**Focus Map Function Documentation**

The code begins with a commented out line that says the following: -\*- coding: utf-8 -\*-

This line was once removed but Canopy gave an error saying that the line was required, so it stayed. Why it’s needed is unknown.

**Packages**

Most of the packages imported were recycled from the MakeImageStack function written by Zack. However, four new packages were added: numpy, scipy, csv, and ast.

Numpy is used to convert the .raw files captured into arrays in TakeImgs.

The misc module is loaded form the scipy package. It is used to save the arrays given by numpy above into images. See TakeImgs.

The csv package is used to create a csv file from the list calculated in FocusMap.

The ast package is used when reading csv files from the present working directory. It makes sure Python can read the csv file.

**Outside Code**

The first piece of code after the packages are loaded is one that connects us to the SEM. Once a connection is established, the program tells us so.

Next up are important variables that we can change before running the code. They’re split up into different categories. Most are sufficiently self-explanatory, so not all of them will be mentioned here.

The variables z\_min and z\_max are two important numbers. Here we’ve set a global minimum/maximum value in mm for z—if a z-value happens to go above or below that, it will change so the SEM does not move the stage into hazardous territory.

There’s also WD\_0, which is the original working distance. Ideally, the user will manually change the working distance until the sample is clear and then move “WD and Z” to the value set for WD\_0. In our case, we set WD\_0 to 25mm. The original z-position is saved as z\_0 once the user has focused and moved “WD and Z” to WD\_0.

The variable args is an easy way to input values into later functions. It is not required for use of the functions.

**Functions**

Dashed lines have been added to separate two pieces of code. The first chunk is a slightly modified version of the code written by Zack. ReadMessage has been cleaned to remove any unnecessary lines, but remains essentially the same as when Zack wrote it. Honestly, I’m not too sure what it does.

WriteImage is another slightly modded version of a function written by Zack. It create a file for the .raw images and writes values into them.

TakeImgs is a heavily modified version of Zack’s main function. The global variables defined here allow the WriteImage function to use them as names for the files. This function also reads csv files named “coordinates.csv” and creates a list of them. The ast package is used to correctly interpret the file as a 1-D list. The while loop below that moves the SEM stage to the list of coordinates and creates file names for each image. After enabling detectors to different channels, scanning is stopped to allow remote-control of the SEM. The scan mode is changed accordingly and the stage moves to the location and writes an image. The variable res is very important as it holds the data taken by the SEM—it is transferred over to the WriteImage function. The last part of the while loop uses the numpy package to manipulate the .raw file data as an array. After the array is shaped accordingly, the misc part of the scipy package saves the image as a .tiff file.

The functions below the dashed line are new pieces of code.

The calc\_coords function allows the user to see what values the SEM will move the stage to. It takes 7 variables, 5 of which are default values. 4 of the default values, the mins and maxs, limit the movement of the custom stage used to analyze the foils. The default variable “delta” sets the distance (in mm) the SEM will move the stage before each scan.

The function mainly takes two arguments, x\_0 and y\_0. This is the first coordinate the SEM will move the stage to. The function takes this coordinate and calculates a grid of points that will later be used as stage-movement coordinates. This grid is calculated by adding to x until x\_max is reached, then resetting x back to x\_0 and moving y down by an amount delta. The grid is calculated in a left-to-right, top-to-bottom fashion. Because of this, smaller grids are made by having x\_0 closer to its maximum and y\_0 closer to its minimum. The largest area calculable is if x\_0 = x\_min and y\_0 = y\_max.

This function does not involve any SEM commands whatsoever. It is used in the FocusMap function, or can be called as a standalone function to see where the SEM will move in the x-y plane.

FocusMap is the meat of the program. This program first assigns detectors to certain channels and switches to resolution mode. The program then calls calc\_coords to create a list of x,y coordinates. Once the list is had, another while loop is defined. The loop has the SEM move the stage to each coordinate in the calc\_coords list. At each point, a built-in function called AutoWD is run. The SEM then calculates the best WD for the sample at that coordinate. Once the working distance is found, it is used to calculate the sample height (z\_n) using the initial height of the stage (z\_0), the original working distance (WD\_0) and the working distance calculated from AutoWD (WD\_n). Here, the minimum and maximum values of z are put in place—if z\_n ends up above or below our global z\_min or z\_max values, it is reset to the closest limiting value to avoid possible collisions. Once our z\_n values are calculated, they are added to the list of x and y values calculated by calc\_coords. This new list now has the form of [(x, y, z)], where the z value is how far the stage will move in the z-axis to keep the WD constant. This list is essentially the focus map.

This list is then saved as a csv file for user to view or edit if the user feels the need to remove any coordinates. This csv file is the file read by TakeImgs to take more images with the focus map.