Source codes for image analysis

ImageJ macro for conversion of RGB files to 8 bit, followed by orientationJ analysis:

```
_// "BatchOrientationJ_Distribution"
//
// Questions? Contact Ian Mellis (ian.mellis at gmail.com)
//
// Based on architecture of
https://imagej.nih.gov/ij/macros/BatchProcessFolders.txt
//
// This macro uses pre-specified OrientationJ Distribution settings to batch-
process
// all samples in a single experiment. Expects file structure, e.g.,
//
//      /path/to/EXPERIMENT1/SAMPLE1NAME/SAMPLE1NAME_Snapshot1.tif
//
// When prompted, select the /path/to/EXPERIMENT1/ folder containing all
sample
// folders. This macro will then loop through all samples and process all images
// for each sample using the OrientationJ Distribution function.
//
// Results will be saved in a new folder, e.g.,
//
//
/path/to/EXPERIMENT1/BatchOrientationJ_Distribution_Results_20170101_10h
30m/
//
// Current OrientationJ_Distribution settings, as specified by Aman Kaur:
// - sigma = 3
// - gradient = 4 // i.e., Gaussian
// - energy = on
// - orientation = on
// - coherency = on
// - color-survey = on
// - hue = Orientation
// - sat = Coherency
// - bri = Original-Image
//
// If you want to change OrientationJ settings, edit orientFile() below.

dir = getDirectory("Choose an Experiment Directory ");
getDateAndTime(year, month, dayOfWeek, dayOfMonth, hour, minute, second,
msec)
```

```
timeStamp = ""+year+"-"+(month+1)+"-"+dayOfMonth+"_"+hour+"h"+minute+"m"
resultsDir = dir+"/BatchOrientationJ_Distribution_Results_"+timeStamp+"/"
File.makeDirectory(resultsDir);

sampCount = 0;
imgCount = 0;
countSamplesAndImages(dir, resultsDir);
processSamples(dir, resultsDir);
print(sampCount+" samples processed");
print(imgCount+" total images processed");

function countSamplesAndImages(dir, resultsDir) {
   list = getFileList(dir);
   for (i=0; i<list.length; i++) {
       if (endsWith(list[i], "/")) {
           if (!startsWith(list[i], "BatchOrientationJ_Distribution_Results")){
               sampCount++;
               countSamplesAndImages(""+dir+list[i], resultsDir);
           }
       } else {
          if(endsWith(list[i], ".tif")) {
             imgCount++;
          }
       }
   }
}

function processSamples(dir, resultsDir) {
   list = getFileList(dir);
   for (i=0; i<list.length; i++) {
      if (endsWith(list[i], "/")) {
          if (!startsWith(list[i], "BatchOrientationJ_Distribution_Results")){
          // for a sample folder containing multiple snapshots
          File.makeDirectory(resultsDir+list[i]); // make subfolder in results
          processSamples(""+dir+list[i], resultsDir+list[i]); // call processSamples for
snapshots in this sample folder
          }
      } else {
          // for a snapshot in a sample folder
          path = dir+list[i]; // full path of snapshot file
          snapName = File.getName(list[i]); // snap basename
          snapBase = replace(snapName, ".tif", "");
          orientFile(path, snapBase, resultsDir); // call orientFile on snapshot
       }
   }
}
```

```
function orientFile(path, snapName, resultsDir) {
   if (endsWith(path, ".tif")) {
      thisSampsResultsFile = resultsDir+snapName+"_distribution.xls";
      oriCall = "log=0.0 tensor=3.0 gradient=4 min-coherency=0.0 min-energy=0.0
harris-index=on color-survey=on s-distribution=on hue=Orientation
sat=Coherency bri=Original-Image filename='"+thisSampsResultsFile+"'";
      open(path);
      run("8-bit");
      run("OrientationJ Distribution", oriCall);
      run("Close All");
   }
}
```

R code for normalization of images obtained from orientation analysis:

```
#call libraries
library(dplyr)
library(xlsx)
library(readxl)

#working directory to import/export files (keep setting as needed)
setwd("path name here")
getwd()

#import files
dat <- read.delim("example.xls")

# Copy the code into the console
a <- dat[which(dat$Y == max(dat$Y)),]      # get the x and Y values for row with max
Y
print a                                    # if a=0, the code will not work; simply
export the file as xlsx
af <- as.numeric(a$X)                      # get the value of X with largest Y in a
numeric class matrix
Xm <- dat[,1]                              # extract column X
Ym <- dat[,2]                              # extract column Y
Yf <- data.frame(Ym)                       # convert column Y values to data frame
to combine with X later
Xm2 <- Xm-af                               # modify Xm to subtract mode value
Xm2m <- matrix(Xm2)                        # convert Xm to matrix

if (af>0) {
  #Part 1 --> #if af>0,
  Xm3 <-  Xm2m[1:af,]
  Xm4 <- Xm3+180                                   # vector for manipulating mode
with 180 values
  Xm4m <- matrix(Xm4)                       # convert Xm4 from integer to matrix
values
  Xf <- replace(Xm2m,1:af, Xm4m)                   # new manipulated values for
Column X
  Xfm <- data.frame(Xf)                     # convert Xf from matrix to data.frame for
combining columns
} else {
  #Part 2 --> if af<0, use -180
  negaf<- 181+af
  Xm3 <-  Xm2m[negaf:180,]
  Xm4 <- Xm3-180                            # vector for manipulating mode with 180
values
  Xm4m <- matrix(Xm4)                       # convert Xm4 from integer to matrix
values
  Xf <- replace(Xm2m,negaf:180, Xm4m)              # new manipulated values for
Column X
  Xfm <- data.frame(Xf)                     # convert Xf from matrix to data.frame for
combining columns
}

#Combining columns and plotting here
filenew<- bind_cols(data.frame(Xfm), data.frame(Yf) # combine columns of X and Y
to make a new data frame
cname<- c("X", "Y")                                 # make vector with column
names
colnames(filenew) <- cname                               # tag column names to
matrix
filef <- filenew[order(filenew$X),]
plot(filef)                                        # a. plot data file

#rename file name as desired for output and view table if desired
write.xlsx(filef, "example.xlsx")                # export results to excel


#combine files to form a master file with all reading per sample
filenames <- list.files(pattern = ".xlsx")                # prepares a list of all
filenames with the pattern in them
comb <- as.data.frame(lapply(filenames, read_excel))      # combines files found in
filenames list as a data frame
View(comb)                                                # shows the combined data
frame in the RStudio
fulldata <- comb[,c(2,3,6,9,12,15,18,21,24,27)]           # extracts desired columns
from the data frame
```

```
write.xlsx(fulldata, "GM01948_AG11726aged.xlsx")   # export final table to excel
```