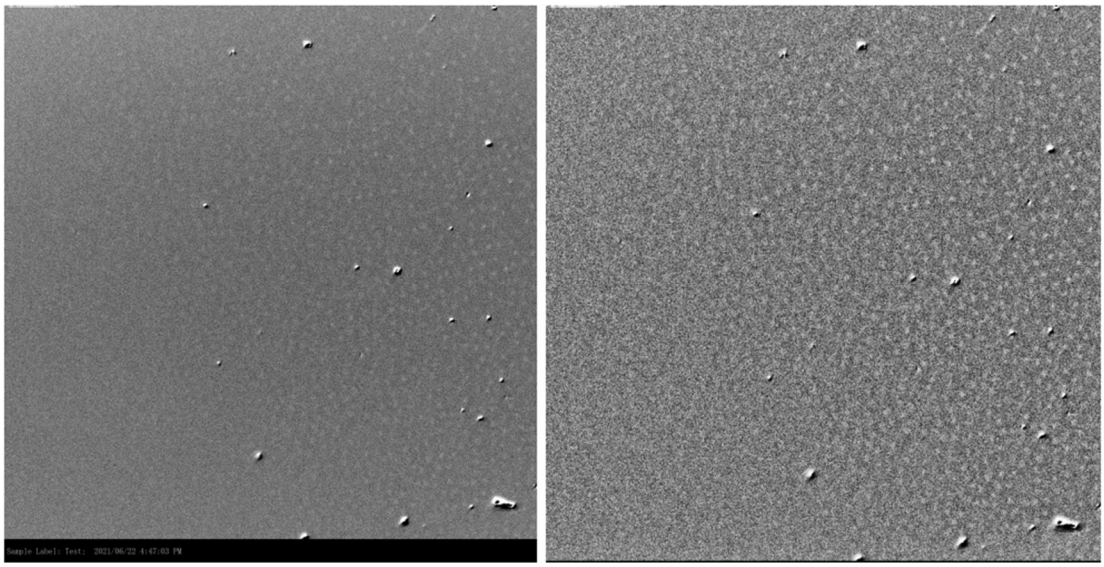
Image after CLAHE,
block size = 28, clipping = 3.0

4. Rolling ball algorithm

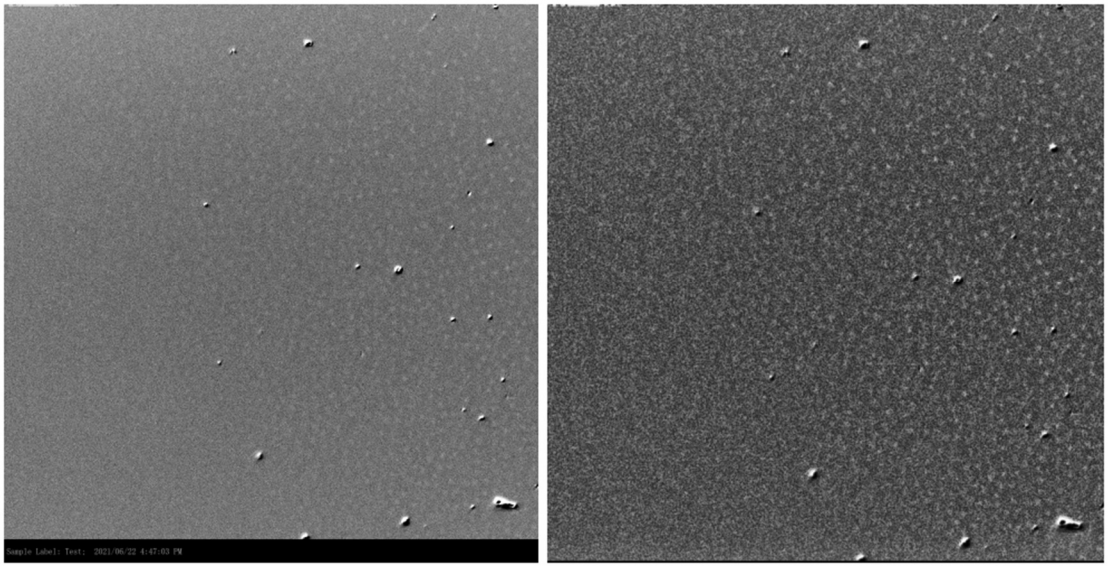
It is a method to correct for uneven illuminated background. It is useful to process image that includes skyrmion that it can distinguish the skyrmion from the background, making the skyrmion seemingly more outstanding. There is usually third dimension (surface height) for a 2D grayscale image which is intensity. The algorithm is intuitive. To get the intensity of the background, image submerging sphere under the surface at the desired position. Once it is completely covered by the blocks, the ball determines the intensity of the background at the position. Then we can roll this ball around below the surface to get the background values for the entire image. Therefore, the radius of the rolling ball should fit the size of the skyrmion.

✧ However, the computation of this algorithm is slow. You might wait a few seconds.

Example on Series of skyrmion figures/-0.0441.jpg:



Before rolling ball algorithm

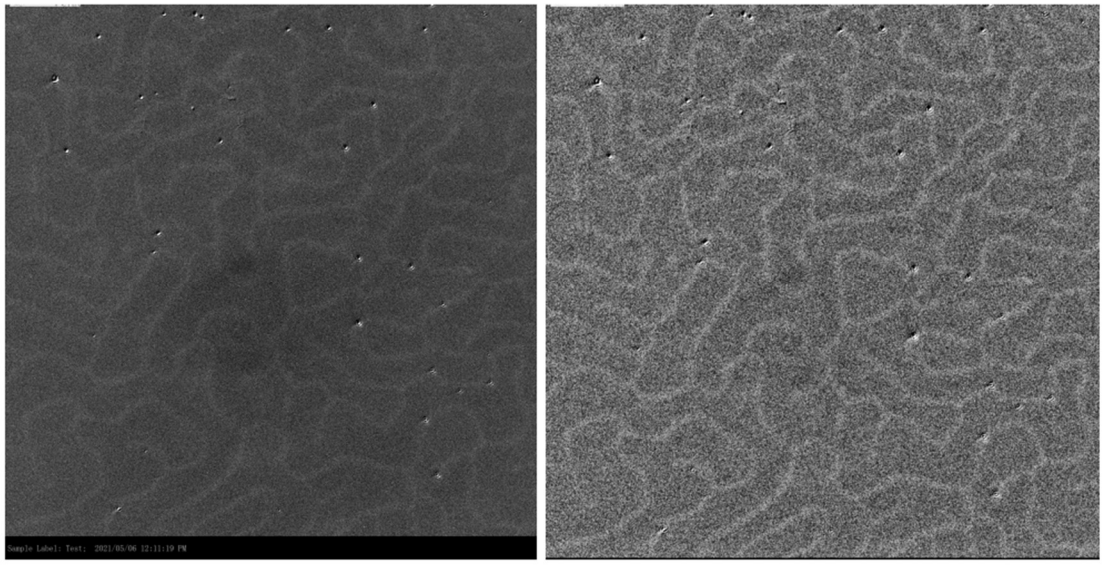


After rolling ball algorithm, radius = 20

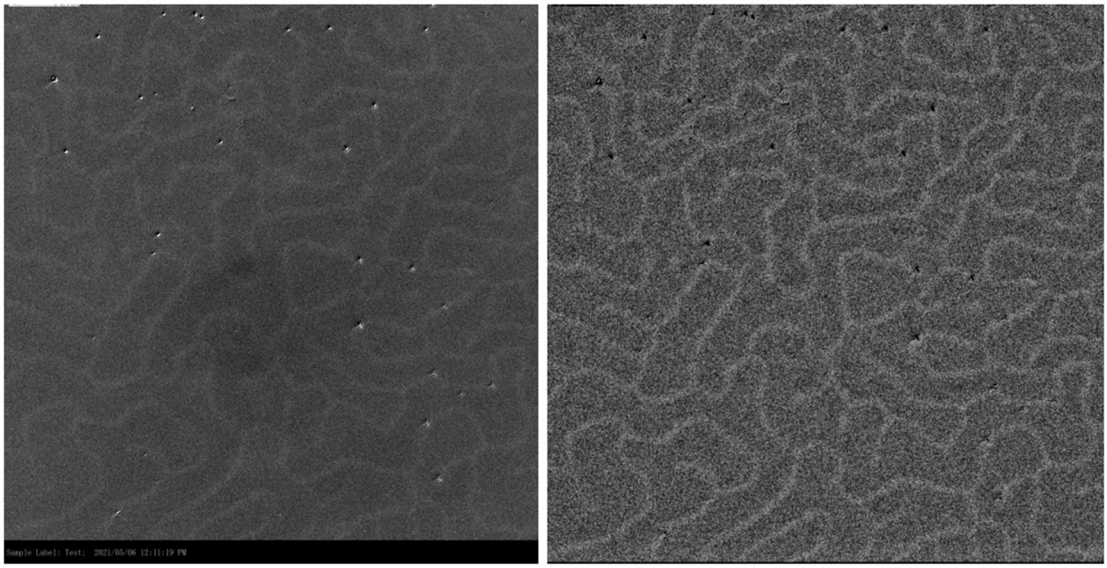
5. Saturated state subtraction

It is observed that there is some common shadow or noise appears in the same series of images. A way to remove these untruth signals is to subtract the image from the saturated state, which is usually the darkest image. The background intensity is usually shifting with different magnetic field strength. The subtraction also unifies the background intensity.

Example on 2021 5.6 skyrmion +to-/-0.0398.jpg:



Before subtraction

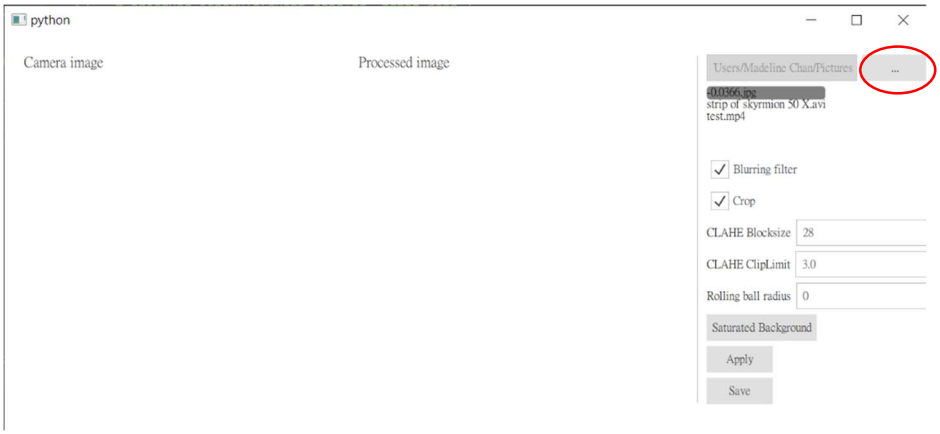


After subtraction

Image used as saturated background:
2021 5.6 skyrmion +to-/-0.0500.jpg

Steps:

1. Choose a folder directory where the raw images locate



2.1. Open the Kerr Microscope System to capture MOKE images. The newly created image in the directory will be automatically processed

OR

2.2. Or choose the image/videos from the file list

*3. Adjust the image operations parameters

4. Save the corresponding images. Format of jpeg, tif and png are available. For videos, only avi is available.

*Please note that if you want to adjust the parameters of image operations, the “Apply” button will apply the image operation on the processed image. If you want to reload the raw image with new parameters of image operations, you can just reselect the image in the file directory part.

3. Further development and Room for Improvement:

Capturing real-time video

A PCO camera is used to take MOKE images. A python package is used to drive the camera. For more details, you may visit: https://pypi.org/project/pco/. In the process, we mainly use the record(), images(), image_average() and set_exposure_time(). There are several modes for record(), including ‘”sequence”, “ring buffer” and “fifo”. For captureing real-time video, FIFO mode is suggested since it is non-blocking, and it can continuously capture images until you tell it to stop. And exposure time is set to be 0.2s by the set_exposure_time() function.

Approach:

Step1. Process the raw images capture from the camera so that it could show the pattern by MOKE (the following steps are suggested by Engineer Yang)

1.1 Get the background image

1.1.1 Set kepco to 2.0V and capture 4 images and call the image_average() to get an averaged background image.

1.2 Get real-time image

1.2.1 Set kepco to other value, say -0.035. Capture 4 images and average them to get a final real-time image.

1.3 Processing the background and real-time image

1.3.1 Convert both image from 16 bit unsinged image into 8 bit signed image

1.3.2 background image subtracted from real-time image

1.3.3 Normalize the pixel value of resulted image to a smaller range (such as -10 to 10)

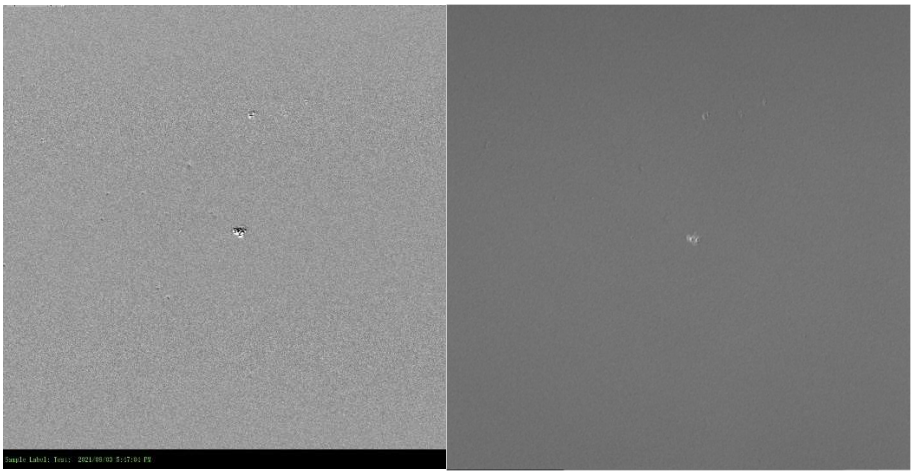
1.3.4 Convert it back to 8 bit unsigned image and normalize the resulted image back to 0 to 255

Step2. measure the framerate during the processing. Turn all the frames into a video with the measured framerate by VideoWriter in OpenCV.

Result:

The result did not live up to expectation. Different further image operations are tried, such as CLAHE, different kernel convolution and gamma correction. The result still could not capture the pattern. There is always a between framerate and camera setting, such as exposure time and averaging size. The exposure time will cause a low framerate.

Also, convolutional filters are not suggested because they slow down the computation.



Expected result Actual result