Overview:

Polycrystalline electron diffraction patterns are often difficult to interpret. It was therefore my intent to see if it is possible to present the diffraction pattern in a different way to see if that helps with interpretation. What many people are familiar with is X-ray diffraction, as such this script aims to emulate the appearance of X-ray powder diffraction results.

Requirements:

Digital Micrograph with python functionality enabled. Numpy installed in the python environment, this should come preinstalled with the python functionality in DM.

The script is primarily meant to be used with polycrystalline diffraction patterns.

This script has only been tested on DM 64-bit version 3.43.3213.0, as such, no guarantee is made that the script will function on any other version.

Installation:

a) Option 1:
   1. Open the script file “Rotational Unfurling” in Digital Micrograph.
   2. Navigate to the file menu and press “Install Script”.
   3. A window will pop up, enter what the name of the script should be and in which menu (and optionally sub-menu) it should be installed in.

b) Option 2:
   1. Open Digital Micrograph.
   2. Navigate to the file menu and press “Install Script File”.
   3. Select the file you wish to install in the file explorer that opens.
   4. A window will pop up, enter what the name of the script should be and in which menu (and optionally sub-menu) it should be installed in.

Usage:

1. Ensure that python is installed in your digital micrograph installation. If not, install python. The python installation procedure can be found within the folder the GMS installer came in.
2. Open the diffraction pattern that should be transformed.
3. If the diffraction pattern has a predefined center, as found in Image Info - Calibration – Origin, the script uses that as the rotational center. If not, the script will require a rectangular or oval ROI. If a ROI is supplied, the script will use the center of the ROI as the center of rotation, regardless of whether there is a predefined center or not.
4. Run the script.
5. When the script is run, a filtered unfurling around the center, as defined above, and a line profile along the radial direction of the unfurling are shown, how these are created is shown below. From these it should be possible to get the distances in the image, as well as make it easier to see how the image changes in the angular direction.

More detail:

When the script is run, an unfurling around the center, as defined above, is performed with a radial resolution of 1600 pixels. As such, one pixel in the radial direction in the unfurling is equal to

$$px_{size,radial} = \frac{\sqrt{\max(x_0, x_{max} - x_0)^2 + \max(y_0, y_{max} - y_0)^2}}{1600}$$

Pixels of the original image. Where x_0 and y_0 are the x- and y-coordinates of the specified center. The radial dimension is directed along the x-axis of the unfurling. The angular resolution is 0.33 degrees, and as such 1080 pixels, and is shown in the y-direction.

After this unfurling has been calculated, the median for each radial pixel column is calculated. This is done via python through numpy since there is no built-in function in the digital micrograph scripting language, and my attempted implementation was very slow. When the median has been calculated, which takes a couple of seconds, the unfurled image is filtered so that if a pixel in the unfurled image is equal to or greater than 1.1 times the median of that pixel-column, it is set to be equal to the value of the pixel minus the median of the column. If a pixel is less than 1.1 times the median of the pixel column, then that pixel is set to 0, and as such only diffraction spots should be visible in the filtered image.

Summary of functions used:

Image RotationalUnfurling(image img, number cx, number cy)

This function transforms the input image “img” by warping it around the center of the image, as defined by the numbers “cx” and “cy”.

Void FilterMedian(image img)

This function filters the input image “img” by comparing with the median of each column in the image, with a tunable limit. The image “img” is filtered vs the median of each column, if the value in a pixel is larger than limit*median of that column, the value in the pixel is set to be the median. If the value is smaller than limit*median, then the value is set to the proper value, where these values are set as a new temporary image “medWarp”. After this, “medWarp” is subtracted from “img”, which leads to all values above limit*median will be positive, and all other values will be 0.

Number MedianPy (image temp)

This function is a hybrid function that makes use of both Digital Micrograph Scripting and Python. The function takes an image “temp” as an input and calculates the median of the image using numpy.median() in python. In this script, the function is used to calculate the median of a column of pixels, but should work for an array of dimension n, see documentation for numpy.median().