3D magnetometry calculation from XPEEM images

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 technique can be used to obtain magnetic contrast images using X-ray magnetic circular dichroic effect (XMCD). The XMCD effect is a polarization dependent absorption process on a given magnetization, i.e. the the X-ray abosption changes with respect to different directions of the magnetic moments for a circular polarized light; and this property is exploited to obtain the magnetization vector of the magnetic domains from XMCD images. In order to obtain the 3D magentometry data of the XMCD images, one should acquire XMCD images in three different angles. From the data one can solve the linear dependence of the X-ray absorption with magnetization and extract the M$_x$, M$_y$ and M$_z$ values. In this document the fundamentals of the 3D magnetometry in XPEEM will be explained along with the required codes from Matlab suitable to carry out the entire process.

2 3D magnetometry in XPEEM

An XMCD image is obtained by acquiring an image with C_+ and C_- polarization; the local magnetization of sample have different absorption with C_+ and C_- polarization. By performing C_+ - C_-/ C_+ + C_-, one can obtain the XMCD value which has a linear relation with the spin and orbital magnetic moment. The intensity of absorption changes with direction of magnetization as I $\propto \vec{\alpha} \cdot \vec{M} \cos\theta$, where $\vec{\alpha}$ is the X-ray propagation vector, $\vec{M}$ is the magnetization vector and θ is the angle between them. In XPEEM, the angle of incidence of X-ray on the sample is 16°, which makes the XPEEM more sensitive to in plane (IP) magnetization than out-of-plane (OOP) magnetization. In order to identify the presence of IP and OOP magnetization, one can acquire XMCD images at two or more number of azimuthal angle (typically 0°, 45° and 90° between the sample and direction of X-ray. In the XMCD images taken at different angles, one can find the magnetic contrast of IP magnetic domains changes with angle and the XMCD contrast of OOP magnetic domains remains unchanged. The measuring geometry is shown in Fig 1, where the X-ray incidents at an angle ψ and incidents on a spot with local magnetization indicated by blue vector, the corresponding M_x, M_y and M_z are given by yellow, green and red arrows respectively. The IP angle θ determines the angle between X-ray propagation vector and IP magnetic components (vector of M_x and M_y), and *phi* is the angle between the perpendicular M_z component with respect to the horizontal plane represented in brown. With the fundamental theory of XMCD and the measuring geometry shown in Fig 1, we can extract the XMCD dependence as a function of magnetization vector or the angle of rotation θ as following;

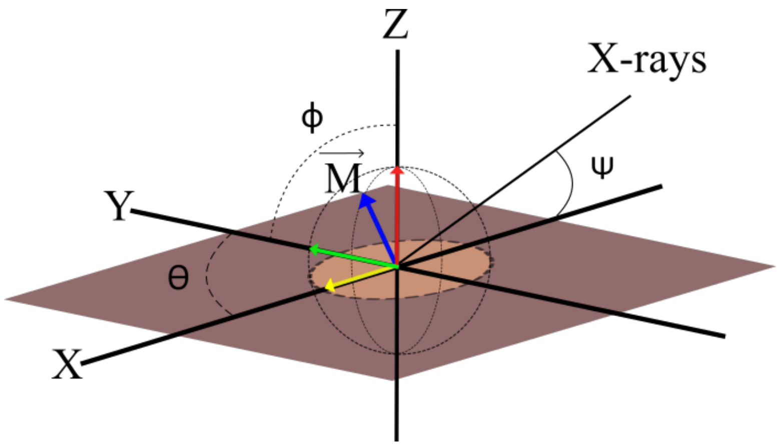

Figure 1: Measuring geometry

The general expression for the XMCD(I), M$_x$,M$_y$ and M$_z$ with respect to different angles can be written in the following form;

$$I_{\psi\phi\theta} \propto M_x \cos\psi \cos\theta + M_y \cos\psi \cos\theta + M_z \sin\phi$$

From here we can calculate the XMCD (I) as a function different values of θ with incident angle ψ of 16°.

At 0°,

$$I_{0^\circ} \propto M_x \cos(16^\circ)\cos(0^\circ) + M_y \cos(16^\circ)\cos(90^\circ) + M_z \sin(16^\circ),$$

$$I_0 \propto M_x \cos(16^\circ) + M_z \sin(16^\circ) \tag{1}$$

At 45°,

$$I_{45^\circ} \propto M_x \cos(16^\circ)\cos(45^\circ) + M_y \cos(16^\circ)\cos(45^\circ) + M_z \sin(16^\circ) \tag{2}$$

At 90°,

$$I_{90^\circ} \propto M_x \cos(16^\circ)\cos(90^\circ) + M_y \cos(16^\circ)\cos(0^\circ) + M_z \sin(16^\circ),$$

$$I_{90^\circ} \propto M_y \cos(16^\circ) + M_z \sin(16^\circ) \tag{3}$$

Therefore, by obtaining XMCD images with different angles and solving the equations(1,2,3) one can obtain the values of M$_x$, M$_y$ and M$_z$.

3 Lens distortion correction

To calculate the 3D magnetization vector of the XMCD images, one should correct for the lens distortions. Even though the image sequence of XMCD images are corrected for drift, the lens distortion produced when imaged at different angles is inevitable. With well defined magnetic structures, the distortion correction is less complicated. With complex domain structure and few reference objects, lens distortion corrections are not trivial. Here we present a lens distortion correction available in Matlab, which can be suitable for correcting the distortion for the XMCD images and it has a file name "Distortion_correction.m" in the repository. One can keep any of the three angles as reference image and correct the images of the other two angles. The Matlab function used for this process is called "firgeotrans" and the corresponding codes are obtained from ref [1]. The first step is to identify the points which moved from the reference image, we use the Matlab function “cpselect”, this function opens a window showing the reference image and moving image side by side and one can pick a point from the reference image by clicking on the point and can select the same reference point on the moving image. Similarly, many such points can be selected, it is important that one should select first the point from the reference image and then from the moving image. After this step, it is possible to export these points as “movingPoints” and “fixedPoints” by opening the file drop down menu in the corner of the window. After this step all the selected points will be exported into Matlab work space for future use. For more information on how to use the “cpselect” function refer [2].

```
% To choose the points which moved from the reference image
h  = cpselect('reference.png','moving.png');
%%
% Close the tool.
close(h)
%%
```

Once the fixed and moving points are exported, the following set of functions performs the appropriate lens correction operation. Here we use the Matlab function “geotrans”. The images are first loaded using the following command and converting the images to double may not be necessary but optional in case of RBG type images.

```
moving = imread('moving_image.png');

fixed = imread('fixed_image.png');

moving = im2double(moving);

fixed = im2double(fixed);
```

The distortion correction is performed in the following set of codes. In “fitgeotrans” function, the last parameter determines which type of transformation to be performed, for lens distortion correction ”affine” method works, however,depending on the complexity of the distortion one can consider using other methods which can be obtained from the link given below the code or form Matlab “help”. The code corrects for the distortion on the moving image and save it as “tform”; however, this image is not in the same dimension of the reference image. Therefore, the next line “ref = imref2d(size(fixed));” converts the coordinates of the fixed reference image into a global coordinates, and finally in the last line “moving_image _registered = imwarp(moving,tform,’OutputView’,ref);” the transformed image is resized into the same dimension as the reference image. The two images, the corrected image and the fixed reference image are then displayed, overlapping each other in the “imshow” function in the next line. If the distortion is not perfect one can repeat the “cpselect” to tune the position of the fixed and moving coordinates, if the image is still not corrected properly, one can consider changing the type of transformation in “fitgeotrans” function. If non of this works, the problem is usually from poor definition of the fixed and the moving points, which should be optimized further. One alternative to solve this problem can be to extract the domain walls and then perform “cpselect” function where one can define the points with better accuracy; the moving and the fixed points determined from the domain wall image can be saved and reused for the XPEEM images for better correction. The codes to extract the domain can be obtained from ref [3]

```
%%
%Perform appropriate transformation using fitgeotrans function
% Fot other types of transformation check the link
% https://ch.mathworks.com/help/images/ref/fitgeotrans.html
tform = fitgeotrans(fixedPoints,movingPoints,'affine');

% refers image coordinate into world coordinates
ref = imref2d(size(fixed));

% Transformed image resized in the same dimension as the world coordinates
fortyfivedeg_registered = imwarp(moving,tform,'OutputView',ref);

% Displays figure with distortion corrected image with reference image
figure, imshowpair(fortyfivedeg_registered,fixed,'blend')
%%
```

All the images should be corrected for lens distortion, and one must observe that the domains and/or the reference structure should overlap on top of each other in all the angles. The images distortion also moves and changes the values of the pixels, in such cases one can perform the drift correction on individual C_+ and C_- images and later produce the XMCD images. Once the images are corrected, magnetization components can be calculated, which is described in the next section.

4 Calculation of the 3D magnetization vectors

The first step is to load the images, and convert into double, as sometimes Matlab can change the image format, and one should make sure to have the XMCD values not changed with loaded in Matlab. Then empty matrix to store M_x, M_y and M_z is created to store the solution. The following set of codes can be found in in the file "3Dmagnetometry_analysis.m".

```
 %% Load image to matlab

I_0=imread('0.tif');
I_45=imread('45.tif');
I_90=imread('90.tif');
%I_180=imread('180deg_20FOV.tif');

%% Convert into double just in case

I_0=double(I_0);
I_45=double(I_45);
I_90=double(I_90);
%I_180=double(I_180);
```

```
%% create space for storing the solved Mx,My and Mz values


Mx=double.empty();

My=double.empty();

Mz=double.empty();
```

After loading the images the three equations (1,2 and 3) is solved using the following set of codes which computes the M_x, M_y and M_z values.

```
A = cell(length(I_0(1,:)),length(I_0(:,1)));


%Add offset if needed otherwise comment out

 for i=1:length(I_0(1,:))

     for j = 1:length(I_0(:,1))

          A{i,j}={I_0(i,j)+0.395 I_45(i,j) I_90(i,j)};

     end

end




a=[ cosd(IX)*cosd(I_one) cosd(IX)*cosd(90-I_one) sind(IX)

     cosd(IX)*cosd(I_two) cosd(IX)*cosd(90-I_two) sind(IX)

     cosd(IX)*cosd(I_three) cosd(IX)*cosd(90-I_three) sind(IX)];
```

```
%a=[ cosd(16) 0 sind(16);
%    cosd(16)*cosd(45) cosd(16)*sind(45) sind(16);
%    0 cosd(16) sind(16)];


X=double.empty();
MX=double.empty();
MY=double.empty();
MZ=double.empty();



 for i=1:length(I_0(1,:))
     for j = 1:length(I_0(:,1))


         B = cell2mat(A{i,j});
         B = B';
         X = linsolve(a,B);
         MX(i,j) = X(1);
         MY(i,j) = X(2);
         MZ(i,j) = X(3);


     end
 end
```

Then the M$_X$,M$_Y$ and M$_Z$ is stored in forms of images and text file.

```
%% Writes the MX, MY and MZ in form of .tif (the values in preserved in this format)
imwrite(MX, 'Mx.tif');
imwrite(MY, 'My.tif');
imwrite(MZ, 'Mz.tif');

%% Writin Mx, My and Mz as text file, still saves as a matrix

dlmwrite('vectorMx.txt',MX,'delimiter',' ','newline','pc');
dlmwrite('vectorMy.txt',MY,'delimiter',' ','newline','pc');
dlmwrite('vectorMz.txt',MZ,'delimiter',' ','newline','pc');

%%
```

The separate M$_X$, M$_Y$ and M$_Z$ can be further analysed according to the requirements. Important: to carry out the codes described below and in the following section the work space of "3Dmagnetometry_analysis.m" need to be open, therefore it is important that the workspace is also saved. This is done in the lines shown below,

```
Filename='something.mat'; % change name
save(Filename); % save workspace for later user
```

To visualize the magnetization vector in matlab "quiver" function can be used. However, a micromagnetic software such as "OOMF micromagnetics" can provide better images. Therefore the processed data should be converted into a suitable format, one of the format which OOMF micromagnetics recognizes is ,ovf(ordinary vector format) which is formerly called .svf (standard vector format). Such file has the x,y and z coordinates and the corresponding M$_x$, M$_y$ and M$_z$ with certain header lines. "ForOOMF_wholemage.m" Matlab file creates .svf file for the entire image from the processed data, the code can be used as such if the variable names are not changed in "3Dmagnetometry_analysis.m", it may take few minutes to write all the lines. A representation of 3D magnetization vector map obtained from OOMF is shown in Fig.2. More information on the .svf or .ovf format can be found in ref [4].

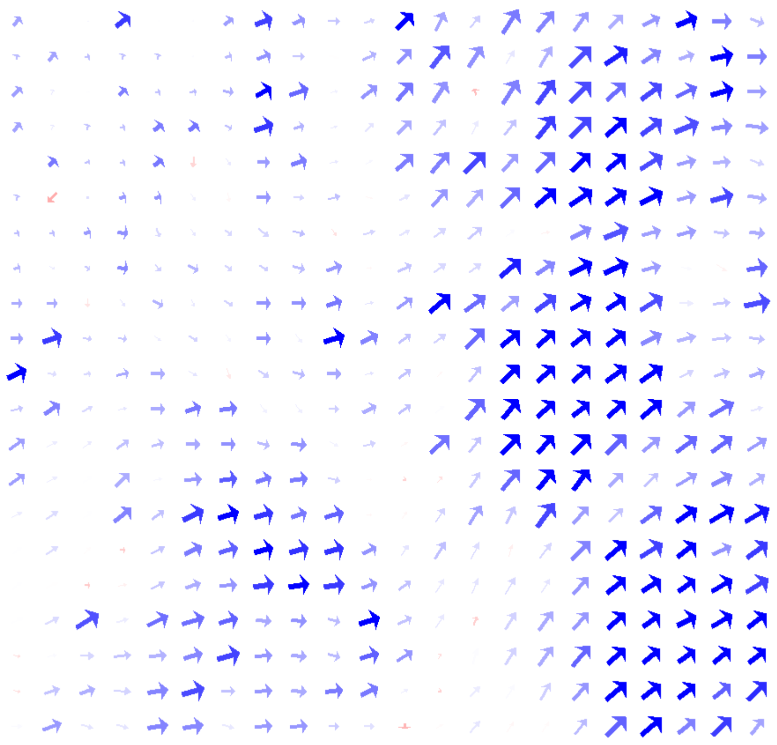

Figure 2: OOMF magnetization vector diagram, where dark blue represents OOP magentization

5 Converting into a format for 3D visualization

Since OOMF micromagentic simulation can only provide a 2D image, to create a 3D visualization of the magnetization vectors the data can be converted into a .vtk(visualization tool kit) format [5] and a 3D image can be created using an open source software "paraview" [6] . The .vtk file has similar information as .svf file, but with different syntax and ordering. Please note that this code do not convert a .svf file to .vtk and only convert the Mx, My and Mz processed by "3Dmagnetometry_analysis.m" to the appropriate .vtk format. "Forvtk_wholeimage.m" Matlab file can be used for this purpose to convert the whole image data into a .vtk format, this process can also take few minutes. The 3D representation of the Fig.2 after producing in paraview is show in Fig.3. Sometimes larger files are difficult to handle and analyse in OOFM or in paraview, therefore, extracting certain part of the image will be useful, in such cases code described in the following section can be followed.

In the image, the arrows represent the magnitude and the colour represents the scalar of the value of the M_z value, and one can change this parameter in the Forvtk_wholeimage.m in the section shown below, the M_x, M_y and M_z values are stored as cutX(i,4), cutX(i,5) and cutX(i,6) respectively and one can change that in the line "fprintf(fid,'%d
n',abs(cutX(i,6)));".

```
% To get the scalar of Mx replace cutX(i,6) to cutX(i,4) and to get
%My replace cutX(i,6) with cutX(i,5)
for i=1:length(cutX)
```

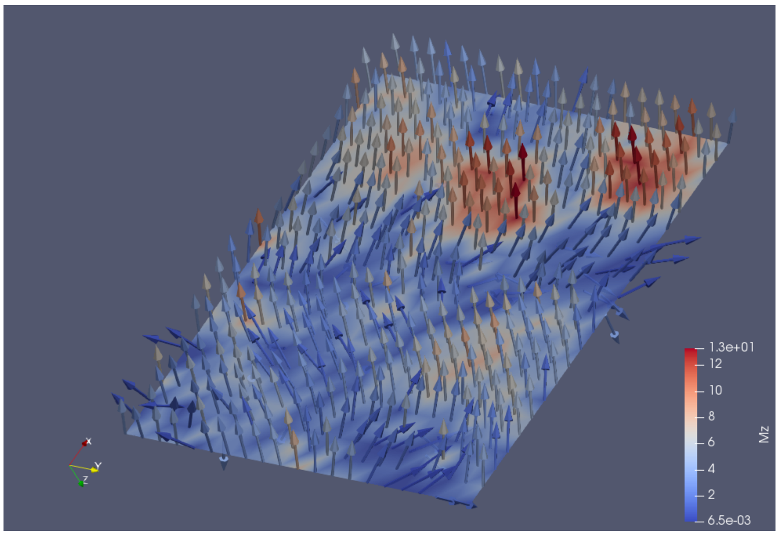

Figure 3: 3D visualization on Paraview

```
    fprintf(fid,'%d\n',abs(cutX(i,6)));
end
fclose(fid);
```

5.1 Convert to OOM for selective part of the images

To extract certain regions from the matrix, we simply create a separate matrix with the same dimension as the XPEEM image with “zeros” and then define a square on the matrix representing the coordinates of the region to be extracted as “1”, an example of such masks created using imageJ is shown in Fig.4. Many such mask images can be produced at once and stored for analysis. It is really important that the dimension of the mask image is same as the XPEEM image. By running the code ForOOMF.m or Forvtk.m one can obtain the .svf and .vtk files

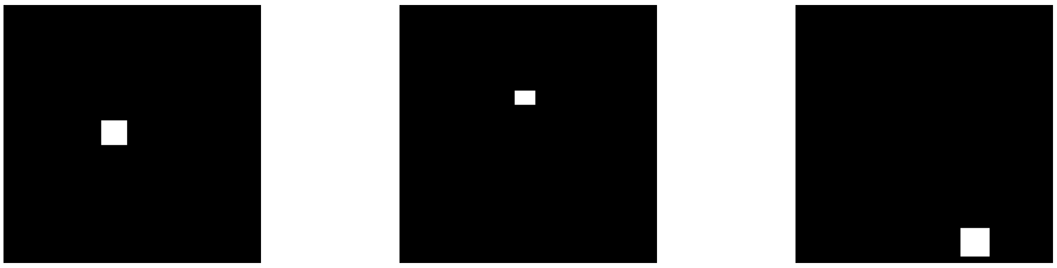
Figure 4: Mask for extracting certain regions from the image

respectively from the regions marked as 1 from different masked images.

To run these codes one needs to first define where the images are stored, therefore one should open the code and fill up the starting letter of the mask images, one can store names like "Mask_region _1,2..."; and define the starting letter in the line as "srcFiles = dir(' I:\Mask*');" for example.

For .vtk files, in case the scalar values need to be changed, one can replace the appropriate values in the lines shown below,

```
% To get the scalar of Mx replace cutX(i,6) to cutX(i,4) and to get My replace
% cutX(i,6) with cutX(i,5)


for i=1:length(cutX)
    fprintf(fid,'%d\n',abs(cutX(i,6)));
end
fclose(fid);
```

These lines are same as the one described in the previous section, please refer the end of previous section.

References

[1] https://ch.mathworks.com/help/images/ref/fitgeotrans.html.

[2] https://ch.mathworks.com/help/images/start-the-control-point-selection-tool.html.

[3] https://github.com/jaianthv/xpeem-image-analysis-domain-wall-fluctuations.

[4] https://math.nist.gov/oommf/doc/userguide12a5/userguide/ovf_0.0_format.html.

[5] https://www.vtk.org/wp-content/uploads/2015/04/file-formats.pdf.

[6] https://www.paraview.org/.