//Preprocess\_3D

//input = "C:/Users/User/Downloads/test\_panc1/crop/";

//output = "C:/Users/User/Downloads/test\_panc1/before\_th/";

// test sauvola

//input = "C:/Users/User/Downloads/test\_panc1/before\_th/";

//output = "C:/Users/User/Downloads/test\_panc1/sauvola/";

// test mean

//input = "C:/Users/User/Downloads/test\_panc1/before\_th/";

//output = "C:/Users/User/Downloads/test\_panc1/mean/";

//test Objects\_Counter\_3D

//input = "C:/Users/User/Downloads/test\_panc1/sauvola/";

//output = "C:/Users/User/Downloads/test\_panc1/3D\_object\_count/results/";

//output2 = "C:/Users/User/Downloads/test\_panc1/3D\_object\_count/objects\_map/";

//output3 = "C:/Users/User/Downloads/test\_panc1/3D\_object\_count/final\_th/";

//adapTH\_3D\_sauvola

//input = "C:/Users/User/Desktop/20210812\_PANC1\_redo/before\_th\_no\_bc/";

//output = "C:/Users/User/Desktop/20210812\_PANC1\_redo/sauvola/";

//adapTH\_3D

//input = "C:/Users/User/Desktop/20210812\_PANC1\_redo/before\_th\_no\_bc/";

//output = "C:/Users/User/Desktop/20210812\_PANC1\_redo/mean/";

//Objects\_Counter\_3D

//input = "C:/Users/User/Desktop/20210812\_PANC1\_redo/sauvola/";

//output = "C:/Users/User/Desktop/20210812\_PANC1\_redo/3D\_object\_count/results/";

//output2 = "C:/Users/User/Desktop/20210812\_PANC1\_redo/3D\_object\_count/objects\_map/";

//output3 = "C:/Users/User/Desktop/20210812\_PANC1\_redo/3D\_object\_count/final\_th/";

//do\_skeletonize

//input = "C:/Users/User/Desktop/20210812\_PANC1\_redo/3D\_object\_count/final\_th/";

//output = "C:/Users/User/Desktop/20210812\_PANC1\_redo/skeleton/";

//Analyze\_Skeleton

//input = "C:/Users/User/Desktop/20210812\_PANC1\_redo/skeleton/";

//output = "C:/Users/User/Desktop/20210812\_PANC1\_redo/data/Skeleton\_Results/";

//Analyze\_Particle\_3D

//input = "C:/Users/User/Desktop/20210812\_PANC1\_redo/3D\_object\_count/objects\_map/";

//output = "C:/Users/User/Desktop/20210812\_PANC1\_redo/data/Analyze\_Particle\_3D/";

setBatchMode(true);

list = getFileList(input);

list2 = newArray(list.length);

// delete the ".tif" from the name and save them in the list2 array

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

list2[i] = substring(list[i], 0, lengthOf(list[i]) - 4);

}

// select some of the following for loops to run

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

// preprocess\_2D(input, output, list[i]);

//}

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

// preprocess\_3D(input, output, list[i]);

//}

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

// preprocess\_3D\_old(input, output, list[i]);

//}

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

// bc\_only(input, output, list[i]);

//}

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

// adapTH\_3D\_sauvola(input, output, list[i]);

//}

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

// Analyze\_Particle\_Results(input, output, list[i], list2[i]);

//}

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

// getFinalTH(input, output, list[i], list2[i]);

//}

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

// getFinalTH\_with\_invert(input, output, list[i], list2[i]);

//}

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

// do\_skeletonize(input, output, list[i], list2[i]);

//}

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

// Analyze\_Skeleton(input, output, list[i], list2[i]);

//}

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

// adapTH\_2D\_mean(input, output, list[i]);

//}

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

// adapTH\_2D\_niblack(input, output, list[i]);

//}

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

// adapTH\_2D\_sauvola(input, output, list[i]);

//}

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

// Objects\_Counter\_3D(input, output, list[i], list2[i]);

//}

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

// adapTH\_3D(input, output, list[i]);

//}

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

Analyze\_Particle\_3D(input, output, list[i], list2[i]);

}

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

// get\_threhsold\_for\_skeleton(input, output, list[i], list2[i]);

//}

setBatchMode(false);

function Analyze\_Particle\_Results(input, output, filename, filename\_without\_extension) {

open(input + filename);

run("Analyze Particles...", "size=0.20-Infinity show=Outlines display clear include summarize record in\_situ");

selectWindow("Summary");

saveAs("results", output2 + filename\_without\_extension + "\_Summary" + ".csv");

selectWindow("Results");

saveAs("results", output + filename\_without\_extension + "\_Results" + ".csv");

selectWindow(filename);

saveAs("Tiff", output3 + filename\_without\_extension + "\_Outline" + ".tif");

selectWindow("Results");

run("Close");

selectWindow(filename\_without\_extension + "\_Summary" + ".csv");

run("Close");

}

function getFinalTH(input, output, filename, filename\_without\_extension){

open(input + filename);

run("Analyze Particles...", "size=0.20-Infinity show=Masks clear in\_situ");

selectWindow(filename);

saveAs("Tiff", output + filename\_without\_extension + "\_FinalTH" + ".tif");

run("Close");

}

function getFinalTH\_with\_invert(input, output, filename, filename\_without\_extension){

open(input + filename);

run("Duplicate...", "title=mask.tif");

selectWindow("mask.tif");

run("Analyze Particles...", "size=0.20-Infinity show=Masks clear include in\_situ");

run("Create Selection");

selectWindow(filename);

run("Analyze Particles...", "size=0.20-Infinity show=Masks clear in\_situ");

run("Restore Selection");

run("Invert");

run("Analyze Particles...", "size=0.05-Infinity show=Masks clear in\_situ");

run("Invert");

selectWindow(filename);

run("Select None");

selectWindow("mask.tif");

close();

selectWindow(filename);

saveAs("Tiff", output + filename\_without\_extension + "\_FinalTH" + ".tif");

run("Close");

}

function preprocess\_2D(input, output, filename) {

open(input + filename);

run("Subtract Background...", "rolling=30");

run("8-bit");

run("Add Noise");

run("Sigma Filter Plus", "radius=2 use=2 minimum=0.2 outlier");

run("Enhance Local Contrast (CLAHE)", "blocksize=64 histogram=256 maximum=1.25 mask=\*None\*");

// if you want to do gamma:

// run("Gamma...", "value=0.80");

// run("8-bit");

// if you want to do b&c intensity normalization:

// BCauto();

saveAs("Tiff", output + filename);

close();

}

function preprocess\_3D\_old(input, output, filename) {

open(input + filename);

run("Subtract Background...", "rolling=10");

run("Sigma Filter Plus", "radius=1 use=2 minimum=0.2 outlier");

// CLAHE\_all\_stack\_start

blocksize = 64;

histogram\_bins = 256;

maximum\_slope = 3;

mask = "\*None\*";

fast = true;

process\_as\_composite = true;

getDimensions( width, height, channels, slices, frames );

isComposite = channels > 1;

parameters =

"blocksize=" + blocksize +

" histogram=" + histogram\_bins +

" maximum=" + maximum\_slope +

" mask=" + mask;

if ( fast )

parameters += " fast\_(less\_accurate)";

if ( isComposite && process\_as\_composite ) {

parameters += " process\_as\_composite";

channels = 1;

}

for ( f=1; f<=frames; f++ ) {

Stack.setFrame( f );

for ( s=1; s<=slices; s++ ) {

Stack.setSlice( s );

for ( c=1; c<=channels; c++ ) {

Stack.setChannel( c );

run( "Enhance Local Contrast (CLAHE)", parameters );

}

}

}

// CLAHE\_all\_stack\_end

// if you want to do gamma:

run("Gamma...", "value=0.90");

// if you want to do b&c intensity normalization:

// run("Brightness/Contrast...");

// run("Enhance Contrast", "saturated=0.35");

// run("Apply LUT");

run("8-bit");

BCauto\_3D();

saveAs("Tiff", output + filename);

close();

}

function bc\_only(input, output, filename) {

open(input + filename);

BCauto\_3D();

saveAs("Tiff", output + filename);

close();

}

function preprocess\_3D(input, output, filename) {

open(input + filename);

run("Subtract Background...", "rolling=10");

// run("Add Noise", "stack");

run("Sigma Filter Plus", "radius=2 use=2 minimum=0.2 outlier");

// CLAHE\_all\_stack\_start

blocksize = 64;

histogram\_bins = 256;

maximum\_slope = 1.25;

mask = "\*None\*";

fast = true;

process\_as\_composite = true;

getDimensions( width, height, channels, slices, frames );

isComposite = channels > 1;

parameters =

"blocksize=" + blocksize +

" histogram=" + histogram\_bins +

" maximum=" + maximum\_slope +

" mask=" + mask;

if ( fast )

parameters += " fast\_(less\_accurate)";

if ( isComposite && process\_as\_composite ) {

parameters += " process\_as\_composite";

channels = 1;

}

for ( f=1; f<=frames; f++ ) {

Stack.setFrame( f );

for ( s=1; s<=slices; s++ ) {

Stack.setSlice( s );

for ( c=1; c<=channels; c++ ) {

Stack.setChannel( c );

run( "Enhance Local Contrast (CLAHE)", parameters );

}

}

}

// CLAHE\_all\_stack\_end

// if you want to do gamma:

run("Gamma...", "value=0.90");

// if you want to do b&c intensity normalization:

// run("Brightness/Contrast...");

// run("Enhance Contrast", "saturated=0.35");

// run("Apply LUT");

run("8-bit");

// BCauto\_3D();

saveAs("Tiff", output + filename);

close();

}

function adapTH\_2D\_mean(input, output, filename) {

open(input + filename);

setOption("ScaleConversions", true);

run("8-bit");

run("adaptiveThr ", "using=Mean from=137 then=-23");

run("Despeckle");

run("Remove Outliers...", "radius=2 threshold=50 which=Bright");

saveAs("Tiff", output + filename);

close();

}

function adapTH\_2D\_niblack(input, output, filename) {

open(input + filename);

setOption("ScaleConversions", true);

run("8-bit");

run("Auto Local Threshold", "method=Niblack radius=25 parameter\_1=0.2 parameter\_2=-10 white");

run("Despeckle");

run("Remove Outliers...", "radius=3 threshold=50 which=Bright");

saveAs("Tiff", output + filename);

close();

}

function adapTH\_2D\_sauvola(input, output, filename){

open(input + filename);

setOption("ScaleConversions", true);

//run("8-bit");

run("Auto Local Threshold", "method=Sauvola radius=60 parameter\_1=-0.45 parameter\_2=128 white");

run("Despeckle");

run("Remove Outliers...", "radius=3 threshold=50 which=Bright");

saveAs("Tiff", output + filename);

close();

}

function adapTH\_3D(input, output, filename) {

open(input + filename);

run("8-bit");

run("Auto Local Threshold", "method=Mean radius=50 parameter\_1=-20 parameter\_2=0 white stack");

run("Despeckle", "stack");

run("Remove Outliers...", "radius=2 threshold=50 which=Bright stack");

//run("3D Fill Holes");

saveAs("Tiff", output + filename);

close();

}

function adapTH\_3D\_sauvola(input, output, filename) {

open(input + filename);

run("8-bit");

run("Auto Local Threshold", "method=Sauvola radius=25 parameter\_1=-0.5 parameter\_2=128 white stack");

run("Despeckle", "stack");

run("Remove Outliers...", "radius=3 threshold=50 which=Bright stack");

//run("3D Fill Holes");

saveAs("Tiff", output + filename);

close();

}

function do\_skeletonize(input, output, filename, filename\_without\_extension) {

open(input + filename);

run("Skeletonize (2D/3D)");

saveAs("Tiff", output + filename\_without\_extension + ".tif");

close();

}

function Analyze\_Skeleton(input, output, filename, filename\_without\_extension) {

open(input + filename);

run("Analyze Skeleton (2D/3D)", "prune=none");

selectWindow("Tagged skeleton");

close();

selectWindow("Results");

saveAs("results", output + filename\_without\_extension + "\_Skeleton\_Results" + ".csv");

}

function Objects\_Counter\_3D(input, output, filename, filename\_without\_extension) {

open(input + filename);

run("Duplicate...", "title=dupli1.tif duplicate");

run("Duplicate...", "title=dupli2.tif duplicate");

run("3D OC Options", "volume surface nb\_of\_obj.\_voxels nb\_of\_surf.\_voxels integrated\_density mean\_gray\_value std\_dev\_gray\_value median\_gray\_value minimum\_gray\_value maximum\_gray\_value centroid mean\_distance\_to\_surface std\_dev\_distance\_to\_surface median\_distance\_to\_surface centre\_of\_mass bounding\_box show\_masked\_image\_(redirection\_requiered) dots\_size=5 font\_size=10 show\_numbers white\_numbers store\_results\_within\_a\_table\_named\_after\_the\_image\_(macro\_friendly) redirect\_to=dupli1.tif");

selectWindow("dupli2.tif");

run("3D Objects Counter", "threshold=128 slice=6 min.=60 max.=9524844 objects statistics");

run("3D Objects Counter", "threshold=128 slice=6 min.=60 max.=9524844 objects");

selectWindow("Statistics for " + "dupli2.tif redirect to dupli1.tif");

saveAs("results", output + filename\_without\_extension + "\_Results" + ".csv");

run("Close");

selectWindow("Objects map of " + "dupli2.tif redirect to dupli1.tif");

saveAs("Tiff", output2 + filename);

run("Close");

// save final\_th

selectWindow("Masked image for dupli2.tif redirect to dupli1.tif");

saveAs("Tiff", output3 + filename);

run("Close");

close("\*");

}

function Analyze\_Particle\_3D(input, output, filename, filename\_without\_extension) {

open(input + filename);

run("Analyze Regions 3D", "volume surface\_area mean\_breadth sphericity euler\_number bounding\_box centroid equivalent\_ellipsoid ellipsoid\_elongations max.\_inscribed surface\_area\_method=[Crofton (13 dirs.)] euler\_connectivity=26");

selectWindow("Log");

run("Close");

selectWindow(filename\_without\_extension + "-morpho");

saveAs("results", output + filename\_without\_extension + "\_Analyze\_Particle\_3D" + ".csv");

run("Close");

close();

}

function BCauto(){

AUTO\_THRESHOLD = 5000;

getRawStatistics(pixcount);

limit = pixcount/10;

threshold = pixcount/AUTO\_THRESHOLD;

nBins = 256;

getHistogram(values, histA, nBins);

i = -1;

found = false;

do {

counts = histA[++i];

if (counts > limit) counts = 0;

found = counts > threshold;

} while ((!found) && (i < histA.length-1))

hmin = values[i];

i = histA.length;

do {

counts = histA[--i];

if (counts > limit) counts = 0;

found = counts > threshold;

} while ((!found) && (i > 0))

hmax = values[i];

setMinAndMax(hmin, hmax);

//print(hmin, hmax);

run("Apply LUT");

}

function BCauto\_3D(){

AUTO\_THRESHOLD = 5000;

getRawStatistics(pixcount);

limit = pixcount/10;

threshold = pixcount/AUTO\_THRESHOLD;

nBins = 256;

getHistogram(values, histA, nBins);

i = -1;

found = false;

do {

counts = histA[++i];

if (counts > limit) counts = 0;

found = counts > threshold;

} while ((!found) && (i < histA.length-1))

hmin = values[i];

i = histA.length;

do {

counts = histA[--i];

if (counts > limit) counts = 0;

found = counts > threshold;

} while ((!found) && (i > 0))

hmax = values[i];

setMinAndMax(hmin, hmax);

//print(hmin, hmax);

run("Apply LUT", "stack");

}

// remove small holes to avoid over skeletonized

function get\_threhsold\_for\_skeleton(input, output, filename, filename\_without\_extension) {

open(input + filename);

for (n=1; n<=nSlices; n++) {

setSlice(n);

run("Duplicate...", "title=mask.tif");

selectWindow("mask.tif");

run("Analyze Particles...", "size=0.20-Infinity show=Masks clear include in\_situ");

run("Create Selection");

selectWindow(filename);

run("Analyze Particles...", "size=0.20-Infinity show=Masks clear in\_situ slice");

run("Restore Selection");

run("Invert", "slice");

run("Analyze Particles...", "size=0.05-Infinity show=Masks clear in\_situ slice");

run("Restore Selection");

run("Invert", "slice");

selectWindow(filename);

run("Select None");

selectWindow("mask.tif");

close();

}

selectWindow(filename);

saveAs("Tiff", output + filename\_without\_extension + "\_th\_for\_skeleton" + ".tif");

run("Close");

}

// Closes the "Results" and "Log" windows and all image windows

function cleanUp() {

requires("1.30e");

if (isOpen("Results")) {

selectWindow("Results");

run("Close" );

{

if (isOpen("Log")) {

selectWindow("Log");

run("Close" );

}

while (nImages()>0) {

selectImage(nImages());

run("Close");

}

}