**Supplementary methods**

*Full script for ImageJ analysis of MVD using CD31-stained slides*

// macro count microvesseldensity on CD31 stained slides

// as well as count number of blue (Hematox) stained nuclei

// Author Jeroen A.M. Beliën

// date: 23-08 till 02-09-2010

// updated algorithm 20-01-2014 v4, included the run("Options...",...) and run("Colors...",...), and replaced run("Invert"); on original places

// included relativeprecentage area brown/blue

//

// This is the batch version as originally created 05-05-2014 by JAMB and translated to English 22-10-2018

//

// plugins needed:

// thresholdcolour: with RGBtoLab\_.class/.java in plugin folder

// ij-ImageIO\_.jar needed for run("Open ...")

// folder colour functions that contains Colour\_Deconvolution.class/.java

// ij-plugins\_toolkit.jar needed for multiband Sobel

// Multi\_Thresholder.jar for MultiThresholder tool

var OriginalImage, LumenImage, MeasurementImage, ExclFromMeasurementImage, ExclFromEverythingImage;

var red\_exclude\_low\_th, lumen\_low\_th;

var totarea\_lumen,totarea\_tissue;

var image\_name;

requires("1.44e");

run("Options...", "iterations=1 black count=1");

run("Colors...", "foreground=white background=black selection=yellow");

dir1 = getDirectory("Choose Source Directory ");

dir2 = getDirectory("Choose Destination Directory ");

list1 = getFileList(dir1);

setBatchMode(true);

for (my\_list1=0; my\_list1<list1.length; my\_list1++) {list = getFileList(dir1+list1[my\_list1]);

showProgress(my\_list1, list1.length+1);

if (list.length>0) {

overall\_result\_file\_area\_percentage = File.open(dir2+"Results\_of\_"+"\_"+list[0]+"\_till\_"+list[list.length - 1]+".txt");

print(overall\_result\_file\_area\_percentage, "Imagename;Tot\_area\_brown;Tot\_area\_demarcacted\_area;Percentage");

for (my\_list=0; my\_list<list.length; my\_list++) {

run("Open ...", "image=["+dir1+list1[my\_list1]+list[my\_list]+"] image=["+dir1+list1[my\_list1]+list[my\_list]+"]");

image\_name = File.name;

overall\_result\_file\_vessels = dir2+"Results\_of\_"+image\_name+"\_microvessels.xls";

overall\_result\_file\_nuclei = dir2+"Results\_of\_"+image\_name+"\_nuclei.xls";

OriginalImage = getImageID();

AssessCD31stained\_microvesselsAndNuclei();

//close();

// close all to clean up

wlist = getList("window.titles");

if (wlist.length==0) {

// nothing to do

}

else {

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

//print(" "+wlist[i]);

selectWindow(wlist[i]);

run("Close");

}

}

if (nImages==0) {

// nothing to do

}

else {

while (nImages>0) {

selectImage(nImages);

close();

}

}

}

File.close(overall\_result\_file\_area\_percentage);

}

}

function AssessCD31stained\_microvesselsAndNuclei() {

// Parameters saved are:

// area as Area

// centroid as X Y

// center (center of mass) as XM YM

// perimeter as Perim.

// bounding (bounding rectangle) as BX BY Width Height

// fit (fit elipse) as Major Minor Angle

// shape (shape descriptors) as Circ. AR Round Solidity

red\_exclude\_low\_th=190;

lumen\_low\_th=240;

// set scale so it will not use an old scaling

// depending on scalebar (and compression set when making snapshot or exporting) set it accordingly

// Default -> run("Set Scale...", "distance=0 known=0 pixel=1 unit=pixel global");

run("Set Scale...", "distance=108 known=200 pixel=1 unit=mu global");

// This image will hold the tissue on which measurements will be done

selectImage(OriginalImage);

run("Duplicate...", "title=Measurement\_area");

MeasurementImage = getImageID();

// Get rid of red demarcated areas but keep the orange part

selectImage(OriginalImage);

run("Duplicate...", "title=ExclFromMeasurement\_area"); // orange part

ExclFromMeasurementImage = getImageID();

selectImage(OriginalImage);

run("Duplicate...", "title=ExclFromEverything"); // red part

ExclFromEverythingImage = getImageID();

// select red annotations

selectImage(ExclFromEverythingImage);

min=newArray(3);

max=newArray(3);

filter=newArray(3);

a=getTitle();

run("RGBtoLab ");

run("RGB Stack");

run("Convert Stack to Images");

selectWindow("Red");

rename("0");

selectWindow("Green");

rename("1");

selectWindow("Blue");

rename("2");

min[0]=0;

max[0]=255;

filter[0]="pass";

min[1]=red\_exclude\_low\_th;

max[1]=255;

filter[1]="pass";

min[2]=0;

max[2]=255;

filter[2]="pass";

for (i=0;i<3;i++){

selectWindow(""+i);

setThreshold(min[i], max[i]);

run("Make Binary", "thresholded remaining");

if (filter[i]=="stop") {

run("Invert");

}

}

imageCalculator("AND create", "0","1");

imageCalculator("AND create", "Result of 0","2");

for (i=0;i<3;i++){

selectWindow(""+i);

close();

}

selectWindow("Result of 0");

close();

selectWindow("Result of Result of 0");

rename(a);

// Colour Thresholding------------

run("8-bit");

//setThreshold(10, 109);

//run("Convert to Mask");

run("Fill Holes"); // Detected area inside red demarcated area has value 255

run("Invert"); // So invert (not invert LUT): this area should be excluded from all measurements

// Now threshold lumen

selectImage(OriginalImage);

run("Duplicate...", "title=lumen\_area");

LumenImage = getImageID();

selectImage(LumenImage);

a=getTitle();

// Colour Thresholding------------

run("8-bit");

setThreshold(lumen\_low\_th,255);

run("Convert to Mask");

run("Invert"); // So invert (not invert LUT): Needed to exclude this area from annotations with respect to calculations;

// Also remove area that needs to be excluded from the lumen area

imageCalculator("AND","lumen\_area", "ExclFromEverything");

// Now obtain DAB positive objects

selectImage(OriginalImage);

run("Set Scale...", "distance=0 known=0 pixel=1 unit=pixel global");

run("Duplicate...", "title=[brown]");

// Obtain number of brown objects (microvessels)

run("Colour Deconvolution", "vectors=[H DAB]");

selectWindow("Colour Deconvolution");

close();

selectWindow("brown-(Colour\_3)");

close();

selectWindow("brown-(Colour\_2)");

run("Duplicate...", "title=[brown\_sobel\_input]");

selectWindow("brown-(Colour\_2)");

setThreshold(1, 220);

run("Convert to Mask");

imageCalculator("AND","brown-(Colour\_2)", "lumen\_area");

imageCalculator("AND","brown-(Colour\_2)", "ExclFromEverything");

run("Duplicate...", "title=[Brown\_area]");

// get rid off possible wrong result by dialtion on upper and left part of image (probably edge effects)

selectWindow("brown-(Colour\_2)");

makeRectangle(1, 1, getWidth(), getHeight());

run("Crop");

// seed image

run("Duplicate...", "title=[seed]");

//Make seed binary

selectWindow("seed");

run("8-bit");

setThreshold(1, 255); //create binary image

run("Convert to Mask");

// 04-02-2014 Take brown colour image as sobel input and not original image

selectWindow("brown\_sobel\_input");

run("Multiband Sobel edges");

setAutoThreshold("Default dark");

//setThreshold(100.7446, 1167.7217);

run("8-bit");

run("MultiThresholder", "Moments");

run("Convert to Mask");

run("Invert");

// get rid off possible wrong result by dialtion on upper and left part of image (probably edge effects)

makeRectangle(1, 1, getWidth(), getHeight());

run("Crop");

imageCalculator("AND create", "brown", "brown\_sobel\_input - Sobel edges"); // mask image

run("Duplicate...", "title=[mask]");

//Make mask binary

run("8-bit");

setThreshold(1, 255); //create binary image

run("Convert to Mask");

selectWindow("seed");

run("Duplicate...", "title=[orig\_seed]");

selectWindow("seed");

run("Dilate");

imageCalculator("AND", "seed","mask");

selectWindow("seed");

imageCalculator("Subtract create", "seed", "orig\_seed");

selectWindow("Result of seed");

getHistogram(values,counts,256);

threshold\_counts = counts[255]/200;

if (threshold\_counts < 50) {

threshold\_counts = 50;

}

//print("aantal witte 255 pixels = "+ counts[255]);

selectWindow("Result of seed");

close();

selectWindow("orig\_seed");

close();

while (counts[255] > threshold\_counts) {

selectWindow("seed");

run("Duplicate...", "title=[orig\_seed]");

selectWindow("seed");

run("Dilate");

imageCalculator("AND", "seed","mask");

selectWindow("seed");

imageCalculator("Subtract create", "seed", "orig\_seed");

selectWindow("Result of seed");

getHistogram(values,counts,256);

//print("aantal witte 255 pixels = "+ counts[255]);

selectWindow("Result of seed");

close();

selectWindow("orig\_seed");

close();

}

run("Set Measurements...", "area centroid center perimeter bounding fit shape limit redirect=None decimal=3");

selectWindow("seed");

run("Duplicate...", "title=[Brown\_results]");

selectWindow("seed");

imageCalculator("AND","seed", "lumen\_area");

imageCalculator("AND","seed", "ExclFromEverything");

run("Analyze Particles...", "size=100-Infinity circularity=0.00-1.00 show=Masks display clear");//size from 1000 back to 100 after attempt on analyzing 1:8 scaled image

//print("Number of brown objects corrected = "+nResults);

saveAs("Results", overall\_result\_file\_vessels);

// Parameters returned:

// area as Area

// centorid as X Y

// center (center of mass) as XM YM

// perimeter as Perim.

// bounding (bounding rectangle) as BX BY Width Height

// fit (fit elipse) as Major Minor Angle

// shape (shape descriptors) as Circ. AR Round Solidity

// Obtain number of nuclei

// selectImage(OriginalImage);

// run("Colour Deconvolution", "vectors=[H DAB]");

selectWindow("brown-(Colour\_1)");

run("Duplicate...", "title=[Blue\_area]");

selectWindow("brown-(Colour\_1)");

setAutoThreshold("Default");

setThreshold(0, 100);// 100 almost OK for blue component

run("Convert to Mask");

run("Set Measurements...", "area centroid center perimeter bounding fit shape limit redirect=None decimal=3");

imageCalculator("AND","brown-(Colour\_1)", "lumen\_area");

imageCalculator("AND","brown-(Colour\_1)", "ExclFromEverything");

run("Analyze Particles...", "size=10-Infinity circularity=0.00-1.00 show=Masks display clear"); //reduced 100 size criterium to 10

//print("Number of blue objects uncorrected at threshold 100 = "+nResults);

saveAs("Results", overall\_result\_file\_nuclei);

//Calculate blue and brown tissue area (so lumen and exclude are taken away!!!)

// Brown area is already thresholded at 220

total\_brown\_area=0;

selectWindow("Brown\_area");

// Everything with value 255 (binary image) will be measured as object!

run("Set Measurements...", "area limit redirect=None decimal=3"); // General settings

run("Analyze Particles...", "size=0-Infinity circularity=0.00-1.00 show=Nothing clear");

for (i=0; i<nResults; i++) {

total\_brown\_area = total\_brown\_area + getResult("Area",i);

}

//print("Total area brown staining in mu = "+total\_brown\_area);

//print("Number of brown objects uncorrected at threshold 220 = "+nResults);

// Threshold blue image at 220

selectWindow("Blue\_area");

setThreshold(0, 220);// 220 OK for blue area component

run("Convert to Mask");

run("Invert");

total\_blue\_area=0;

// Everything with value 255 (binary image) will be measured as object!

run("Set Measurements...", "area limit redirect=None decimal=3"); // General settings

run("Analyze Particles...", "size=0-Infinity circularity=0.00-1.00 show=Nothing clear");

for (i=0; i<nResults; i++) {

total\_blue\_area = total\_blue\_area + getResult("Area",i);

}

//print("Total area blue staining in mu = "+total\_blue\_area);

//print("Number of blue objects uncorrected at threshold 220 = "+nResults);

selectImage(LumenImage);

// lumen is already inverted so we obtain tissue/stroma component

// Everything with value 255 (binary image) will be measured as object!

run("Set Measurements...", "area limit redirect=None decimal=3"); // General settings

run("Analyze Particles...", "size=0-Infinity circularity=0.00-1.00 show=Nothing clear");

for (i=0; i<nResults; i++) {

total\_demarcated\_tissue\_area = total\_demarcated\_tissue\_area + getResult("Area",i);

}

//print("Total total\_demarcated\_tissue\_area in mu = "+total\_demarcated\_tissue\_area);

area\_percentage = 0;

area\_percentage = (total\_brown\_area/total\_demarcated\_tissue\_area)\*100;

//print("percentage brown/demarcated\_area = "+area\_percentage);

print(overall\_result\_file\_area\_percentage,dir1+list[my\_list]+";"+total\_brown\_area+";"+total\_demarcated\_tissue\_area+";"+area\_percentage);

}