Image fluorescence processing sudo code

1. Get target folder
2. Get images type
3. Loop through opening files
   1. Extract filename, lens, current, exposure, etc.
   2. Loop through drawing number of lines based on file type
      1. Starting guess based on file type
      2. Move until confirmed line found
      3. Take profile of line
      4. Take data on background and noise
      5. Find sets of peaks
      6. Take mean, min and max of peaks
      7. Store filename, lens, current, exposure, location/descriptor, mean background, stdDev background, and each concentration’s mean, min and max

Functions:

Findline

Find peaks

Stable

macro "GALT [a]" {

var dir = getString("This macro takes a file path which is the target directory. It loops through the files in this target directory to perform analysis. Please enter the target directory file path.", "D:/GALT FL Images/16mm lens/vertical rows center");

if ( !endsWith(dir, "/") ) {

dir = dir + "/";

}

function append(arr, value) {

arr2 = newArray(arr.length+1);

for (i=0; i<arr.length; i++) {

arr2[i] = arr[i];

}

arr2[arr.length] = value;

return arr2;

}

res = newArray;

if (File.isDirectory(dir)) {

files = getFileList(dir); // get array of files

// loop through files

for (i=0; i<files.length; i++) {

// open file

var filename = files[i];

res = append(res, filename);

var pth = dir + filename;

open(pth);

// starting guess parameters for line finding

var lineTop = 700;

var lineBottom = 2300;

var lineColumn = 2487;

var pixels = 20; // number of pixels between centers is about 14

// find nearby line with brighest pixel

var lastMax = 0;

for (j=0; j<pixels; j++) {

makeLine(lineColumn+j, lineBottom, lineColumn+j, lineTop);

vals = getProfile();

Array.getStatistics(vals, min, max, mean, stdDev);

if (max < lastMax) {

makeLine(lineColumn+j-1, lineBottom, lineColumn+j-1, lineTop);

vals = getProfile();

break;

} else {

lastMax = max;

}

}

// show line profile

Plot.create("Profile " + filename, "Row", "Intensity", vals);

Plot.show();

// characterize noise

var length = vals.length;

endNoise = Array.slice(vals, length-300, length);

Array.getStatistics(endNoise, minNoise, maxNoise, meanNoise, stdDevNoise);

res = append(res, meanNoise);

res = append(res, stdDevNoise);

// find sets

var setLength = 100; // max number of vertical pixels a set takes up

var tol = mean + 3\*stdDevNoise;

maxima = Array.findMaxima(vals, tol);

var expectedPeaks = 5; // expected number of wells per set

for (k=0; k<maxima.length-expectedPeaks+1; k++) {

set = Array.slice(maxima, k, k+expectedPeaks);

Array.getStatistics(set, minSet, maxSet, meanSet, stdDevSet);

if ( (maxSet - minSet) <= setLength ) {

res = append(res, vals[set[4]]);

k = k + expectedPeaks-1;

}

}

// let user look before closing

waitForUser;

close(filename);

}

Array.print(res);

} else {

waitForUser("Error: Not a valid directory entry.", "It appears the file path you've entered is not recognized as a directory. Please check your file path and run this macro again.");

}

}

macro "GALT [a]" {

var dir = getString("This macro takes a file path which is the target directory. It loops through the files in this target directory to perform analysis. Please enter the target directory file path.", "C:/Gener8/GLT/GALT FL Images/16mm lens/vertical rows center");

if ( !endsWith(dir, "/") ) {

dir = dir + "/";

}

function append(arr, value) {

arr2 = newArray(arr.length+1);

for (i=0; i<arr.length; i++) {

arr2[i] = arr[i];

}

arr2[arr.length] = value;

return arr2;

}

Table.create("Fluorescence");

if (File.isDirectory(dir)) {

files = getFileList(dir); // get array of files

Table.setColumn("File Name", files);

// loop through files

for (i=0; i<files.length; i++) {

// open file

var filename = files[i];

var pth = dir + filename;

open(pth);

// starting guess parameters for line finding

var lineTop = 700;

var lineBottom = 2300;

var lineColumn = 2487;

var pixels = 20; // number of pixels between centers is about 14

// find nearby line with brighest pixel

var lastMax = 0;

for (j=0; j<pixels; j++) {

makeLine(lineColumn+j, lineBottom, lineColumn+j, lineTop);

vals = getProfile();

Array.getStatistics(vals, min, max, mean, stdDev);

if (max < lastMax) {

makeLine(lineColumn+j-1, lineBottom, lineColumn+j-1, lineTop);

vals = getProfile();

break;

} else {

lastMax = max;

}

}

// show line profile

Plot.create("Profile " + filename, "Row", "Intensity", vals);

Plot.show();

// characterize noise

var length = vals.length;

endNoise = Array.slice(vals, length-300, length);

Array.getStatistics(endNoise, minNoise, maxNoise, meanNoise, stdDevNoise);

Table.set("Mean Noise", i, meanNoise);

Table.set("Noise Standard Deviation", i, stdDevNoise);

// find sets

var setLength = 100; // max number of vertical pixels a set takes up

var tol = mean + 3\*stdDevNoise;

// var tol = maxNoise\*2;

Table.set("Max Noise", i, maxNoise);

maxima = Array.findMaxima(vals, tol);

Array.sort(maxima);

Array.print(maxima);

var expectedPeaks = 5; // expected number of wells per set

res = newArray;

for (k=0; k<maxima.length-expectedPeaks+1; k++) {

set = Array.slice(maxima, k, k+expectedPeaks);

Array.getStatistics(set, minSet, maxSet, meanSet, stdDevSet);

if ( ((maxSet - minSet) <= setLength) && {

res = append(res, vals[set[4]]);

k = k + expectedPeaks-1;

}

}

for (m=0; m<res.length; m++) {

Table.set("Set " + m, i, res[m]);

}

// let user look before closing

waitForUser;

close(filename);

}

Table.update;

} else {

waitForUser("Error: Not a valid directory entry.", "It appears the file path you've entered is not recognized as a directory. Please check your file path and run this macro again.");

}

}