```
macro "PAGIS_v09_i"{

    //THANK YOU FOR DOWNLOADING THE THESIS ITERATON OF THE
    //PHASE ANALYSIS OF GRAYSCALE IMAGES CODE, PAGIS
    //VERSION: 09.i.
    verNo = "09_i";
    //PLEASE DO NOT RE-DISTRIBUTE
    //WILL BE DULY RELEASED UNDER:
    //Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)


    //THIS CODE CAN BE RUN WITH AN IMAGEJ PACKAGE SUCH AS FIJI
    //VIEWING OF CODE IS RECCOMMENDED WITH A TEXT READER SUCH AS NOTEPAD++
    //JavaScript LANGUAGE SETTINGS PROVIDE A SUITABLE MARKUP STYLE




    //Feature request: CHECK HERE FOR VALID STARTING FILE TYPE
    myFileName=getInfo ("image.filename");

    //get path to file dir
    myFileDir=getDir("image");
    ogDirNames = getFileList(myFileDir);
    myFilePath=myFileDir + myFileName;

    //Create a local temp dir and save image, clearing any redundant one
    tmp = getDirectory("temp");
    if (tmp=="")
        exit("No temp directory available");
    tempDir = tmp+"fijiTemp"+File.separator;

    //Old temp folder check and deletion
    if (File.exists(tempDir)){
        oldTempFilenames = getFileList(tempDir);
        for (index= 0; index < oldTempFilenames.length; index++) {
            File.delete(tempDir+oldTempFilenames [index]);
        }

        File.delete(tempDir);
    }
        File.makeDirectory(tempDir);
    if (!File.exists(tempDir))
        exit("Unable to create temp directory");

    //Remove scalebar, adjustable for any user image resolution/edge scalebar location
    print("\\Clear");
    run("Close");
    setTool("rectangle");
    waitForUser("Select image area excluding scalebar and hit OK");
    getSelectionBounds(x,y,width,height);
    makeRectangle(x, y, width, height);
    run("Crop");
    run("Options...", "iterations=1 count=1 black");

    //Histogram generation
        getStatistics(area, mean, min, max, std, histogram);
        values = newArray(256);
        histoLog = newArray(256);
        histoLogNorm = newArray(256);
        histoNorm = newArray(256);
        value = min;
        binWidth = (max-min)/256;

    //Generaion of array with log data
           for (i=0; i<256; i++) {
               values[i] = value;
               value += binWidth;
               //check + ignore 0/-ve values
               if (histogram[i] < 1) {
                    histoLog[i]= 0;
```

```
                }
                 else{
                     histoLog[i]= log(histogram[i]);
                }
            }

    //Normalize both arrays
    Array.getStatistics(histoLog,min,LogMx);
    Array.getStatistics(histogram,min,HistoMx);
            for (i=0; i<256; i++) {
                histoLogNorm[i]= (histoLog[i]/LogMx)*100;
                histoNorm[i]= (histogram[i]/HistoMx)*100;
                }
    Plot.create("Detecting Histogram Minima", "Gray Value", "Normalised Pixel Count",
    values, histoNorm);
    Plot.setFrameSize(600, 450);
    Plot.setLimits(0, 255, 0, 100)
    Plot.setLineWidth(2);
    Plot.setColor("blue");
    Plot.add("line",histoLogNorm);
    Plot.setColor("black");
    Plot.setFontSize(24, "bold");
    Plot.setAxisLabelSize(24, "bold");
    run("RGB Color");
    Plot.show;
    Plot.setLegend("Histogram\nlog Histogram");

    //Phase number definition and refinement
    Dialog.create("Number of phases");
    Dialog.addMessage("Please define the number of phases,\nincluding things like
    pores\n present in the sample");
    Dialog.addNumber("Number Present",3,0,1,"Phases");
    Dialog.show();

    //Timing bookmark to measure analysis duration
    t1 = getTime();

    //Calculation of histogram minima
    PhaseNum = Dialog.getNumber();

    //Feature request: ADD USER ADJUSTABLE SLIDER AND INTERATIVE CONFIRMATION OF
    PHASE SELECTION
    //Minima sensitivity paramater
    SensT = 100;


    minLocs= Array.findMinima(histogram, SensT);
    Array.sort(minLocs);
    print("\\Clear");
    print("Minima:");

    //Plot minima locations on histogram
    for (index= 0; index < minLocs.length; index++) {

        x= minLocs[index];
        y = histoNorm[x];
        print("x= ", x);
        toUnscaled(x,y);
        setColor("RED");
        fillOval(x-10, y-10, 20, 20);
    }

    //Duration uptdate for analysis time
    t2 = getTime();
    print("\nProcessing time 1 =", d2s((t2-t1)/1000,3), "seconds");

    //Create array to store area results
    areaResults = newArray(minLocs.length - 1);

    //Threshold and area measurement loop
    for (index= 0; index < minLocs.length -1; index++) {
```

```
140
141             //Fresh file open and scalebar removal***
142             open(myFilePath);
143             makeRectangle(0, 0, width, height);
144             run("Crop");
145             run("Options...", "iterations=1 count=1 black");
146
147             //Set threshold variables from minima calculations
148             leftThreshold = minLocs[index];
149             rightThreshold = minLocs[index+1];
150
151             //Set to correct range to measure
152             setThreshold(leftThreshold, rightThreshold);
153             setOption("BlackBackground", true);
154             run("Convert to Mask");
155
156             //Save to temp directory
157             Layer_name = myFileName+"_"+index;
158             saveAs ("Tiff",tempDir+Layer_name);
159
160             //Measurement and storage of results in an array
161             run("Set Measurements...", "area mean min area_fraction limit display
                redirect=None decimal=3");
162             run("Measure");
163             areaResults [index] = getResult("%Area",index);
164             run ("Select None");
165         }
166
167         //Store names of temp files in an array
168         layerFilenames = getFileList(tempDir);
169
170         //Merge composite, flatten and save to source dir
171         bigString = "";
172
173         for (index=0; index<layerFilenames.length; index++) {
174             bigString = bigString+" c"+index+1+"="+layerFilenames[index];
175         }
176         run("Merge Channels...", bigString+" create keep");
177         run("Flatten");
178
179         //1st processing step time check
180         t2 = getTime();
181
182         //User phase labelling
183         colourLbls = newArray("RED","GREEN","BLUE","GRAY","CYAN","MAGENTA","YELLOW");
184
185         Dialog.create("Assign a label to each identified phase");
186         for (index=0; index<layerFilenames.length; index++) {
187             Dialog.addString("Label for "+colourLbls[index]+" phase:","");
188             Dialog.addToSameRow();
189             Dialog.addCheckbox("Is real",true);
190         }
191
192         Dialog.show();
193
194         userReality = newArray(layerFilenames.length);
195         userLabels = newArray(layerFilenames.length);
196
197         //Stores and deletes labels wrt "reality index"
198         t3 = getTime();
199         for (index=0; index<layerFilenames.length; index++) {
200             userReality [index] = Dialog.getCheckbox();
201             userLabels [index] = Dialog.getString();
202         }
203
204
205         //Draws labels and keys
206         lpCnter = 0;
207         for (index=0; index<layerFilenames.length; index++) {
208             if (userReality [index] > 0){
209                 makeRectangle(50,50+(lpCnter*150),120,120);
```

```
            setColor("black");
            run("Fill", "slice");
            setColor(colourLbls [index]);
            makeRectangle(60,60+(lpCnter*150),100,100);
            run("Fill", "slice");
            setColor("black");
            setFont("Arial",60,"Bold");
            setFont("Arial",60,"Bold");
            drawString(userLabels [index],200,150+(lpCnter*150),"white");
            lpCnter = lpCnter + 1;
            }
        else {
            close(layerFilenames [index]);
            File.delete(tempDir+layerFilenames [index]);
            }
    }

    //Flatten and save version of first composite image
    saveAs ("Tiff", myFileDir+myFileName+"_phasecompositelabel");
    run("Images to Stack", "name=Stack title=[] use");

    rowCnter = 2;
    if (lpCnter < 2) {;
    }
         else if (lpCnter < 4) {
            rowCnter = 3;
            }
            else if (lpCnter < 6) {
                rowCnter = 4;
                }
                else if (lpCnter > 5) {
                    rowCnter = 5;
                    }
    run("Make Montage...", "columns=2 rows="+rowCnter+" scale=0.50 border=5 label");
    close("\\Others");
    saveAs ("Tiff", myFileDir+myFileName+"_phaseselection-montage");

    //Processing time checkpoint
    t4 = getTime();
    print("Image 1 processing time =", d2s(((t4-t3)+(t2-t1))/1000,3), "seconds");
    print("\n");

    //Update temp directory file list
    layerFilenames = getFileList(tempDir);

    //normalization factor and results printing
    areaResNorm = areaResults;
    normFactor = 0;
    phaseCounter = 0;
    phaseNumber = newArray(areaResults.length);
    Array.fill(phaseNumber,0);

    //Delete data from non-real phases
    for (index=0; index<areaResults.length; index++) {
        if(userReality [index] <1){
            areaResNorm[index] = 0;
        }
        else{
            normFactor = normFactor + areaResNorm[index];
            phaseCounter = phaseCounter + 1;
            phaseNumber[index] = phaseCounter;
        }
    }

    //Normalize remaining data and dialogs with labels, confirmation window presented
        Dialog.create("Confirm phase selection and set source");
    for (index=0; index<areaResults.length; index++) {
        if(userReality [index] >0) {
            areaResNorm[index] = (areaResNorm[index]/normFactor)*100;
            Dialog.addMessage("Phase "+phaseNumber[index]+", "+colourLbls[index]+": "+
            userLabels[index]+", "+d2s(areaResNorm[index],1)+"% (norm.)");
```

```
        }
    }
    Dialog.addDirectory("Please select the directory of image set", myFileDir);
    Dialog.show();
    dataSource = Dialog.getString();

    //Time marker
    t5 = getTime();

    //Collect filenames in datasource folder, open all and create stack
    dataList = getFileList(dataSource);
    close("*");

    //Measuere stack
    //Feature request: CHECK FOR VALID PHASES ONLY
    dataSetResults = newArray(areaResults.length);

    for (index= 0; index < minLocs.length-1; index++) {
        for (j=0; j<dataList.length; j++) {
            open(dataSource+"/"+dataList[j]);
        }

        run("Images to Stack", "name=Stack title=[] use");

        makeRectangle(0, 0, width, height);
        run("Crop");
        run("Options...", "iterations=1 count=1 black");

        //Set threshold variables from minima calculations
        leftThreshold = minLocs[index];
        rightThreshold = minLocs[index+1];

        //Set to correct range to measure
        setThreshold(leftThreshold, rightThreshold);
        setOption("BlackBackground", true);
        run("Convert to Mask", "method=Default background=Dark black");


        //Measurement and storage of results in an array
        run("Set Measurements...", "area mean min area_fraction limit display
        redirect=None decimal=3");
        run("Measure");
        //Feature request: ADD (SLOWER) INDIVIDUAL MEASURMENT AS TO ALLOW FOR AVERAGE
        OF STACK AND ERROR CALC
        dataSetResults [index] = getResult("%Area",index);
        print(userLabels[index]+" phase, measured between ("+leftThreshold+"-"+
        rightThreshold+"): " +d2s(dataSetResults[index],1)+"%");
        run ("Select None");
        close("*");
    }

    //Total set processing time checkpoint
    t6 = getTime();
    print("\nImage SET processing time =", d2s((t6-t5)/1000,3), "seconds");

    print("\nThank you for using\nP.hase A.nalysis G.rayscale I.image
    S.cript\nVersion_"+verNo);

```