

High-Content Scan 2 (HCSan2) Users Guide

Everett Ramer

ramer@udel.edu

Contents

<i>HCSan2</i> Images.....	3
How to Run <i>HCSan2</i>	4
<i>HCSan2</i> Results	7
Appendix A Assay Parameters.....	11
Appendix B Macro Code	15

High-Content Scan2 (*HCSan2*) is an *ImageJ* macro that performs an automated high-content analysis of two-channel fluorescence images. In the interactive mode the user can use the ROI Manager to view the cell/ROI corresponding to each measurement, a feature that facilitates the optimization of assay parameters. An Excel-compatible report and zipped folder containing images of each cell/ROI are generated, allowing users to archive and publish results at the cellular level.

High-content screening is based on a cell-by-cell analysis of multi-channel images similar to the one shown in Fig. 1:

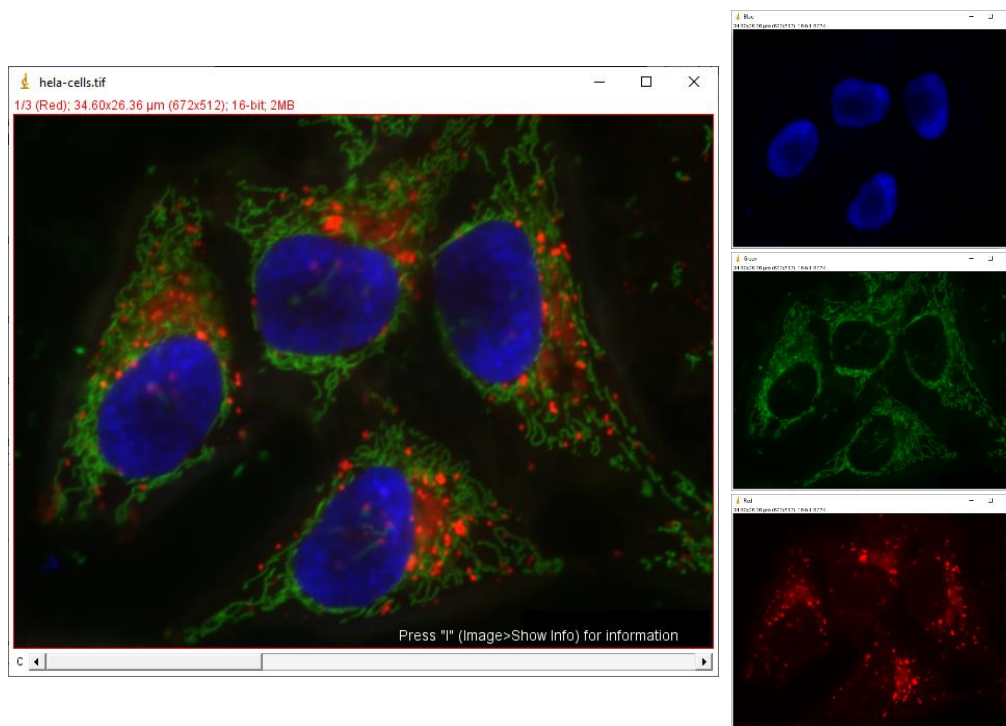


Figure 1 Three-channel fluorescence image of HeLa cells with blue nucleus, green mitochondria, and red lysosomes.

Left: 16-bits/channel composite color image, Right: Individual channel images.

Image from ImageJ: *Files>>Open Samples* courtesy of Tony Collins, creator of the *ImageJ for Microscopy* collection of plugins at <http://www.macbiophotonics.ca/imagej/>.

The key components of high-content analysis are:

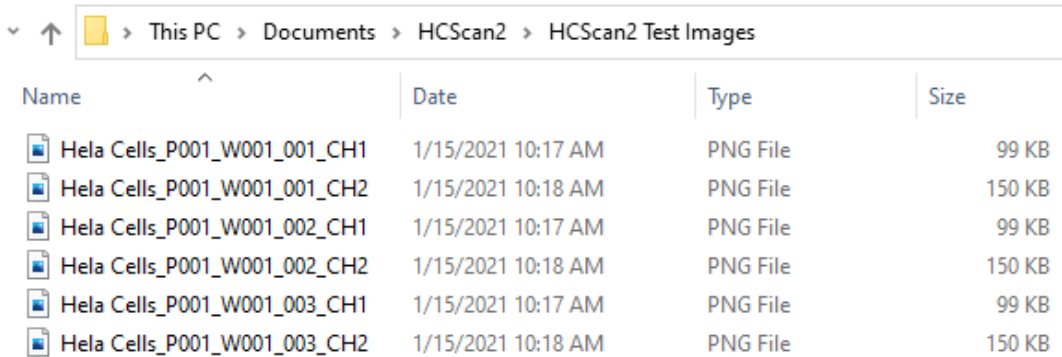
1. *CH1* images contain fluorescence-labeled nuclei. These are used to locate the cells. As Fig. 1 illustrates, in addition to being very bright, the labeled nuclei are relatively large and fairly circular in shape. These morphological parameters are used to distinguish them from other bright objects, e.g., fibers from wipes.
2. *CH2* images contain fluorescence-labeled target compounds. These appear as small spots or diffuse areas of luminance confined to the nucleus or cytoplasm of the cell.
3. Circle and Ring masks are created from the nuclei boundaries. The intensities measured under these masks in the *CH2* images indicate the concentrations of the target compounds in those regions of the cell.

HCScan2 Images

Prepare your multi-channel images in a file folder with the channels stored as separate, 8-bit, gray-scale images and titled as

name1_CH1.type
name1_CH2.type
name2_CH1.type
name2_CH2.type
.
.
.

The *name* field can be any valid file name characters, including sequence number, and the *type* field can be any file type supported by *ImageJ*. The name fields must match for each image pair. For example, your image folder might appear as in Fig. 2.









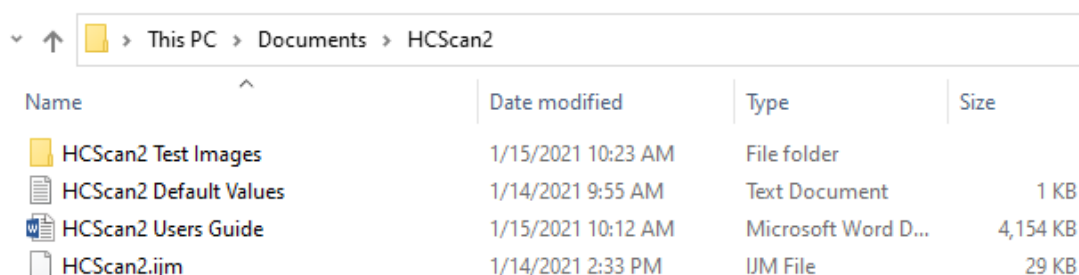
Name	Date	Type	Size
 Hela Cells_P001_W001_001_CH1	1/15/2021 10:17 AM	PNG File	99 KB
 Hela Cells_P001_W001_001_CH2	1/15/2021 10:18 AM	PNG File	150 KB
 Hela Cells_P001_W001_002_CH1	1/15/2021 10:17 AM	PNG File	99 KB
 Hela Cells_P001_W001_002_CH2	1/15/2021 10:18 AM	PNG File	150 KB
 Hela Cells_P001_W001_003_CH1	1/15/2021 10:17 AM	PNG File	99 KB
 Hela Cells_P001_W001_003_CH2	1/15/2021 10:18 AM	PNG File	150 KB

Figure 2 Image folder.
Image name includes Plate, Well and Sequence numbers

How to Run HCSan2

Copy the *HCSan2* folder to your *Documents* folder. This folder contains the *HCSan2* macro (.ijm), the Users Guide (.docx), a file of default assay parameters (.txt), and a set of test images, as shown in Fig. 3.



Name	Date modified	Type	Size
HCSan2 Test Images	1/15/2021 10:23 AM	File folder	
HCSan2 Default Values	1/14/2021 9:55 AM	Text Document	1 KB
HCSan2 Users Guide	1/15/2021 10:12 AM	Microsoft Word D...	4,154 KB
HCSan2.ijm	1/14/2021 2:33 PM	IJM File	29 KB

Figure 3 Copy *HCSan2* folder to *Documents*.

If you have multiple users on your computer there may be problems with other users accessing your *Documents* folder.

To install *HCSan2*, launch *ImageJ* and click on the *Plugins>> Macros>>Install* menu item, see Fig. 4.

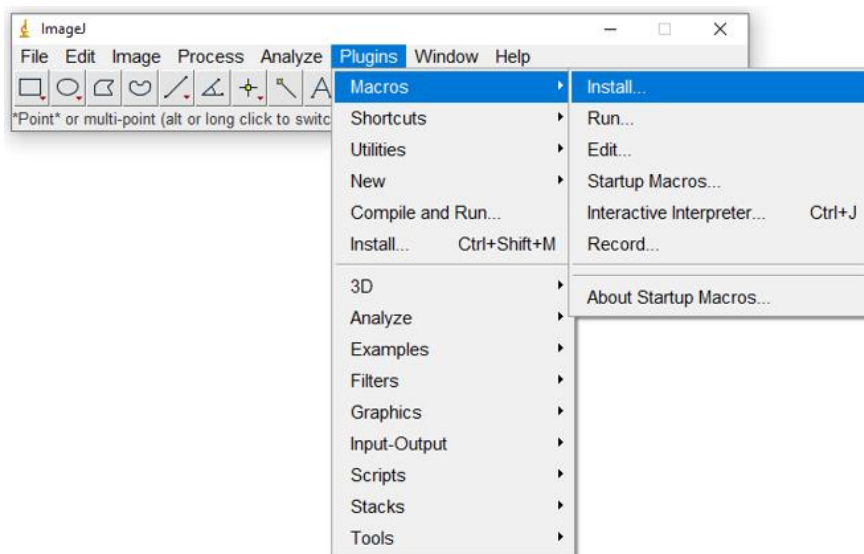


Figure 4 In *ImageJ*, select the *Plugins>> Macros>>Install* menu item.

The *Install Macros* dialog will appear and allow you to browse to the *HCSan2* folder and select *HCSan2.ijm*, see Fig. 5.

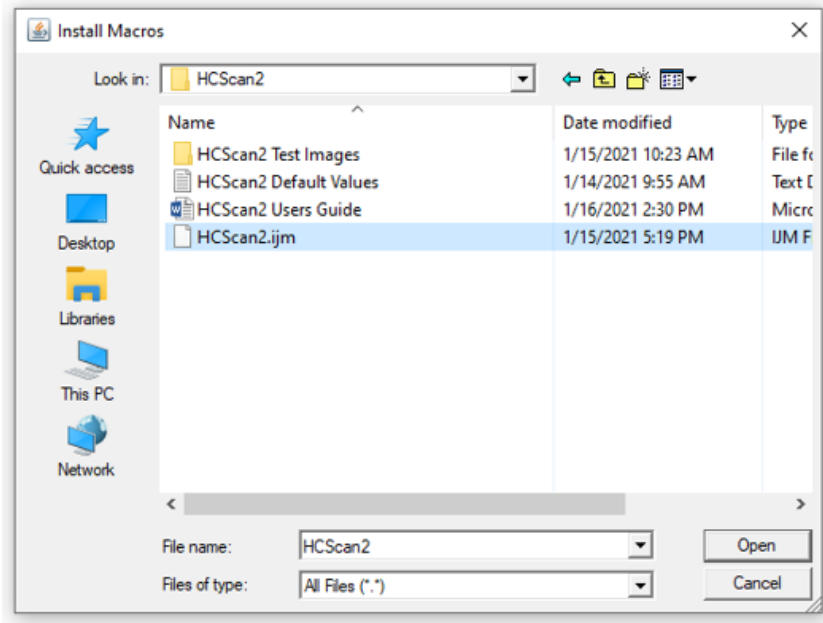


Figure 5 In the *Install Macros* dialog browse to the *HCSan2* folder and click on *HCSan2.ijm*.

Installing *HCSan2* will put it in the *Plugins>>Macros* list, see Fig. 5, and you can launch it by clicking this entry.

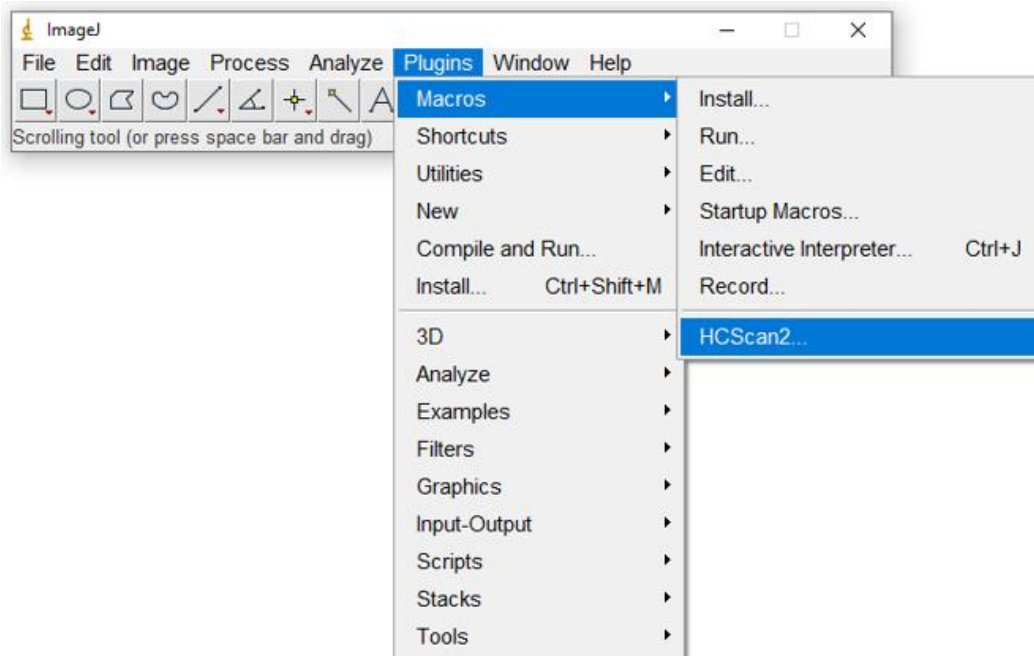


Figure 5 After *HCSan2* is installed, you can run it by clicking on *HCSan2* in the *Plugins>>Macros* list.

After *HCSan2* is launched you will see the *Assay Parameters* dialog shown in Fig. 6. These parameters are described in Appendix A, at the end of this document.

The values displayed are the ones currently stored in the file *HCSan2 Default Values.txt*. Check the *Load Default Values* box to load values from a different file, then click the *OK* button to execute that command. The *Open Defaults File* dialog will allow you to browse to the file you want to load. The *Assay Parameters* dialog will reappear after you select the file and click the *Open* button, and will contain the assay parameters you just loaded.

You can edit the values in the *Assay Parameters* dialog. To save them for future analyses, check the *Save as Default Values* box, then click the *OK* button. The *Save Defaults File* dialog will allow you to browse to the folder you want and save the default values in the file of your choice. The *Assay Parameters* dialog will reappear after you select the file and click the *Open* button.

Figure 6 *HCSan2* assay parameters.

To begin the analysis, click *OK* with both *Load Default Values* and *Save Default Values* unchecked. Clicking *Cancel* will stop the macro without beginning the analysis.

At the beginning of the analysis *HCSan2* will ask you to select an image file. Use the *Open File* select window to browse to the folder containing the image you want to analyze, select it and click *Open*, see Fig. 7. It does not matter which channel of the two-channel image pair you select, *HCSan2* will open both of them.

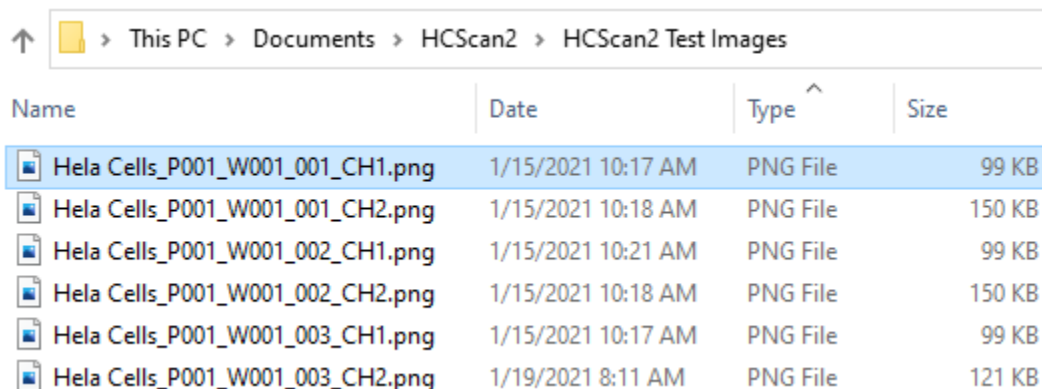


Figure 7 HCSan2 image selection window. Select one file of the two-channel image pair and click *Open*.

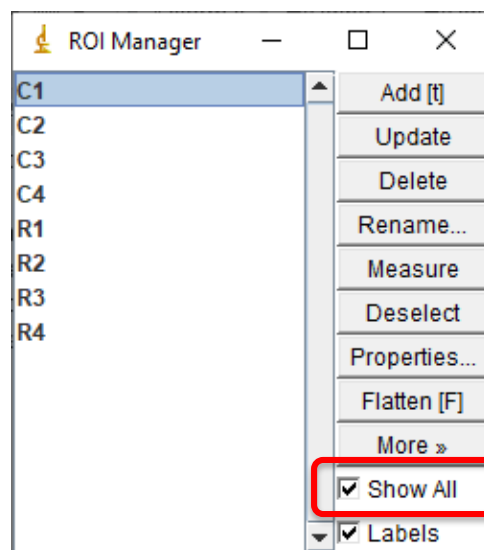
HCSan2 Results

When the analysis is finished your desktop will display the following

1. *ImageJ* menu and tool bar
2. *CH1* image
3. *CH2* image
4. *Results* table
5. *ROI Manager*

The *ROI Manager* contains a combined list of circle and ring ROI's. The circle ROI's are listed first, and are labeled C1, C2,... The ring ROI's have the same labels, but begin with "R". You can select an ROI by clicking, as shown in Fig. 8.

Figure 8 *ROI Manager* used to control the display of ROI's on the analyzed images.



The selected ROI is selected is highlighted on the active image, as shown in Fig. 9. If the *Show All* box of the *ROI Manager* is, unchecked, only the selected ROI is displayed. If *Show All* is checked, all ROI's are displayed and the selected ROI is

highlighted in a contrasting color. Note that checking *Show All* displays the ring ROI's because they are at the end of the list and are drawn on top of the circle ROI's. The *ROI Manager* has a number of tools, listed on the right side, that can be used to further control ROI display.

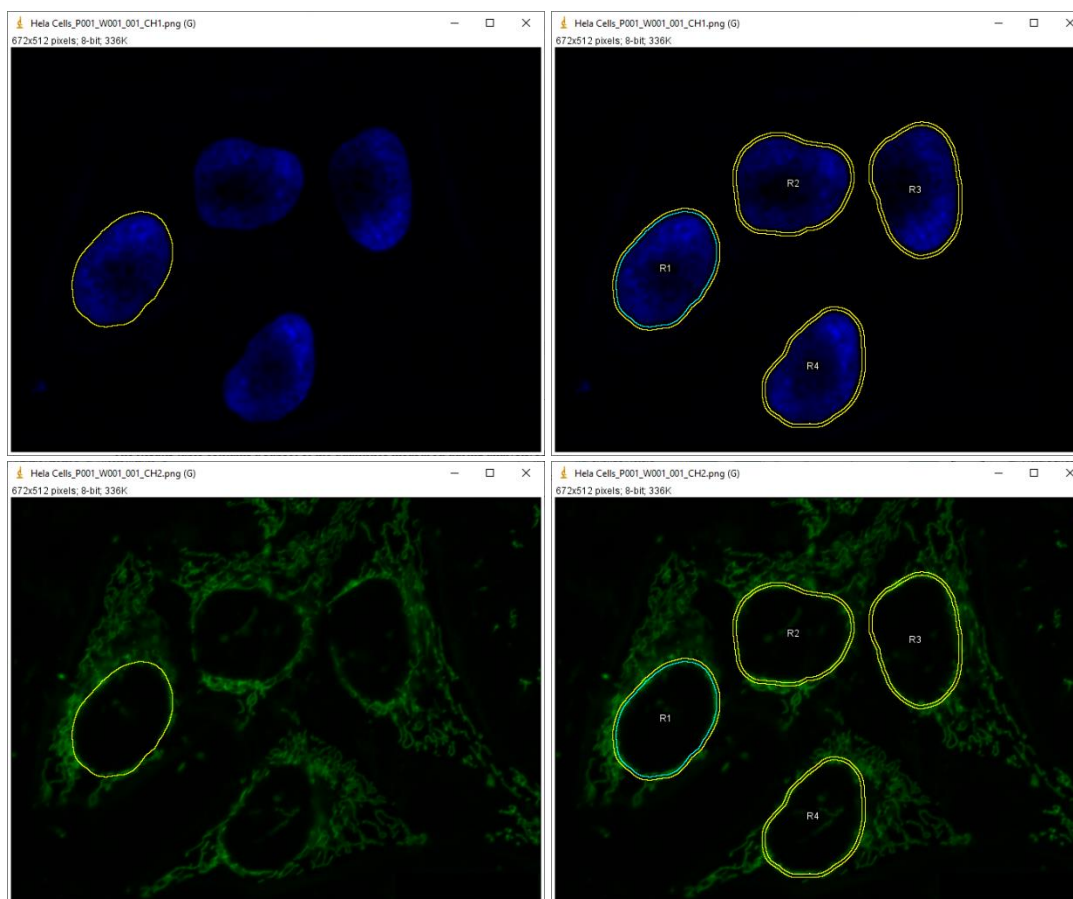
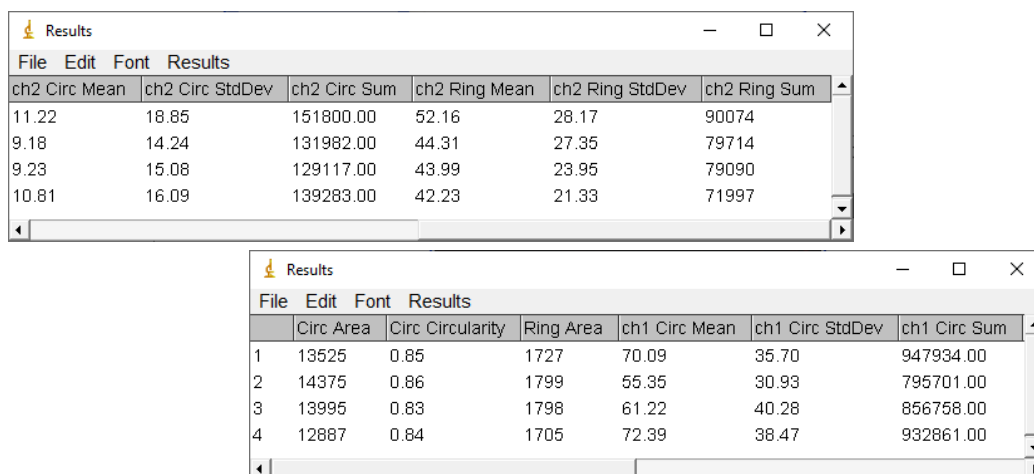


Figure 9 Overlays displayed for ROI C1 selected (as shown in Fig. 8).
Left: *Show All* box unchecked, Right: *Show All* box checked

The *Results* table contains a subset of the quantities measured during analysis, see Fig. 10. The first column is label of the object as it appears in the *ROI Manager*. The next columns give the area and circularity of its circle mask followed by the area of its ring mask. The area and circularity are the morphological parameters used to select valid objects. The intensity data are next, first the circle mask intensities in *CH1*, which are also used to select valid objects, followed by the circle and ring mask intensities in *CH2*, which are the target measurements.

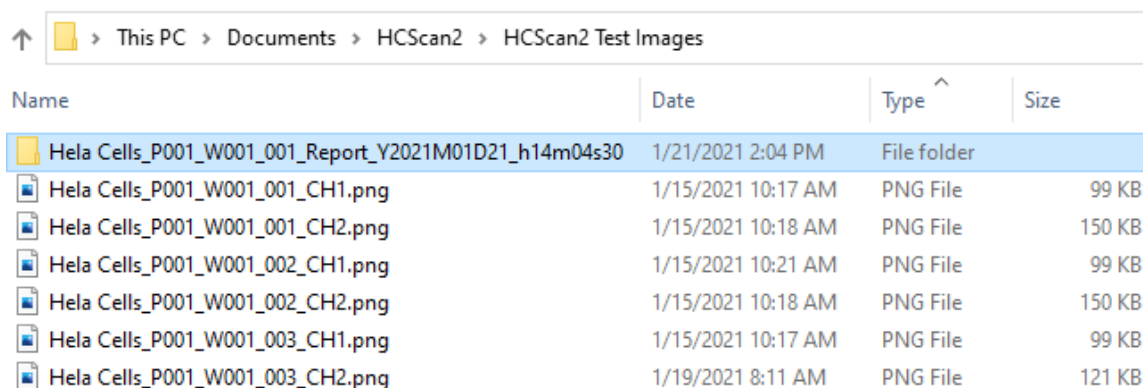


ch2 Circ Mean	ch2 Circ StdDev	ch2 Circ Sum	ch2 Ring Mean	ch2 Ring StdDev	ch2 Ring Sum
11.22	18.85	151800.00	52.16	28.17	90074
9.18	14.24	131982.00	44.31	27.35	79714
9.23	15.08	129117.00	43.99	23.95	79090
10.81	16.09	139283.00	42.23	21.33	71997

	Circ Area	Circ Circularity	Ring Area	ch1 Circ Mean	ch1 Circ StdDev	ch1 Circ Sum
1	13525	0.85	1727	70.09	35.70	947934.00
2	14375	0.86	1799	55.35	30.93	795701.00
3	13995	0.83	1798	61.22	40.28	856758.00
4	12887	0.84	1705	72.39	38.47	932861.00

Figure 10 Results table contains a subset of the measured quantities.

A new report folder will appear in the folder that contained the image you just analyzed, see Fig. 11.

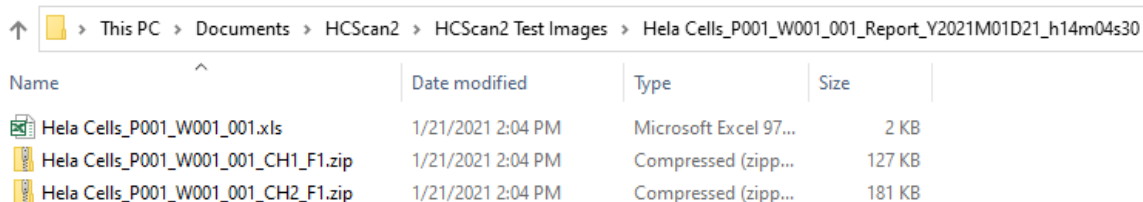


Name	Date	Type	Size
Hela Cells_P001_W001_001_Report_Y2021M01D21_h14m04s30	1/21/2021 2:04 PM	File folder	
Hela Cells_P001_W001_001_CH1.png	1/15/2021 10:17 AM	PNG File	99 KB
Hela Cells_P001_W001_001_CH2.png	1/15/2021 10:18 AM	PNG File	150 KB
Hela Cells_P001_W001_002_CH1.png	1/15/2021 10:21 AM	PNG File	99 KB
Hela Cells_P001_W001_002_CH2.png	1/15/2021 10:18 AM	PNG File	150 KB
Hela Cells_P001_W001_003_CH1.png	1/15/2021 10:17 AM	PNG File	99 KB
Hela Cells_P001_W001_003_CH2.png	1/19/2021 8:11 AM	PNG File	121 KB

Figure 11 Analysis report folder in folder containing image.

The report folder will contain the following files, as shown in Fig. 12

1. An *Excel* compatible file of analysis results
2. A *.zip* file containing the *CH1* image fields with ROI's
3. A *.zip* file containing the *CH2* image fields with ROI's



Name	Date modified	Type	Size
Hela Cells_P001_W001_001.xls	1/21/2021 2:04 PM	Microsoft Excel 97...	2 KB
Hela Cells_P001_W001_001_CH1_F1.zip	1/21/2021 2:04 PM	Compressed (zipp...	127 KB
Hela Cells_P001_W001_001_CH2_F1.zip	1/21/2021 2:04 PM	Compressed (zipp...	181 KB

Figure 12 Contents of analysis report folder.

All of the results, including assay parameters are saved to the *Excel*-readable file (.xls). The report file may be opened by double clicking; however, you will have to click *YES* in response to the *Excel* question.

The two .zip files that contain fields of the analyzed image with ROI's. These images can be opened using *ImageJ* and look just like those shown in Fig. 9, except that the *ROI Manager* cannot be used to select individual ROI's. You can copy and paste these images into Excel or Word documents.

Note About Memory

Unlike other Windows applications ImageJ will only use the memory allocated to it. You can change the allocated memory to equal 75% of total RAM via the menu command: "*Edit/Options/Memory*". Specifying more than 75% of real RAM results in virtual RAM being used causing ImageJ to become very slow and unstable. See <http://rsb.info.nih.gov/ij/docs/install/>.

Appendix A Assay Parameters

The parameters *HCSan2* uses to configure the analysis are described below:

1. *Field Rows* and *Field Columns*: Used to specify the number of rows and columns of non-overlapping fields into which the image is divided for analysis. This feature allows the macro to be run on computers that do not have enough memory to analyze an entire image. Fields are numbered 1 through $Rows \cdot Columns$ from left-to-right and top-to-bottom, as shown in Fig. 11. Setting both parameters to “1” will analyze the entire image as one field.

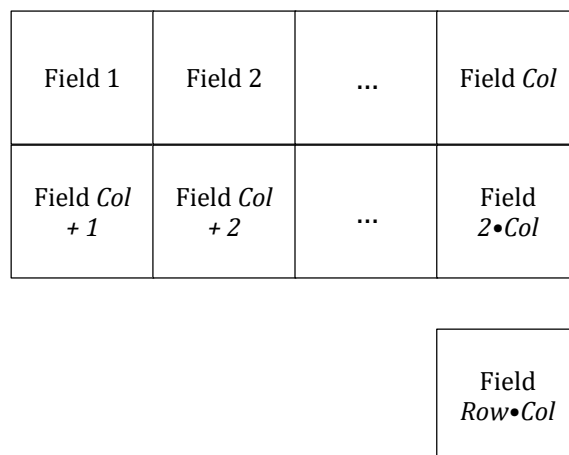


Figure 11 Field numbering scheme.

2. *Analyze Field*: Used to analyze a single, specified field of the row-column array described above. Setting this parameter to 0 will analyze all fields of the array. A parameter value other than 0 will allow you to review the analysis of the specified field.
3. *Stop Analysis after _____*: Stops the macro at points in the analysis so you can *interactively set* various assay parameters. For example, if you stop after the images are loaded you can manually execute the background removal tool with different rolling ball radii, then manually try different threshold methods and values.
4. *Stop for Analysis Review*: Checking this box will stop the analysis after the specified image field. The macro will leave the *CH1* and *CH2* images on the desktop so you can examine the results using the ROI Manager.
5. *Rolling Ball Radius*: Used to remove the background on both *CH1* and *CH2* images. The *Before* image in Fig. 12 has an uneven background that is higher on the left than on the right. The plot below the image represents the pixel values in the horizontal strip slightly below the center of the image. Imagine rolling a ball along the underneath surface of the image and setting every pixel touched by the ball to zero. The result is shown on the right: All that remains in the intensity profile of the image are the peaks that were too small for the ball to enter. The

radius of the ball must be larger than the radius of the smallest object of interest in *CH1*.

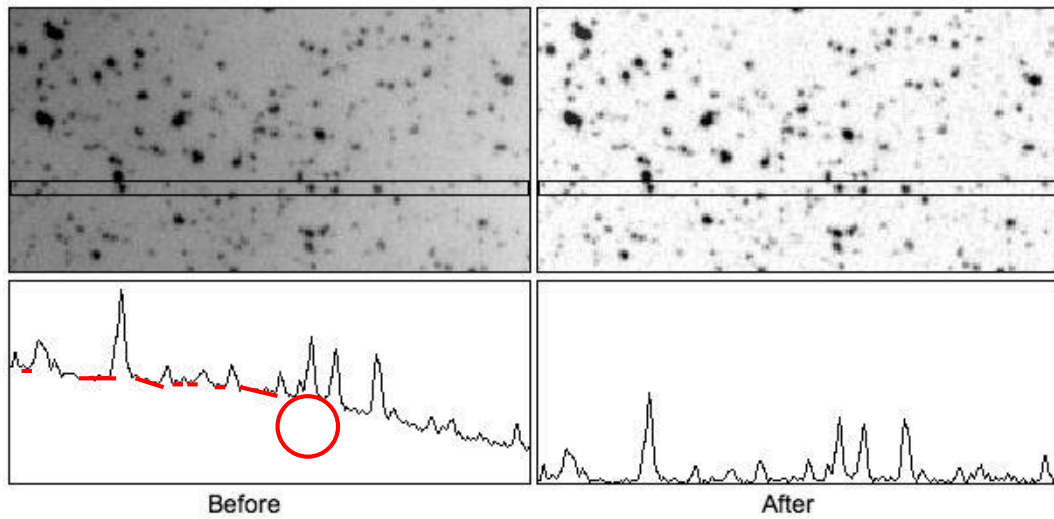


Figure 12 Rolling ball background removal.

(Figure from http://imagejdocu.tudor.lu/doku.php?id=gui:process:subtract_background)

6. *Smooth Radius*: Used to agglomerate *CH1* objects inside smooth boundaries. Object boundaries must be drawn in regions where object pixel intensities are very close to background pixel intensities, see Figure 13. Thus, boundary shape is strongly affected by noise. Smoothing an image before thresholding results in smoother object boundaries.

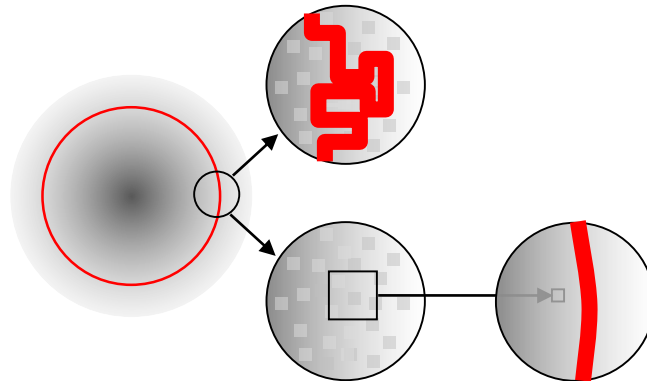


Figure 12 Particle boundary smoothing.

Left: Particle boundary lies in region of image strongly affected by noise.

Top: Thresholding results in jagged boundary.

Bottom: Smoothing the image before thresholding

Right: results in smoother boundary

7. *Min Area*: Used to reject small objects in *CH1*. This is the minimum area of valid circle masks.

One of the results measured in this analysis is the standard deviation of pixel values within a mask. Since the standard deviation is not defined for less than two pixels the minimum area of circle and ring masks is internally constrained to be greater than two pixels.

8. *Max Area*: Used to reject large objects in *CH1*. This is the maximum area of valid circle masks.

The maximum area of circle and ring masks is internally constrained to be less than $image\ width/2 \cdot image\ height/2$.

9. *Min Circularity*: Used to reject *CH1* objects based on shape. Circularity is defined as

$$Circularity = 4\pi \frac{Area}{Perimeter^2},$$

and varies between 0 (fiber) and 1 (circle), see Fig. 14. Since *CH1* objects are assumed to be circular, only the *Min Circularity* parameter is configurable.

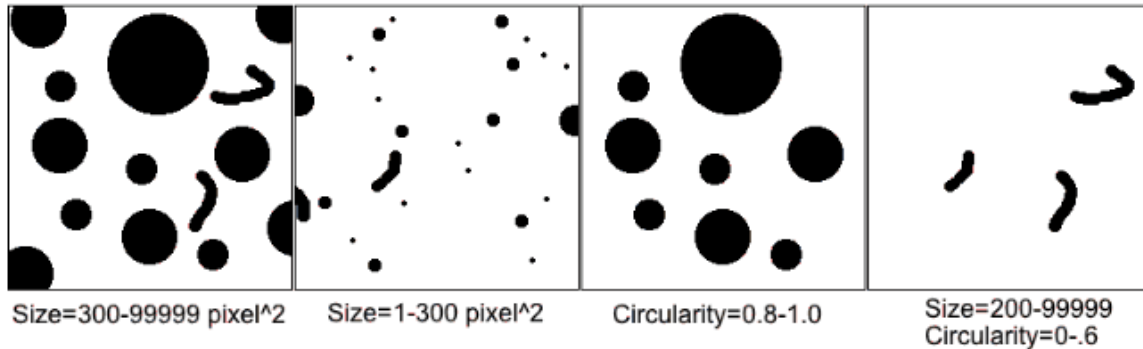


Figure 14 Effect of size and shape parameters on rejection of *CH1* objects.

(Figure from http://imagejdocu.tudor.lu/doku.php?id=gui:analyze:analyze_particles)

10. *Ring Width*: Specifies the width of the ring mask used to measure target intensity in *CH2* that is outside the *CH1* object.

Image Segmentation

Automatic thresholding is used to segment images into objects and background. Both isodata and triangle methods are commonly used in bioassays, see Fig. 15. Isodata is a valley-finding method and is recommended when the number of background and object pixels is comparable ($mBack$ and $mObj$ are the means of the background and objects, respectively). Triangle is a cliff-finding method that is useful when object pixels are sparse.

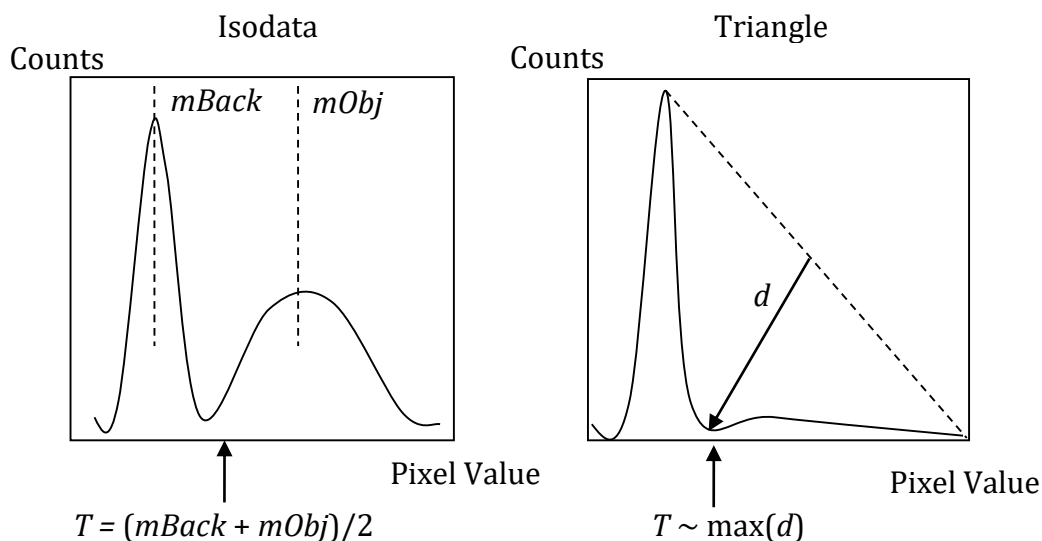


Figure 15 Automatic image segmentation methods.

Appendix B Macro Code

```
//Global variables
//Assay parameters
var nFieldRows;      //int 1 - 8
var nFieldCols;      //int 1 - 8
var analyzeField;    //int 0 - 64
//Channels 1 & 2
var rollingBallRadius; //pixels
var thresholdMethod; //string
var thresholdAdjust;  // % (or lower level if manual threshold method)
//Channel 1
var smoothRadius;    //pixels
var minArea;         //pixels
var maxArea;         //pixels
var minCircularity;  //float 0.0 - 1.0
//Channel 2
var ringWidth;       //pixels
//Flags
var loadDefaults;    //Boolean
var saveDefaults;    //Boolean
var stopAfterLoadingImages; //Boolean
var stopAfterValidObjects; //Boolean
var stopAfterLabeledMasks; //Boolean
var stopForAnalysisReview; //Boolean
//var valid;         //Boolean

//Other globals
//var historyPath = getDirectory("temp") + "\\HCSan2\\HCSan2 History.txt";
var defaultsPath = getDirectory("home") + "\\Documents\\HCSan2\\HCSan2
Default Values.txt";
var imagePath;
var imageTitle;
var imageType;
var analysisDateStamp;
var analysisTimeStamp;
var reportDirectory;
var reportXLS;
var reportFile;
var openReportFile = true; //Boolean
var currentField;
var fieldX1, fieldY1, fieldWidth, fieldHeight; //Field Upper left corner (X1, Y1) and
size - units are pixels
var thresholdLower, thresholdUpper;

macro "HCSan2..." {
  //Get assay parameter values-----
  openReportFile = true;
```

```

do {
    loadDefaultValues();
    showAssayParamsDialog();
    if (loadDefaults) defaultsPath = File.openDialog("Open Defaults File");
    else if (saveDefaults) saveDefaultValues();
} while ( loadDefaults || saveDefaults );
minArea = maxOf(minArea, 2); //Always have at least 2 circ pixels
minRingArea = maxOf(floor(3 * (sqrt(minArea) + ringWidth * ringWidth)), 2);
//Always have at least 2 ring pixels
cleanupAfterAnalysis = !(stopAfterLoadingImages || stopAfterValidObjects ||
stopAfterLabeledMasks);

//Get image path, title, and type
close("*"); //Close all images
run("Open..."); //Propmt user to open either CH1 or CH2 of image
run("Set Scale...", "distance=0 known=0 pixel=1 unit=pixel global"); //Set all units
to pixels
//Get analysis date & time stamp
analysisDateStamp = getDateStamp();
analysisTimeStamp = getTimeStamp();
//Get path
imagePath = File.directory;
imageTitleAndType = getTitle();
//Get title and type
index = lastIndexof(imageTitleAndType, ".");
if (index > 4) {
    imageTitle = substring(imageTitleAndType, 0, index - 4);
    imageType = substring(imageTitleAndType, index + 1,
lengthOf(imageTitleAndType));
    ch1Title = imageTitle + "_CH1";
    ch2Title = imageTitle + "_CH2";

//Open image pair
close("*"); //Close all images
open(imagePath + "/" + ch1Title + "." + imageType);
rename(ch1Title);
open(imagePath + "/" + ch2Title + "." + imageType);
rename(ch2Title);

currentField = 1;
if (nFieldRows == 1 && nFieldCols == 1) {
    fieldX1 = 0;
    fieldY1 = 0;
    fieldWidth = getWidth();
    fieldHeight = getHeight();
    analyze(ch1Title, ch2Title);
}
else {

```



```

    stopAfterFirstField = stopAfterLoadingImages || stopAfterValidObjects ||
stopAfterLabeledMasks;
    stopAfterFirstField = stopAfterFirstField || stopForAnalysisReview ||
analyzeField != 0;
    for (r = 0; r < nFieldRows; r++) {
        for (c = 0; c < nFieldCols; c++) {
            if (analyzeField == 0 || analyzeField == currentField) {
                analyze(cropImage(ch1Title, c, r), cropImage(ch2Title, c, r));
                if (stopAfterFirstField) {
                    r = nFieldRows + 1; //This will stop the row & column loops
                    c = nFieldCols + 1;
                }
            }
            currentField++;
        }
    }
}
else print("Invalid image file name");

if (!openReportFile) File.close(reportFile);
if (cleanupAfterAnalysis) cleanup("Analysis Complete", ch1Title, ch2Title);
}

function analyze(ch1Title, ch2Title) {
    // valid = true;
    if (stopAfterLoadingImages) {
        return;
    }

    //Remove background in both channels
    selectWindow(ch1Title);
    run("Subtract Background...", "rolling=" + rollingBallRadius + " disable"); //Disable
smoothing
    selectWindow(ch2Title);
    run("Subtract Background...", "rolling=" + rollingBallRadius + " disable"); //Disable
smoothing
    //Smooth
    selectWindow(ch1Title);
    run("Duplicate...", "title=Circle");
    //Smooth to aggregate objects and smooth their boundaries
    run("Gaussian Blur...", "sigma=" + smoothRadius);

    //Segment objects from background
    threshold("Circle", thresholdMethod, thresholdAdjust);

    //Create binary masks of valid objects-----
    //Circles

```

```

maxArea = minOf(maxArea, getWidth() / 2.0 * getHeight() / 2.0);
roiManager("reset");
run("Clear Results");
run("Set Measurements...", "redirect=None decimal=2");
run("Analyze Particles...", "size=" + minArea + "-" + maxArea + " pixel circularity="
+ minCircularity + "-1.00 show=Masks clear add exclude include in_situ");
nValidObjects = roiManager("count");
print("nValidObjects " + nValidObjects);

if (stopAfterValidObjects) {
    return;
}

if (nValidObjects > 0) {
    //Voronoi - Used to separate adjacent circles after their dilation
    selectWindow("Circle");
    run("Duplicate...", "title=Voronoi");
    selectWindow("Voronoi");
    run("Options...", "count=1 edm=Overwrite do=Nothing");
    run("Voronoi");
    setThreshold(1, 255);
    run("Convert to Mask");
    //run("Options...", "iterations=1 count=1 edm=Overwrite do=Nothing");
    //run("Dilate");
    run("Invert");

    //Rings
    selectWindow("Circle");
    run("Duplicate...", "title=Ring");
    //Dilate the circles
    run("Maximum...", "radius=" + ringWidth);
    //Separate any that might have touched during dilation
    imageCalculator("AND", "Ring", "Voronoi");

    //Create labeled Voronoi-----
    selectWindow("Voronoi");
    //Set min Voronoi size to minArea pixels to ignore debris created during circ
dilation and Voronoi creation
    roiManager("reset");
    run("Clear Results");
    run("Set Measurements...", "redirect=None decimal=2");
    run("Analyze Particles...", "size=" + minArea + "-Infinity circularity=0.00-1.00
show=Nothing display clear add include in_situ");
    //run("Analyze Particles...", "size=10-Infinity circularity=0.00-1.00 show=Nothing
display clear add include in_situ");
    nVoronoi = roiManager("count");
    print("nVoronoi " + nVoronoi);
    if (nVoronoi != nValidObjects) {

```

```

    // valid = false;
    message = "Field " + currentField + ": nVoronoi (" + nVoronoi + ") does not equal
nValidObjects (" + nValidObjects + "): Analysis aborted";
    if (cleanupAfterAnalysis) {
        cleanup(message, ch1Title, ch2Title);
    }
    return;
}
//Fill Voronoi regions with a gray scale value equal to their ROI index
run("16-bit");
for (i = 0; i < nValidObjects; i++) {
    roiManager("Select", i);
    setColor(i + 1); //Add 1 because index starts at 0, but 0 is background value
    fill();
}
run("Select None");

//Create circle ROI's with Voronoi labels-----
selectWindow("Circle");
roiManager("reset");
run("Clear Results");
run("Set Measurements...", "modal redirect='Voronoi' decimal=0");
run("Analyze Particles...", "size=0-Infinity circularity=0.00-1.00 show=Nothing
clear add include");
//combineROIfragments(0, nVoronoi);
//Change circle ROI labels to Voronoi labels
nCirc = roiManager("count");
print("nCirc " + nCirc);
if (nCirc != nVoronoi) {
    // valid = false;
    message = "(Field " + currentField + "): nCirc (" + nCirc + ") does not equal
nVoronoi (" + nVoronoi + "): Analysis aborted";
    print(message);
    if (cleanupAfterAnalysis) {
        cleanup(message, ch1Title, ch2Title);
    }
    return;
}
circIndices = newArray(nCirc); //Indices of circ masks in ROI Manager
renameROIs('C', 0, nCirc, circIndices);

//Create circle ROI's with Voronoi labels-----
selectWindow("Ring");
run("Set Measurements...", "modal redirect='Voronoi' decimal=0");
run("Analyze Particles...", "size=0-Infinity circularity=0.00-1.00 show=Nothing
add include");
//combineROIfragments(nCirc, nVoronoi);
//Change ring mask labels to Voronoi labels

```

```

nRing = roiManager("count") - nCirc; //Ring indices run nCirc..nRing + nCirc
print("nRing " + nRing);
if (nRing != nVoronoi) {
    //      valid = false;
    message = "(Field " + currentField + "): nRing (" + nRing + ") does not equal
nVoronoi (" + nVoronoi + "): Analysis aborted";
    if (cleanupAfterAnalysis) {
        cleanup(message, ch1Title, ch2Title);
    }
    return;
}
ringIndices = newArray(nRing); //Indices of ring masks in ROI Manager
renameROIs('D', nCirc, nRing, ringIndices); //Use D because it follows C in
alphabetical sort
//The temporary 'D' ROI's have only outer boundaries
//XOR with circle ROI's to make true ring (with inner & outer boundaries)
selectedROIs = newArray(2);
for (i = 0; i < nRing; i++) {
    selectedROIs[0] = i; //circle
    selectedROIs[1] = nCirc + i; //ring      //Ring indices run nCirc..nRing + nCirc
    roiManager("Select", selectedROIs);
    roiManager("XOR");
    roiManager("Add", "ffff00");
    iCombined = nCirc + nRing + i;
    roiManager("Select", iCombined);
    roiManager("Rename", leftPad("R", i + 1, nRing));
}
//Delete temporary 'D' rings
for (i = nCirc; i < nCirc + nRing; i++) { //Ring indices run nCirc..nRing + nCirc
    roiManager("Select", nCirc); //Use nCirc because ROI manager list shifts down as
entries are deleted
    roiManager("Delete");
}

if (stopAfterLabeledMasks) {
    return;
}

//Validate ring and circ labels before performing the analysis
run("Set Measurements...", "modal redirect='Voronoi' decimal=0");
run("Clear Results");
roiManager("select", circIndices);
roiManager("Measure");
roiManager("select", ringIndices);
roiManager("Measure");
for (i = 0; i < nVoronoi; i++) {
    circLabel = getResult("Mode", i);
    ringLabel = getResult("Mode", nCirc + i); //Ring indices run nCirc..nRing + nCirc

```

```

    if (circLabel != ringLabel) {
        //          valid = false;
        message = "Field " + currentField + ": circ and ring labels for object " + i + " do
not agree: Analysis aborted";
        if (cleanupAfterAnalysis) cleanup(message, ch1Title, ch2Title);
        return;
    }
}

//Analyze-----
//Circles - Channel 1
selectWindow(ch1Title);
//run("Set Measurements...", "area feret's shape mean standard min max
integrated add redirect="" + ch1Title + " decimal=2");
run("Set Measurements...", "area feret's shape mean standard min max integrated
add redirect=None decimal=2");
run("Clear Results");
roiManager("select", circIndices);
roiManager("Measure");
run("Remove Overlay");
//saveAs("Measurements", CircleXLS);
circArea = newArray(nCirc);
circFeretDiameter = newArray(nCirc);
circCircularity = newArray(nCirc);
ch1CircMean = newArray(nCirc);
ch1CircStdDev = newArray(nCirc);
ch1CircMin = newArray(nCirc);
ch1CircMax = newArray(nCirc);
ch1CircSumPixels = newArray(nCirc);
for (i = 0; i < nCirc; i++) {
    circArea[i] = getResult("Area", i);
    circCircularity[i] = getResult("Circ.", i);
    circFeretDiameter[i] = getResult("Feret", i);
    ch1CircMean[i] = getResult("Mean", i);
    ch1CircStdDev[i] = getResult("StdDev", i);
    ch1CircMin[i] = getResult("Min", i);
    ch1CircMax[i] = getResult("Max", i);
    ch1CircSumPixels[i] = getResult("RawIntDen", i);
}

//Circles - Channel 2
selectWindow(ch2Title);
run("Set Measurements...", "mean standard min max integrated redirect=None
decimal=2");
run("Clear Results");
roiManager("select", circIndices);
roiManager("Measure");
run("Remove Overlay");

```

```

//saveAs("Measurements", CircleXLS);
ch2CircMean = newArray(nCirc);
ch2CircStdDev = newArray(nCirc);
ch2CircMin = newArray(nCirc);
ch2CircMax = newArray(nCirc);
ch2CircSumPixels = newArray(nCirc);
for (i = 0; i < nCirc; i++) {
    ch2CircMean[i] = getResult("Mean", i);
    ch2CircStdDev[i] = getResult("StdDev", i);
    ch2CircMin[i] = getResult("Min", i);
    ch2CircMax[i] = getResult("Max", i);
    ch2CircSumPixels[i] = getResult("RawIntDen", i);
}

if (cleanupAfterAnalysis) {
    selectWindow("Circle");
    close();
}

//Rings - Channel 2
selectWindow(ch2Title);
//run("Set Measurements...", "area mean standard min max integrated add
redirect="" + ch2Title + "" decimal=2");
run("Set Measurements...", "area mean standard min max integrated add
redirect=None decimal=2");
run("Clear Results");
roiManager("select", ringIndices);
roiManager("Measure");
run("Remove Overlay");
//saveAs("Measurements", ch2RingXLS);
ringArea = newArray(nRing);
ch2RingMean = newArray(nRing);
ch2RingStdDev = newArray(nRing);
ch2RingMin = newArray(nRing);
ch2RingMax = newArray(nRing);
ch2RingSumPixels = newArray(nRing);
for (i = 0; i < nRing; i++) {
    ringArea[i] = getResult("Area", i);
    ch2RingMean[i] = getResult("Mean", i);
    ch2RingStdDev[i] = getResult("StdDev", i);
    ch2RingMin[i] = getResult("Min", i);
    ch2RingMax[i] = getResult("Max", i);
    ch2RingSumPixels[i] = getResult("RawIntDen", i);
}

if (cleanupAfterAnalysis) {
    selectWindow("Ring");
    close();
}

```

```

    selectWindow("Voronoi");
    close();
}

//Display overlays on ch1 & ch2 images-----
roiManager("UseNames", "true");
selectWindow(ch1Title);
run("Overlay Options...", "stroke=red width=2 fill=none apply");
roiManager("Show All with labels");
selectWindow(ch2Title);
run("Overlay Options...", "stroke=green width=2 fill=none apply");
roiManager("Show All with labels");
}
else {
    if (cleanupAfterAnalysis) {
        selectWindow("Circle");
        close();
    }
}

//Create Results table-----
run("Clear Results");
for (i = 0; i < nValidObjects; i++) {
    //setResult("Label",i,circLabel[i]);
    setResult("Circ Area", i, circArea[i]);
    setResult("Circ Circularity", i, circCircularity[i]);
    setResult("Ring Area", i, ringArea[i]);
    setResult("ch1 Circ Mean", i, ch1CircMean[i]);
    setResult("ch1 Circ StdDev", i, ch1CircStdDev[i]);
    setResult("ch1 Circ Sum", i, ch1CircSumPixels[i]);

    //setResult("Ring Label",i,ringLabel[i]);
    setResult("ch2 Circ Mean", i, ch2CircMean[i]);
    setResult("ch2 Circ StdDev", i, ch2CircStdDev[i]);
    setResult("ch2 Circ Sum", i, ch2CircSumPixels[i]);
    setResult("ch2 Ring Mean", i, ch2RingMean[i]);
    setResult("ch2 Ring StdDev", i, ch2RingStdDev[i]);
    setResult("ch2 Ring Sum", i, ch2RingSumPixels[i]);
}
updateResults();

//Create report-----
if (openReportFile) {
    openReportFile = false;
    reportDirectory = imagePath + imageTitle + "_Report" + "_" + analysisDateStamp +
    "_" + analysisTimeStamp;
    File.makeDirectory(reportDirectory)
    reportXLS = reportDirectory + "\\\" + imageTitle + ".xls";
}

```

```

reportFile = File.open(reportXLS);
print(reportFile, "Image:\t" + imagePath + imageTitle + "." + imageType);
print(reportFile, "Field:\t" + currentField + "\n");
print(reportFile, "Analysis Date:\t" + analysisDateStamp + "\n");
print(reportFile, "Analysis Time:\t" + analysisTimeStamp + "\n");
print(reportFile, "\n");

print(reportFile, "Analysis Parameters:\n");
print(reportFile, "Field Rows\t" + nFieldRows + "\n");
print(reportFile, "Field Cols\t" + nFieldCols + "\n");
print(reportFile, "Rolling Ball Radius (pixels)\t" + rollingBallRadius + "\n");
print(reportFile, "Threshold Method\t" + thresholdMethod + "\n");
print(reportFile, "Threshold Adjust (%) \t" + thresholdAdjust + "\n");
print(reportFile, "Threshold Lower\t" + thresholdLower + "\n");
print(reportFile, "Threshold Upper\t" + thresholdUpper + "\n");
print(reportFile, "Smooth Radius (pixels)\t" + smoothRadius + "\n");
print(reportFile, "Min Area (pixels)\t" + minArea + "\n");
print(reportFile, "Max Area (pixels)\t" + maxArea + "\n");
print(reportFile, "Min Circularity\t" + minCircularity + "\n");
print(reportFile, "Ring Width (pixels)\t" + ringWidth + "\n");
print(reportFile, "\n");
}

print(reportFile, "Analysis Field:\t" + currentField + "\n");
print(reportFile, "Upper Left Corner\t (" + fieldX1 + ", " + fieldY1 + ")\n");
print(reportFile, "Width\t" + fieldWidth + "\n");
print(reportFile, "Height\t" + fieldHeight + "\n");
print(reportFile, "\n");

print(reportFile, "Analysis Results:\n");
headerLine = "";
headerLine = headerLine + "ROI\t";
headerLine = headerLine + "Circ Area\t";
headerLine = headerLine + "Circ Feret Diameter\t";
headerLine = headerLine + "Circ Circularity\t";
headerLine = headerLine + "Ring Area\t";
headerLine = headerLine + "ch1 Circ Mean\t";
headerLine = headerLine + "ch1 Circ StdDev\t";
headerLine = headerLine + "ch1 Circ Min\t";
headerLine = headerLine + "ch1 Circ Max\t";
headerLine = headerLine + "ch1 Circ Sum Pixels\t";
headerLine = headerLine + "ch2 Circ Mean\t";
headerLine = headerLine + "ch2 Circ StdDev\t";
headerLine = headerLine + "ch2 Circ Min\t";
headerLine = headerLine + "ch2 Circ Max\t";
headerLine = headerLine + "ch2 Circ Sum Pixels\t";
headerLine = headerLine + "ch2 Ring Mean\t";
headerLine = headerLine + "ch2 Ring StdDev\t";

```



```

headerLine = headerLine + "ch2 Ring Min\t";
headerLine = headerLine + "ch2 Ring Max\t";
headerLine = headerLine + "ch2 Ring Sum Pixels\n";
print(reportFile, headerLine);

for (i = 0; i < nValidObjects; i++) {
    line = "";
    line = line + (i + 1) + "\t";
    line = line + circArea[i] + "\t";
    line = line + circFeretDiameter[i] + "\t";
    line = line + circCircularity[i] + "\t";
    line = line + ringArea[i] + "\t";
    line = line + ch1CircMean[i] + "\t";
    line = line + ch1CircStdDev[i] + "\t";
    line = line + ch1CircMin[i] + "\t";
    line = line + ch1CircMax[i] + "\t";
    line = line + ch1CircSumPixels[i] + "\t";
    line = line + ch2CircMean[i] + "\t";
    line = line + ch2CircStdDev[i] + "\t";
    line = line + ch2CircMin[i] + "\t";
    line = line + ch2CircMax[i] + "\t";
    line = line + ch2CircSumPixels[i] + "\t";
    line = line + ch2RingMean[i] + "\t";
    line = line + ch2RingStdDev[i] + "\t";
    line = line + ch2RingMin[i] + "\t";
    line = line + ch2RingMax[i] + "\t";
    line = line + ch2RingSumPixels[i] + "\t";
    print(reportFile, line);
}
print(reportFile, "\n");

//Save images with ROI's
selectWindow(ch1Title);
ch1Image = reportDirectory + "\\\" + imageTitle + "_CH1_F" + currentField;
saveAs("zip", ch1Image);
rename(ch1Title);
selectWindow(ch2Title);
ch2Image = reportDirectory + "\\\" + imageTitle + "_CH2_F" + currentField;
saveAs("zip", ch2Image);
rename(ch2Title);

if (cleanupAfterAnalysis) {
    if (!stopForAnalysisReview) {
        if (isOpen(ch1Title)) {
            selectWindow(ch1Title);
            close();
        }
        if (isOpen(ch2Title)) {

```

```

        selectWindow(ch2Title);
        close();
    }
}
}
}

/*
function loadHistory() {
    // Get path to temp directory
    hp = getDirectory(historyPath);
    if (hp=="") {
        // Create a history directory in temp
        File.makeDirectory(historyPath);
        if (!File.exists(historyPath + historyName))
            exit("Unable to create history directory");
    }
    if (File.exists(historyName))
        historyString = File.openAsString(defaultsPath + defaultsName);
        historyLines = split(defaultsString, "\n");
        defaultsPath = defaultsLines[0];
        defaultsName = defaultsLines[1];
        imagePath = defaultsLines[2];
    }

function saveHistory(fullName) {
    historyString = "";
    historyString = historyString + defaultsPath + "\n";
    historyString = historyString + defaultsName + "\n";
    historyString = historyString + imagePath + "\n";
    File.saveString(historyString, fullName);
}
*/
}

//Helper Functions-----
function loadDefaultValues() {
    defaultsString = File.openAsString(defaultsPath);
    defaultsLines = split(defaultsString, "\n");
    nFieldRows = parseInt(defaultsLines[0]);
    nFieldCols = parseInt(defaultsLines[1]);
    analyzeField = parseInt(defaultsLines[2]);
    rollingBallRadius = parseInt(defaultsLines[3]);
    smoothRadius = parseInt(defaultsLines[4]);
    thresholdMethod = defaultsLines[5];
    thresholdAdjust = parseFloat(defaultsLines[6]);
    minArea = parseInt(defaultsLines[7]);

```

```

maxArea = parseInt(defaultsLines[8]);
minCircularity = parseFloat(defaultsLines[9]);
ringWidth = parseInt(defaultsLines[10]);
}

```

```

function saveDefaultValues() {
  defaultsString = "";
  defaultsString = defaultsString + nFieldRows + " nFieldRows\n";
  defaultsString = defaultsString + nFieldCols + " nFieldCols\n";
  defaultsString = defaultsString + analyzeField + " analyzeField\n";
  defaultsString = defaultsString + rollingBallRadius + " rollingBallRadius\n";
  defaultsString = defaultsString + smoothRadius + " smoothRadius\n";
  defaultsString = defaultsString + thresholdMethod + "\n";
  defaultsString = defaultsString + thresholdAdjust + " thresholdAdjust\n";
  defaultsString = defaultsString + minArea + " minArea\n";
  defaultsString = defaultsString + maxArea + " maxArea\n";
  defaultsString = defaultsString + minCircularity + " minCircularity\n";
  defaultsString = defaultsString + ringWidth + " ringWidth\n";
  defaultsPath = File.openDialog("Save Defaults File");
  File.saveString(defaultsString, defaultsPath);
}

```

```

function showAssayParamsDialog() {
  Dialog.create("Assay Parameters");
  Dialog.addNumber("# of Field Rows:", nFieldRows, 0, 8, "images");
  Dialog.addNumber("# of Field Cols:", nFieldCols, 0, 8, "images");
  Dialog.addNumber("Analyze Field:", analyzeField, 0, 8, "(0 = All Fields)");
  Dialog.addCheckbox("Stop after Loading Images", false);
  Dialog.addCheckbox("Stop after Valid Objects", false);
  Dialog.addCheckbox("Stop after Labeled Masks", false);
  Dialog.addCheckbox("Stop for Analysis Review", false);
  Dialog.addNumber("Rolling Ball Radius:", rollingBallRadius, 0, 8, "pixels");
  Dialog.addNumber("Smoothing Radius:", smoothRadius, 0, 8, "pixels");
  Dialog.addChoice("Threshold Method:", newArray("MaxEntropy", "Triangle",
"IsoData", "Manual"), thresholdMethod);
  Dialog.addNumber("Threshold Adjust:", thresholdAdjust, 0, 8, "%/level");
  Dialog.addNumber("Min Area:", minArea, 0, 8, "pixels");
  Dialog.addNumber("Max Area:", maxArea, 0, 8, "pixels");
  Dialog.addNumber("Min Circularity:", minCircularity, 2, 8, "");
  Dialog.addNumber("Ring Width:", ringWidth, 0, 8, "pixels");
  Dialog.addCheckbox("Load Default Values", false);
  Dialog.addCheckbox("Save Default Values", false);
  Dialog.show();
  nFieldRows = Dialog.getNumber();
  nFieldCols = Dialog.getNumber();
  analyzeField = Dialog.getNumber();
  stopAfterLoadingImages = Dialog.getCheckbox();
  stopAfterValidObjects = Dialog.getCheckbox();
}

```

```

stopAfterLabeledMasks = Dialog.getCheckbox();
stopForAnalysisReview = Dialog.getCheckbox();
rollingBallRadius = Dialog.getNumber();
smoothRadius = Dialog.getNumber();
thresholdMethod = Dialog.getChoice();
thresholdAdjust = Dialog.getNumber();
minArea = Dialog.getNumber();
maxArea = Dialog.getNumber();
minCircularity = Dialog.getNumber(); //Enforce this value to be between 0.0 and
1.0
ringWidth = Dialog.getNumber();
loadDefaults = Dialog.getCheckbox();
saveDefaults = Dialog.getCheckbox();
}

function cropImage(imageTitle, c, r) {
    colrowString = "" + leftPad("_R", r, nFieldRows) + leftPad("_C", c, nFieldCols);
    selectWindow(imageTitle);
    fieldTitle = imageTitle + colrowString;
    run("Duplicate...", "title=[" + fieldTitle + "]");
    h = getHeight();
    w = getWidth();
    fieldX1 = abs(floor(c * w / nFieldCols));
    fieldY1 = abs(floor(r * h / nFieldRows));
    //print(x1); print(y1);
    fieldWidth = abs(floor(w / nFieldCols));
    fieldHeight = abs(floor(h / nFieldRows));
    //print(width);print(height);
    makeRectangle(fieldX1, fieldY1, fieldWidth, fieldHeight);
    run("Crop");
    return fieldTitle;
}

function threshold(chTitle, method, adjust) {
    selectWindow(chTitle);
    if (method == "Manual") {
        setThreshold(adjust, 255);
    }
    else {
        setAutoThreshold(method + " dark");
        getThreshold(lower, upper);
        lower = lower * (1.0 + adjust / 100.0);
        setThreshold(lower, 255);
    }
    getThreshold(thresholdLower, thresholdUpper);
    run("Convert to Mask");
}

```

```

function leftPad(label, value, maxValue) { //Formats numbers with leading zeros
    maxValueString = "" + maxValue;
    valueString = "" + value;
    for (i = lengthOf(valueString); i < lengthOf(maxValueString); i++) {
        label = label + 0;
    }
    s = label + value;
    return s;
}

```

```

function getDateStamp() {
    getDateAndTime(year, month, dayOfWeek, dayOfMonth, hour, minute, second,
msec);
    dateString = "Y" + year + "M";
    month++;
    if (month < 10) {
        dateString = dateString + "0";
    }
    dateString = dateString + month + "D";
    if (dayOfMonth < 10) {
        dateString = dateString + "0";
    }
    dateString = dateString + dayOfMonth;
    return dateString;
}

```

```

function getTimeStamp() {
    getDateAndTime(year, month, dayOfWeek, dayOfMonth, hour, minute, second,
msec);
    timeString = "h";
    if (hour < 10) {
        timeString = timeString + "0";
    }
    timeString = timeString + hour + "m";
    if (minute < 10) {
        timeString = timeString + "0";
    }
    timeString = timeString + minute + "s";
    if (second < 10) {
        timeString = timeString + "0";
    }
    timeString = timeString + second;
    return timeString;
}

```

```

function combineROIfragments(offset, nVoronoiLabels) {
    //offset skips the first part of the ROI manager list
    nROIfragments = roiManager("count");
}

```

```

//Combine ROI fragments that have identical Voronoi labels
selectedROIs = newArray(2);
//Create a list of combined ROI indices, one for each label
combinedROIIndices = newArray(nVoronoiLabels);
Array.fill(combinedROIIndices, -1); //-1 => no ROI fragment assigned to this label
//Scan list of ROI fragments and assign to label
for (i = offset; i < nROIfragments; i++) {
  roiLabel = getResult("Mode", i);
  if (combinedROIIndices[roiLabel - 1] == -1) {
    //No combined ROI started for this label => assign current fragment
    combinedROIIndices[roiLabel - 1] = i;
  }
  else {
    //Combined ROI already started for this label=> combine current ROI fragment
    with previous one
    selectedROIs[0] = combinedROIIndices[roiLabel - 1];
    selectedROIs[1] = i;
    roiManager("Select", selectedROIs);
    roiManager("Combine");
    roiManager("Add", "ffff00");
    combinedROIIndices[roiLabel - 1] = roiManager("index");
  }
}
//roiManager("index") doesn't work this way
}
//ROI manager contains both original fragments and combined fragments---keep
only combined ones
nROIManagerEntries = roiManager("count");
i = offset;
do {
  if (!isMember(i, combinedROIIndices, nVoronoiLabels)) {
    roiManager("select", i);
    roiManager("Delete");
    nROIManagerEntries--;
  }
  else i++;
} while ( i < nROIManagerEntries );
}

function isMember(target, array, arraySize) {
  flag = false;
  i = 0;
  do {
    if (array[i] == target) flag = true;
    i++;
  } while ( flag == false && i < arraySize );
  return flag;
}

```

```

function renameROIs(ROIType, offset, nVoronoi, arrayIndices) {
  //offset allows skipping first part of ROI manager list
  for (i = 0; i < nVoronoi; i++) {
    arrayIndices[i] = offset + i;
    VoronoiLabel = getResult("Mode", i);
    roiManager("Select", offset + i); //Circle or ring label to be renamed
    roiManager("Rename", leftPad(ROIType, VoronoiLabel, nVoronoi));
  }
  roiManager("Sort"); //Sort ROI manager list by label
}

```

```

function cleanup(message, ch1Title, ch2Title) {
  print(ch1Title + " " + message);
  if (!stopForAnalysisReview) {
    if (isOpen(ch1Title)) {
      selectWindow(ch1Title);
      close();
    }
    if (isOpen(ch2Title)) {
      selectWindow(ch2Title);
      close();
    }
  }
  if (isOpen("Circle")) {
    selectWindow("Circle");
    close();
  }
  if (isOpen("Ring")) {
    selectWindow("Ring");
    close();
  }
  if (isOpen("Voronoi")) {
    selectWindow("Voronoi");
    close();
  }
}

```