Appendix 2: Python code for generating downscaled images

This code requires the python module in appendix 3

This code was run in a python notebook on a windows PC. Linux PCs seem to have an issue with the code.

```python
# Importing the necesarry packages
import os.path
from PIL import Image
import numpy as np
import csv
import histogram_for_MCC
from cmath import sqrt

# Defining scales to be used for downscaling, 200 means a 1:200 scale
scales = [200, 100, 50, 25, 10, 5, 1]

# Defining the dolomite threshold as a fraction
hit_ratio = 0.08

# Defining the grayscale threshold for grayscale imput
# Imput range [0 - 255]
# Currently any value works as the imput is black or white, not gray
grayscale_threshold = 50

# Defining number of permutations for the permutation test
itterations = 100000

# Defining whether to create a histogram
# ["Yes" / "No"]
plot_yn = "No"

# Defining whether to recreate the downscaled images if they are already created
# ["Yes" / "No"]
zoom_yn = "No"

# Defining the values for "Hit", "Miss" and "Out of bounds"
hit_rgb = (255,255,255,0)
miss_rgb = (0,0,0,255)
oob_rgb = (255,0,0,255)

# Creating headers for the data set
# included are:
# - all settings possible for each image
# - Confusion matrix results
# - Statistical performance metrics
# - 0.5th and 99.5th percentile value for each performance metrics
lines = [['Case', 'surface', 'Cut off', 'Frequency', 'Illumination', 'Zoom','True Positive', 'True
Negative',
            'False Positive', 'False Negative', 'Out of Bounds', 'MCC_observe', 'X5_MCC', 'X95_MCC',
            'sensitivity_observe', 'X5_sensitivity', 'X95_sensitivity',
            'specificity_observe', 'X5_specificity', 'X95_specificity',
            'error_rate_observe', 'X5_error_rate', 'X95_error_rate',
            'accuracy_observe', 'X5_accuracy', 'X95_accuracy',
            'precision_observe', 'X5_precision', 'X95_precision',
            'false_positive_rate_observe', 'X5_false_positive_rate', 'X95_false_positive_rate']]

# Itterating over all scales to be used for downscaling
for scale in scales:

    # Creating scale based suffix for files
    zoom = '_' + (3-len(str(scale)))*'0' + str(scale) + 'xZoom.png'

    # Listing the names of all files to be downscaled
    file_names   = ['case_01_black_white_co_095_freq_30_Q0c_Illu_45',
                    'case_01_black_white_co_095_freq_30_Q1c_Illu_45',
                    'case_01_black_white_co_095_freq_30_Q2c_Illu_45',
                    'case_01_black_white_co_095_freq_30_Q3c_Illu_45',
                    'case_01_black_white_co_095_freq_30_Q4c_Illu_45',
                    'case_02_black_white_co_115_freq_60_Q0c_Illu_45',
                    'case_02_black_white_co_115_freq_60_Q1c_Illu_45',
                    'case_02_black_white_co_115_freq_60_Q2c_Illu_45',
                    'case_02_black_white_co_115_freq_60_Q3c_Illu_45',
                    'case_02_black_white_co_115_freq_60_Q4c_Illu_45',
                    'case_03_black_white_co_125_freq_90_Q0c_Illu_45',
                    'case_03_black_white_co_125_freq_90_Q1c_Illu_45',
                    'case_03_black_white_co_125_freq_90_Q2c_Illu_45',
                    'case_03_black_white_co_125_freq_90_Q3c_Illu_45',
                    'case_03_black_white_co_125_freq_90_Q4c_Illu_45',
                    'case_04_black_white_co_067_freq_30_Q0t_Illu_25_whole',
                    'case_04_black_white_co_067_freq_30_Q1t_Illu_25_whole',
                    'case_04_black_white_co_067_freq_30_Q2t_Illu_25_whole',
                    'case_04_black_white_co_067_freq_30_Q3t_Illu_25_whole',
                    'case_04_black_white_co_067_freq_30_Q4t_Illu_25_whole',
                    'case_05_black_white_co_045_freq_30_Q0t_Illu_15_whole',
                    'case_05_black_white_co_045_freq_30_Q1t_Illu_15_whole',
                    'case_05_black_white_co_045_freq_30_Q2t_Illu_15_whole',
                    'case_05_black_white_co_045_freq_30_Q3t_Illu_15_whole',
                    'case_05_black_white_co_045_freq_30_Q4t_Illu_15_whole',
                    'case_06_black_white_co_058_freq_30_Q0c_Illu_45',
                    'case_06_black_white_co_058_freq_30_Q1c_Illu_45',
                    'case_06_black_white_co_058_freq_30_Q2c_Illu_45',
                    'case_06_black_white_co_058_freq_30_Q3c_Illu_45',
                    'case_06_black_white_co_058_freq_30_Q4c_Illu_45',
                    'case_07_black_white_co_125_freq_30_Q0c_Illu_45',
                    'case_07_black_white_co_125_freq_30_Q1c_Illu_45',
                    'case_07_black_white_co_125_freq_30_Q2c_Illu_45',
                    'case_07_black_white_co_125_freq_30_Q3c_Illu_45',
                    'case_07_black_white_co_125_freq_30_Q4c_Illu_45',
                    'case_08_black_white_co_165_freq_30_Q0c_Illu_45',
                    'case_08_black_white_co_165_freq_30_Q1c_Illu_45',
                    'case_08_black_white_co_165_freq_30_Q2c_Illu_45',
                    'case_08_black_white_co_165_freq_30_Q3c_Illu_45',
                    'case_08_black_white_co_165_freq_30_Q4c_Illu_45',
                    'case_09_black_white_co_045_freq_30_Q0t_Illu_15_whole',
                    'case_09_black_white_co_045_freq_30_Q1t_Illu_15_whole',
                    'case_09_black_white_co_045_freq_30_Q2t_Illu_15_whole',
                    'case_09_black_white_co_045_freq_30_Q3t_Illu_15_whole',
                    'case_09_black_white_co_045_freq_30_Q4t_Illu_15_whole',
                    'case_10_black_white_co_058_freq_30_Q0t_Illu_15_whole',
                    'case_10_black_white_co_058_freq_30_Q1t_Illu_15_whole',
                    'case_10_black_white_co_058_freq_30_Q2t_Illu_15_whole',
                    'case_10_black_white_co_058_freq_30_Q3t_Illu_15_whole',
                    'case_10_black_white_co_058_freq_30_Q4t_Illu_15_whole',
                    'case_11_black_white_co_080_freq_30_Q0t_Illu_15_whole',
                    'case_11_black_white_co_080_freq_30_Q1t_Illu_15_whole',
                    'case_11_black_white_co_080_freq_30_Q2t_Illu_15_whole',
                    'case_11_black_white_co_080_freq_30_Q3t_Illu_15_whole',
                    'case_11_black_white_co_080_freq_30_Q4t_Illu_15_whole',
                    'Model Q0c Dolo',
                    'Model Q0t Dolo',
                    'Model Q1c Dolo',
                    'Model Q1t Dolo',
                    'Model Q2c Dolo',
                    'Model Q2t Dolo',
                    'Model Q3c Dolo',
                    'Model Q3t Dolo',
                    'Model Q4c Dolo',
                    'Model Q4t Dolo'
                    ]

    # Itterating over files to be downscaled
    for files in file_names:
```

```
# Checking if downscaled image exists and wether to overwrite if so
if  os.path.exists("Zoomed/" + files + zoom) == True and zoom_yn == "No":

    # Not overwriting the downscaled image
    cool = "Cool"

# Creating a downscaled image
else:

    # Finding horizon identifier of the image file
    depth = files.find('Q')

    # Finding and opening model image of the same surface as the selected image file
    im_mask = Image.open('Source/Model ' + files[depth:depth+3] + ' Dolo.png')
    pix_mask = im_mask.load
    pixel_mask = list(im_mask.getdata())

    # Opening the selected image file
    im = Image.open("Source/" + files +'.png')
    pix = im.load
    pixel_values = list(im.getdata())

    # Creating objects to store temporary image data
    binary_list = []
    binary = []

    # Itterating over the pixels in the
    for i in range(len(pixel_values)):

        # If the pixel is not grayscale (Red) in the model image,
        # the pixel is considered out of bounds "[0,0,1]"
        if pixel_mask[i][0] != pixel_mask[i][1]:
            binary_list.append (np.array([0,0,1]))

        # If the pixel's Red value (in RGB) is over the theshold,
        # the pixel is considered dolomite/high amplitude "[0,1,0]"
        elif pixel_values[i][0] <= grayscale_threshold:
            binary_list.append (np.array([0,1,0]))

        # If the pixel's Red value (in RGB) is under the theshold,
        # the pixel is considered no dolomite/low amplitude "[1,0,0]"
        elif pixel_values[i][0] > grayscale_threshold:
            binary_list.append (np.array([1,0,0]))

    # Making the list into an array
```

```
for i in  range(im.size[1]):
    binary.append(binary_list[i*im.size[0]:(i+1)*im.size[0]])
    # print(binary)

# Defining objects used for scaling
half_new_zoom = []
new_zoom = []

# Itterating over the rows of pixels
for x in binary:

    # Creating an object to store horontally tallied pixels per section at the zoom scale
    row = []

    # Deviding X axis into downscaled sections
    for y in range(int(im.size[0]/scale)):

        # Tallying pixel types within each section
        row.append(sum(x[y*scale:(y+1)*scale]))

    # Creating horizontally tallied image
    half_new_zoom.append(row)

# Converting to a numpy array
temp = np.array(half_new_zoom)

# Deviding Y axis into downscaled sections
for x in range(int(im.size[1]/scale)):

    # Combining tallies vertically at the zoom scale
    zoom_array = np.sum(temp[x*scale:(x+1)*scale],0)

    # Creating an row object for the final image
    new_zoom_array= []

    # Itterating over the tallies per section for the downscaled image (rows)
    for (i,value) in enumerate(zoom_array):

        # Checking whether the threshold value is met and there are any dolomite pixels
        if value[0] >= hit_ratio*value[1] and value[0] > 0:

            # Defining the section is a hit
            new_zoom_array.append(1)

        # If the threshold value is not met, but there are pixels not "out of bounds"
        elif value[0] + value[1] > 0:

            # Defining section as a miss
            new_zoom_array.append(0)

        # If there are only "out of bounds" values
        else:
            # defining section as "out of bounds"
            new_zoom_array.append(2)

    # adding the row of hit/miss/oob identifiers to scaling object
    new_zoom.append(new_zoom_array)

# Creating an object for image generation
final = []

# Itterating over lines with hit/miss/oob identifiers
for y in new_zoom:

    # Temporary line object for RGB values
    tuppler = []

    # Itterating over hit/miss/oob identifiers per line
    for (i,value) in enumerate(y):

        # Assigning RGB values to the hit/miss/oob categories for the new image
        if y[i] == 1:
            tuppler.append(hit_rgb)
        elif y[i] == 0:
            tuppler.append(miss_rgb)
        elif y[i] == 2:
            tuppler.append(oob_rgb)

    # Compiling image array
    final.append(tuppler)

# Creating new image with correct scale
img_new = Image.new('RGBA', [int(im.size[0]/scale),int(im.size[1]/scale)], 255)
data = img_new.load()

# Colloring pixels individually
for x in range(img_new.size[0]):
    for y in range(img_new.size[1]):
        data[x,y] = final[y][x]
        # Saving the scaled image with a scal suffix
        img_new.save("Zoomed/" + files + zoom, "PNG")

# Defining images to calculate MCCs for
hits         = ['case_01_black_white_co_095_freq_30_Q0c_Illu_45',
                'case_01_black_white_co_095_freq_30_Q1c_Illu_45',
                'case_01_black_white_co_095_freq_30_Q2c_Illu_45',
                'case_01_black_white_co_095_freq_30_Q3c_Illu_45',
                'case_01_black_white_co_095_freq_30_Q4c_Illu_45',
                'case_02_black_white_co_115_freq_60_Q0c_Illu_45',
                'case_02_black_white_co_115_freq_60_Q1c_Illu_45',
                'case_02_black_white_co_115_freq_60_Q2c_Illu_45',
                'case_02_black_white_co_115_freq_60_Q3c_Illu_45',
                'case_02_black_white_co_115_freq_60_Q4c_Illu_45',
                'case_03_black_white_co_125_freq_90_Q0c_Illu_45',
                'case_03_black_white_co_125_freq_90_Q1c_Illu_45',
                'case_03_black_white_co_125_freq_90_Q2c_Illu_45',
                'case_03_black_white_co_125_freq_90_Q3c_Illu_45',
                'case_03_black_white_co_125_freq_90_Q4c_Illu_45',
                'case_04_black_white_co_067_freq_30_Q0t_Illu_25_whole',
                'case_04_black_white_co_067_freq_30_Q1t_Illu_25_whole',
                'case_04_black_white_co_067_freq_30_Q2t_Illu_25_whole',
                'case_04_black_white_co_067_freq_30_Q3t_Illu_25_whole',
                'case_04_black_white_co_067_freq_30_Q4t_Illu_25_whole',
                'case_05_black_white_co_045_freq_30_Q0t_Illu_15_whole',
                'case_05_black_white_co_045_freq_30_Q1t_Illu_15_whole',
                'case_05_black_white_co_045_freq_30_Q2t_Illu_15_whole',
                'case_05_black_white_co_045_freq_30_Q3t_Illu_15_whole',
                'case_05_black_white_co_045_freq_30_Q4t_Illu_15_whole',
                'case_06_black_white_co_058_freq_30_Q0c_Illu_45',
                'case_06_black_white_co_058_freq_30_Q1c_Illu_45',
                'case_06_black_white_co_058_freq_30_Q2c_Illu_45',
                'case_06_black_white_co_058_freq_30_Q3c_Illu_45',
                'case_06_black_white_co_058_freq_30_Q4c_Illu_45',
                'case_07_black_white_co_125_freq_30_Q0c_Illu_45',
                'case_07_black_white_co_125_freq_30_Q1c_Illu_45',
                'case_07_black_white_co_125_freq_30_Q2c_Illu_45',
                'case_07_black_white_co_125_freq_30_Q3c_Illu_45',
                'case_07_black_white_co_125_freq_30_Q4c_Illu_45',
                'case_08_black_white_co_165_freq_30_Q0c_Illu_45',
                'case_08_black_white_co_165_freq_30_Q1c_Illu_45',
                'case_08_black_white_co_165_freq_30_Q2c_Illu_45',
                'case_08_black_white_co_165_freq_30_Q3c_Illu_45',
                'case_08_black_white_co_165_freq_30_Q4c_Illu_45',
                'case_09_black_white_co_045_freq_30_Q0t_Illu_15_whole',
                'case_09_black_white_co_045_freq_30_Q1t_Illu_15_whole',
                'case_09_black_white_co_045_freq_30_Q2t_Illu_15_whole',
                'case_09_black_white_co_045_freq_30_Q3t_Illu_15_whole',
                'case_09_black_white_co_045_freq_30_Q4t_Illu_15_whole',
                'case_10_black_white_co_058_freq_30_Q0t_Illu_15_whole',
                'case_10_black_white_co_058_freq_30_Q1t_Illu_15_whole',
                'case_10_black_white_co_058_freq_30_Q2t_Illu_15_whole',
                'case_10_black_white_co_058_freq_30_Q3t_Illu_15_whole',
                'case_10_black_white_co_058_freq_30_Q4t_Illu_15_whole',
                'case_11_black_white_co_080_freq_30_Q0t_Illu_15_whole',
                'case_11_black_white_co_080_freq_30_Q1t_Illu_15_whole',
                'case_11_black_white_co_080_freq_30_Q2t_Illu_15_whole',
                'case_11_black_white_co_080_freq_30_Q3t_Illu_15_whole',
                'case_11_black_white_co_080_freq_30_Q4t_Illu_15_whole',
                ]

# Itterating over images to be analyzed
for hit in hits:

    # Fetching image horizon number
    start_Q = hit.find('Q')
    q = hit[start_Q:start_Q+3]

    # Fetching image case number
    start_case_number = hit.find('case')
    case_number = hit[start_case_number+5:start_case_number+7]

    # Determining model image to compare to
    solution = 'Model ' + q + ' Dolo' + zoom
    generated = hit + zoom

    # Opening model image and storing pixel data
    im2 = Image.open("Zoomed/" + solution)
    pix2 = im2.load
    pixels2 = list(im2.getdata())

    # Creating/resetting confusion matrix tallies
    t_n  = 0 # true negative
    t_p  = 0 # true positive
    f_n  = 0 # false negative
    f_p  = 0 # false positive
    oob  = 0 # out of bounds

    # Opening the RMS amplitude image and storing pixel data
    im1 = Image.open("Zoomed/" +generated)
```

```
pix1 = im1.load
pixels1 = list(im1.getdata())

# Running MCC script (see the created "histogram_for_MCC.py")
# The MCC script generates the performance metric
[MCC_observe, X5_MCC, X95_MCC, sensitivity_observe, X5_sensitivity, X95_sensitivity,
specificity_observe, X5_specificity, X95_specificity, error_rate_observe, X5_error_rate,
X95_error_rate, accuracy_observe, X5_accuracy, X95_accuracy, precision_observe,
X5_precision, X95_precision, false_positive_rate_observe, X5_false_positive_rate,
X95_false_positive_rate] = histogram_for_MCC.histogram_for_MCC(generated,solution,pixels1,
                                                     pixels2,itterations, hit_ratio, plot_yn)

# Tallying confusion matrix values
for (i,value) in enumerate(pixels1):

    # Positive for prediction:
    if value == hit_rgb:
        # Positive for Actual case
        if pixels2[i] == hit_rgb:
            t_p  += 1
        # Negative for Actual case
        elif pixels2[i] == miss_rgb:
            f_p += 1

    # Negative for hits:
    if value == miss_rgb:
        # Positive for Actual case
        if pixels2[i] == hit_rgb:
            f_n += 1
        # Negative for Actual case
        elif pixels2[i] == miss_rgb:
            t_n  += 1

    # Out of bounds tally
    if pixels2[i] == oob_rgb:
        oob += 1

# Finding case parameters
start_co = hit.find('co')
start_freq = hit.find('freq')
start_Illu = hit.find('Illu')

# Creating data points per surface + case + scale combination
lines.append([case_number,hit[start_Q:start_Q+2],'0.'+hit[start_co+3:start_co+6],
hit[start_freq+5:start_freq+7],hit[start_Illu+5:start_Illu+7], zoom[1:4],t_p,t_n,f_p,f_n,
        oob,MCC_observe, X5_MCC, X95_MCC, sensitivity_observe, X5_sensitivity, X95_sensitivity,
        specificity_observe, X5_specificity, X95_specificity, error_rate_observe, X5_error_rate,
        X95_error_rate,  accuracy_observe, X5_accuracy, X95_accuracy, precision_observe, X5_precision,
        X95_precision, false_positive_rate_observe, X5_false_positive_rate, X95_false_positive_rate])

# Creating a CSV file with all the data
with open('Hits.csv', 'w+', newline='') as newfile:
    writer = csv.writer(newfile, quoting=csv.QUOTE_ALL)
    writer.writerows(lines)
```