Extraction of domain wall position from magnetic contrast images obtained from X-ray photoemission electron microscopy

Jaianth Vijayakumar*[1] and C.A.F. Vaz[1]

[1]Swiss light source, Paul Scherrer Institut, Villigen 5232, Switzerland

**cptjaianth@gmail.com*

1 Introduction

X-ray photoemission electron microscopy (XPEEM) is a material characterization technique where one can acquire spatially resolved X-ray absorption spectra [1] [2]. In combination with the X-ray magnetic circular dichoric effect (XMCD), XPEEM allows one to acquire images of magnetic domains with lateral resolution down to 30 nm. XMCD images of magentic domains on different systems measured at surface interface microscopy [3] can be found in [] [] [].The stability of the magnetic domain depends on a number of energy contributions including the presence of thermal excitation. In some instances, one can observe fluctuations in domain walls [4] [5] or even domain wall nucleation events. XPEEM offers the possibility to observe such domain wall fluctuations with nm spatial resolution. The rate at which the domain wall fluctuates depends on many factors that are sometimes challenging to quantify and to analyse. Here we collect suitable codes available in Matlab and combined them into a sequential algorithm by which long sequence of XMCD images can be analysed and from which we extract informations on the domain wall position; our algorithm can be used as a starting point for more complex analysis of the dynamics of domain walls. As an example, we present here the nucleation process in Pt/Co/Pt heterostructures and explain the extraction of the domain wall step by step using the Matlab code published here.

2 Image analysis procedure and description of the code

A magnetic contrast/XMCD image is acquired in XPEEM by acquiring images with C_+ and C_- X-ray polarization and dividing them pixel-wise. A sequence of many XMCD images are acquired to improve the signal to noise ratio. To determine the fluctuation rate, it is important to acquire image sequences with the correct integration time, i.e. with a acquisition time shorter than the typical fluctuation time. Once the images are acquired the following steps can be used to extract the domain walls.

2.0.1 Step 1: Drift correction

The image sequence should first be corrected for drift. One can use drift correction method such as “Linear stack Alignment with SIFT” method available in imageJ, for example. An example of an image averaged over all images in the image sequence before and after drift correction is shown in Fig.1.

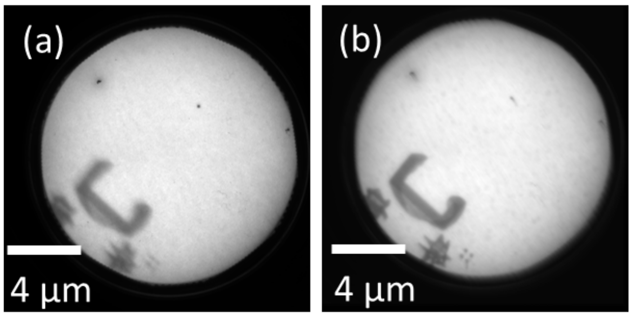

Figure 1: Image before (a) and after (b) drift correction.

2.0.2 Step 2: Loading images

From this step onwards the Matlab codes can be used. For loading the images into Matlab, a “srcFiles” Matlab function was used to the load path for all the images in the image sequence, other functions such as “Tall” can also be used in case of loading large number of images. We store all the processed image in Matlab memory in Cell “A”, however, this is optional. Enter the number of images in the image sequence in “Noimage = ” variable. An example of a raw XMCD image (averaged) to be processed is shown in Fig.2. From the image sequence shown in “Readme.md” one can observe the nucleation of domain walls.

```
Noimage = ; % enter number of images in your image sequence
A=cell(Noimage,1,1);
% Load images and store it in srcFiles, refer matlab for more information on
% 'srcFiles'
srcFiles = dir('Z:\image*');
```

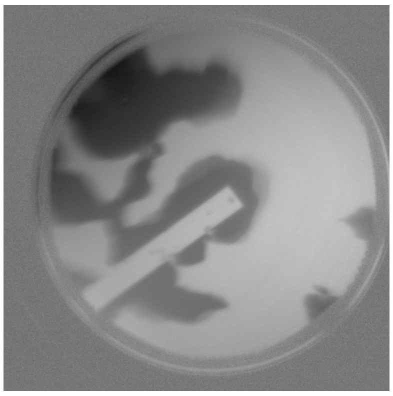

Figure 2: Raw XMCD image.

2.0.3 Step 3: removing unwanted area

In this step we clear the outside part of the image from the analysis, since usually XPEEM images have camera frame on the surrounding area similar to those shown in Fig.1. which is not needed in the analysis. This section of the codes is taken from [6] [7]. One can also use imageJ to perform this function. For this operation, one has to make sure that the loaded image "I" is in class "double". Then a circle is defined to mark the region to be extracted in the line "center = [x ,y ,r]", where the x,y,r is the row position, column position and radius (r) of the circle in pixels respectively. Once this line and the remaining lines shown below are executed, an output image is produced named as “croppedImage”. The cropped image obtained from Fig.2 is shown in Fig.3.

```
for i = 1 : length(srcFiles)


% Clearing the outside part of the image, such as detector/sensor frame etc.
    I=imread(srcFiles(i).name);
    imageSize = size(I);
    center = [256, 256, 190]; % center and radius of circle ([c_row, c_col,
%radius])
    [x,y] = ndgrid((1:imageSize(1))-center(1),(1:imageSize(2))-center(2));
    mask = ((x.^2 + y.^2)<center(3)^2); % in case it does not works use the next
% line
    mask = uint8((x.^2 + y.^2)<center(3)^2); % uncomment in the main code
    I=double(I);
    croppedImage=I.*mask;
```

2.0.4 Step 4: extracting the domain wall

Once the image is cropped, the next step is to extract the domain wall position. Here we use the Canny edge detection method. Matlab has other edge detection methods which one can consider [8]. Here the code is obtained from the example given in Matlab library. In the line “BWs = edge(croppedImage,’canny’,0.0105);” one needs to adjust the last parameter which is

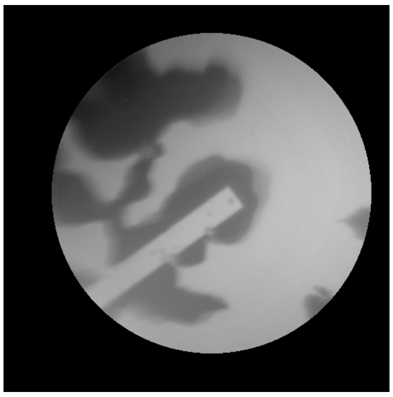

Figure 3: Image after removing the outside frame.

the threshold value to obtain the appropriate edges. The edge is detected by creating a gaussian gradient between the high and low intensity region, the threshold value basically defines the sharpness of the gaussian gradient, a threshold value with more sharpness results in picking up edges from unwanted area creating noises in the image. The details on the threshold paramters and further details on Canny edge detection method can be found in Ref. [9]. The images with the extracted domain wall is stored in cell “A”. The sum of all images whose domain walls were extracted is shown in Fig4, where one can see that the domain wall position changes, and from the image sequence shown in “readme.md” one can find the domain wall motion with time.

```
BWs = edge(croppedImage,'canny',0.0105);% adjust the threshold value to obtain
%a good edge
imshow(BWs);
A{[i]}  = BWs; % stores the image in cell "A"
```

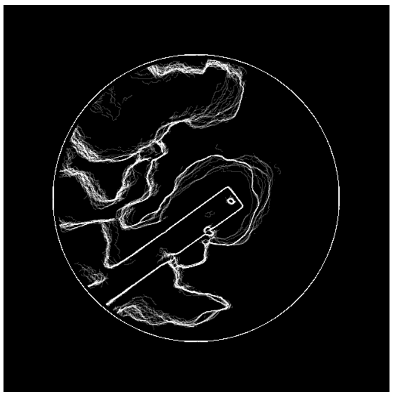

Figure 4: Summed image sequences representing the positing of the domain wall in each images.

2.0.5 Step 5: Save images

The processed images can be simultaneously stored on a local drive instead of the Matlab workspace. In this code however we store first in Matlab work space have the option to save the image in each loop or store later after all images are processed. Uncomment whichever is not required.

Use the code listed below to save the images with a prefix “extractedimage_”the number of loop”.

```
mkdir Processed image;
cd Processed image;
% save image simultaneously while looping, uncomment if not needed
    baseFileName = sprintf('extractedimage_%d.png', i);
    imwrite(A{i,1},baseFileName);
cd ..
```

To save the processed image with the same name as the source image, use the following code; the name of the file is obtained by removing the file extension in the line "baseFileName = erase(src.Files(i).name,".tif")"; here the images have ".tif" extension, but this should be changed accordingly. Then "Processed _image" is added to the new file name as prefix and it is done in the line "imwrite(A{i, 1},'Processed_image_%s.tif',baseFileName);". The images are stored in the same folder with the source images, in case of storing then in another folder use "cd" or "mkdir".

```
% to save the image with same file name used + processed, uncommment if not
%needed
    baseFileName = erase(src.Files(i).name,".tif") % erase the extension to get
%filename
    imwrite(A{i,1},'Processed_image_%s.tif',baseFileName);
end
```

In case the threshold parameter need to be changed many times, it is not required to store these images in the loop each time, therefore one can comment them out and use the following code to save the images in the end from cell "A" after the loop.

```
for i = 1 : length(A)
mkdir Processed image;
cd Processed image;
   baseFileName = sprintf('cropped_image_%d.png', i);
   imwrite(A{i,1},baseFileName);
```

```
cd ..
end
```

References

[1] J. Stohr and S. Anders. X-ray spectro-microscopy of complex materials and surfaces. *IBM Journal of Research and Development*, 44(4):535–551, July 2000.

[2] J Stohr. Exploring the microscopic origin of magnetic anisotropies with X-ray magnetic circular dichroism (XMCD) spectroscopy. *Journal of Magnetism and Magnetic Materials*, 200:470–497, 1999.

[3] U. Flechsig, F. Nolting, A. Fraile Rodríguez, J. Krempaský, C. Quitmann, T. Schmidt, S. Spielmann, and D. Zimoch. Performance measurements at the sls sim beamline. *AIP Conference Proceedings*, 1234(1):319–322, 2010.

[4] M. Kronseder, T. N.G. Meier, M. Zimmermann, M. Buchner, M. Vogel, and C. H. Back. Real-time observation of domain fluctuations in a two-dimensional magnetic model system. *Nature Communications*, 6:1–7, 2015.

[5] W. Kuch, K. Fukumoto, J. Wang, F. Nolting, C. Quitmann, and T. Ramsvik. Thermal melting of magnetic stripe domains. *Physical Review B - Condensed Matter and Materials Physics*, 83(17):1–4, 2011.

[6] https://ch.mathworks.com/matlabcentral/answers/266039-how-to-get-a-crop-a-circluar-region.

[7] http://stackoverflow.com/questions/19992097/how-to-do-circular-crop-using-matlab.

[8] https://ch.mathworks.com/help/images/ref/edge.htmlbuo5g3w-1-threshold.

[9] John Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.